

Microsoft
.net

Microsoft
.net

Microsoft
.net

CAPÍTULO 1- PLATAFORMA .NET

OBJETIVOS

Conhecer a plataforma DotNet(.net) e suas funcionalidades.

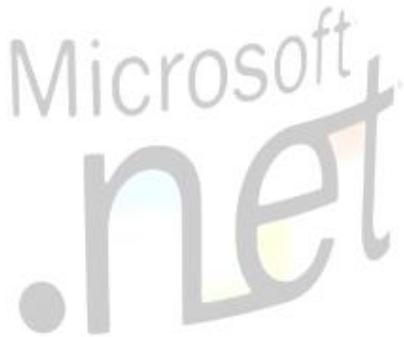
CONTEÚDO

Plataforma .net

.net Framework

Arquitetura .net

CLR (Common Language Runtime)



INTRODUÇÃO

Segundo ACCORSI (2008) a Microsoft .NET é uma iniciativa da Microsoft em que visa uma plataforma única para desenvolvimento e execução de sistemas e aplicações. Todo e qualquer código gerado para .NET, pode ser executado em qualquer dispositivo ou plataforma que possua um framework: a “Plataforma .NET” (.NET Framework). Com idéia semelhante à plataforma Java, o programador deixa de escrever código para um sistema ou dispositivo específico, e passa a escrever para a plataforma .NET.

NET FRAMEWORK E CLR

O que é .NET FRAMEWORK?

O .NET Framework é um modelo de programação de código gerenciado da Microsoft para criar aplicativos em clientes, servidores e dispositivos móveis ou incorporados ao Windows.

- Conjunto rico de bibliotecas com os mais variados usos;
- Facilidade de desenvolvimento de aplicações desde as mais simples até as mais complexas;
- Facilidade na instalação e na distribuição de aplicações;
- Alta escalabilidade para ambientes de missão crítica;
- Interoperabilidade entre plataformas e componentes desenvolvidos em outras linguagens .NET;
- Orientada a objetos;
- Tecnologia baseada em máquina virtual;

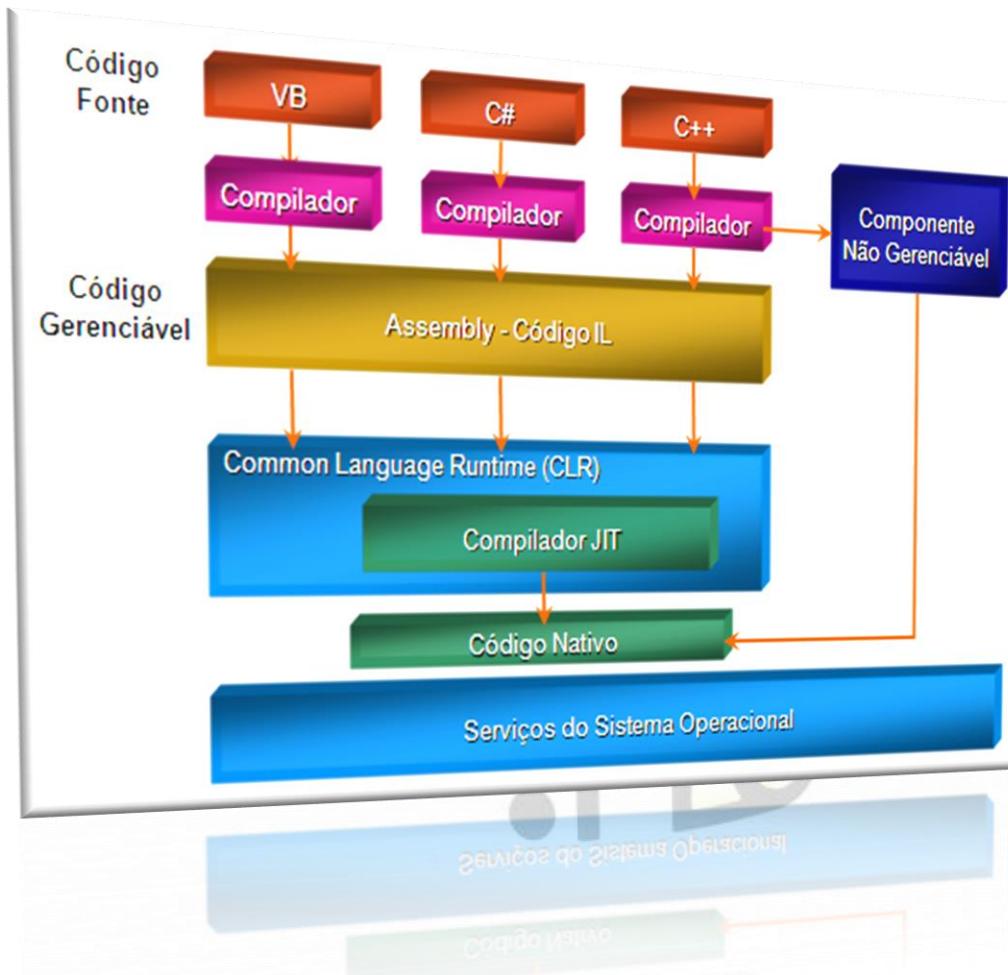
O que posso fazer com o .net?

O .NET permite desenvolver soluções como:

- Aplicativos Web
- Aplicativos para Servidores
- Aplicativos Smart Client
- Aplicativos de Console
- Aplicativos de Banco de Dados
- Serviços Windows
- Web Services e muito mais

ARQUITETURA .NET

O framework é instalado junto com o Visual Studio. A versão 1.1 (segunda versão) é instalada junto com o XP SP1. Usamos o Visual Studio (IDE) somente como uma interface de programação das bibliotecas .NET.

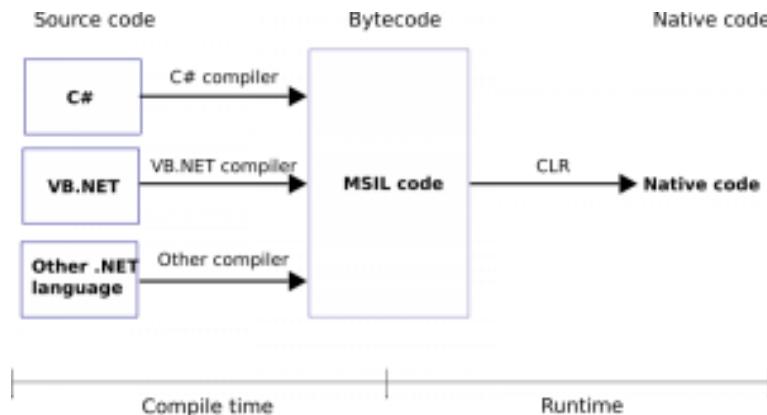


CLR (Common Language Runtime)

A plataforma .NET é executada sobre uma CLR (Common Language Runtime — Ambiente de Execução Independente de Linguagem) interagindo com uma Coleção de Bibliotecas Unificadas, que juntas formam o framework. Esta CLR é capaz de executar, atualmente, mais de vinte diferentes linguagens de programação, interagindo entre si como se fossem uma única linguagem.

O CLR (Common Language Runtime), é uma aplicação similar a uma máquina virtual que se encarrega de providenciar a execução das aplicações para ela escritas. São oferecidos a estas aplicações numerosos serviços que facilitam seu desenvolvimento e manutenção que favorece sua confiança e segurança. O CLR é o verdadeiro responsável pela integração entre as linguagens suportadas pela plataforma .NET. O compilador de cada linguagem segue uma série de padrões (Common Language Specification – CLS) para compilar seus códigos, por

isso as outras linguagens conseguem “entender” as classes e os métodos, dentre outras informações, que essa linguagem definiu.



Serviços presentes no CLR

- Serviços de gerenciamento de memória
- Serviços de tratamento de exceções
- Serviços de compilação
- Serviços de segurança, etc.
- Fornece suporte às linguagens de programação do
 - + .Net:
 - + VB.Net;
 - + C#;
 - + C++;
 - + J#;

Integração Multi-Linguagem

Segundo DANTAS(2006) a integração multi-linguagem é possível através do MSIL (código intermediário) pense na seguinte situação, você cria um componente escrito em C# e quer consumir esse componente em um programa escrito em Visual Basic.NET, isso é possível, pois todo código em .NET é compilado duas vezes, a primeira vez pelo compilador da própria linguagem onde o resultado dessa compilação é um código intermediário (MSIL) e a segunda compilação é feita pelo JIT (Just In Time Compiler) onde o resultado dessa compilação é um código nativo de CPU ai sim esse código é passado para o comando do CLR e o programa é executado, ou seja, no final tudo é IL independente da linguagem.

ATIVIDADES PROPOSTA

OBJETIVO:	Identificar e descrever conceitos sobre a plataforma .net
METODOLOGIA:	Questões dissertativas.
DATA ENTREGA:	DATA: ___/___/ - entrega manual para docente.
ATV_PP.1	

RESPONDA AS SEGUINTE QUESTÕES.

ATIVIDADE 01

01.a O que você entende por .net framework?

01.b Cite 2 exemplos de aplicativos desenvolvidos utilizando a plataforma .net.

01.c O que é CLR? Cite 03 características.

CAPÍTULO 2- WINDOWS FORMS

OBJETIVOS

Conhecer a IDE Windows Forms.

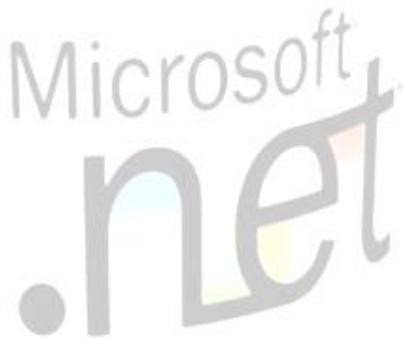
CONTEÚDO

Área de trabalho do Windows Forms

Propriedades, Métodos e Eventos

Objetos:

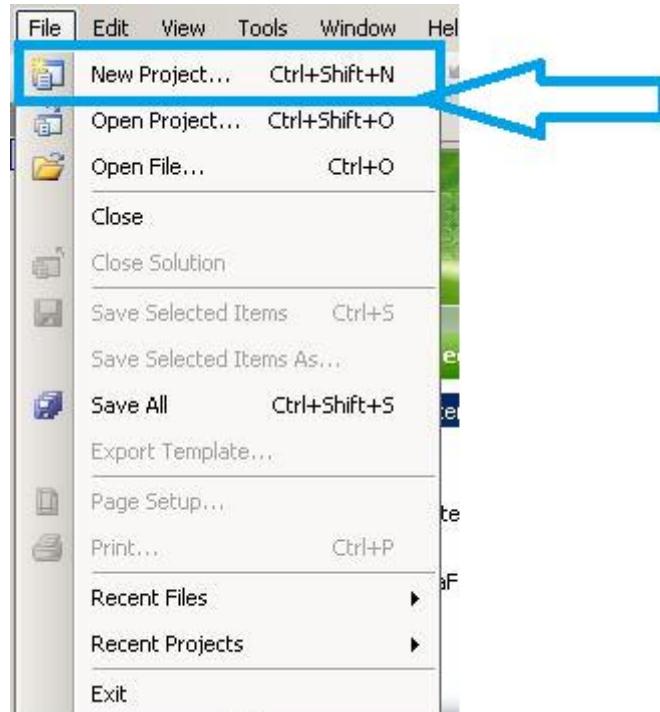
- ✓ Forms
- ✓ TextBox
- ✓ Button
- ✓ Picture Box



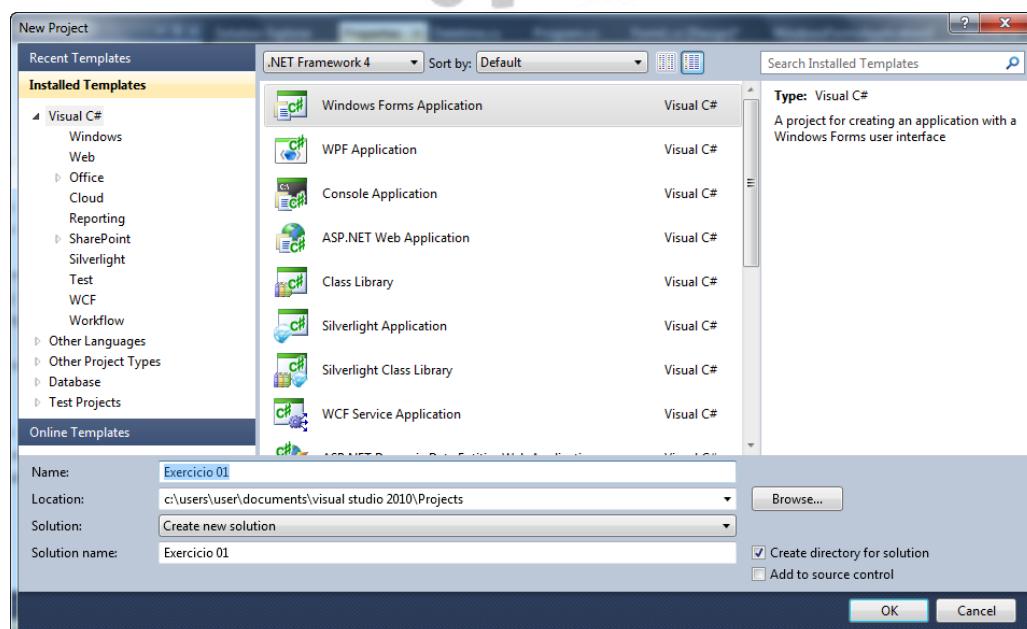
CRIAR UMA SOLUÇÃO

O Visual Studio trabalha com a notação de solução para efetuar um projeto.

CRIAR UMA SOLUÇÃO → FILE/ NEW PROJECT



ESCOLHER O TIPO DE PROJETO → WINDOWS FORMS APPLICATION

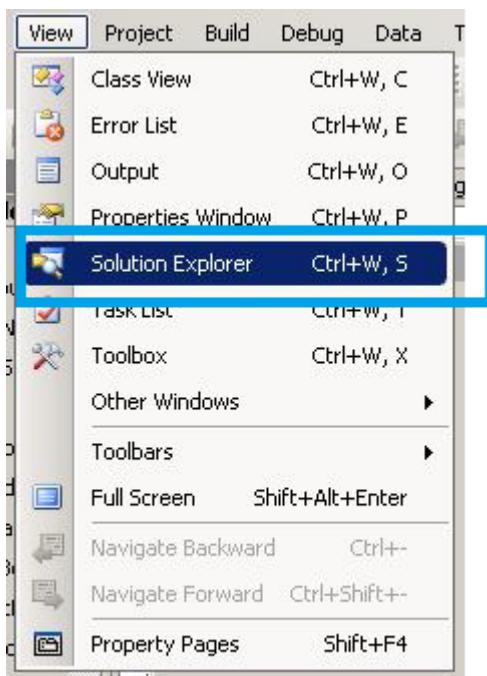


NÃO ESQUEÇA !!!

*Os arquivos das soluções do Visual Studio ficam salvos na pasta
meus documentos/visualstudio2010/projects/nomeprojeto*

*- Quando você for fazer backup de seus projetos seleccione a pasta com
sua solução e efetue a cópia para sua mídia de armazenamento.*

VISUALIZAR A ESTRUTURA DE UMA SOLUÇÃO. → VIEW/SOLUTIONEXPLORER



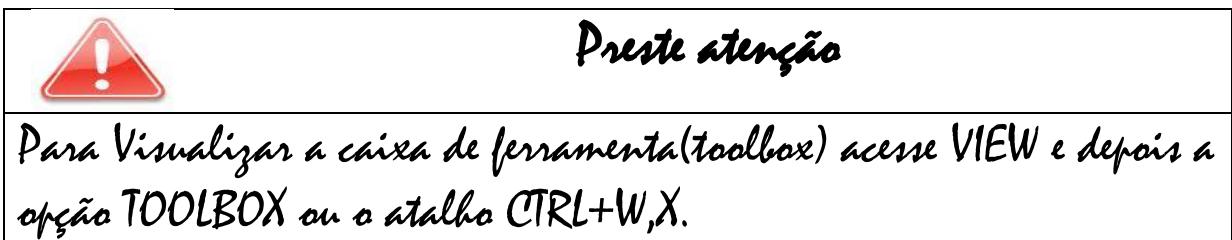
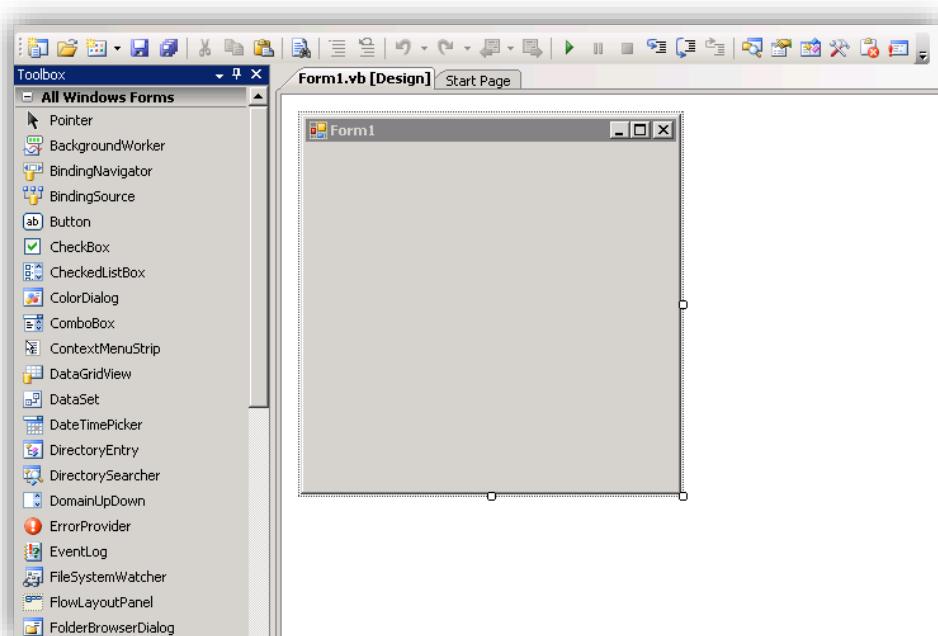
WINDOWS FORMS

O Windows Forms é um editor com os estilos de interação e manipulação direta para criação de aplicações Windows;

- A representação de manipulação da interface (tempo de projeto), é a mesma representação que a aplicação cria (tempo de execução);
- Você "enxerga" o que está programando;
- É difícil de representar algumas situações:
- Usuário age diretamente sobre os objetos da aplicação;
- Geralmente utiliza-se de mouse para drag & drop.

CONTROLES E COMPONENTES

- O Windows Forms é baseado em controles ou componentes que são elementos de uma interface gráfica com os quais o usuário interage de maneira direta;
- Exemplos de Controles: Janelas, Formulários, TextBox, Botão, botão de rádio, caixa de seleção, menu, etc.
- Os componentes no WinForms podem ser arrastados da Toolbox para dentro de um container, como, por exemplo, dentro de um formulário.
- Esses componentes gráficos ficam na *Toolbox*, no lado esquerdo da tela. Expanda a seção “*All Windows Forms*” dessa barra para ver a lista de todos os controles que você tem à disposição (mas nem todos são visíveis em tempo de execução).



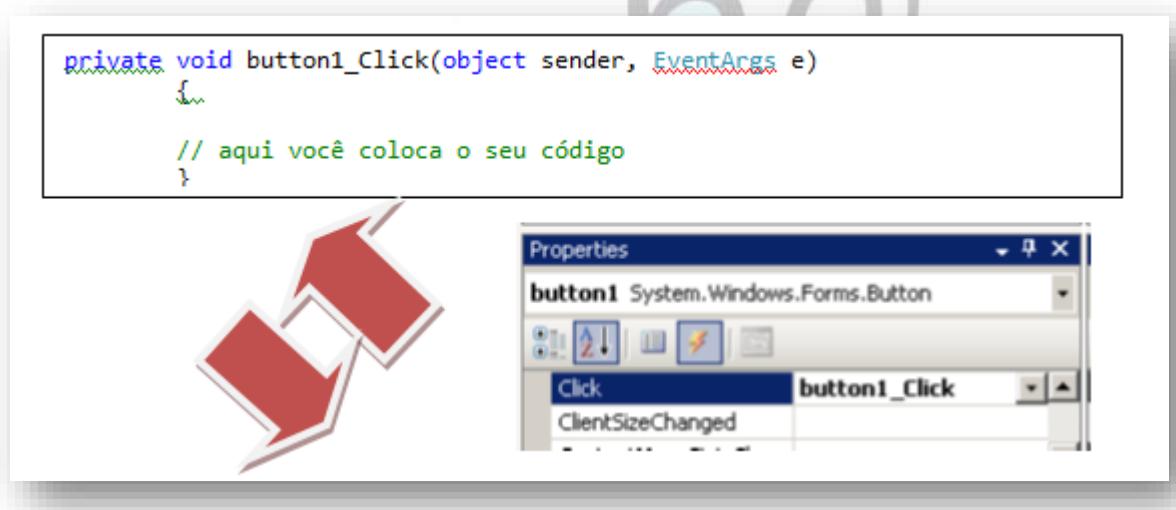
Propriedades, Métodos e Eventos

Observe que todos os componentes podem ser modificados em tempo de projeto através da aba de propriedades ou em tempo de execução, via código;

- Além disso, você pode associar eventos a um ou mais objetos;

Eventos

- Eventos são ações realizadas quando uma certa condição for satisfeita;
- Tal condição pode ser, por exemplo:
 - O clique de um botão;
 - O digitar de uma tecla;
 - O arrastar do componente;
 - A mudança de valor do componente;
- No Visual Studio, ao clicar duas vezes em um componente você gera o evento padrão associado a ele;
- Para visualizar os demais eventos, selecione a aba de eventos;
- O código do evento é gerado na classe [Nome do Form].cs;
- "sender" representa o objeto que chamou o evento;
- "e" contém informações sobre o evento chamado.

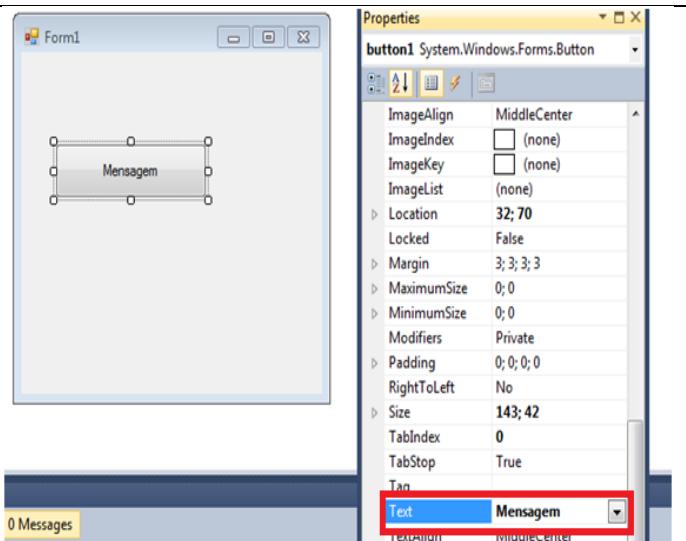


Propriedades

Propriedades são características dos objetos, as quais podem ser alteradas em tempo de design através da caixa de Properties.

Para alterar as propriedades de um objeto basta selecionar o mesmo e alterar a propriedade desejada.

No exemplo ao lado temos o objeto Button, o qual alteramos sua propriedade TEXT para “Mensagem”.



Preste atenção

Para Visualizar a caixa de Properties acesse VIEW e depois a opção PROPERTIES WINDOW ou o atalho CTRL+W,P.

Para ver as propriedades de um objeto, após selecionar o mesmo aperte <F4> e escolha a propriedade desejada; caso queira implementar algum evento clique no botão .

Métodos

Segundo OLIVEIRA(2009) Um método é uma sequência de instruções. Cada método tem um nome e um corpo. O nome deve ajudar a identificar o seu propósito e corpo contém as instruções que vão ser executadas quando o método for chamado.

A maioria dos métodos recebe dados, processa-os e devolve uma informação ou um resultado.

Exemplo de um método:

```
valorDevolvido nomeDoMetodo (listaDeParametros)
```

```
{
```

```
//corpo do método
```

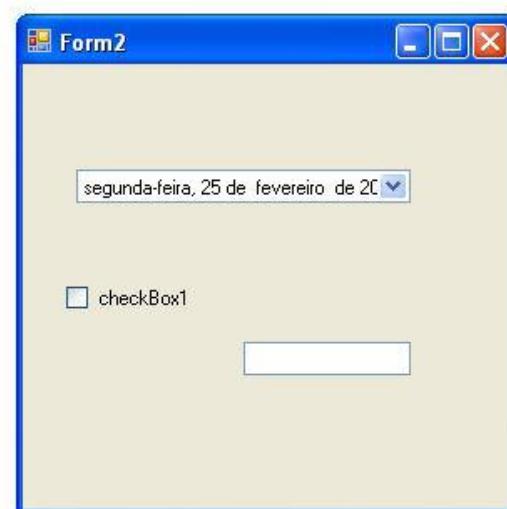
```
}
```

- valorDevolvido - é o tipo de dado que vai ser devolvido pelo método. Pode ser o nome de qualquer tipo, como int ou string. Se o método não devolver nenhum valor tem de ser do tipo void.

- nomeDoMetodo - é o nome que vai ser usado para chamar o método.
- listaDeParametros - onde é inserido o tipo e o nome da variável do método. Este parâmetro é opcional. Se o método tiver mais do que um parâmetro, deve-se separá-los com vírgulas.
- corpo do método - é onde estão as linhas de código, que vão ser executadas quando o método for chamado.

Tipos de Componentes:

- Visuais: caixas de texto, botões;
- Não visuais: dataset, timer, serial port;
- No exemplo ao lado temos 3 componentes visuais e 3 componentes não visuais.



- O WinForms utiliza C# para representar a interface;
- No momento que você arrasta um componente na tela é criado automaticamente um código que gera a interface;
- Esse código é criado dentro da função InitializeComponent dentro do arquivo [Nome do Form].Designer.cs;
- Em C# uma classe pode estar espalhada por mais de um arquivo.
- Cada componente é um objeto de uma classe específica e possui Propriedades, Métodos e Eventos .
- Por exemplo, cada Textbox é uma instância da classe System.Windows.Forms.TextBox;

Objeto: FORM (formulário)

É o principal objeto de uma solução, é onde serão inseridos os demais controles e a partir da onde será executado os projetos.

Principais propriedades do FORM

- ❖ **AutoScroll** : Habilita barra de rolagem se algum componente estiver em algum lugar não visível do form.
- ❖ **BackColor e ForeColor** :Determina as cores do form.
- ❖ **BackgroundImage**: Determina uma imagem como background no form.
- ❖ **ControlBox**: Maximizar, minimizar, fechar e help no caption do form.
- ❖ **FormBorderStyle**: Especifica como a borda do form será apresentada
- ❖ **IsMdiContainer** : Determina se o form é um MDIContainer ou não.
- ❖ **Opacity**: Habilita a transparência do form. 0% é transparente e 100% é opaco.
- ❖ **Size** : Determina as dimensões iniciais do form.
- ❖ **StartPosition**: Posição inicial do form na área de trabalho.
- ❖ **Text**: Texto que será mostrado na caption bar.
- ❖ **Window State**: É o estado inicial da janela (minimizado, normal ou maximizado).

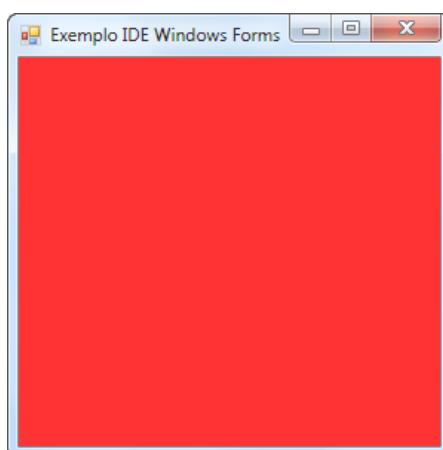
Exemplo 01

Faça um projeto utilizando C# e a IDE Windows Forms alterando com os objetos abaixo e suas propriedade:

Nome da solução: **exemplo01**

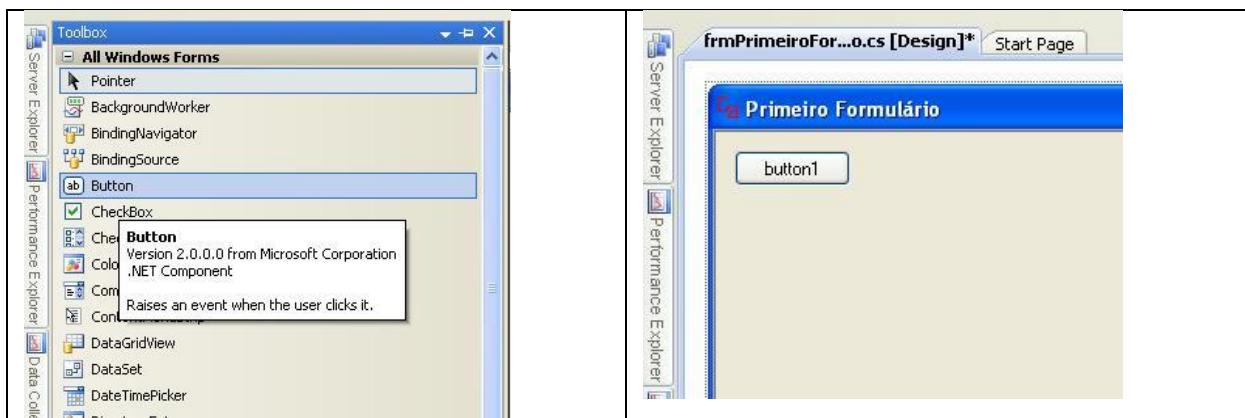
Objeto:	Propriedade:	Valor:
Form	name	frm exemplo 1
	backColor	Red
	Opacity	40%
	Text	Exemplo IDE Windows Forms

Resultado em execução:



Objeto: BUTTON (botão)

Objeto utilizado para disparar ações a solução, como por exemplo, exibir mensagem, efetuar cálculos, abrir formulários entre outros.



Principais propriedades do BUTTON

- ❖ **BackColor:** é a cor de fundo do botão, ou a cor da parede.
- ❖ **ForeColor:** a cor das fontes ou das palavras que aqui escreveremos.
- ❖ **BackgroundImage:** ler imagens do tipo GIF, JPG, BMP e PNG.
- ❖ **BackgroundImageLayout:** define como a figura será carregada no fundo do botão, em modo título, em modo esticado, etc.
- ❖ **Dock:**, define se queremos o componente literalmente grudado em um ou mais cantos.
- ❖ **Enabled:** define se este botão está habilitado ou não, ou seja, se o usuário pode usá-lo ou não.
- ❖ **Image:**, semelhante ao BackgroundImage, só que aqui colocamos uma imagem que não cubrirá o fundo todo, apenas uma pequena imagem, como um ícone.
- ❖ **Location:** define a posição Top (distância do botão em relação a margem superior do formulário) e a posição Left (distância do botão em relação a margem esquerda do formulário)
- ❖ **Name:** é o nome do controle, nome que usaremos para referenciá-lo quando estivermos codificando. Aqui chamei de btnHelloWorld.
- ❖ **Size:** define a largura e a altura do botão.
- ❖ **TabIndex:** define a ordem que este controle está em relação aos demais controles do formulário quando apertamos a tecla TAB para mudar de controle.

- ❖ **TabStop:** define se ao apertarmos a tecla TAB o FOCO irá para neste controle ou se para para o próximo ignorando a TabIndex.
- ❖ **Text:** texto apresentado ao usuário de modo a ele entender para que serve o botão.
- ❖ **Visible:** define se este botão está visível ou não, ou seja, se o usuário pode usá-lo ou não.

Principais eventos do BUTTON

ONCLICK: este evento é disparado, quando o usuário dá um click no botão.

Para acessar o evento de um objeto:

- Selecionar o objeto;
- Pressionar F4 (propriedades);
- Selecionar o “raio” (lista de eventos);
- Clique 2x no evento Click.

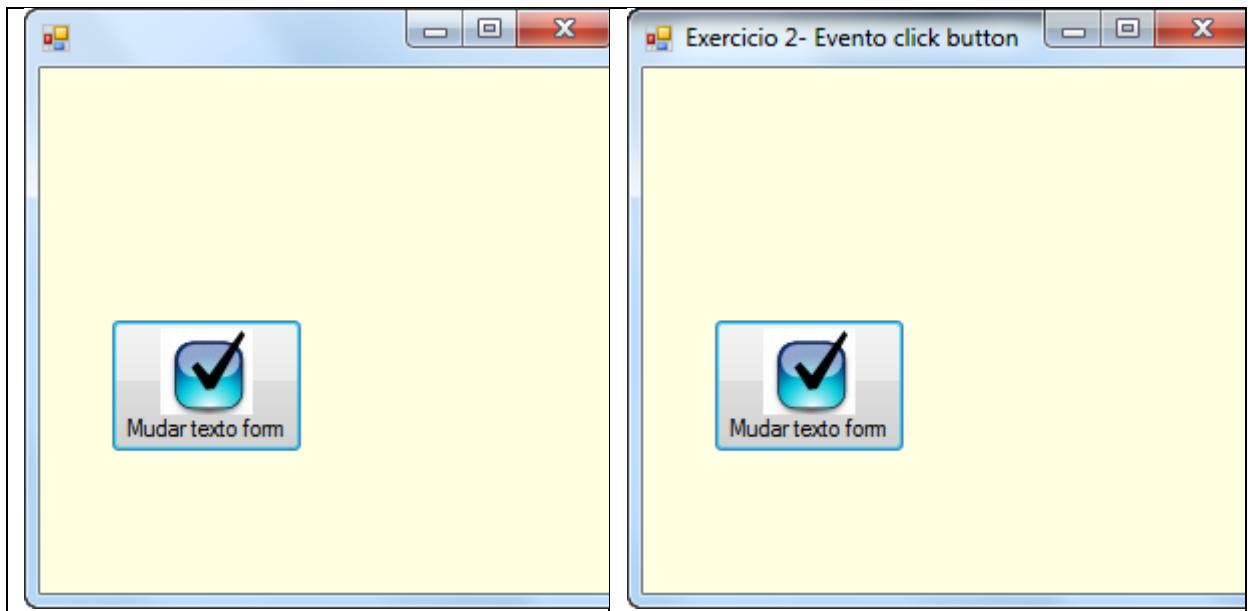
Exemplo 02

Faça um projeto utilizando C# e a IDE Windows Forms com as seguintes características:

Nome da solução: exemplo02

Objeto:	Propriedade:	Valor:
Form	name	Frmexemplo2
	backColor	Info
	Text	<deixar em branco>
Button	name	Btnmsg
	backColor	ActiveCaption
	image	Img01
	Text	Mudar texto form
	TextImageRelation	ImageAboveText

Evento: Quando o usuário clicar no botão o texto do formulário(caixa de título) deverá mudar para “Exercicio 2- Evento click button”.



Após alterar as propriedades dos objetos, precisamos implementar o evento click do botão, para isso então:

- *Selecionar o botão;*
- *Pressionar F4 (propriedades);*
- *Selecionar o “raio” (lista de eventos);*
- *Clique 2 x no evento Click*

Irá abrir a janela para implementação do código direto dentro do evento:

```
WindowsFormsApplication5.Form1
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace WindowsFormsApplication5
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Btnmsg_Click(object sender, EventArgs e)
        {
        }
    }
}
```

Dentro do método iremos digitar o seguinte código:

	<pre>private void Btnmsg_Click(object sender, EventArgs e) { this.Text = "Exercicio 2- Evento click button"; }</pre>
--	--

Explicando o código:

this → quando queremos fazer referência a um formulário em C#, usamos a palavra this.

Text → nome da propriedade do formulário que queremos alterar, no nosso caso a propriedade Text(texto que irá na barra de título).

= → quando queremos atribuir um valor a um objeto ou variável em C# utilizamos o sinal de igual, em seguida colocamos a string. (sempre utilizando “ ” ou o valor.

Para finalizar nossa instrução utilizamos o ponto e vírgula(;).

Resumindo:

Sempre que queremos alterar a propriedade de um objeto, colocamos o nome do objeto(btnclick, label1, etc) . (ponto) e o nome da propriedade(text, background, forecolor, etc) = valor ;(ponto e vírgula)

nome_do_objeto(minúsculo) .(ponto)Propriedade(Maiusculo) =(igual) valor da propriedade;(ponto e vírgula)

btsalvar . BackColor = red ;

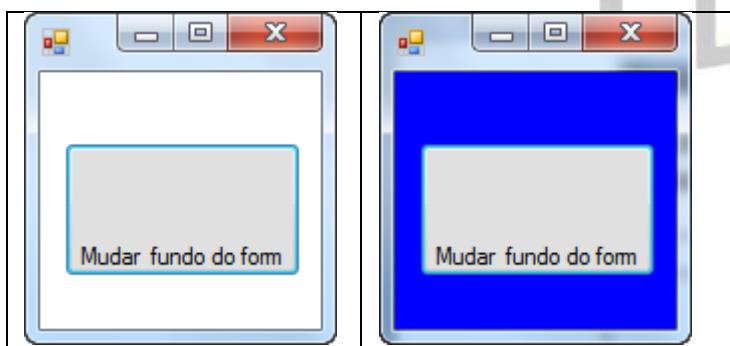
Exemplo 03

Faça um projeto utilizando C# e a IDE Windows Forms com as seguintes características:

Nome da solução: exemplo03

Objeto:	Propriedade:	Valor:
Form	Name	Frmexemplo3
	backColor	White
	Text	Exemplo 3- trabalhando com formulários
Button	Name	Btnalteracor
	backColor	224; 224; 224
	Text	Mudar fundo do form

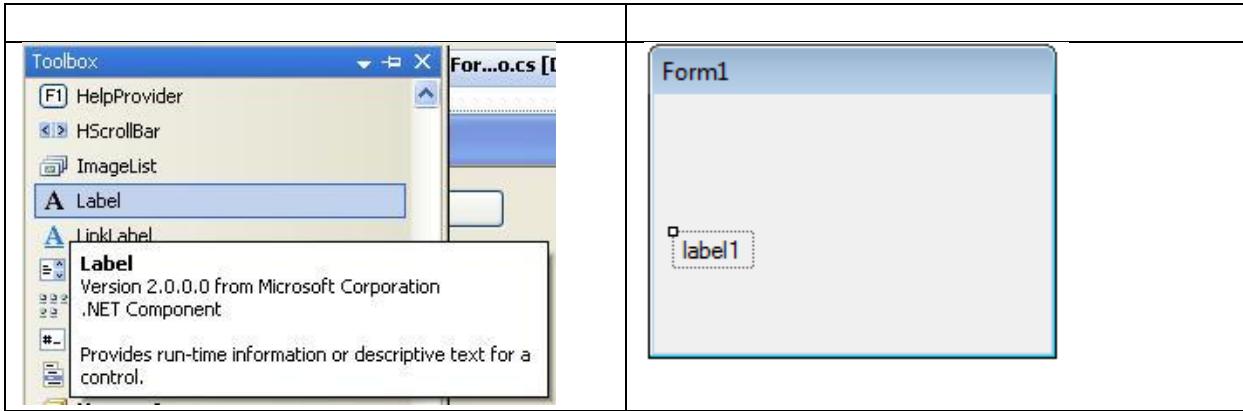
Evento: Quando o usuário clicar no botão irá mudar a cor do fundo do formulário para azul.



```
private void Btnalteracor_Click(object sender, EventArgs e)
{
    this.BackColor = Color.Blue;
}
```

Objeto: LABEL (rótulo)

Objeto utilizado para exibir textos os quais não podem ser selecionado ou alterado pelo usuário.



- ❖ **BackColor:** é a cor de fundo do formulário, ou a cor da parede, e **ForeColor** a cor das fontes ou das palavras que aqui escreveremos.
- ❖ **Dock:** define se queremos o componente literalmente grudado em um ou mais cantos.
- ❖ **Enabled:** define se esta label está habilitada ou não, ou seja, se o usuário pode usá-lo ou não.
- ❖ **Image:** semelhante ao **BackgroundImage**, só que aqui colocamos uma imagem que não cubrirá o fundo todo, apenas uma pequena imagem, como um ícone.
- ❖ **Location:** define a posição **Top** (distância do controle em relação a margem superior do formulário) e a posição **Left** (distância do controle em relação a margem esquerda do formulário)
- ❖ **Name:** é o nome do controle, nome que usaremos para referenciá-lo quando estivermos codificando. Aqui chamei de *lblNome*.
- ❖ **Size:** define a largura e a altura do controle.
- ❖ **Text:** é o texto que está na caixa, através desta propriedade podemos colocar ou pegar a informação.
- ❖ **TextAlign:** define se o texto será colocado a direita ou a esquerda ou centralizado.

Exemplo 04

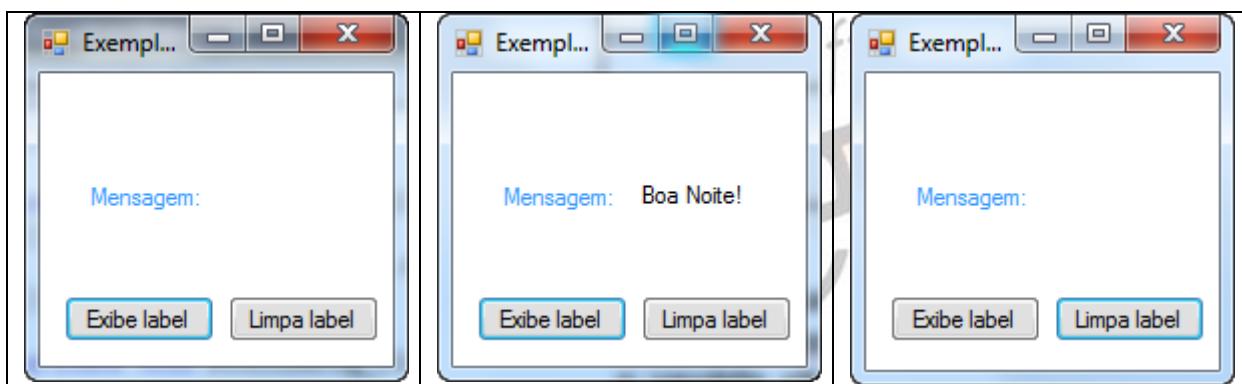
Faça um projeto utilizando C# e a IDE Windows Forms com as seguintes características:

Nome da solução: exemplo04

Objeto:	Propriedade:	Valor:
Form	Name	Frmexemplo4
	backColor	White
	Text	Exemplo 4- trabalhando com label
Button	Name	BtnExibeMsg
	backColor	224; 224; 224

	Text	Exibe label
Button	Name	BtnLimpaTexto
	backColor	224; 224; 224
	Text	Limpar label
Label	Name	lblindicanome
	ForeColor	Highlight
	Text	Mensagem:
Label	Name	lblvalnome
	ForeColor	Black
	Text	<deixar sem valor>

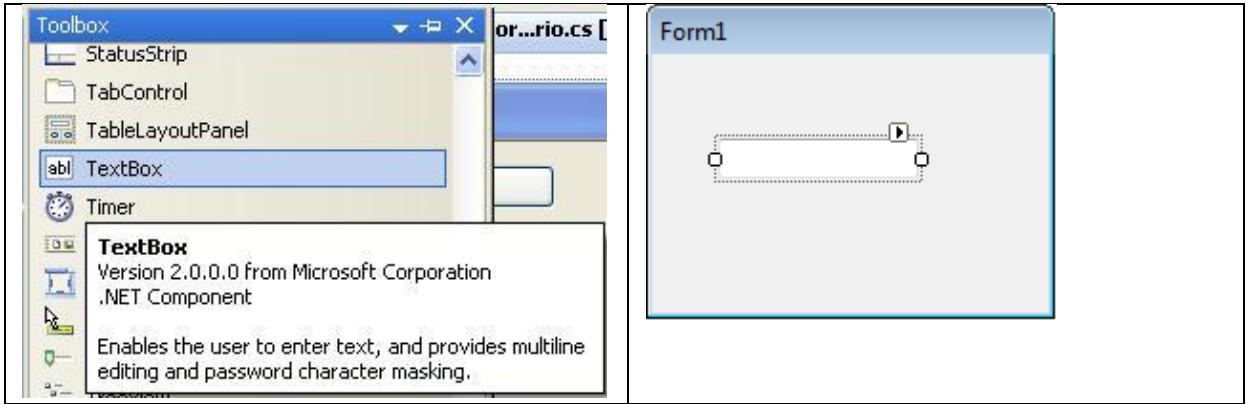
Evento: Quando o usuário clicar no primeiro botão a segunda label(lblvalnome) irá receber um texto com a mensagem “Boa Noite!”, quando o usuário clicar no segundo botão o conteúdo da segunda label(lblvalnome) irá voltar ao normal, isto é, ficará sem conteúdo.



	<pre> private void BtnExibeMsg_Click(object sender, EventArgs e) { lblvalnome.Text = "Boa Noite!"; } private void BtnLimpaTexto_Click(object sender, EventArgs e) { lblvalnome.Text = ""; } </pre>
--	---

Objeto: TEXTBOX (caixa de texto)

Permite a apresentação e a entrada de dados do usuário.



- ❖ **BackColor:** é a cor de fundo do formulário, ou a cor da parede, e **ForeColor** a cor das fontes ou das palavras que aqui escreveremos.
- ❖ **CharacterCasing:** muda a forma como o texto é apresentado, todo o texto em letras maiúsculas, ou letras minúsculas, ou como o usuário digitar.
- ❖ **Dock:** define se queremos o componente literalmente grudado em um ou mais cantos.
- ❖ **Enabled:** define se esta **TextBox** está habilitada ou não, ou seja, se o usuário pode usá-la ou não.
- ❖ **Image:** semelhante ao **BackgroundImage**, só que aqui colocamos uma imagem que não cubrirá o fundo todo, apenas uma pequena imagem, como um ícone.
- ❖ **Location:** define a posição **Top** (distância do **TextBox** em relação a margem superior do formulário) e a posição **Left** (distância do **TextBox** em relação a margem esquerda do formulário)
- ❖ **MaxLength:** tamanho máximo em número de caracteres.
- ❖ **Multiline:** abrange o uso da tecla <enter> para que seja possível o uso de quebra de linha.
- ❖ **Name:** é o nome do controle, nome que usaremos para referenciá-lo quando estivermos codificando. Aqui chamei de **txtNome**.
- ❖ **PasswordChar:** caractere que é utilizado quando estamos usando o **TextBox** para coletar informações de senha, ou seja, quando não queremos mostrar o que está sendo escrito.
- ❖ **ReadOnly:** mostra o texto mas não permite que seja alterado.
- ❖ **TabIndex:** define a ordem que este controle está em relação aos demais controles do formulário quando apertamos a tecla TAB para mudar de controle.

Principais eventos do TEXTBOX

CLICK: este evento é disparado quando o usuário dá um click na caixa de texto;

ENTER: este evento é disparado quando o usuário entra na caixa de texto.

MOUSELEAVE: este evento é disparado quando o usuário tira o mouse da caixa de texto.

MOUSEMOVE: este evento é disparado quando o usuário passa o mouse sobre a caixa de texto.

TEXTCHANGED: este evento é disparado, quando o usuário está editando o seu conteúdo(texto).

Exemplo 05

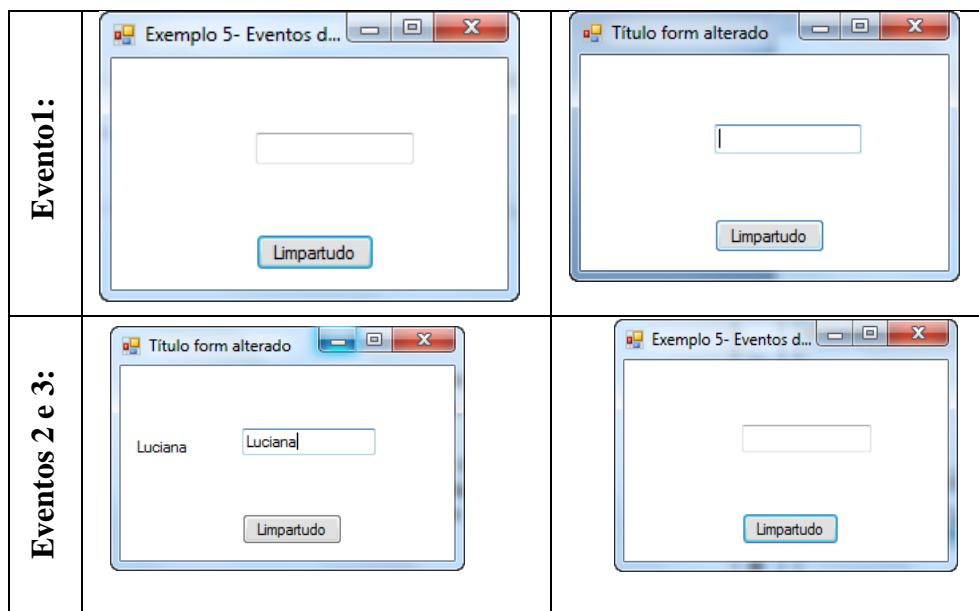
Faça um projeto utilizando C# e a IDE Windows Forms com as seguintes características:

Nome da solução: exemplo05

Objeto:	Propriedade:	Valor:
Form	Name	Frmexemplo5
	backColor	White
	Text	Exemplo 5- Eventos da textbox
Label	Name	lblconteudo
	ForeColor	Black
	Text	<deixar sem valor>
Textbox	Name	txtnome
	ForeColor	Black
	Text	<deixar sem valor>
Button	Name	btnclick
	ForeColor	Black
	Text	Limpar tudo

Eventos:

- 1. Quando o usuário clicar na caixa de texto(txtnome) o texto da barra de título do formulário irá mudar para: “Título form alterado”.**
- 2. Quando o usuário estiver digitando na caixa de texto , a label(lblconteudo) irá receber o conteúdo da caixa de texto.**
- 3. Quando o usuário clicar no botão limpar os conteúdos voltarão ao normal, isto é, o título do form voltará para “Exemplo 5- Eventos da textbox”, o texto da label voltará a ficar sem valor, e a caixa de texto ficará sem valor.**

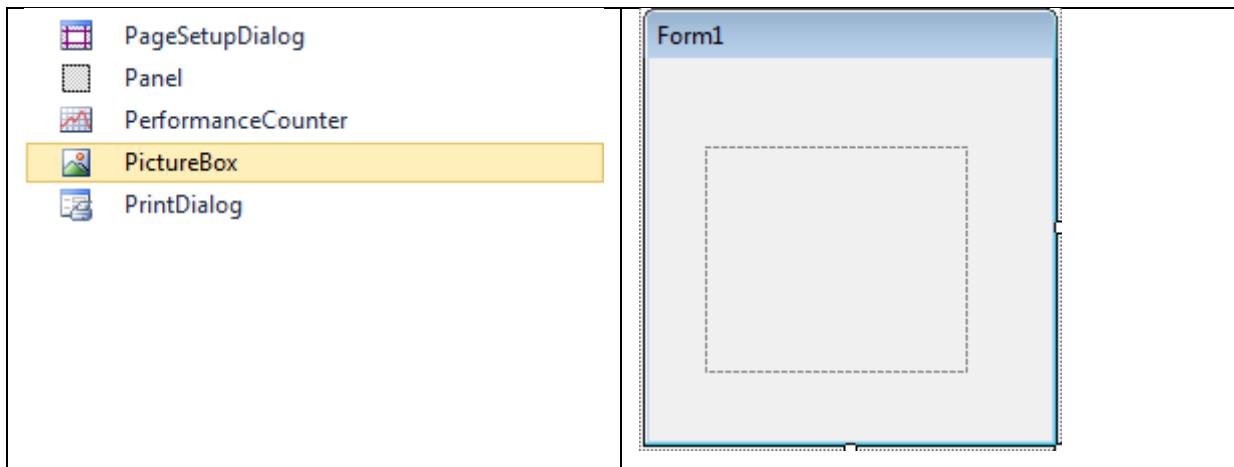


Codificação:

- ✓ No item 1, para que o título do form seja alterado quando o usuário clicar na caixa de texto utilizamos o evento CLICK da caixa de texto(txtname).
- ✓ No item 2, para o conteúdo da label receber o conteúdo da caixa de texto enquanto estamos digitando algo na caixa utilizamos o evento TEXTCHANGED da caixa de texto(txtname).
- ✓ No item 3 utilizamos o evento CLICK do botão para limpar os conteúdos da label(lblconteudo) e da textbox(txtname) e voltar o titulo anterior do form(frm exemplo5).

	<pre> private void BtnLimpaTexto_Click(object sender, EventArgs e) { this.Text = "Exemplo 5- Eventos da textbox"; lblconteudo.Text = ""; txtname.Text = ""; } private void textBox1_TextChanged(object sender, EventArgs e) { lblconteudo.Text = txtname.Text; } private void textBox1_Enter(object sender, EventArgs e) { this.Text = "muda conteudo"; } </pre>
---	--

Objeto: PICTUREBOX (caixa de imagem)



Este controle é utilizado para quando se tem o objetivo de mostrar imagens, fazendo a leitura a partir de um arquivo. É então possível posicionar a imagem num local desejado ou redimensioná-la de acordo com a área pretendida.

- ❖ **Name:** é o nome do controle, nome que usaremos para referenciá-lo quando estivermos codificando.
- ❖ **Image:** Define a imagem a carregar no controle.
- ❖ **SizeMode:** Define como a imagem será carregada no formulário.
- ❖ **Top:** Define a altura(inferior/superior) do objeto sobre o formulário.
- ❖ **Left:** Define a posição(direita/esquerda) do objeto sobre o formulário.

Exemplo 06

Faça um projeto utilizando C# e a IDE Windows Forms com as seguintes características:

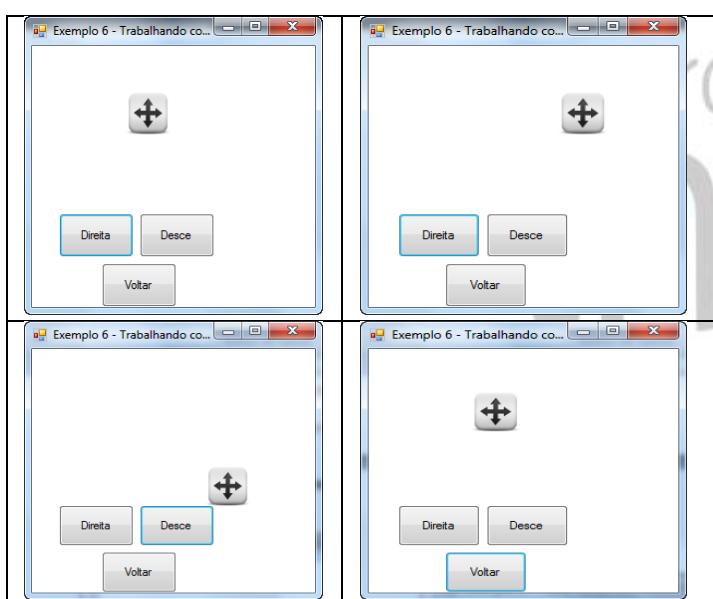
Nome da solução: exemplo06

Objeto:	Propriedade:	Valor:
Form	Name	Frmexemplo06
	backColor	White
	Text	Exemplo 6- Trabalhando com imagens
PictureBox	Name	picImagen1
	Image	Move.png
	SizeMode	StretchImage
	Location	88;39
Button	Name	btndesce
	ForeColor	Black
	Text	Desce
Button	Name	btndireita

	ForeColor	Black
	Text	Direita
Button	Name	btndesce
	ForeColor	Black
	Text	Desce

Eventos:

1. Quando o usuário clicar no primeiro botão(btndesce) a imagem irá descer.
2. Quando o usuário clicar no segundo botão(btndireita) a imagem irá se mover à direita.
3. Quando o usuário clicar no terceiro botão(btninicio) a imagem voltará a posição anterior.



Codificação:

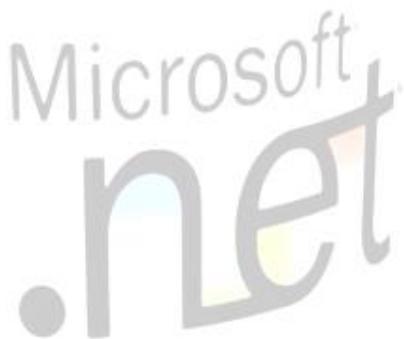
- ✓ Para mover os objetos usamos as propriedades **left**(esquerda/direita) e **top**(subir/descer).



```
private void btndireita_Click(object sender, EventArgs e)
{
    picImagen1.Left = picImagen1.Left + 80;
}

private void btndesce_Click(object sender, EventArgs e)
{
    picImagen1.Top = picImagen1.Top + 80;
}

private void btninicio_Click(object sender, EventArgs e)
{
    picImagen1.Left = picImagen1.Left - 80;
    picImagen1.Top = picImagen1.Top - 80;
}
```



CONCATENAR TEXTOS/VALORES EM C#.

Durante o desenvolvimento dos projetos em diversas linguagens, em determinadas ocasiões é preciso concatenar(juntar) textos/valores.

No C# utilizamos o sinal de adição(+) para efetuar esta operação.

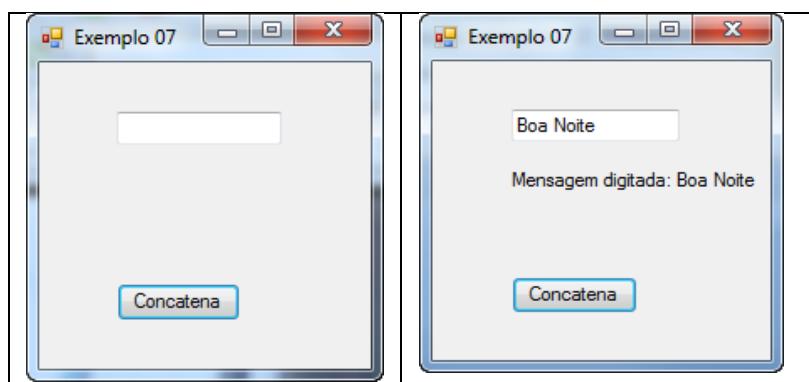
Exemplo 07

Faça um projeto utilizando C# e a IDE Windows Forms com as seguintes características:

Objeto:	Propriedade:	Valor:
Form	Name	Frmexemplo7
	backColor	White
	Text	Exemplo 7- Concatenando valores
Textbox	Name	txtmensagem
	Text	<deixar em branco>
Label	Name	lblrecebemsg
	ForeColor	Black
	Text	<deixar em branco>
Button	Name	btnconcatena
	ForeColor	Black
	Text	Concatena Valores

Eventos:

1. O usuário vai digitar uma mensagem na caixa de texto(txtmensagem);
2. Quando o usuário clicar no botão(btnconcatena); no texto da label(lblrecebemsg) vai aparecer o texto “Mensagem digitada: ” e a mensagem digitada pelo usuário.



	<pre>private void btnconcatena_Click(object sender, EventArgs e) { lblrecebemsg.Text = "Mensagem digitada: " + txtmensagem.Text; }</pre>
--	--

OBJETIVO:	Propriedades e eventos objetos
METODOLOGIA:	Desenvolvimento de projetos
DATA ENTREGA:	DATA: ___ / ___ / ___
LISTA_ATV_PP.2	

Desenvolva os seguintes projetos no Visual Studio, utilizando a linguagem C#.

ATIVIDADE 02

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- Form
- Button
- Label

Eventos

- Quando o usuário clicar no botão irá aparecer uma mensagem na label, que inicialmente estará sem texto.

ATIVIDADE 03

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- Form
- 05 Button's (4 button para alterar as cores do form e 1 button para voltar a cor original)

Eventos

- Quando o usuário clicar em cada um dos 4 botões irá mudar a cor do form.(a cor ficará a sua escolha)
- Quando o usuário clicar no ultimo botão a cor inicial do form irá voltar a cor original.



ATIVIDADE 04

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

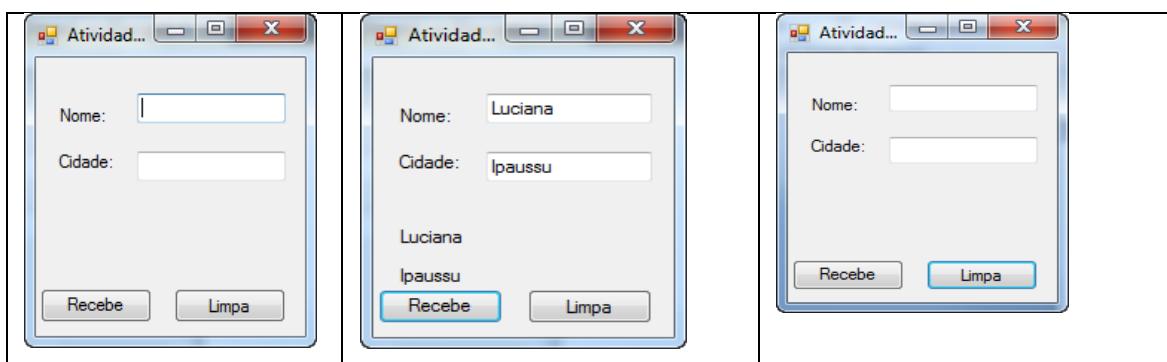
- 01 Form
- 04 label
- 02 textbox
- 02button

Configure os objetos conforme descrição abaixo:

Objeto:	Propriedade:	Valor:
Form	Name	Frmatv04
	backColor	Control
	Text	Atividade 04
Label	Name	Lblnome
	ForeColor	Black
	Text	Nome:
Label	Name	Lblcidade
	ForeColor	Black
	Text	Cidade:
Textbox	Name	Txt nome
	Text	<deixar em branco>
Textbox	Name	Txt cidade
	Text	<deixar em branco>
Button	Name	btnRecebe
	ForeColor	Black
	Text	Recebe valor
Button	Name	btnLimpar
	ForeColor	Black
	Text	Limpar

Eventos

- O usuário deverá digitar seu nome e sua cidade nas caixas de texto(txt nome, txt cidade).
- Quando o usuário clicar no botão(btnrecebe) as label's(lblcidade, lblnome) irá receber os conteúdos das caixas de texto correspondente.
- Quando o usuário clicar no botão(btnlimpar) as caixas de textos e as labels que receberam o nome e a cidade deverão ficar sem valor.



ATIVIDADE 05

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 04 button
- 01 picturebox

Configure os objetos conforme descrição abaixo:

Objeto:	Propriedade:	Valor:
Form	Name	FrmAtv05
	backColor	White
	Size	550;380
	StartPosition	CenterScreen
	Text	Atividade 05
Button	BackColor	LightSkyBlue
	BackgroundImageLayout	Center
	Image	SetaCima.bmp
	Name	btnSobe
Button	BackColor	LightSkyBlue
	BackgroundImageLayout	Center
	Image	SetaBaixo.bmp
	Name	btnDesce
Button	BackColor	LightSkyBlue
	BackgroundImageLayout	Center
	Image	SetaEsquerda.bmp
	Name	btnEsquerda
Button	BackColor	LightSkyBlue
	BackgroundImageLayout	Center
	Image	SetaDireita.bmp
	Name	btnDireita
PictureBox	Name	picNave
	BackColor	White

Eventos

- Quando o usuário clicar nos botões a nave deverá se deslocar para cima,baixo,direita e esquerda de acordo com cada botão.

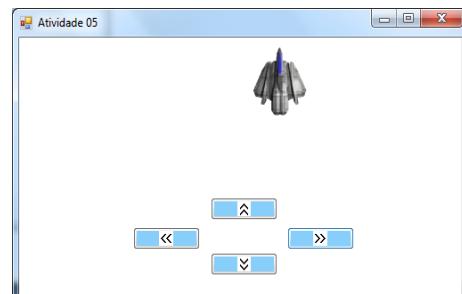
ATIVIDADE 06

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 03 label
- 03 button
- 03 textbox

Configure os objetos conforme descrição abaixo:



Objeto:	Propriedade:	Valor:
Form	Name	FrmAtv06
	backColor	White
	StartPosition	CenterScreen

	Text	Atividade 06
Button	BackColor	Control
	Text	>>
	Name	Btnval1
	BackColor	Control
Button	Text	>>
	Name	Btnval1
	BackColor	Control
	Text	>>
Button	Name	Btnval2
	Text	<deixar em branco>
Label	Name	Lblval1
	Text	<deixar em branco>
Label	Name	Lblval2
	Text	<deixar em branco>
Label	Name	Lblval3
	Text	<deixar em branco>
TextBox	Name	txtval1
	Text	<deixar em branco>
TextBox	Name	txtval2
	Text	<deixar em branco>
TextBox	Name	txtval3
	Text	<deixar em branco>

Eventos

- O usuário vai digitar 03 valores nas caixas de texto(txtval1,txtval2,txtval3);
- Quando o usuário clicar nos botões de cada caixa de texto , a label correspondente irá receber o texto “Valor digitado” mais o valor digitado.



3

CAPÍTULO

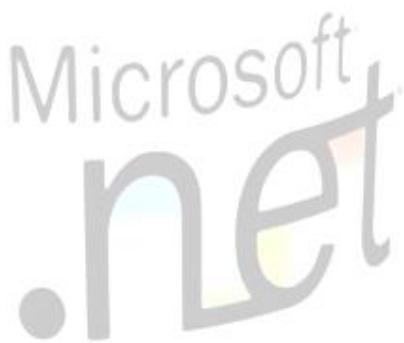
CAPÍTULO 3- VARIÁVEIS EM C#

OBJETIVOS

Conhecer os tipos de dados e funções de conversão.

CONTEÚDO

- ✓ Tipos de variáveis
- ✓ Escopo de variáveis
- ✓ Conversão de dados



Variáveis na linguagem C#

Durante a execução de um programa, dados são armazenados temporariamente na memória. Uma variável é um nome que se dá a um local na memória que armazena certo tipo de dado, assim cada variável está associada com um tipo de dado e um valor.

A linguagem C#, assim como toda linguagem de programação faz o uso de variáveis e exige que todas as variáveis sejam declaradas.

Toda variável deve ter um tipo que define qual o tipo de dado que deverá ser armazenada, dessa forma toda variável possui regras que determinam seu uso. Podemos dizer que existem cinco categorias básicas de variáveis: inteiros, números de ponto flutuante, booleanas, datas e strings (variáveis do tipo texto).

Nas linguagens de programação orientada a objetos existem dois tipos diferentes de variáveis: os tipos intrínsecos à linguagem, também conhecido como tipos primitivos, e aqueles que são criados pelos programadores, conhecidos como classes.

No C# a declaração de variáveis é obrigatória, assim como a declaração de seus tipos.

Nomeando e declarando uma variável

A documentação do Microsoft .Net Framework dá as seguintes recomendações para a nomeação das variáveis:

- Evite usar underline;
- Não crie variáveis que apenas se diferenciem apenas pela sua forma. Exemplo: *minhaVariavel* e outra chamada *MinhaVariavel*;
- Procure iniciar o nome com uma letra minúscula;
- Evite usar todas as letras maiúsculas;
- Quando o nome tiver mais que uma palavra, a primeira letra de cada palavra após a primeira deve ser maiúscula (por ex: valProdutoVista)

Declaração de variáveis

```
[TIPO DA VARIÁVEL] [NOME DA VARIÁVEL] = [VALOR];  
OU  
[TIPO] [NOME DA VARIÁVEL];
```

Tipos de variáveis

	Tipo	Size(em bits)	Variação
Inteiro	Byte	8	0 to 255
	Short	16	-32,768 a 32,767
	Int	32	-2,147,483,648 a 2,147,483,647
	Long	64	-9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
Flutuante	Decimal	128	$\pm 1.0 \times 10^{-28}$ a $\pm 7.9 \times 10^{28}$
	Double	64	$\pm 5.0 \times 10^{-324}$ a $\pm 1.7 \times 10^{308}$
	Float	32	$\pm 1.5 \times 10^{-45}$ a $\pm 3.4 \times 10^{38}$
string e caracter	Char	16	Um caracter
	String	16 bits por caracter	Seqüência de caracter

Tipos de ponto flutuante são usados quando se necessita para realizar as operações que exijam representações fracionária. No entanto, para cálculos financeiros, o tipo decimal é a melhor opção, porque você pode evitar erros de arredondamentos.

Escopo das variáveis

Dentro de um mesmo form podemos trabalhar com variáveis *locais* e *publicas*.

Variáveis Locais

Qualquer variável criada dentro do corpo do método faz parte do escopo do método. Estas variáveis são chamadas de **variáveis locais** porque são locais ao método onde são declaradas. Elas não podem ser usadas no escopo nenhum outro método, por isso você não pode usar variáveis locais para armazenar informações entre métodos. Quando um método acaba sua execução ele finaliza as variáveis que ele criou.

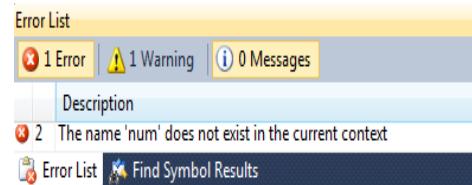
Exemplo 08

```
private void button1_Click(object sender, EventArgs e)
{
    int num = 85;
}
```

No exemplo acima a variável num do tipo int foi declarada dentro do evento click do button1, é local a este método, isto é, não podemos utilizar esta variável em outro método.

Se tentarmos utilizar a variável num no outro evento, conforme abaixo irá apresentar um erro:

```
private void textBox1_TextChanged(object sender,  
EventArgs e)  
{  
    MessageBox.Show("Teste" + num);  
}
```



Variáveis Públicas

Para utilizar variáveis em métodos diferentes dentro do mesmo formulário declaramos na área publica do formulário.

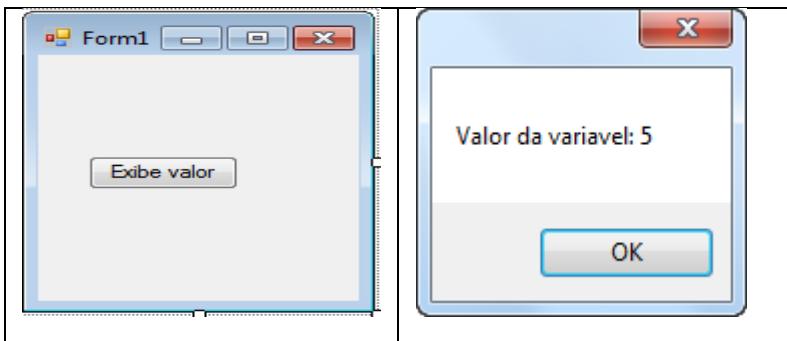
The screenshot shows the 'Form1.cs' code editor. A red box highlights the declaration of a public variable 'num' within the 'Form1' class. The code is as follows:

```
using System;  
using System.Collections.Generic;  
using System.ComponentModel;  
using System.Data;  
using System.Drawing;  
using System.Linq;  
using System.Text;  
using System.Windows.Forms;  
  
namespace exemplo_variaveis  
{  
    public partial class Form1 : Form  
    {  
        int num = 5;  
        public Form1()  
        {  
            InitializeComponent();  
        }  
        private void button1_Click(object sender, EventArgs e)  
        {  
            MessageBox.Show("Teste" + num);  
        }  
    }  
}
```

A variável num criada dentro do método público do form pode ser utilizada em outros métodos do formulário.

Exemplo 09

Declare uma variável pública do tipo int e exiba seu valor usando dentro do método click do botão.




```

using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace exemplo_variaveis
{
    public partial class Form1 : Form
    {
        int num = 5;
        public Form1()
        {
            InitializeComponent();
        }

        private void btnexibe_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Valor da variavel: " + num);
        }
    }
}

```

- Declaramos a variavel **num** dentro da area publica do form:

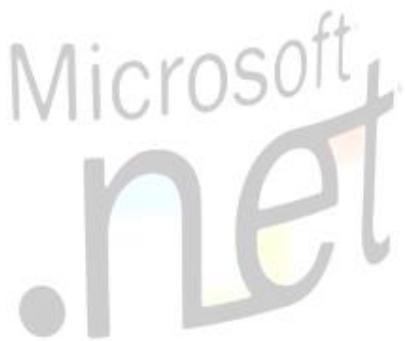
```

public partial class Form1 : Form
{
    int num = 5;
}

```

- E utilizamos no evento click do Button

```
private void btnexibe_Click(object sender, EventArgs e)
{
    MessageBox.Show("Valor da variavel: " + num);
}
```



OBJETIVO:	Identificar variáveis e tipos
METODOLOGIA:	
DATA ENTREGA:	DATA: ___ / ___ / ___
LISTA_ATV_PP.3	

ATIVIDADE 07

Considere as seguintes declarações em C#, coloque V(verdadeiro) ou F(falso) :

- | | |
|------------------------------------|--------------------------------|
| a) () int idade=12.4; | f) () float total; |
| b) () decimal VALPRODUTO=12.4; | g) () double valProduto=12.4; |
| c) () decimal valProduto=12.4; | h) () byte quant=500; |
| d) () string nomeCliente="Maria"; | |
| e) () char sexo="F"; | |

ATIVIDADE 08

Considere a seguinte trecho de código em C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace exemplo_variaveis
{
    public partial class Form1 : Form
    {
        int a, num, idade;
        double valVenda = 120.40;

        public Form1()
        {
            InitializeComponent();
        }
        private void btnbotaoa_Click(object sender, EventArgs e)
        {
            double x = 123.2345;
            double valCompra = 90.12;
        }
        private void btnbotaoab_Click(object sender, EventArgs e)
        {
            int y;
            double valor = 12.455;
        }
    }
}
```

Quais variáveis são locais e quais são públicas ?

Variáveis locais: _____

Variáveis públicas: _____

ATIVIDADE 09

Considere o seguinte trecho de código em C#:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace exemplo_variaveis
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int idade = 24;

        }

        private void button2_Click(object sender, EventArgs e)
        {
            MessageBox.Show("A idade é = " + idade);
        }
    }
}
```

O programa ao ser executado apresentou o seguinte erro:

Error List		
	Description	
✖ 2	The name 'idade' does not exist in the current context	

Explique o porque do erro e qual seria a solução para este problema.

Conversão de tipos

Converter um tipo de dado em um número ou literal é comum em situações de programação.

Os casos mais típicos são as conversões de valores contidos em TextBoxes, a propriedade text só permite strings.

Possui conversões implícitas e explícitas.

Exemplo 10

```
double x = 10;
```

```
int y = 10;
```

```
x = y;
```

```
y = x;
```

- X = Y é uma conversão implícita. O C# cuida da conversão sem necessidade de intervenção do programador.
- Y = X não pode ser feita implicitamente. O espaço reservado em memória para um double é maior do que para um inteiro.

Conversões implícitas permitem comparações e operações diretas.

```
if (x == y)
```

```
MessageBox.Show("Iguais");
```

Funções de conversão

São métodos estáticos e estão presentes na classe System.Convert.

- ToBoolean – converte um valor para boolean.
- ToByte – converte um valor para um inteiro sem sinal de 8 bits
- ToChar – converte um valor para um caracter Unicode.
- ToDateTime – converte um valor para um DateTime.
- ToDecimal – converte um valor para Decimal
- ToDouble – converte um valor para ponto flutuante com precisão Double.
- ToInt16 – converte um valor para inteiro com sinal de 16 bits (compatível com short).
- ToInt32 – converte um valor para um inteiro com sinal de 32 bits (compatível com int).
- ToInt64 – converte um valor para um inteiro com sinal de 64 bits (compatível com long).
- ToSingle – converte um valor para um ponto flutuante com precisão single (compatível com float).

- ToString – converte um valor para uma string.
- ToUInt16 – converte um valor para um inteiro sem sinal de 16 bits (compatível com ushort).
- ToUInt32 – converte um valor para um inteiro sem sinal de 32 bits (compatível com uint).
- ToUInt64 – converte um valor para um inteiro sem sinal de 64 bits (compatível com ulong).

Operadores Aritméticos

Em C# temos os seguintes operadores aritméticos:

Operador	Descrição
+	Adicao
-	Subtracao
*	Multiplicacao
/	Divisao
%	Resto/Modulo

Operadores de atribuição

Em C# temos os seguintes operadores de atribuição:

Operador	Descrição
=	Atribuição simples
+=	Atribuição aditiva
-=	Atribuição Subtrativa
*=	Atribuição Multiplicativa
/=	Atribuição de divisão
%=	Atribuição de módulo

Operadores relacionais

Em C# temos os seguintes operadores relacionais:

Operador	Descrição
==	Igualdade
>	Maior
<	Menor
<=	Menor igual
>=	Maior igual
!=	Diferente

Operadores lógicos

Em C# temos os seguintes operadores lógicos:

Operador	Descrição
&&	E
	OU

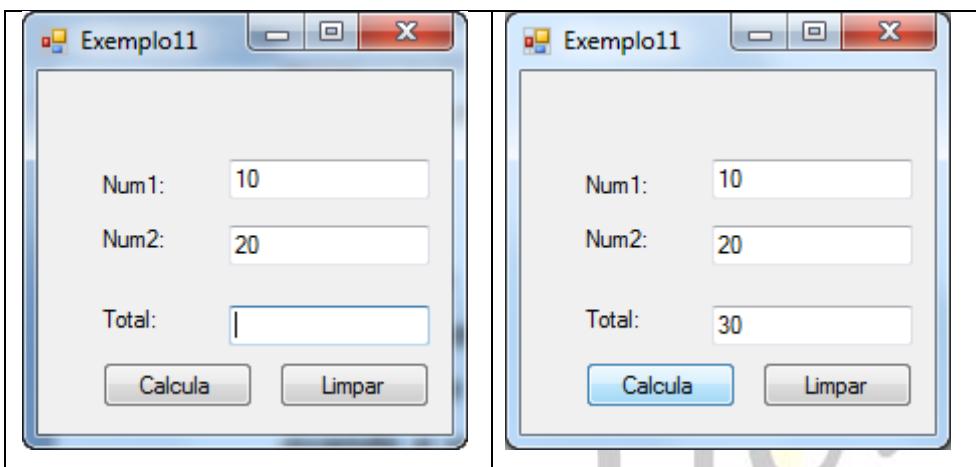
Exemplo 11

Faça uma solução em C#, utilizando a IDE Windows Forms, que tenha um form (frmexemplo11), três label (lblnum1, lblnum2, lbltotal), três textbox (txtnum1, txtnum2, txttotal), dois Button (btncalcula, btnlimpa).

Eventos

O usuário irá entrar com dois números inteiros nas caixas de texto (txtnum1, txtnum2), quando o usuário clicar no button (btncalcula) será feita a soma entre estes números e exibir na caixa de texto (txttotal). Para finalizar quando o usuário clicar no button (btnlimpa) o conteúdo das caixas de textos ficará sem valor algum.

As labels servirão para indicar qual dado será digitado em cada textbox.



Neste exemplo precisamos declarar três variáveis (num1, num2 e total). As variáveis serão do tipo int, pois o usuário vai entrar com valores inteiros.

Quando o usuário entra com dados em objetos, a linguagem C# interpreta estes dados como string, por isso precisamos efetuar a conversão de dados.

- Para converter dados de uma caixa de texto para o tipo inteiro utilizamos a função Convert.ToInt32.

Ex: num2=Convert.ToInt32(txtnum2.Text);

- Depois de efetuado os cálculos precisamos exibir os dados em uma caixa de texto, por isso precisamos converter novamente para string, utilizando a função Convert.ToString.

Ex: txttotal.Text = Convert.ToString(valTotal);



```
private void btncalcul_Click(object sender, EventArgs e)
{
    int num1,num2,total;
    num1=Convert.ToInt32(txtnum1.Text);
    num2=Convert.ToInt32(txtnum2.Text);
    total=(num1+num2);
    txttotal.Text=Convert.ToString(total);
}
private void btnlimpa_Click(object sender, EventArgs e)
{
    txtnum1.Text = "";
    txtnum2.Text = "";
    txttotal.Text = "";
}
```

Exemplo 12

Faça uma solução em C#, utilizando a IDE Windows Forms, que tenha um form (frmexemplo11), três label (lblvalunit, lblquant, lbltotal), três textbox(txtvalunit, txtquant, txttotal), dois Button(btncalcula, btnnovo_calc).

Eventos

O usuário irá entrar com a quantidade na caixa de texto (txtquant) e o valor unitário na caixa de texto (txtvalunit) quando o usuário clicar no button (btncalcula) será feita a multiplicação entre este valores e exibir o total na caixa de texto(txttotal). Para finalizar quando o usuário clicar no button(btnnovo_calc) o conteúdo das caixas de textos ficará sem valor algum.

As labels servirão para indicar qual dado será digitado em cada textbox.

Estado	Valor Unitário	Quantidade	Total
Antes de Calcular	12,30	2	
Depois de Calcular	12,30	2	24,60

Neste exemplo precisamos declarar três variáveis (quant, valUnit, valTotal).

A variável **quant** será do tipo int, pois o usuário vai entrar com valor inteiro, as variáveis **valUnit** e **valTotal** são do tipo decimal pois estamos trabalhando com valor monetário.

- Para converter dados de uma caixa de texto para o tipo inteiro utilizamos a função `Convert.ToInt32`.

Ex: quant = Convert.ToInt32(txtquant.Text);

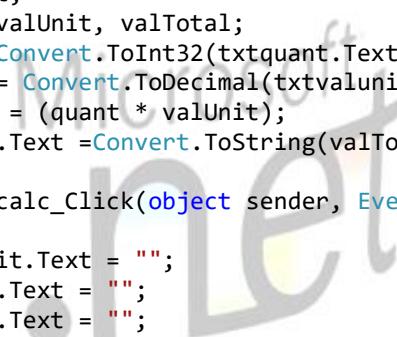
- Para converter dados de uma caixa de texto para o tipo decimal utilizamos a função `Convert.ToDecimal`.

Ex: valUnit = Convert.ToDecimal(txtvalunit.Text);

- Depois de efetuado os cálculos precisamos exibir os dados em uma caixa de texto, por isso precisamos converter novamente para string, utilizando a função `Convert.ToString`.

Ex: txttotal.Text = Convert.ToString(valTotal);



<pre>private void btncalcula_Click(object sender, EventArgs e) { int quant; decimal valUnit, valTotal; quant = Convert.ToInt32(txtquant.Text); valUnit = Convert.ToDecimal(txtvalunit.Text); valTotal = (quant * valUnit); txttotal.Text = Convert.ToString(valTotal); } private void btnnovocalc_Click(object sender, EventArgs e) { txtvalunit.Text = ""; txtquant.Text = ""; txttotal.Text = ""; }</pre>	
---	---

Exemplo 13

Faça uma solução em C#, utilizando a IDE Windows Forms, que tenha um form (frmexemplo13), três label (lblnome, lblidade, lbdados), dois textbox (txtnome, txtidade), dois Button (btnexibe, btnlimpa).

Eventos

O usuário irá entrar com o seu nome na caixa de texto (txtnome) e a sua idade na caixa de texto (txtidade) quando o usuário clicar no button (btnexibe) deverá ser calculado o numero de dias vividos e exibir na label (lbdados) o nome do usuário e o numero de dias vividos. Para finalizar quando o usuário clicar no button (btnlimpa) o conteúdo das caixas de textos ficará sem valor algum.

A label (lbdados) inicialmente deverá ficar sem valor (propriedade text).

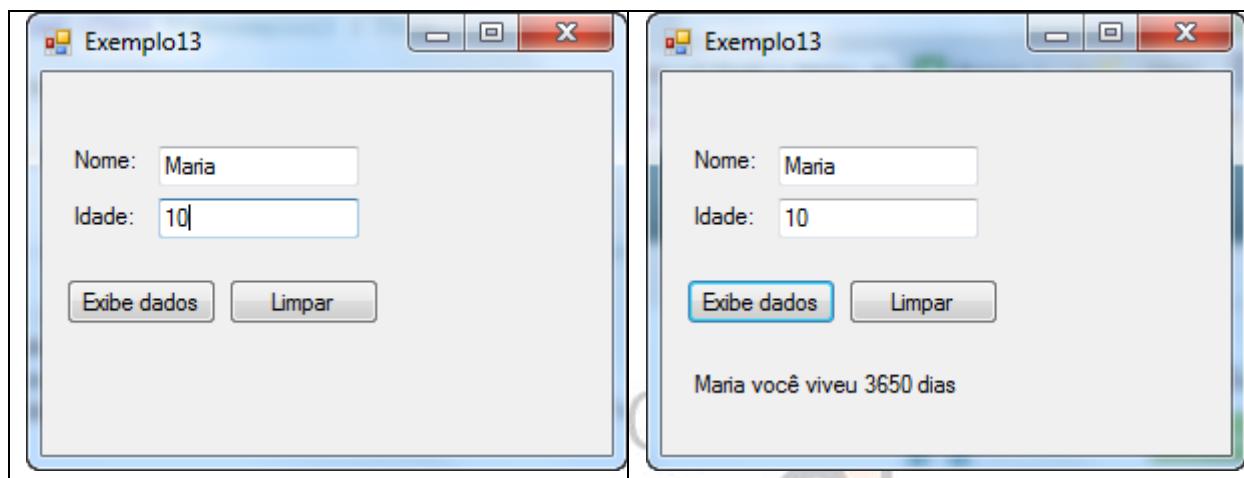
Neste exemplo, precisamos utilizar duas variáveis(idade,nDias)do tipo int.

A variável idade irá receber a idade e nDias irá receber o calculo:

```
nidas= (idade*365);
```

Em seguida precisamos exibir o nome do usuário e a idade, para isso é necessário concatenar dados utilizando o operador “+”.

```
lbldados.Text = txtnome.Text + " você viveu " + Convert.ToString(nDias) + " dias";
```



```
private void btnexibe_Click(object sender, EventArgs e)
{
    int idade, nDias;
    idade = Convert.ToInt32(txtidade.Text);
    nDias = (idade * 365);
    lbldados.Text = txtnome.Text + " você viveu " +
Convert.ToString(nDias) + " dias";

}

private void btnlimpa_Click(object sender, EventArgs e)
{
    txtidade.Text = "";
    txtnome.Text = "";
    lbldados.Text = "";
}
```

CAPÍTULO

3

ATIVIDADES PROPOSTA

OBJETIVO:	Conversão de tipos
METODOLOGIA:	Desenvolvimento de projetos
DATA ENTREGA:	DATA: ___ / ___ / ___
LISTA ATP_PP.3.1	

ATIVIDADE 10

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 02 textbox
- 02 label
- 02 button

Eventos

O usuário irá entrar com dois números inteiros nas caixas de texto (txtnum1,txtnum2), quando o usuário clicar no button (btncalcula) será feita a multiplicação entre os números e exibir na caixa de texto(txtotal). Para finalizar quando o usuário clicar no button(btnlimpa)o conteúdo das caixas de textos ficará sem valor algum.

As labels servirão para indicar qual dado será digitado em cada textbox.

ATIVIDADE 11

Crie um programa em C#, usando recursos de interface gráfica, para conversão de temperatura. O programa converte Fahrenheit para Celsius

- A temperatura Fahrenheit deve ser inserida via teclado usando um *textbox*

Eventos

- Um botão deverá ser usado para efetuar o calculo, utilizando a fórmula:

$$\text{Celsius} = \frac{5}{9} * (\text{Fahrenheit} - 32)$$

- Um *label* deve ser usado para exibir a temperatura convertida.

ATIVIDADE 12

Criar uma interface gráfica em C# com os seguintes objetos:

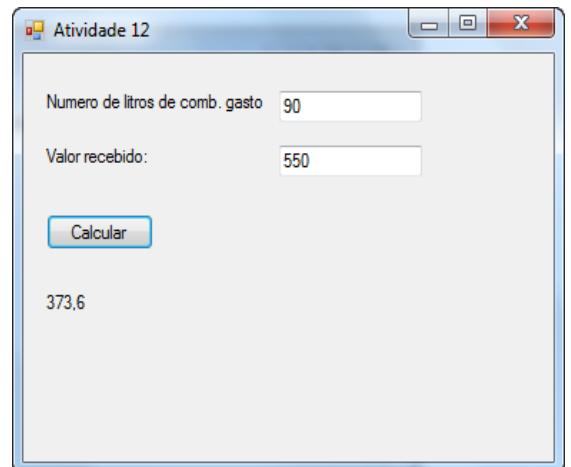
- Um *label* e um campo *textbox* para receber Número de litros de combustível gasto
- Um *label* e um campo *textbox* para receber Valor Recebido dos passageiros

- Um botão calcular
- Um *label* para exibir Lucro Líquido
- Ao clicar no botão calcular deverá ser exibido o lucro líquido de um motorista de táxi, sabendo-se que o preço do combustível é R\$1,96.

Observ.: Sabe- se que o lucro líquido = total recebido – total gasto.

As variáveis para receber o num de litros de combustível precisa ser float e as variáveis para receber o valor recebido é decimal e a variável para calcular o lucro precisa ser também decimal.

A função para converter para float é:
Convert.ToDouble



ATIVIDADE 13

Criar uma interface gráfica em C# com os seguintes objetos:

- Um *label* e um campo *textbox* para receber a nota 1.
- Um *label* e um campo *textbox* para receber a nota 2.
- Um botão calcular
- Um *label* e um campo *textbox* para exibir a média .–

Ao clicar no botão calcular deverá ser exibido a média do aluno.

Sabe-se que a media=(nota1+nota2)/2.

Observ.: as variáveis deverão ser do tipo float.

ATIVIDADE 14

Criar uma interface gráfica em C# com os seguintes objetos:

- Um *label* e um campo *textbox* para receber um numero.
- Um botão calcular
- Um botão limpar
- Um *label* para exibir o resto.
- Ao clicar no botão calcular deverá ser exibido o resto da divisão do numero digitado por 2.

- Ao clicar no botão limpar o conteúdo das caixas de texto e da label do resto deverão estar sem valor.

Observ.: declare a variável para receber o numero como inteiro e utilize o operador % para calcular o resto da divisão.

ATIVIDADE 15

Criar uma interface gráfica em C# com os seguintes objetos:

- Um *label* e um campo *textbox* para receber o valor de um produto.
- Um botão calcular
- Um botão limpar
- Um *label* e uma caixa de texto para exibir o valor à vista.
- Um *label* e uma caixa de texto para exibir o valor à prazo.
- Ao clicar no botão calcular deverá ser calculado o valo do produto à vista e à prazo e exibido nas textbox correspondente.

Sabe-se que para calcular o valor à vista existe um desconto de 10% e à prazo um acréscimo de 12%.

- Ao clicar no botão limpar o conteúdo das caixas de texto deverão estar sem valor.

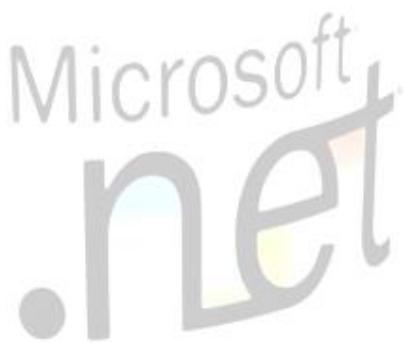
Observ. Declare as variáveis como decimal.

OBJETIVOS

Conhecer os tipos de dados e funções de conversão.

CONTEÚDO

- ✓ Tipos de variáveis
- ✓ Escopo de variáveis
- ✓ Conversão de dados



ESTRUTURAS DE DECISÃO EM C#

O C# assim como outras linguagens apresentam diversas estruturas que controla a execução (fluxo e comportamento) de um programa. Dentre elas podemos citar as estruturas de decisão permitem que escolhamos qual parte do código será executado dado uma expressão booleana (que pode ter como resultado, apenas dois valores: verdadeiro ou falso).

Um exemplo de estrutura de decisão é a estrutura IF (SE), que serve para mudar o fluxo da execução de um programa dado uma expressão booleana. A maioria dos programas tomam decisões que afetam seu fluxo. As declarações que tomam essas decisões são chamadas de declarações de controle.

if sem chaves	if com chaves
if (expressao booleana) comando;	if (expressao booleana) { comando 1; comando 2; comando n; } Obs: Usando quando temos apenas um comando fazendo parte do if

if - else sem chaves	if com chaves
if (expressao booleana) comando1; else comando2; Obs: onde comando1 será somente executado se a expressão booleana for verdadeiro, e caso for falso será executado o comando2	if (expressao booleana) { comandos bloco 1; } else { comandos bloco 2; } Obs: onde os comandos do bloco 1 serão somente executados se a expressão booleana for verdadeiro, e caso for falso será executado os comandos do bloco 2

if – else if – else	Onde:
if (expressao booleana 1) { comandos bloco 1; } else if (expressao booleana 2) { comandos bloco 2; } else if (expressao booleana n) { comandos bloco n; } else { comandos bloco else; }	- Os comandos do bloco 1 serão somente executados se a expressão booleana 1 for verdadeira; - Os comandos do bloco 2 serão somente executados se a expressão booleana 2 for verdadeira; - Os comandos do bloco n serão somente executados se a expressão booleana n for verdadeira; - Os comandos do bloco else serão somente executados se nenhuma expressão booleana anterior for verdadeira.

O uso de chaves é obrigatório quando se quer que mais de um comando seja executado quando uma expressão é verdadeira ou não. Caso o fluxo seja direcionado para apenas um comando, a chaves não se faz necessária.

Switch

O comando switch existe nas principais linguagens de programação (C, C++, Java, C#). Seu propósito é permitir que o valor de uma variável ou expressão controle o fluxo de execução do programa.

Sintaxe:

O comando deve começar com a palavra reservada **switch**, e cada bloco de execução deve conter a palavra reservada **case** acompanhada do valor que a variável de controle deve ter. Para finalizar o bloco, devemos usar o comando **break**.

Switch	
switch (expressão) { case valor1: comandos; break; case valor2: comandos; break; case valorN: comandos; break; default: comandos; break; }	Onde: - Os comandos relacionados ao case valor1 serão somente executados se o valor da expressão for igual ao valor1; - Os comandos relacionados ao case valor2 serão somente executados se o valor da expressão for igual ao valor2; - Os comandos relacionados ao case valorN serão somente executados se o valor da expressão for igual ao valorN; - Os comandos do bloco default serão somente executados se nenhuma expressão tiver um valor definido na estrutura case.

Estrutura Switch com intervalo de valores

<pre>switch (expressao) { case 1: case 2: comando1; break; case 3: case 4: comando2; break; case 5: case 6: comando3; break; }</pre>	
--	--

Caixas de mensagem em C#

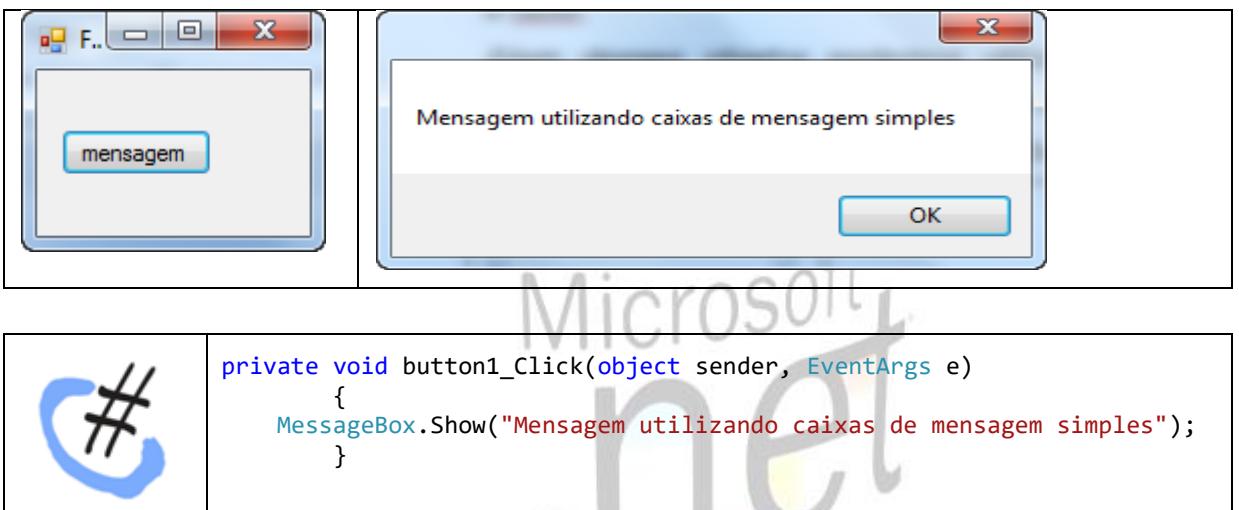
Durante a execução de nossos programas precisamos exibir determinados valores para o usuário normalmente utilizamos objetos para isto, como por exemplo, textbox e label.

Alem desses objetos podemos utilizar caixas de mensagens, o Visual Studio disponibiliza esse recurso de caixa de mensagem.

Para exibir uma caixa de mensagem, basta utilizar a classe `MessageBox.Show`

Para exibir caixas de mensagem simples, basta colocar:

`MessageBox.Show("mensagem");`



A caixa de mensagem simples aparece com uma mensagem e um botão apenas.

Caso você queira personalizar sua caixa de mensagem, podemos utilizar outros parâmetros, como por exemplo colocar um titulo na caixa de mensagem inserir mais botões para o usuário escolher e um ícone na caixa.

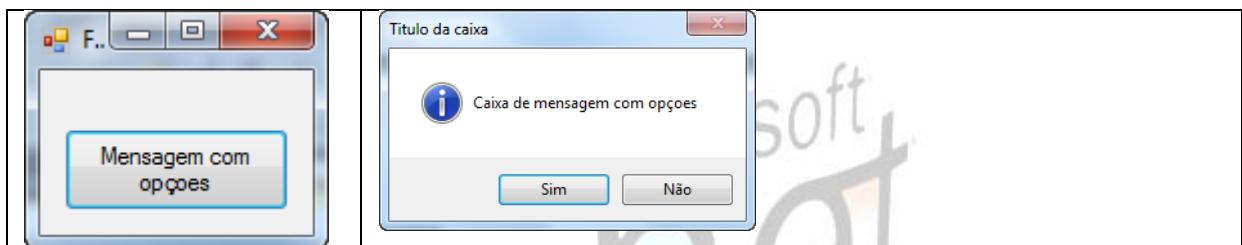
Tipos de botões	
Valor	Botões
<code>AbortRetryIgnore</code>	Abort, Retry, Ignore
<code>OK</code>	<code>OK</code>
<code>OKCancel</code>	<code>OK, Cancel</code>
<code>RetryCancel</code>	<code>Retry, Cancel</code>
<code>YesNo</code>	<code>Yes, No</code>
<code>YesNoCancel</code>	<code>Yes, No, Cancel</code>

Tipos de Ícones	
Valor	Descrição

Asterisk	Mostra um círculo contendo um i
Error	Mostra um círculo vermelho contendo um X
Exclamation	Mostra um triângulo amarelo com um ponto de exclamação.
Hand	Mostra um círculo vermelho contendo um X branco
Information	Mostra um círculo contendo um i
Question	Mostra um círculo contendo um ponto de interrogação
Stop	Mostra um círculo vermelho contendo um X branco
Warning	Mostra um triângulo amarelo contendo um ponto de exclamação

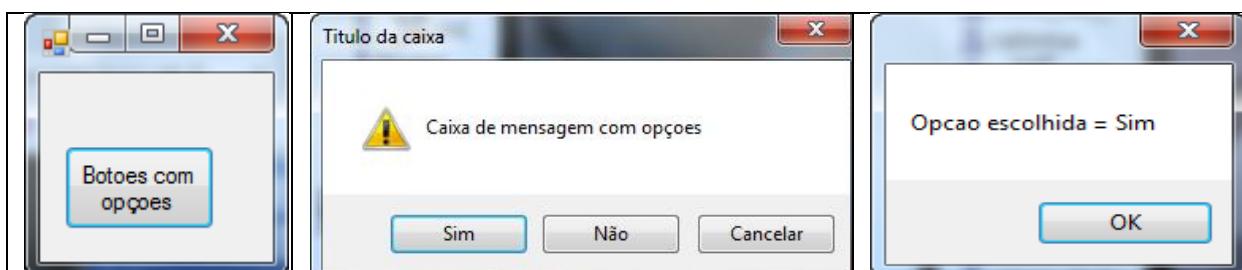
Sintaxe da caixa de mensagem:

```
MessageBox.Show("Mensagem", "Titulo da caixa", MessageBoxButtons.nome_botoes,
MessageBoxIcon.nome_icones);
```



```
private void button1_Click(object sender, EventArgs e)
{
    MessageBox.Show("Caixa de mensagem com opções", "Titulo da caixa",
    MessageBoxButtons.YesNo, MessageBoxIcon.Information);
}
```

Caso precise codificar o botão selecionado pelo usuário utilize um IF e parâmetro DialogResult.nomebotao.



	<pre>private void button1_Click(object sender, EventArgs e) { if (MessageBox.Show("Caixa de mensagem com opções", "Titulo da caixa", MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning)== DialogResult.Yes) MessageBox.Show("Opcão escolhida = Sim"); }</pre>
--	---

Exemplo 14

Criar uma interface gráfica em C# com os seguintes objetos:

- Três *label* e três *textbox* para receber três notas .
- Um botão calcular
- Um botão limpar
- Ao clicar no botão calcular deverá ser exibido a média do aluno e a situação do mesmo em uma caixa de mensagem.
- Ao clicar no botão limpar o conteúdo das caixas de texto deverão estar sem valor.

Observ.:

Para calcular a média considere a seguinte formula

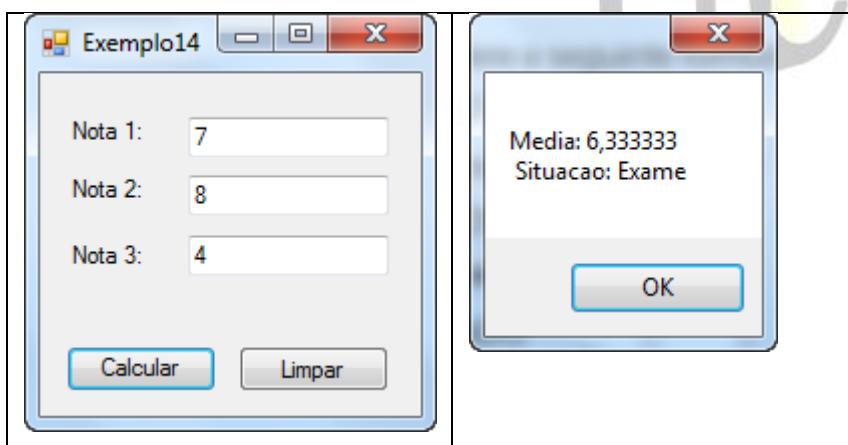
$$\text{Media} = (\text{nota1} + \text{nota2} + \text{nota3}) / 3$$

Para a situação do aluno considere que:

Media $\leq 5 \rightarrow$ “REPROVADO”

Media > 5 e Media $\leq 7 \rightarrow$ “EXAME”

Media $> 7 \rightarrow$ “APROVADO”



- Depois de efetuado o calculo da média para exibir a situação precisamos utilizar uma estrutura de decisão, no caso utilizamos a estrutura IF..., ELSE IF..., ELSE.
- Para exibir o valor da média em uma caixa de mensagem, precisamos também converter o valor da variável media para string, pois as caixas de mensagem foram feitas para receber string.
- Quando queremos exibir uma mensagem em varias linhas em uma caixa de mensagem utilizamos “\n” para efetuar a quebra de linha.



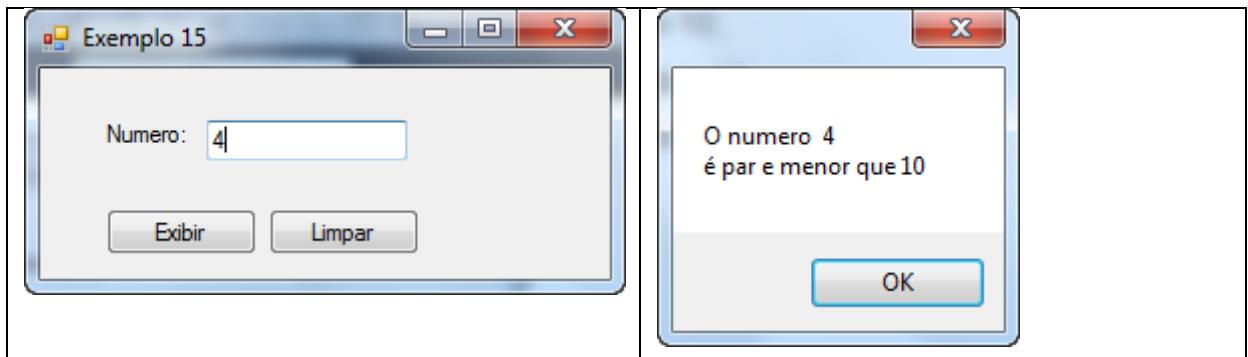
```
private void btncalcular_Click(object sender, EventArgs e)
{
    float nota1,nota2,nota3,media;
    nota1= Convert.ToSingle(txtnota1.Text);
    nota2= Convert.ToSingle(txtnota2.Text);
    nota3= Convert.ToSingle(txtnota3.Text);
    media= (nota1+nota2+nota3)/3;
    if (media <=5)
        MessageBox.Show("Media: " + Convert.ToString(media) + "\n
Situacao: Reprovado");
    else if (media <=7)
        MessageBox.Show("Media: " + Convert.ToString(media) +
"\n Situacao: Exame");
    else
        MessageBox.Show("Media: " + Convert.ToString(media) +
"\n Situacao: Aprovado");
}

private void btnlimpar_Click(object sender, EventArgs e)
{
    txtnota1.Text = "";
    txtnota2.Text = "";
    txtnota3.Text = "";
}
```

Exemplo 15

Criar uma interface gráfica em C# com os seguintes objetos:

- Duas *label* e duas *textbox* para receber dois numeros .
- Um botão exibir
- Um botão limpar
- Ao clicar no botão calcular deverá ser exibido uma mensagem para verificar se o numero
- Par e maior que 10;
- Par e menor que 10;
- Impar e menor que 10;
- Impar e menor que 10;



```

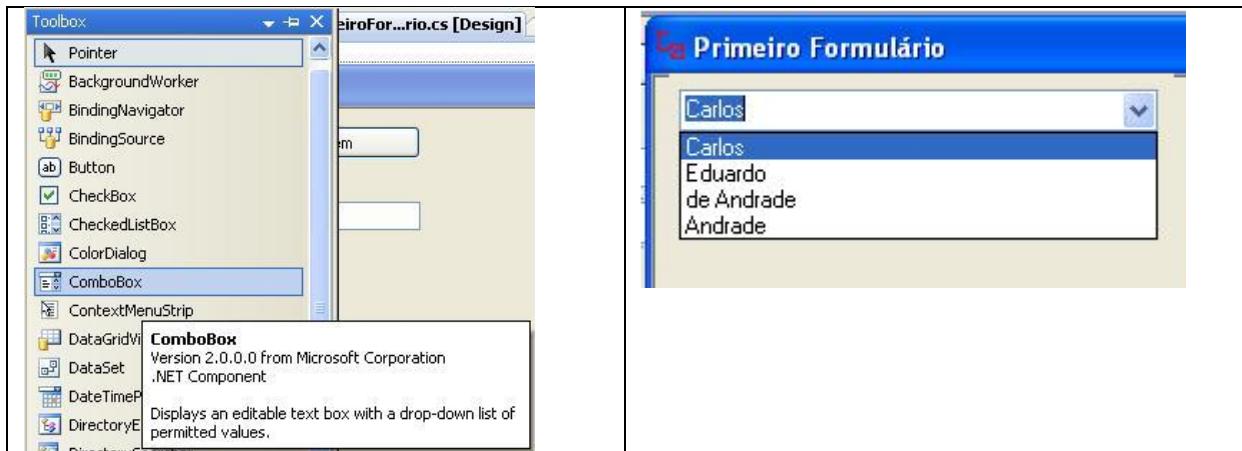
C# icon
private void txtexibir_Click(object sender, EventArgs e)
{
    int numero ;
    numero = Convert.ToInt32(txtnum.Text);
    if (((numero%2) ==0 ) && (numero>= 10))
        MessageBox.Show(" O numero " + Convert.ToString(numero) + " \n é par
e maior ou igual a 10");
    else if (((numero % 2) == 0) && (numero < 10))
        MessageBox.Show(" O numero " + Convert.ToString(numero) + " \n é par
e menor que 10");
    else if (((numero % 2) != 0) && (numero > 10))
        MessageBox.Show(" O numero " + Convert.ToString(numero) + " \n é
ímpar e maior que 10");
    else if (((numero % 2) != 0) && (numero < 10))
        MessageBox.Show(" O numero " + Convert.ToString(numero) + " \n é
ímpar e menor que 10");
}

private void btnlimpar_Click(object sender, EventArgs e)
{
    txtnum.Text = "";
}

```

Objeto: COMBOBOX (caixa de seleção)

Permite a escolha de itens pré definidos.



Principais eventos do COMBOBOX

- ❖ **AutoCompleteMode:** define a forma como o Text do ComboBox sugere um item a ser selecionado.
- ❖ **AutoCompleteSource:** é a lista onde temos os valores que serão usados para a sugestão de auto preenchimento.
- ❖ **BackColor:** é a cor de fundo do formulário, ou a cor da parede, e **ForeColor** a cor das fontes ou das palavras que aqui escreveremos.
- ❖ **DataSource:** se temos uma fonte de dados como uma DataTable, podemos usar aqui.
- ❖ **DisplayMember:** se usarmos a propriedade DataSource, como sugerido acima com um DataTable, teremos várias colunas na tabela, então definimos qual das colunas será a informação que o usuário irá enxergar.
- ❖ **Dock:** define se queremos o componente literalmente grudado em um ou mais cantos.
- ❖ **DropDownStyle:** modo como o combo funciona, sendo uma lista somente leitura, lista com TextBox, ou somente um TextBox.
- ❖ **Enabled:** define se este ComboBox está habilitado ou não, ou seja, se o usuário pode usá-lo ou não.
- ❖ **Items:** lista de valores que o ComboBox disponibiliza ao usuário para seleção.
- ❖ **Location:** define a posição Top (distância do controle em relação a margem superior do formulário) e a posição Left (distância do controle em relação a margem esquerda do formulário)

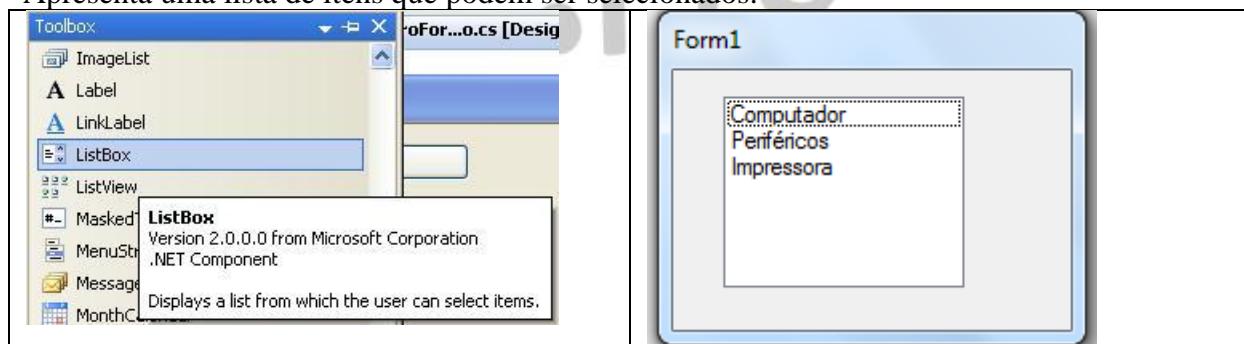
- ❖ **Name:** é o nome do controle, nome que usaremos para referenciá-lo quando estivermos codificando. Aqui chamei de `lblNome`.
- ❖ **Size:** define a largura e a altura do controle.
- ❖ **Text:** é a informação selecionada. Também pode ser usado como se fosse um `TextBox`.
- ❖ **ValueMember:** como a `DisplayMember` define qual informação o usuário ve, aqui definimos qual informação é útil ao sistema, por exemplo o `IdUsuario`. Ou seja o valor que aquela informação representa para o sistema.

Principais eventos do COMBOBOX

Evento	Codificação
Adicionar itens a um combobox:	<code>comboBox1.Items.Add(cboestado.SelectedItem)</code>
Remover item a um combobox	<code>comboBox1.Items.Remove(comboBox1.SelectedItem);</code>
Limpar um combobox (excluir todos os itens)	<code>comboBox1.Items.Clear();</code>
Exibir o índice de um item selecionado	<code>comboBox1.SelectedIndex;</code>
Exibir o valor do índice selecionado	<code>comboBox1.SelectedItem;</code>

Objeto: LISTBOX (lista de seleção)

Apresenta uma lista de itens que podem ser selecionados.



Principais propriedades do LISTBOX

- ❖ **Name:** é o nome do controle, nome que usaremos para referenciá-lo quando estivermos codificando.
- ❖ **BackColor:** é a cor de fundo do formulário, ou a cor da parede, e **ForeColor** a cor das fontes ou das palavras que aqui escreveremos.
- ❖ **DataSource:** se temos uma fonte de dados como uma `DataTable`, podemos usar aqui.

- ❖ **DisplayMember:** se usarmos a propriedade *DataSource*, como sugerido acima com um *DataTable*, teremos várias colunas na tabela, então definimos qual das colunas será a informação que o usuário irá enxergar.
- ❖ **ValueMember:** como a *DisplayMember* define qual informação o usuário ve, aqui definimos qual informação é útil ao sistema, por exemplo o *IdUsuario*. Ou seja o valor que aquela informação representa para o sistema.
- ❖ **Dock:** define se queremos o componente literalmente grudado em um ou mais cantos.
- ❖ **Enabled:** define se este controle está habilitado ou não, ou seja, se o usuário pode usá-lo ou não.
- ❖ **Location:** define a posição *Top* (distância do controle em relação a margem superior do formulário) e a posição *Left* (distância do controle em relação a margem esquerda do formulário)
- ❖ **Items:** lista de valores que o controle disponibiliza ao usuário para seleção.
- ❖ **Size:** define a largura e a altura do controle.
- ❖ **MultiColumn:** se precisarmos separar as informações em mais de uma coluna, bem semelhante a uma grid! (veremos a grid mais a frente)
- ❖ **SelectionMode:** aqui definimos se a lista permitirá escolher um ou mais itens de uma só vez.

Principais eventos do LISTBOX(igual ao do combobox)

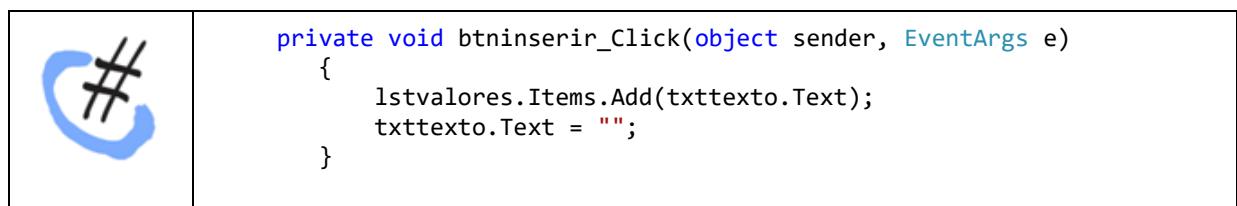
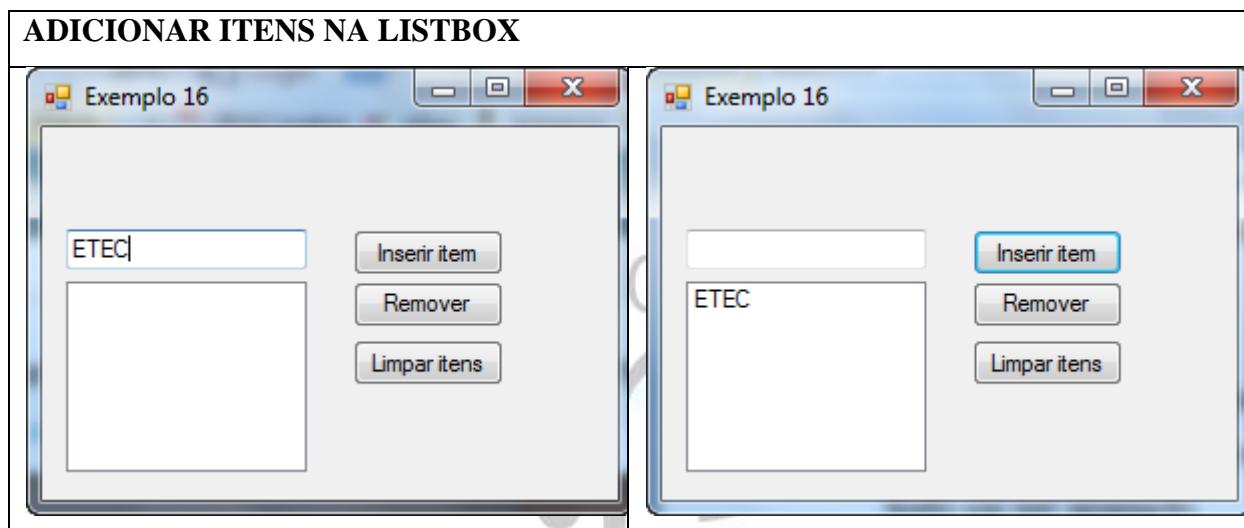
Evento	Codificação
Adicionar itens a um listbox:	<code>listBox1.Items.Add(cboestado.SelectedItem)</code>
Remover item a um listbox	<code>listBox1.Items.Remove(comboBox1.SelectedItem);</code>
Limpar um listbox (excluir todos os itens)	<code>listBox1.Items.Clear();</code>
Exibir o índice de um item selecionado	<code>listBox1.SelectedIndex;</code>
Exibir o valor do índice selecionado	<code>listBox1.SelectedItem;</code>

Exemplo 16

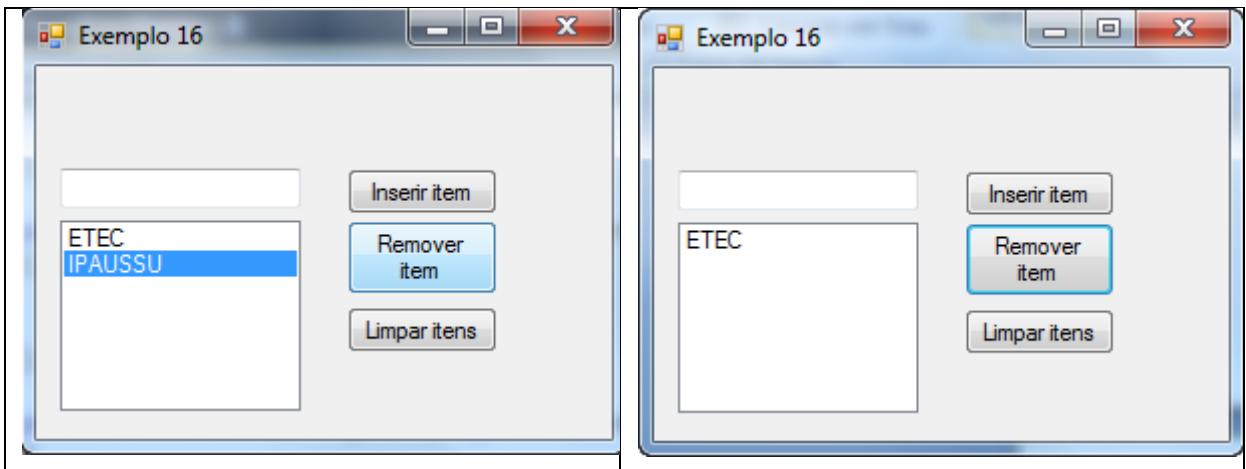
Criar uma interface gráfica em C# com os seguintes objetos:

- Um *textbox* para receber um texto .
- Um *listbox*;
- Um botão Receber texto.
- Um botão Limpar tudo;

- Um botão excluir item;
- Quando o usuário clicar no botão (btninserir) o texto que o usuário digitou na textbox(txttexto) vai ser inserido na listbox(lstvalores) e seguida o texto da caixa de texto vai ser apagado;
- Quando o usuário selecionar um item e clicar no botão (btnremover) o item selecionado será removido;
- Quando o usuário clicar no botão(btnlimpar) todos os itens da listbox(lstvalores) serão excluídos;

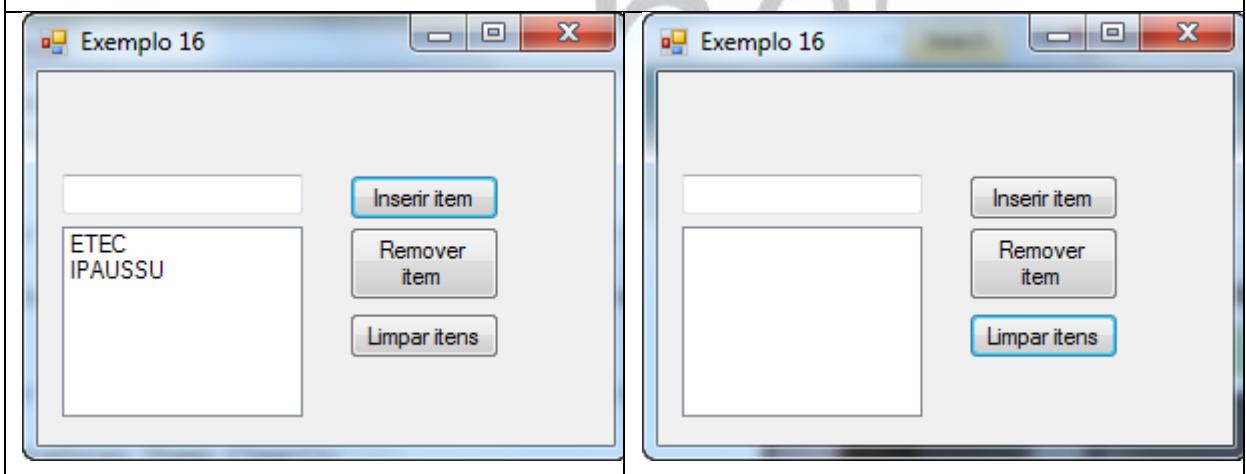


EXCLUIR ITEM DA LISTBOX



```
private void btnremover_Click(object sender, EventArgs e)
{
    lstvalores.Items.Remove(lstvalores.SelectedItem);
}
```

LIMPAR TODOS ITENS DA LISTBOX



```
private void btnlimpar_Click(object sender, EventArgs e)
{
    lstvalores.Items.Clear();
}
```

Observ.:

- Para adicionar itens em uma listbox/comboBox podemos utilizar a propriedade **Items** ou utilizar o evento `.Add("texto");`

- Para excluir um item de uma listbox/combobox, utilizamos o evento **SelectedItem**, pois um item precisa estar selecionado antes de ser removido;
- Para limpar o listbox/combobox não precisamos selecionar nenhum item, pois todos os itens serão excluídos.

Exemplo 17

Criar uma interface gráfica em C# com os seguintes objetos:

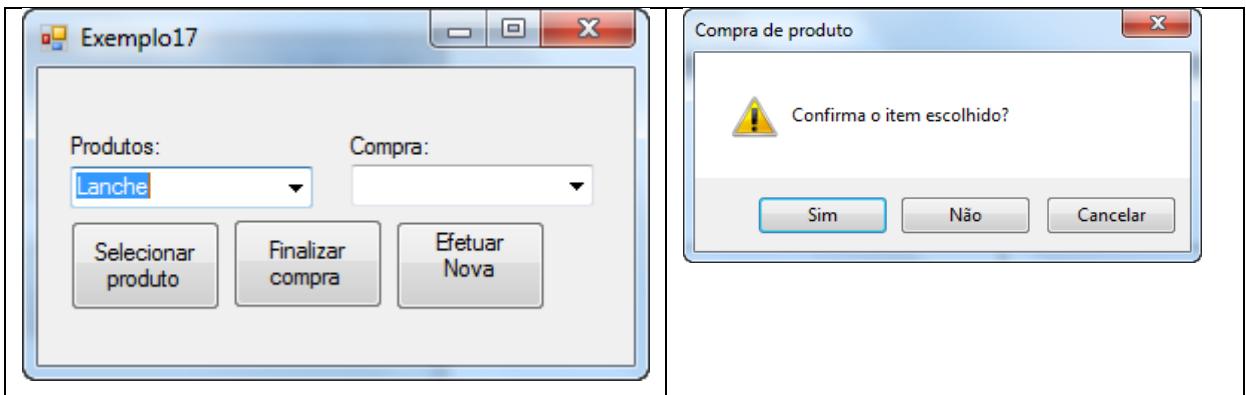
- *Dois combobox;*
- *Um botão Selecionar produto;*
- *Um botão Finalizar compra;*
- *Um botão Efetuar Nova Compra;*
- *Duas labels(para indicar o conteúdo dos combobox)*

- No primeiro combobox(cboprodutos) deverão aparecer os produtos (Lanche, Pizza, Refrigerante);
- Quando o usuário escolher o produto e clicar em no botão Selecionar Produto(btnselecionaprod) vai aparecer uma caixa de mensagem de confirmação do produto, se a resposta for “SIM” o item selecionado automaticamente vai para o combobox(cbocompra);
- Quando o usuário clicar no botão(btnFinalizar) as duas combobox ficarão desabilitadas, isto é, o usuário não vai poder selecionar mais nenhum item;
- Quando o usuário clicar em Efetuar Nova Compra(btnnovacompra), os dois combobox voltarão a ficar habilitados e os itens do combobox da compra (cbocompra) ficará sem item algum.

Codificação:

- Para um combobox(cbocompra) receber o conteúdo de outro combobox(cboprodutos) utilizamos o evento **SelectedItem**;
- Para desabilitar/habilitar um objeto utilizamos a propriedade **enabled** e deixando seu valor false(desabilitar) ou true(habilitar).

ESCOLHENDO UM ITEM



```
private void btnselecionaprod_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Confirma o item escolhido?", "Compra de produto",
        MessageBoxButtons.YesNoCancel, MessageBoxIcon.Warning) == DialogResult.Yes)
        cbocompra.Items.Add(cboprodutos.SelectedItem);

}
```



```
private void btnFinalizar_Click(object sender, EventArgs e)
{
    cbocompra.Enabled = false;
    cboprodutos.Enabled = false;
}

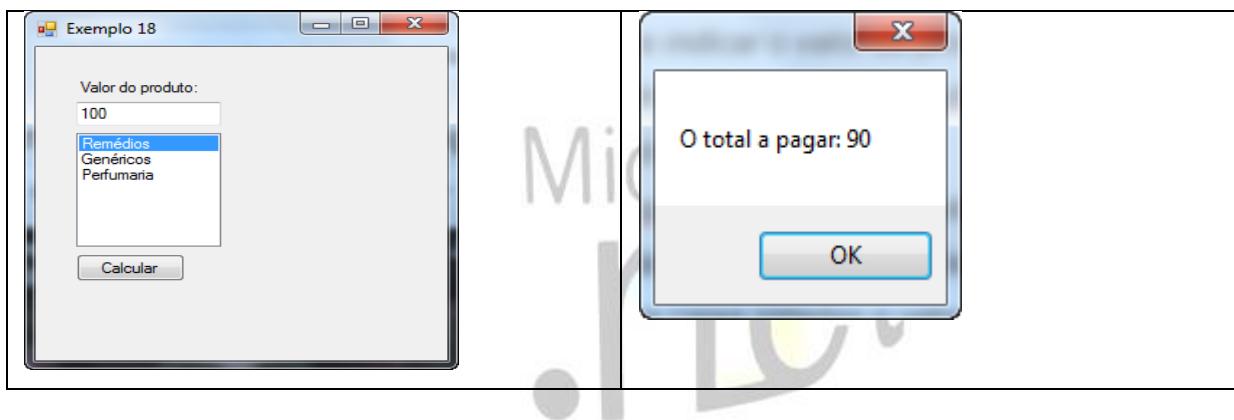
private void btnnovacompra_Click(object sender, EventArgs e)
{
    cbocompra.Enabled = true;
    cboprodutos.Enabled = true;
    cbocompra.Items.Clear();
}
```

Exemplo 18

Criar uma interface gráfica em C# com os seguintes objetos:

- Um textbox
- Um listbox
- Um botão Calcular;
- Uma label (para indicar o valor do produto na textbox)
- No primeiro listbox(lstprodutos) deverão aparecer os produtos (Remédios,Genéricos,Perfumaria);
- No textbox(txtvalorprod) deverá receber o valor do produto digitado pelo usuário;
- Quando o usuário escolher o tipo produto e clicar em no botão Calcular (btncalcula) vai exibir em uma caixa simples o valor a pagar;
- Considere os seguintes descontos para os produtos:

Remédios (10%)/Genéricos(30%)/Perfumaria(5%)



- Para implementar este programa precisamos implementar a propriedade SelectedIndex do listbox; cada item do listbox está associado a um índice(index) e de acordo com este índice, é só colocar nosso código.

	<pre>private void btncalcula_Click(object sender, EventArgs e) { double valprod, valtotal; valprod = Convert.ToDouble(txtvalorprod.Text); if (lstprodutos.SelectedIndex == 0) valtotal = Convert.ToDouble(valprod * 0.9); else if (lstprodutos.SelectedIndex == 1) valtotal = Convert.ToDouble(valprod * 0.7); else valtotal = Convert.ToDouble(valprod * 0.95); MessageBox.Show("O total a pagar: " + Convert.ToString(valtotal));</pre>
--	---

CAPÍTULO

4

ATIVIDADES PROPOSTA

OBJETIVO:	Estruturas de seleção em C#
METODOLOGIA:	Desenvolvimento de projetos
DATA ENTREGA:	DATA: ___ / ___ / ___
LISTA_ATV_P.4	

ATIVIDADE 16

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 01 Textbox
- 01 Button
- 01 Label

Eventos

O usuário irá entrar a idade(textbox) e quando clicar no button exibirá uma mensagem exibindo sua categoria.

A label servirá para indicar que a textbox vai receber a idade.

Considere as seguintes categorias :

- não eleitor (abaixo de 16 anos);
- eleitor obrigatório (entre 18 e 65 anos) e
- eleitor facultativo (entre 16 e 18 anos e acima dos 65 anos).

ATIVIDADE 17

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

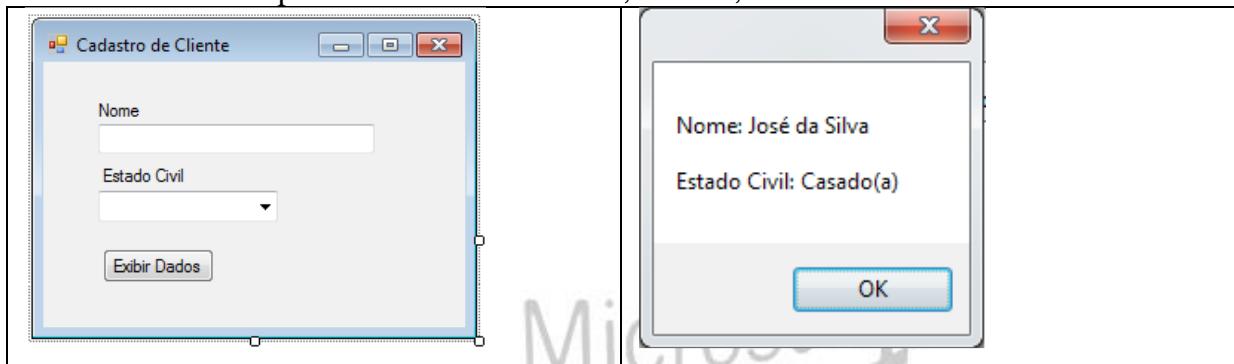
Objetos

- 01 Form
- 01 Textbox
- 01 Button
- 01 combobox
- 02label

Eventos

- O usuário vai entrar com seu nome(textbox) e escolher seu estado civil (combobox)
- Quando o usuário clicar no Button irá aparecer uma caixa de mensagem com seu nome e com a seu estado civil.

Considere os dados para o estado civil: *solteiro, casado, viúvo e divorciado*.



ATIVIDADE 18

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

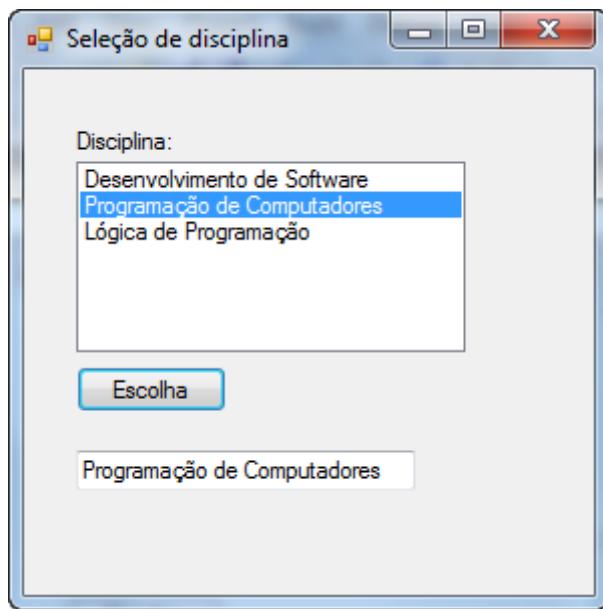
Objetos

- 01 Form
- 01 Textbox
- 01 Button
- 01 listbox
- 01label

Eventos

- Deverão ser inseridos no listbox, as seguintes disciplinas(Desenvolvimento de Software, Programação de computadores e Lógica de Programação) utilizando a propriedade **Items**. Quando o usuário selecionar a disciplina e clicar no botão a disciplina irá aparecer na caixa de texto.

A label servirá para indicar o conteúdo da listbox.



ATIVIDADE 19

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

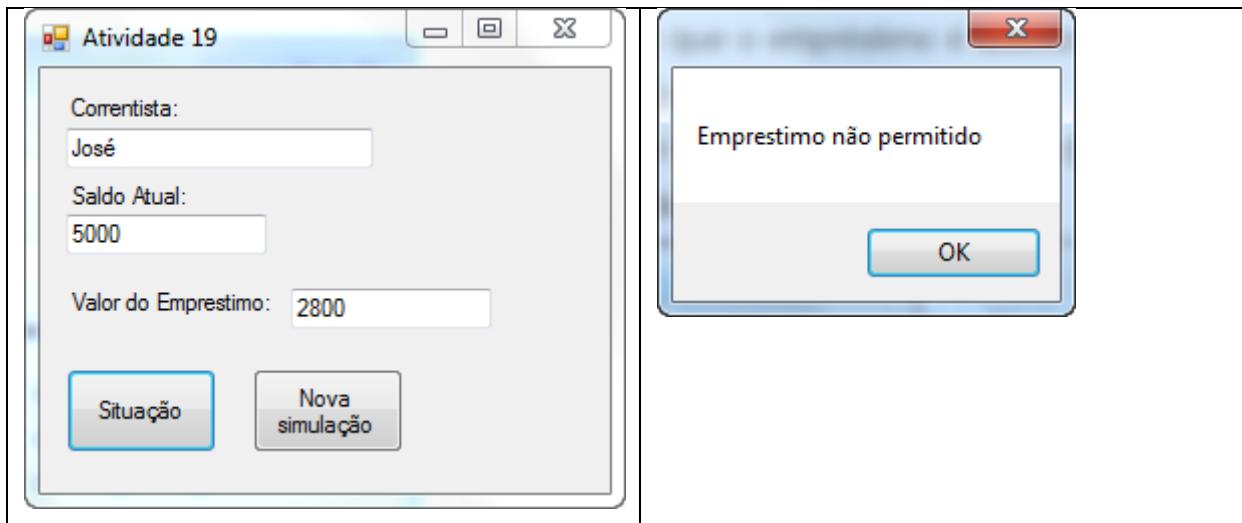
Objetos

- 01 Form
- 03 Textbox
- 02 Button
- 03label

Eventos

- O usuário deverá entrar com o nome do correntista, saldo atual e valor do empréstimo(através das textbox). usuário
- Quando o usuário clicar no botão vai aparecer uma mensagem permitindo ou não o empréstimo, sabendo-se que o empréstimo é autorizado se o valor pretendido for menor ou igual a 30% do saldo atual.
- Quando o usuário clicar nova simulação os valores (textbox) serão apagados.

As labels servirão para indicar o conteúdo da textbox.



ATIVIDADE 20

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 02 Textbox
- 01 Button
- 04label
- 01 combobox

Eventos

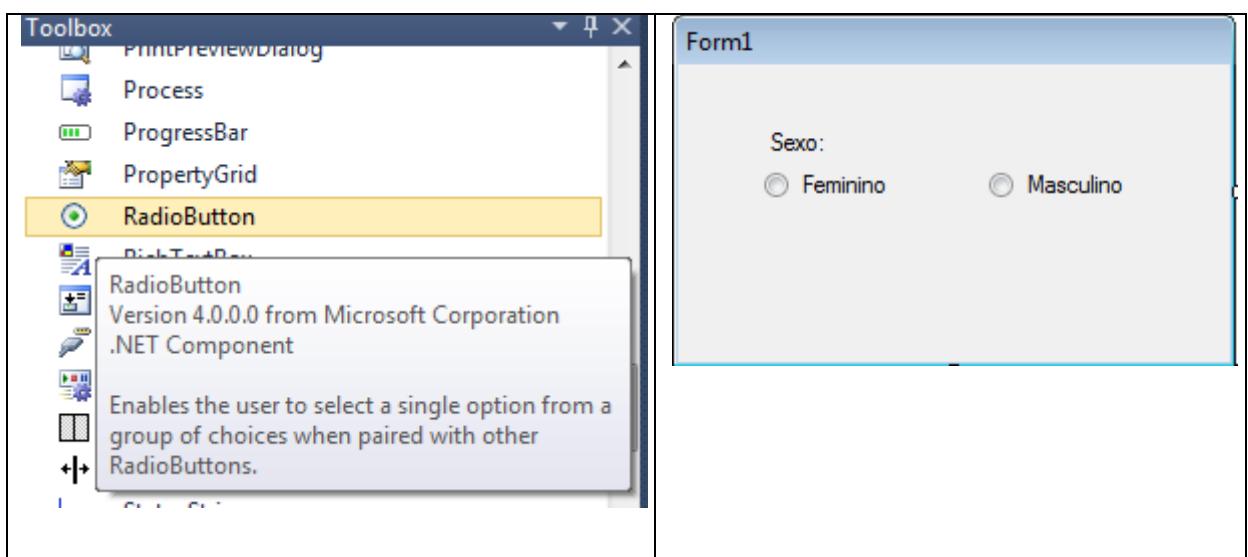
- O usuário deverá entrar com o valor do salário(textbox).
- Em seguida quando o usuário selecionar a faixa de imposto(evento SelectedIndexChanged do combobox), automaticamente será apresentado o valor do imposto no textbox.
- Quando o usuário clicar no botão será apresentado em uma label (que inicialmente estará com sua propriedade text sem valor) o total a receber.
- Considere as seguintes faixas a ser incluídas pela propriedade **Items** do combobox: Faixa 01, Faixa 02,Faixa 03 e Faixa 04.
- Para cada uma das faixas considere os seguintes descontos:
Faixa 01(sem desconto), Faixa 02(5%),Faixa 03(8%) e Faixa 04(11%).

Observação: As variáveis nesta atividades precisam publicas, pois estamos trabalhando com variáveis em eventos diferentes (**SelectedIndexChanged do combobox e click do botão**).



Objeto: RADIOPUSHBUTTON (botão radio)

Permite ao usuário escolher apenas uma opção entre diversas.



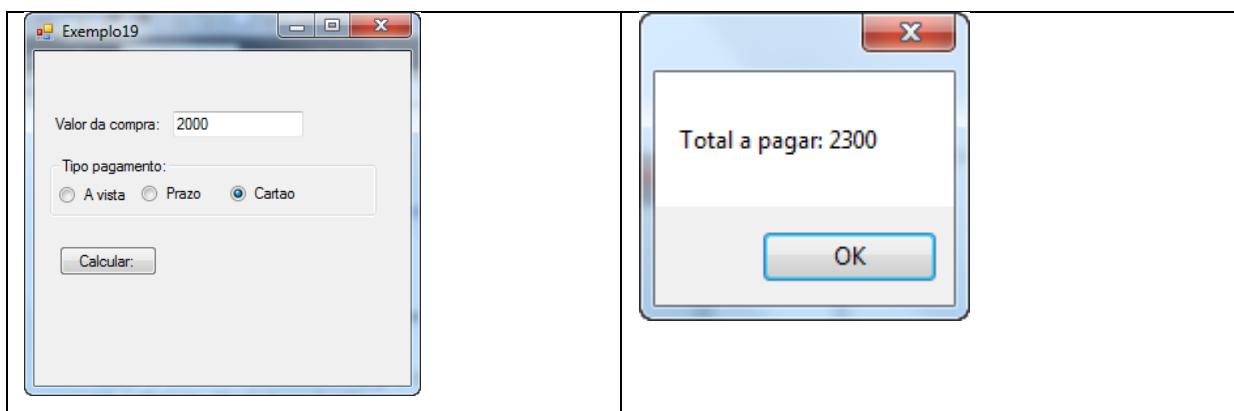
- ❖ *Text:* é o texto que vai estar vinculado ao radio Button.

- ❖ **Checked:** Define se um RadioButton está seleccionado se o valor for True ou se não está seleccionado caso o valor seja False.
- ❖ **Dock:** define se queremos o componente literalmente grudado em um ou mais cantos.
- ❖ **Enabled:** define se este controle está habilitado ou não, ou seja, se o usuário pode usá-lo ou não.
- ❖ **Location:** define a posição Top (distância do controle em relação a margem superior do formulário) e a posição Left (distância do controle em relação a margem esquerda do formulário).
- ❖ **TabIndex:** define a ordem que este controle está em relação aos demais controles do formulário quando apertamos a tecla TAB para mudar de controle.
- ❖ **Name:** propriedade que vai identificar o objeto durante o desenvolvimento de códigos.

Exemplo 19

Criar uma interface gráfica em C# com os seguintes objetos:

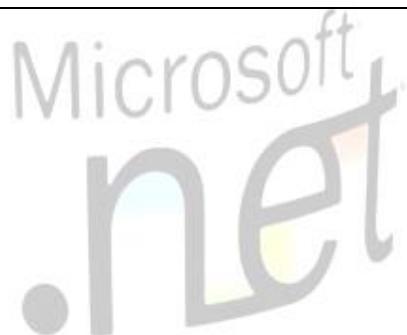
- | | | |
|--|------------------------------|-----------------------------|
| - Um textbox; | - Um button; | |
| - Três radiobutton; | - Um groupbox; | |
| - O usuário vai entrar no textbox o valor da compra; | | |
| - Inserir dentro do groupbox os três radiobutton, para o usuário selecionar o tipo de pagamento(a vista, a prazo, cartão); | | |
| - Quando o usuário clicar no button uma caixa de mensagem vai surgir com o valor a pagar, considerando os seguintes critérios: | | |
| • A vista → 10% de desconto | • A prazo → 20% de acréscimo | • Cartao → 15% de acrescimo |



Para programar este exemplo, precisamos saber qual tipo de pagamento o usuário selecionou. Quando um radiobutton está selecionado sua propriedade checked recebe o valor true e quando não está selecionado fica com o valor false.

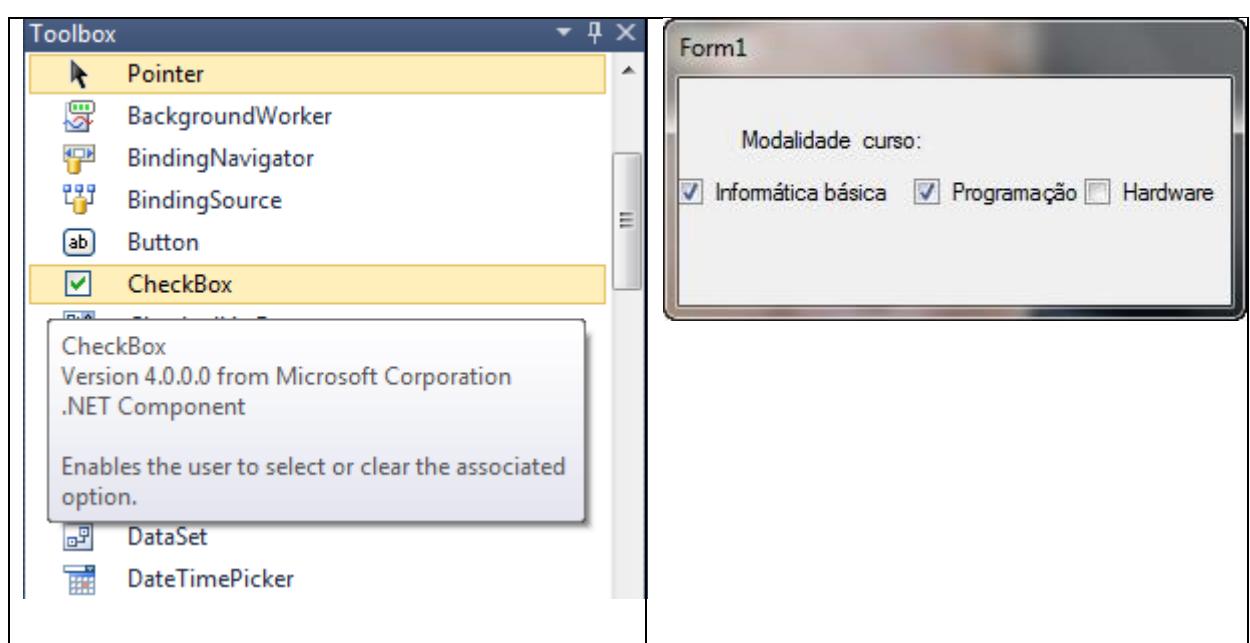


```
private void btncalcular_Click(object sender, EventArgs e)
{
    double valor, total;
    valor = Convert.ToDouble(txtcompra.Text);
    if (rdbvista.Checked == true)
        total = (valor * 0.90);
    else if (rdbprazo.Checked == true)
        total = (valor * 1.2);
    else
        total = (valor * 1.15);
    MessageBox.Show("Total a pagar: " + Convert.ToString(total));
}
```



Objeto: CKECKBOX (caixa de seleção)

Utilizado quando precisamos selecionar mais de uma opção.



- ❖ **CheckAlign:** Define o alinhamento do texto com o objeto.
- ❖ **Checked:** Define se um RadioButton está seleccionado se o valor for True ou se não está seleccionado caso o valor seja False.
- ❖ **CheckState:** Define o tipo de selecao do objeto sendo:

Checked: O CheckBox Exibe uma marca de seleção.

Unchecked : O CheckBox está vazio.

Indeterminate : O CheckBox Exibe uma marca de seleção e está sombreada.

- ❖ **Dock:** define se queremos o componente literalmente grudado em um ou mais cantos.
- ❖ **Enabled:** define se este controle está habilitado ou não, ou seja, se o usuário pode usá-lo ou não.
- ❖ **Location:** define a posição Top (distância do controle em relação a margem superior do formulário) e a posição Left (distância do controle em relação a margem esquerda do formulário).
- ❖ **Name:** propriedade que vai identificar o objeto durante o desenvolvimento de códigos.
- ❖ **TabIndex:** define a ordem que este controle esta em relação aos demais controles do formulário quando apertamos a tecla TAB para mudar de controle.
- ❖ **Text:** é o texto que vai estar vinculado ao radio Button.

Exemplo 20

Criar uma interface gráfica em C# com os seguintes objetos:

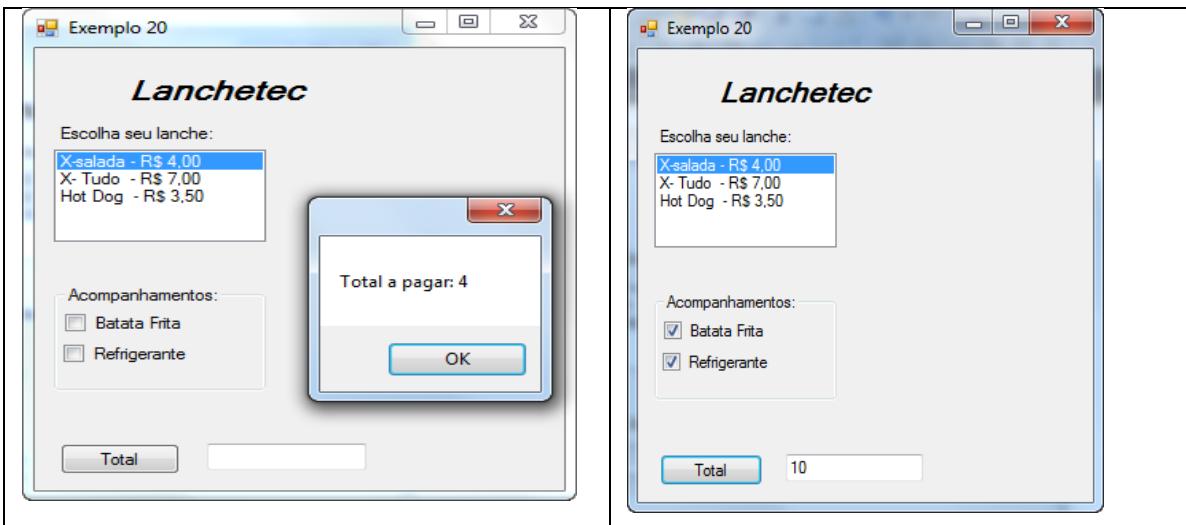
- | | |
|------------------|----------------|
| - Um textbox; | - Um button; |
| - Um listbox; | - Um groupbox; |
| - Dois checkbox; | |

- O usuário seleciona um lanche em um listbox, com as seguintes opções (X-Salada-X-Tudo-HotDog) e o valor do lanche será exibido em uma caixa de mensagem;
- Em seguida o usuário poderá escolher acompanhamentos (Batata Frita e Refrigerante);
- Quando o botão total for clicado será exibido na textbox o total.

Considere os seguintes valores:

X-Salada → 4,00/X-Tudo → 7,00/Hot-Dog → 3,50

Batata Frita → 3,50/ Refrigerante → 2,50



```

C# Microsoft .NET
namespace Exemplo20
{
    public partial class Form1 : Form
    {
        double valLanche, total, acomp1, acomp2;

        private void lstlanches_SelectedIndexChanged(object sender, EventArgs e)
        {
            accomp1 = 0;
            accomp2 = 0;
            if (lstlanches.SelectedIndex == 0)
                valLanche = 4;
            else if (lstlanches.SelectedIndex == 1)
                valLanche = 7;
            else if (lstlanches.SelectedIndex == 2)
                valLanche = 3.5;
            MessageBox.Show("Total a pagar: " +
Convert.ToString(valLanche));
        }

        private void button1_Click(object sender, EventArgs e)
        {

            //total = 0;
            if (chkrefrigerante.Checked == true)
                accomp1 = 2.5;
            if (chkbatata.Checked == true)
                accomp2 = 3.5;
            total=valLanche+acomp1+acomp2;

            txttotal.Text = Convert.ToString(total);
        }
    }
}

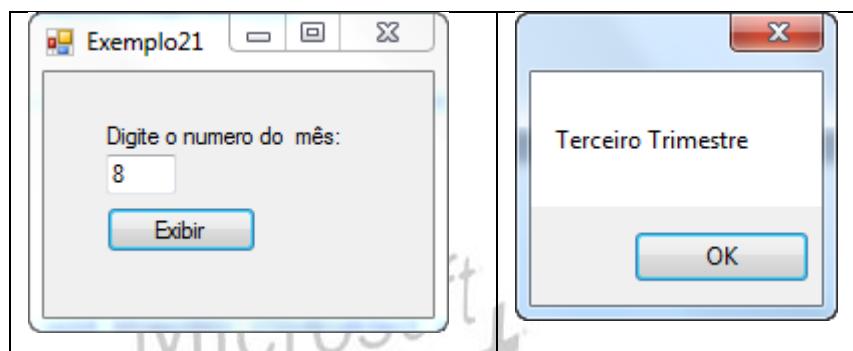
```

Neste exemplo utilizamos variáveis públicas, pois os valores dos lanches serão utilizados no evento SelectedIndexChanged do listbox, e depois no evento click do botão.

Exemplo 21

Criar uma interface gráfica em C# com os seguintes objetos:

- Um form;
- Um button;
- Um textbox
- Um label;
- O usuário vai digitar no textbox um numero referente ao mês e quando clicar no button será exibido uma caixa de mensagem referente a qual trimestre pertence este mês.
- Utilize a estrutura switch ...case



```
private void btnexibir_Click(object sender, EventArgs e)
{
    short mes;
    mes = Convert.ToInt16(txtmes.Text);
    switch (mes)
    {
        case 1:
        case 2:
        case 3:
            MessageBox.Show("Primeiro Trimestre");
            break;
        case 4:
        case 5:
        case 6:
            MessageBox.Show("Segundo Trimestre");
            break;
        case 7:
        case 8:
        case 9:
            MessageBox.Show("Terceiro Trimestre");
            break;
        case 10:
        case 11:
        case 12:
            MessageBox.Show("Quarto Trimestre");
            break;
    }
}
```

Neste exemplo declaramos a variável mês do tipo **short**, que é um tipo inteiro que armazena números inteiros de 0 a 255.

Utilizamos a estrutura switch...case pelo fato de termos muitas opções.

Quando utilizamos esta estrutura precisamos declarar a variável que será avaliada do tipo int,char,short.

Para fazer a conversão do tipo short utilizamos a função Convert.ToInt16.

CAPÍTULO

4

ATIVIDADES PROPOSTA



OBJETIVO:	Estruturas de seleção em C#
METODOLOGIA:	Desenvolvimento de projetos
DATA ENTREGA:	DATA: ___/___/___
LISTA_ATV_P.4	

ATIVIDADE 21

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 01 Groupbox
- 04 radiobutton
- 03 labels
- 01button
- 03 textbox

Eventos

Construa uma calculadora em que o usuário vai entrar com dois numeros (textbox)e escolher uma operação(soma/subtração/multiplicação/divisão) através de um radio button.

O total deverá ser exibido em uma caixa de texto.

Os radiobutton deverão estar dentro de um groupbox.

As labels deverão indicar dos conteúdos das caixas de texto.

ATIVIDADE 22

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

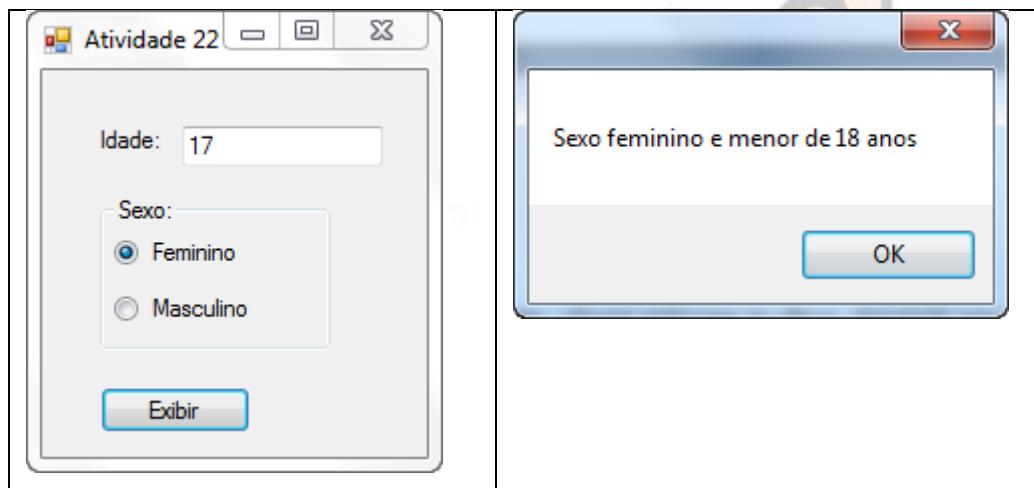
Objetos

- 01 Form
- 02 radiobutton
- 01button
- 01Groupbox
- 01 label
- 01textbox

Eventos

Construa um aplicativo para o usuário entrar com sua idade(textbox) e escolher seu sexo através de radiobutton.

Em seguida quando clicar em um button deverá ser exibido sua idade, seu sexo e se a pessoa é maior ou menor de 18 anos.



ATIVIDADE 23

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 03 checkbox
- 02 button
- 01Groupbox
- 09 label
- 07textbox

Eventos

- Construa um aplicativo para escolher itens de papelaria, o usuário deverá selecionar os itens desejados(caderno,lápis,caneta) através de checkbox e vai digitar a quantidade desejada.
- Quando o usuário clicar no botão calcular vai aparecer o valor de cada item e o valor a pagar.
- Considere os valores unitários e o layout da formulário conforme figura abaixo.

Itens:			
Material	Quantidade	Valor unitario:	Valor do item:
<input checked="" type="checkbox"/> Lapis:	3	0.70	2.1
<input type="checkbox"/> Caneta:		1.20	0
<input checked="" type="checkbox"/> Caderno:	2	12.30	24.6
			Total: 26.7

ATIVIDADE 24

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 02 radiobutton
- 01 button
- 01Groupbox
- 03 label
- 02 textbox

Eventos

- Construa um aplicativo para o usuário entrar com sua altura(textbox) e selecionar seu sexo(radiobutton);
- Quando o usuário clicar no botão deve ser exibida em outra textbox seu peso ideal de acordo com as condições abaixo:
 - Homens: $(72.7 * \text{altura}) - 58.0$
 - Mulheres: $(62.1 * \text{altura}) - 44.7$

ATIVIDADE 25

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 01 button
- 04 radiobutton
- 01Groupbox

Eventos

- Construa um aplicativo para o usuário selecionar o DDD através de um radiobutton e exibir em uma caixa de mensagem o nome da cidade do DDD selecionado.

Considere as seguintes condições

DDD	CIDADE
11	São Paulo.
21	Rio de Janeiro
31	Belo Horizonte
41	Curitiba

CAPÍTULO 5- FORMULÁRIOS E MENU EM C#

CAPÍTULO

5

OBJETIVOS

Conhecer os tipos de dados e funções de conversão.

CONTEÚDO

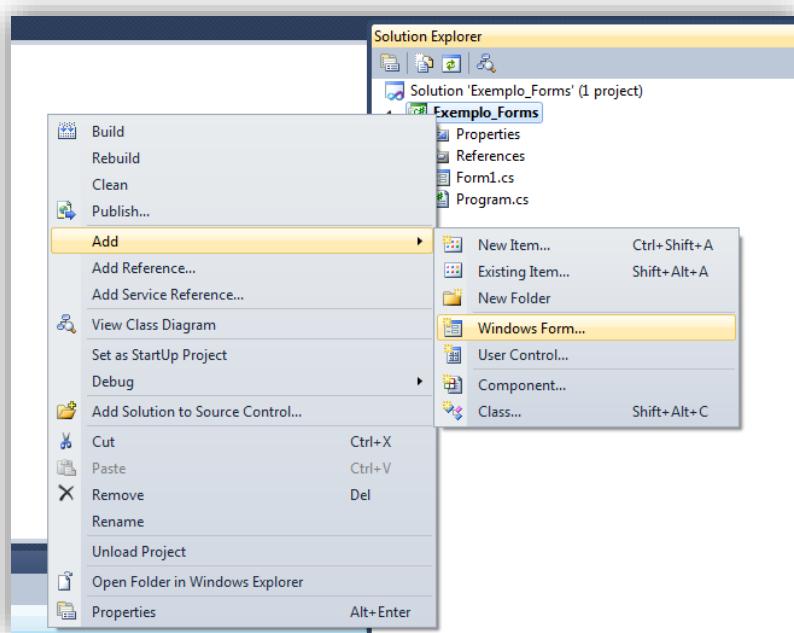
Um programa realmente não é desenvolvido com apenas um formulário(Form). Sempre temos vários deles nos nossos programas, e agora vamos aprender a trabalhar com diversos forms em nossa aplicação.

TRABALHANDO COM MÚLTIPLOS FORMULÁRIOS

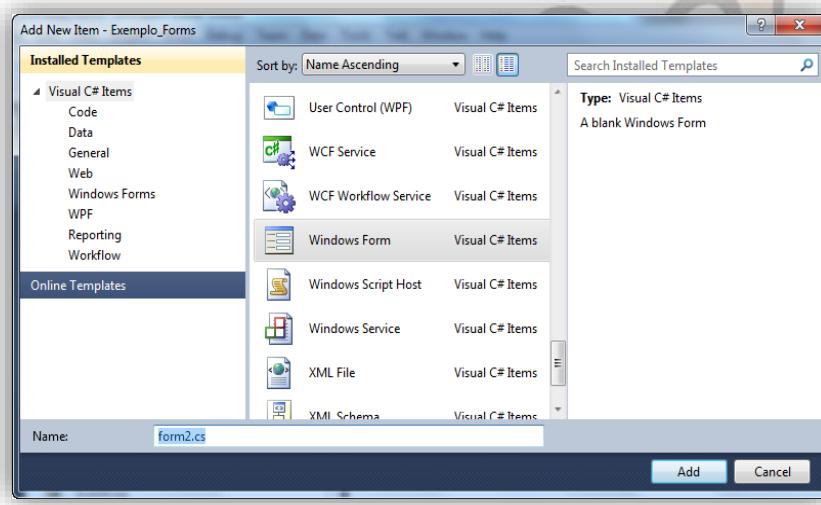
Sempre que criamos uma Windows Application um formulário já é criado por padrão, e nomeado como Form1.

Para adicionar um novo formulário na sua solução você:

1- Na janela Solution Explorer, clique com o botão direito no nome do projeto, selecione Add e clique em Windows Form. Como a imagem seguinte.

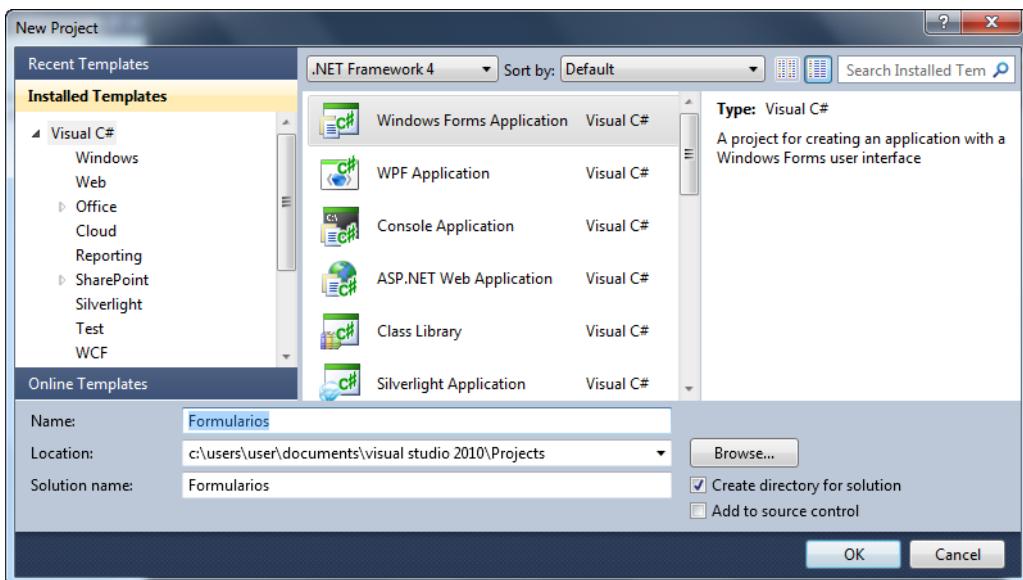


2- Escolha a opção Windows Form e digite um nome para o novo formulário e clique em Add.



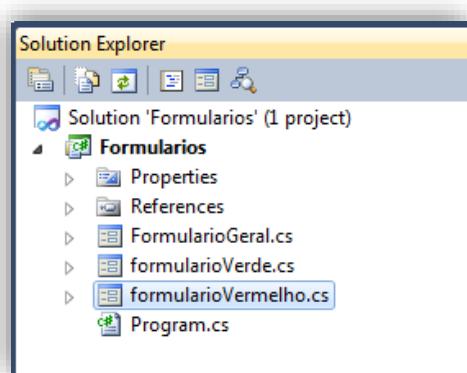
Exemplo 22

1- Crie uma Windows Application chamada Formularios.



- 2- Arraste 2 buttons para o Form1.
- 3- Altere a propriedade name do form1 para frmgeral
- 4- Mude as propriedades name dos Buttons 1 e 2 para btnverde e btnvermelho.
- 5- Mude as propriedades text dos Buttons 1 e 2 para Verde e Vermelho.
- 6- Adicione mais 2 forms no projeto.
- 7- Dê o nome aos arquivos dos formulário de formularioVermelho e formularioVerde.
- 8- Altere a propriedade name dos formulários de frmVermelho e frmVerde.
- 9- Altere a cor dos formulários frmVermelho para vermelho e frmVerde para verde.

Observe como está nosso projeto:



Repare o nome dos arquivos dos formulários (Formularioverde e Forumlariovermelho), caso seus arquivos estejam com nomes diferentes , clique com o botão direito sobre o form e selecione a opcao Rename e altere o nome, conforme figura anterior.

Agora precisamos programar os botões para que os formulários sejam abertos. Veja o código abaixo do evento click dos botões btnvermelho e btnverde.



```
private void btnvermelho_Click(object sender, EventArgs e)
{
    formularioVermelho Vermelho = new formularioVermelho();
    Vermelho.Show();
    this.Hide();
}

private void btnverde_Click(object sender, EventArgs e)
{
    formularioVerde Verde = new formularioVerde();
    Verde.ShowDialog();
    this.Hide();
}
```

Para abrir outro formulário sempre utilizamos o nome do formulário(nome do arquivo).

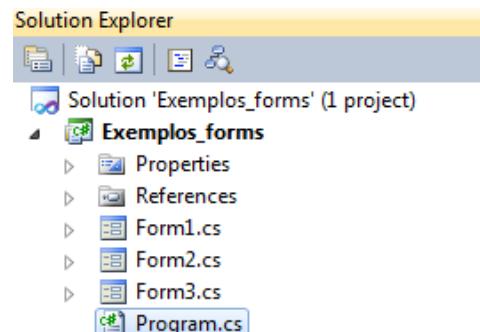
Repare que ao executar o projeto você não consegue voltar para o formulário geral sem antes fechar o formulario Verde, isso porque usamos o metodo Showdialog ao invés do método Show.

ALTERANDO O FORMULÁRIO DE INICIALIZAÇÃO

Quando estamos trabalhando com diversos formulário em nossa aplicação, por padrão o primeiro formulário (form1) será o form de inicialização.

Se quisermos alterar o form de inicialização, precisamos fazer o seguinte:

- No Solution Explorer, selecione o arquivo Program.cs;



- No nosso exemplo, nossa solução (Exemplo_forms) possue 03 forms(form1,form2,form3), por padrão o form1 é carregado inicialmente:
- Vamos alterar para iniciar pelo form3;
- Quando clicamos no program.cs é apresentado a seguinte tela:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Windows.Forms;

namespace Exemplos_forms
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()
        {
            Application.EnableVisualStyles();
            Application.SetCompatibleTextRenderingDefault(false);
            Application.Run(new Form1());
        }
    }
}
```

Na ultima linha onde temos: `Application.Run(new Form1());`

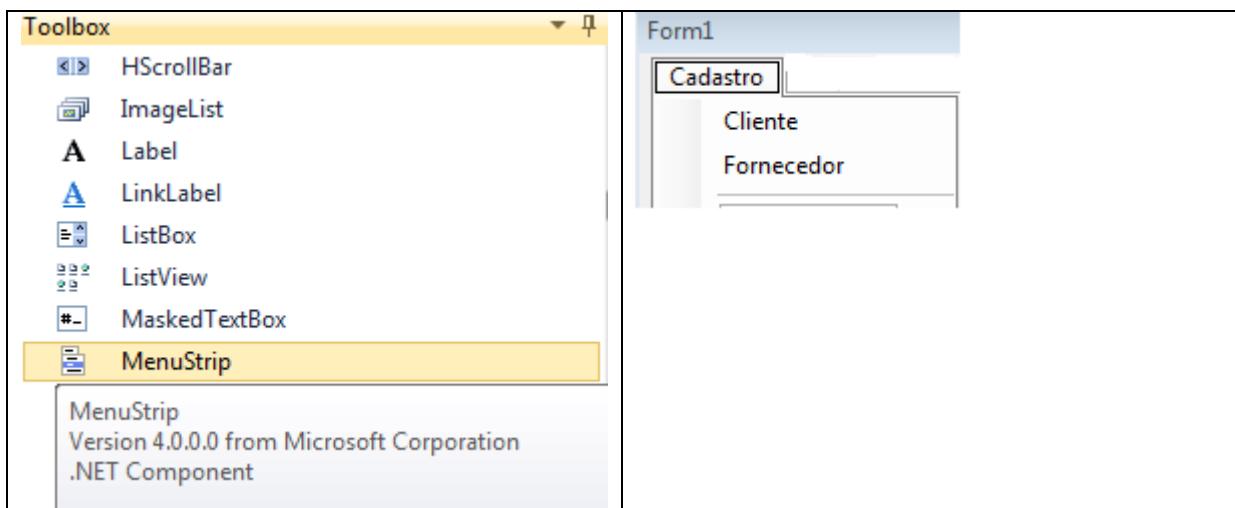
Alteramos e no lugar do Form1, colocamos o nome do form que queremos iniciar.

No nosso caso o form3, ficando assim:

`Application.Run(new Form3());`

Objeto: MENUSTRIPO (menu)

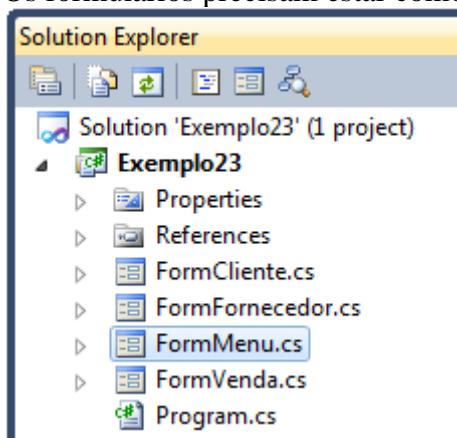
Os menus são colocados no fundo da *form*. De maneira geral, um modo eficiente de apresentar múltiplas opções ao utilizador. É possível adicionar diversos itens a um menu.



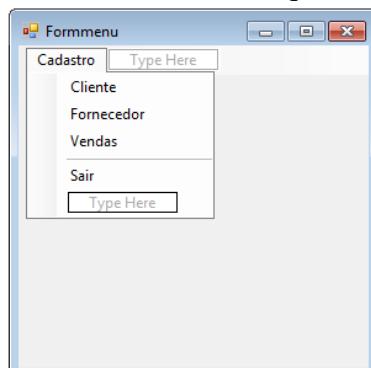
Exemplo 23

- 1- Crie uma Windows Application chamada Exemplo23;
- 2- Renomeie (botão direito no form1 opção Rename) o form1 para Formmenu;
- 3- Adicione mais três formulários e dê o nome as arquivos de FormCliente, FormFornecedor e FormVenda.

Os formulários precisam estar como na figura abaixo:



- 4- No Formmenu coloque um menustrip , deixando como imagem abaixo:



Para inserir a linha divisória e só digitar – e dar <ENTER>

5- Programe que as opções(Cliente,Fornecedor e Venda) do menu abra seu formulário correspondente, e quando os forms forem abertos o formMenu deverá estar descarregado da memória.

6- Em cada formulário deverá ter um botão para voltar para o menu;

7- Quando o usuário selecionar a opção Sair, o aplicativo deverá ser encerrado.

Para programar as opções do menu é só dar um click em cada opção que será aberto a tela de código.



```
private void fornecedorToolStripMenuItem_Click(object sender, EventArgs e)
{
    FormFornecedor Fornecedor = new FormFornecedor();
    Fornecedor.Show();
    this.Hide();
}

private void sairToolStripMenuItem_Click(object sender, EventArgs e)
{
    Application.Exit();
}

private void clienteToolStripMenuItem_Click(object sender, EventArgs e)
{
    FormCliente Cliente = new FormCliente();
    Cliente.Show();
    this.Hide();
}

private void vendasToolStripMenuItem_Click(object sender, EventArgs e)
{
    FormVenda Venda = new FormVenda();
    Venda.Show();
    this.Hide();
}
```

Para fechar nosso aplicativo usamos a seguinte instrução:

`Application.Exit();`

No evento click dos botões dos três formulários (FormCliente,FormFornecedor e FormVenda) colocamos o seguinte código.

```
private void btnmenu_Click(object sender, EventArgs e)
{
    FormMenu Menu = new FormMenu();
    Menu.Show();
    this.Hide();
}
```

6

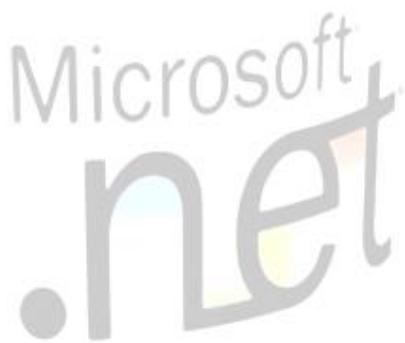
CAPÍTULO

CAPÍTULO 6- FORMATAÇÃO DE DADOS E DATAS

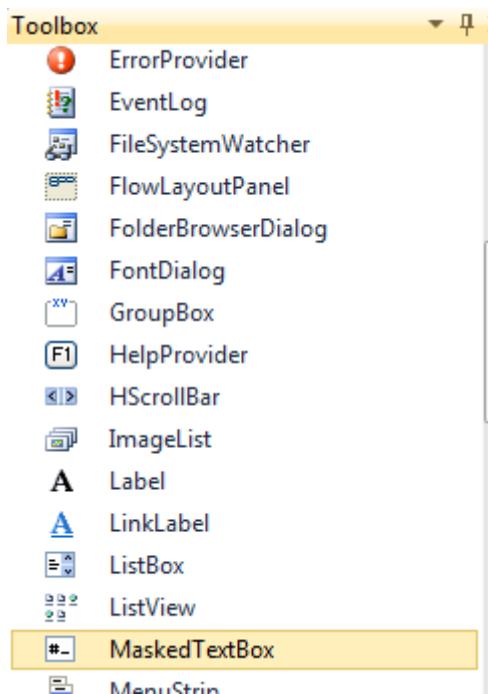
OBJETIVOS

Conhecer os tipos de dados e funções de conversão.

CONTEÚDO



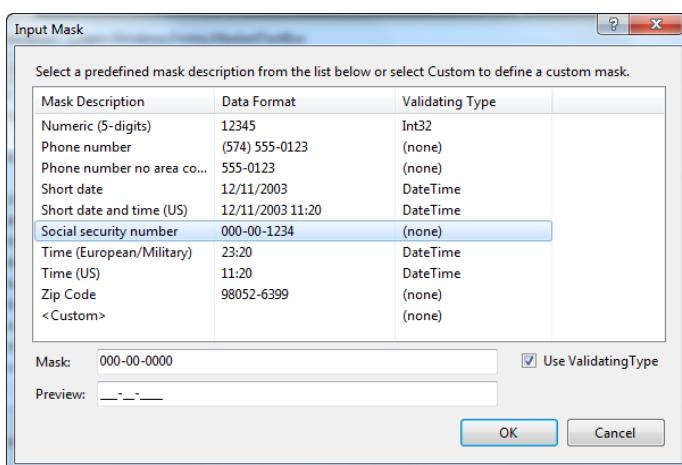
Objeto: MASKEDTEXTBOX (caixas de texto formatada)



PROPRIEDADES

- ❖ **Name:** Define o nome do objeto.
- ❖ **Mask:** Obtém ou define a máscara de entrada para usar em time de execução.

O Visual Studio dispõe de algumas máscaras prontas para utilização.



Podemos também personalizar as máscaras de entrada de acordo com nossa necessidade.

Para isto precisamos utilizar caracteres próprios conforme tabela abaixo:

0	Dígito 0 de entrada obrigatória
---	---------------------------------

9	Dígito de 9 entrada não obrigatória.
L	Caractere de A Z. Entrada obrigatória.
?	Caractere de A Z. Entrada não obrigatória
A	Caractere de A Z, ou dígito de 0 a 9. Entrada não obrigatória.
&	Qualquer caractere. Entrada obrigatória.
C	Qualquer caractere. Entrada não obrigatória
.	Espaço reservado decimal
,	Milhares espaço reservado
:	Separador de time
/	Separador de data
\$	Símbolo de moeda

Exemplos:

CPF: mascara= 000.000.000-00

tels: mascara=(00)0000-0000

cep: mascara= 00000-000

data nascimento: mascara=00/00/0000

moeda: mascara= \$999,999.00



EVENTOS

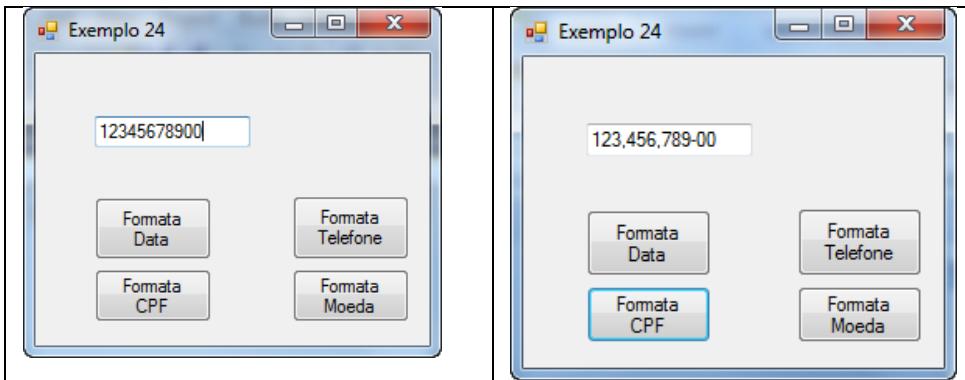
❖ **Leave:** Evento é disparado quando o objeto perde o foco.

❖ **MaskInputRejected:** Evento utilizado para validar dados de um maskedtextbox.

Exemplo 24

Criar uma interface gráfica em C# com os seguintes objetos:

- Um form;
- Um maskedtextbox;
- Três button;
- O usuário vai digitar valores no maskedtextbox(mskvalor) e quando clicar nos botões (btnformatadata, btnformatatelefone, btnformatacpf, btnformatamoeda) vai aparecer na maskedtextbox o valor formatado.



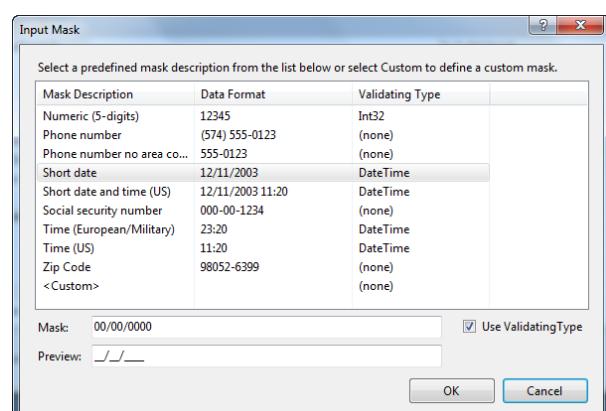
Podemos configurar a máscara em tempo de design através da propriedade mask ou em tempo de execução.

	<pre> private void btnformatadata_Click(object sender, EventArgs e) { mskvalor.Mask = "00/00/0000"; } private void btnformatatelefone_Click(object sender, EventArgs e) { mskvalor.Mask = "(00)0000-0000"; } private void btnformatacpf_Click(object sender, EventArgs e) { mskvalor.Mask = "000.000.000-00"; } private void btnformatamoeda_Click(object sender, EventArgs e) { mskvalor.Mask = "\$0000,00"; } </pre>
--	---

Exemplo 25

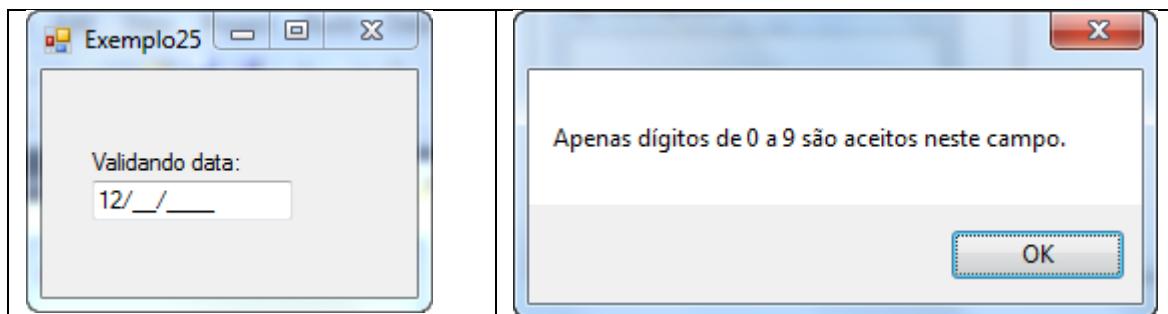
Criar uma interface gráfica em C# com os seguintes objetos:

- Um form;
- Um maskedtextbox;
- Um label
- Formatar a propriedade mask do maskedtextbox(mskdata) para “00/00/0000” ou escolha date short.



- Ao digitar dados no maskedtextbox, caso o usuário digite qualquer valor que não seja numero será exibido uma mensagem de alerta.

A validação de dados em um maskedtextbox é feita através do evento MaskInputRejected, o qual verifica o dado digitado com os caracteres definidos na propriedade mask.



```
private void mskdata_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
{
    MessageBox.Show("Apenas dígitos de 0 a 9 são aceitos neste campo.");
}
```

Formatar String C# (String.Format)

O uso de formatações de expressões nas aplicações é fundamental para uma boa apresentação.

O .NET dispõe da classe String, o qual permite manipular uma string, por exemplo copiar, comparar, concatenar e formatar.

Sintaxe:

```
String padrão = String.Format("Este {0} é o que muda", variável);
```

O .net possuí um método para formatar string chamada Format, dentro da classe String. Este método exige 2 parâmetros de entrada:

1º O formato: Este diz respeito a como será feita a apresentação do dado. Por exemplo se queremos formatar um número para ter 5 casas decimais, é aqui que informamos.

2º O dado a ser formatado: Este parâmetro é do tipo object, suportando qualquer tipo de

dado, Inteiro, Decimal, Double, etc. Porém será gerada uma exception caso a formatação não for possível, por isso é necessário um cuidado especial neste ponto.

Exemplos:

- a) Converter conteúdo de um textbox para numero float(decimal).

```
textBox1.Text = String.Format("{0:f}", Convert.ToSingle(textBox1.Text));
```

- b) Converter conteúdo de um textbox para moeda(currency).

```
textBox1.Text = String.Format("{0:c}", Convert.ToDecimal(textBox1.Text));
```

O parâmetro {0}, {1}, ...{n} significa o índice do valor que queremos formatar, depois o :f, :c é o tipo do dados que queremos formatar, e depois precisamos converter o conteúdo da caixa de texto.

- c) `private void button1_Click(object sender, EventArgs e)`

```
{
```

```
    Double num1;  
    DateTime datahora;  
    datahora = DateTime.Now;  
    num1 = 10;
```

```
    label1.Text = String.Format("O valor formatado para moeda e {0:c} A data atual:  
{1:T}", Convert.ToDecimal(num1), Convert.ToDateTime(datahora));  
}
```

Neste exemplo declaramos duas variaveis num e datahora, depois precisamos exibir seus valores formatados para moeda e data e hora.

A função DateTime.Now exibe a data e hora do sistema atual.

Neste exemplo temos dois índices {0} e {1} pois precisamos dois dados representados pelas variáveis .

No índice {0} utilizamos o caracter de formatação :c, para formatar para moeda(currency).

No índice {1} utilizamos o caracter de formatação :f, para formatar a data com a data por extenso.

Executando nosso exemplo ficará assim:

```
O valor formatado para moeda é R$ 10,00 A data atual: sexta-feira, 6 de janeiro de 2012 14:52
```

Segue abaixo tabela com os caracteres de formatação mais usado:

Caracter	Tipo	Formato	Saída (Ex.: Tipo double 1.2345)
c	Currency (Moeda)	{0:c}	R\$12,45
f	Ponto fixo	{0:f}	1.23
r	Arredondado	{0:r}	1.23

Caracter	Tipo	Saída (Ex.: 19 de Setembro de 1980 14:30:59)
d	Data curta	19/09/1980
D	Data long	19 Setembro 1980
t	Hora Curta	14:30
T	Hora Longa	14:30:59
f	Data e Hora	19 Setembro 1980 14:30
F	Data e Hora completo	19 Setembro 1980 14:30:59
g	Data de Hora padrão	19/09/1980 14:30
G	Data de Hora padrão longo	19/09/1980 14:30:59
M	Dia / Mês	19 Setembro
Y	Mês / Ano	Setembro 1980

Caracter	Tipo	Saída (Ex.: Setembro 19, 1980 14:30:59)

dd	Dia	19
ddd	Nome curto do dia	Sex
dddd	Nome completo do dia	Sexta-feira
hh	2 dígitos para a hora	02
HH	2 dígitos para a hora (24 horas)	14
mm	2 dígitos para o minuto	30
MM	Mês	09
MMM	Nome curto do Mês	Set
MMMM	Nome do Mês	Setembro
ss	Segundos	59
tt	AM/PM	PM
yy	2 dígitos do Ano	80
yyyy	4 dígitos do Ano	1980
:	Separador, ex. {0:hh:mm:ss}	14:30:59
/	Separador, ex. {0:dd/MM/yyyy}	08/06/1970

Trabalhando com Data e Hora

Declarar uma variável de data é uma tarefa extremamente simples, basta informar o tipo `DateTime` e o nome da variável. A variável `DateTime` é representada por uma estrutura na bilbioteca de classes do .NET e possui uma série de métodos estáticos e não estáticos que permitem a sua manipulação. Dentre as possibilidades da estrutura de datas o próprio construtor já apresenta uma série de possibilidades para podermos inicializar uma estrutura de datas.

É possível inicializar a variável com a data e hora atual através dos métodos estáticos `Now` e `Today`. O método `Now` representa a data e hora atuais, já o método `Today` representa apenas a data sem o componente da hora.

Após a inicialização da data é possível obtermos os componentes da data separadamente através das propriedades da variável, conforme o exemplo a seguir.

```
int hora = data1.Hour; //retorna a hora  
int dia = data1.Day;//retorna o dia  
int mes = data1.Month;//retorna o mes  
int ano = data1.Year;//retorna ano
```

A variável que vai pegar parte da data precisar ser do tipo inteiro.

Exemplo:

a) Exibir o mês da data atual

```
private void button2_Click(object sender, EventArgs e)  
{  
    DateTime hora;  
    int hora1,hora2;  
    hora = DateTime.Now;  
    hora1 = hora.Month  
    label1.Text = Convert.ToString(hora1);  
}
```

O exemplo anterior exibimos o mês da data atual.

b) Adicionando componentes a uma data

A partir de uma variável data é possível adicionarmos dias, meses, anos, horas, minutos, etc. Para adicionarmos estas componentes a uma estrutura de datas devemos utilizar os métodos AddXXX correspondentes (ex. AddDays, AddYear, AddMinutes). O exemplo a seguir mostra a utilização destes métodos.

Exemplo:

Adicionar 10 dias a data atual.

```
private void button2_Click(object sender, EventArgs e)  
{  
    DateTime hoje = DateTime.Now;
```

```
        DateTime data_futura = hoje.AddDays(10);
        label1.Text = Convert.ToString(data_futura.Day);
    }
```

c) Textbox receber data atual

```
TextBox1.Text = DateTime.Today.ToString();
```

d) Exibir a data digitada em uma caixa de texto

```
private void button3_Click(object sender, EventArgs e)
{
    DateTime a, b;
    a = DateTime.Now;
    b = Convert.ToDateTime(textBox1.Text);
    MessageBox.Show(Convert.ToString(b));
}
```

DIFERENÇA ENTRE DATAS

Outro método bastante útil na classe DateTime é o método Subtract que permite a Subtração de uma data a partir de outra. Através deste método podemos descobrir quanto tempo transcorreu entre uma data e outra. O resultado desta operação é outra estrutura bastante útil, a estrutura TimeSpan.

A estrutura TimeSpan representa um intervalo de tempo. No caso de uma diferença entre datas, ela representa o tempo transcorrido entre uma data e outra.

```
private void button4_Click(object sender, EventArgs e)
```

```
{  
    DateTime a = DateTime.Now;  
    DateTime b = DateTime.Now.AddDays(180);
```

```
    TimeSpan result = b.Subtract(a);
```

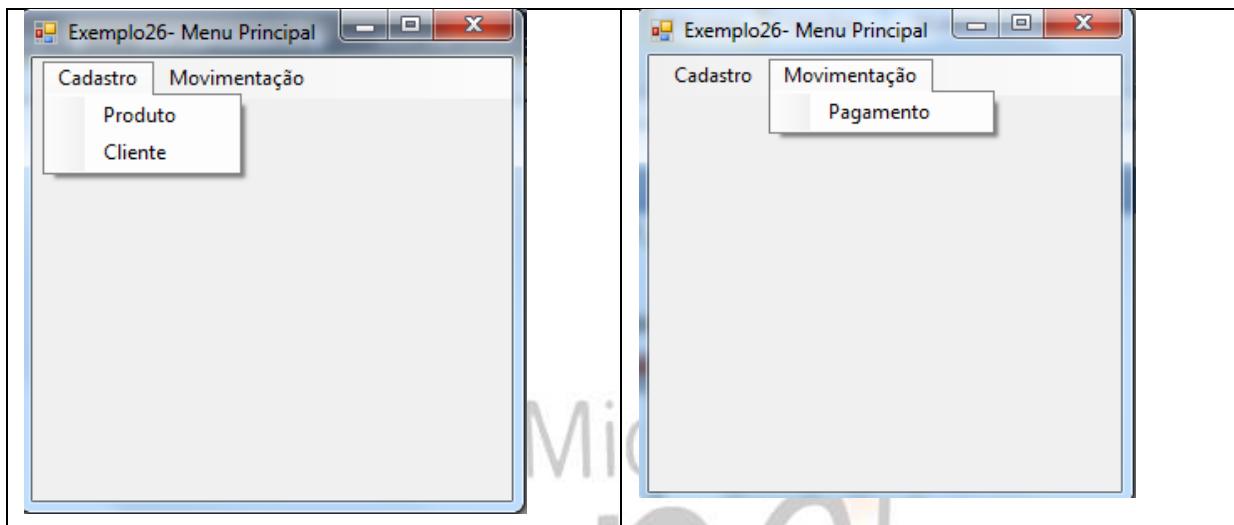
```
    MessageBox.Show("A diferença entre as duas datas" + Convert.ToString(result.TotalDays));  
}
```

O objeto TimeSpan representa um intervalo de tempo, ou duração de tempo, medida como um número positivo ou negativo de dias, horas, minutos, segundos e frações de segundo.

Exemplo 26

Criar uma interface gráfica em C# para o projeto abaixo:

- Um form principal(frmMenu)
- Um menu conforme imagem abaixo:



- Um form para Cadastro de Produto;
- Um form para Cadastro de Cliente;
- Um form para Pagamento de Compra;

Codificação e Layout:

FORM MENU(FrmMenu.cs)

Objeto:	Propriedade:	Valor:
Form	Name	frmMenu
	Nome do arquivo	FrmMenu.cs
	Text	Exemplo26- Menu principal
Form	Name	frmProduto
	Nome do arquivo	frmCadproduto.cs
	Text	Cadastro de Produto
Form	Name	frmCliente
	Nome do arquivo	frmCadCliente.cs
	Text	Cadastro de Cliente

Form	Name	frmCompra
	Nome do arquivo	frmCadCompra.cs
	Text	Compra

No menu principal precisamos abrir os outros formulários, conforme códigos abaixo:



```

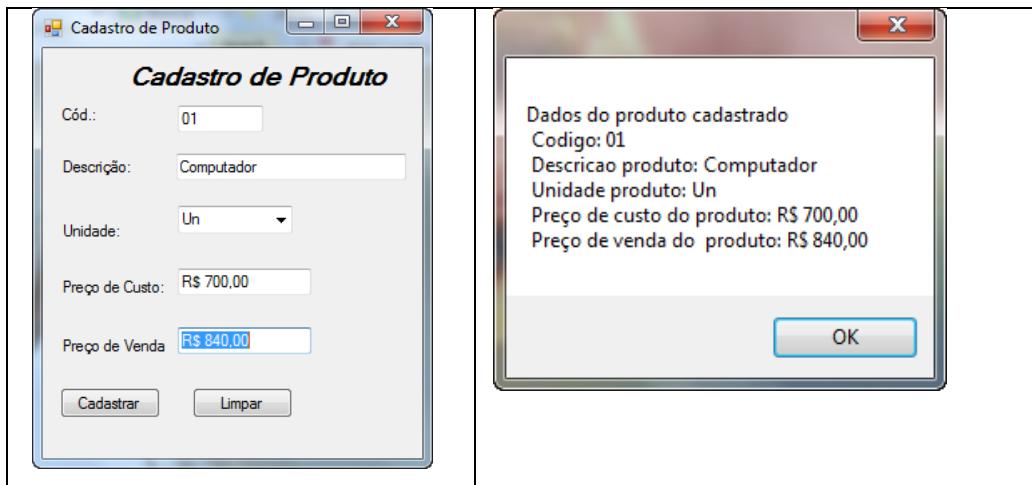
//abrir o formulário de produto
private void produtoToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmProduto produto = new frmProduto();
    produto.Show();
    this.Hide();
}

//abrir o formulário de pagamento de compra
private void pagamentoToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmCompra Compra = new frmCompra();
    Compra.Show();
    this.Hide();
}

//abrir o formulário de cadastro de cliente
private void clienteToolStripMenuItem_Click(object sender, EventArgs e)
{
    frmCliente cadcliente = new frmCliente();
    cadcliente.Show();
    this.Hide();
}
}

```

FORM Produto (frmCadproduto.cs)



No cadastro de produto coloque os seguintes objetos:

Objeto:	Propriedade:	Valor:
TextBox	Name	txtcod
	Name	frmdesc
	Name	txtval_custo
	Name	txtval_venda
Combobox	Name	cbounid
	Items	Kg Pcte Mts Um
Button	Name	btncadastrar
	text	Cadastrar
	Name	btnlimpar
	Text	Limpar

Eventos:

- No textbox(txtval_custo) o usuário vai digitar o preço de custo;
- Quando o usuário sair deste textbox automaticamente :
 - O valor do txtval_custo vai ser formatado para moeda;
 - O valor do txtval_venda vai receber o valor do preço de custo acrescido de 20% e também deverá estar formatado para moeda.
- Quando o usuário clicar no botão (btncadastrar) será exibido uma caixa de mensagem com os dados cadastrados.
- Quando o usuário clicar no botão limpar os objetos deverá estar sem conteúdos;

Programação:

- Depois que o usuário digita um valor em um objeto, no nosso caso textbox é disparado o evento **Leave**, é neste evento que o nosso código e formatação das caixas de texto(txtval_custo,txtval_venda) será implementada.



```

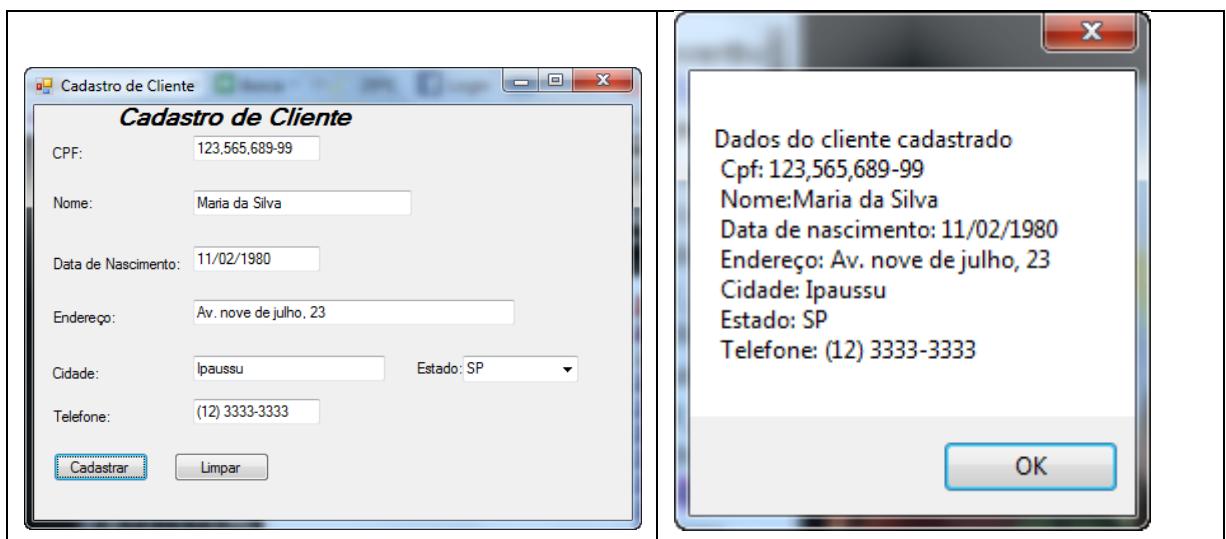
private void txtval_custo_Leave(object sender, EventArgs e)
{
    Double val_custo, val_venda;
    val_custo = Convert.ToDouble(txtval_custo.Text);
    val_venda = val_custo * 1.2;
    txtval_custo.Text =
        String.Format("{0:C}", Convert.ToDouble(txtval_custo.Text));
    txtval_venda.Text = String.Format("{0:C}", Convert.ToDouble(val_venda));
}

private void btncadastrar_Click(object sender, EventArgs e)
{
    MessageBox.Show("Dados do produto cadastrado" + "\n Código: " + txtcod.Text
        + "\n Descrição produto: " + txtdesc.Text + "\n Unidade produto: " +
        cboUnid.Text + "\n Preço de custo do produto: " + txtval_custo.Text + "\n
        Preço de venda do produto: " + txtval_venda.Text);
}

private void btnlimpar_Click(object sender, EventArgs e)
{
    txtcod.Text = "";
    txtdesc.Text = "";
    txtval_venda.Text = "";
    txtval_custo.Text = "";
    cboUnid.Text = "";
}

```

FORM Cliente (frmCadCliente.cs)



No cadastro de cliente coloque os seguintes objetos:

Objeto:	Propriedade:	Valor:
TextBox	Name	txtcidade
	Name	frmendereco
	Name	txtnome
MaskedTextBox	Name	mskcpf
	mask	000.000.000-00
	Name	mskDtnasc
	mask	00/00/0000
	Name	mskTelefone
	mask	(99) 0000-0000
Combobox	Name	cboestado
	Itens	Inserir as siglas dos Estados
Button	Name	Btncadastrar
	Text	Cadastrar
	Name	Btnlimpar
	Text	Limpar

Eventos:

- Nos MaskedTextBox's (mskcpf, mskDtnasc, mskTelefone) será aceito apenas números durante a digitação;
- No MaskedTextBox da data de nascimento precisa ser validado se a data é válida;
- Quando o usuário clicar no botão cadastrar deverá ser exibido uma caixa de mensagem com os dados cadastrados;
- Quando o usuário clicar no botão limpar os objetos deverá estar sem conteúdos;

Programação:

- Para validar os valores para ser aceito apenas aqueles definidos na propriedade mask, utilizamos o evento **MaskInputRejected**, colocamos uma mensagem que o valor digitado não é válido.
- Para ocorrer a validação de datas utilizamos o evento **TypeValidationCompleted** e precisamos fazer a validação de dados de entrada utilizando a estrutura IF, conforme abaixo:

```
if (!e.IsValidInput)
```



```
private void mskcpf_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
{
    MessageBox.Show("Digite apenas numeros" );
}

private void mskDtnasc_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
{
    MessageBox.Show("Digite apenas numeros" );
}

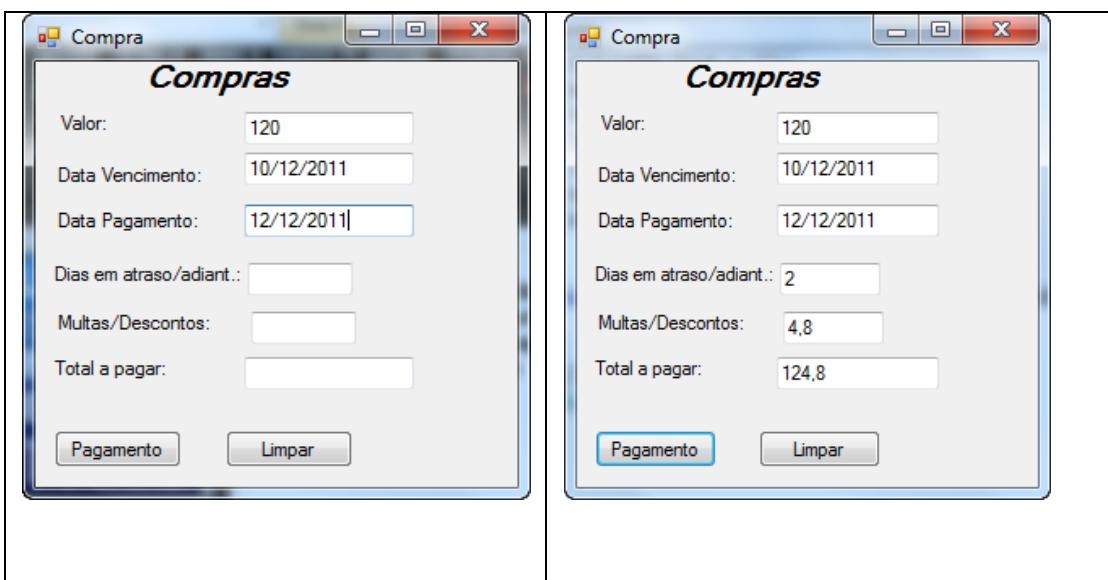
private void msktelefone_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
{
    MessageBox.Show("Digite apenas numeros" );
}

private void btncadastrar_Click(object sender, EventArgs e)
{
    MessageBox.Show("Dados do cliente cadastrado" + "\n Cpf: " +
mskcpf.Text + "\n Nome:" + txtname.Text + "\n Data de nascimento: " +
mskDtnasc.Text + "\n Endereço: " + txtendereco.Text + "\n Cidade: " +
txtcidde.Text + "\n Estado: " + cboestado.Text + "\n Telefone: " +
msktelefone.Text );
}

private void btnlimpar_Click(object sender, EventArgs e)
{
    txtcidde.Text = "";
    mskcpf.Text = "";
    txtname.Text = "";
    txtendereco.Text = "";
    cboestado.Text = "";
    mskDtnasc.Text = "";
    msktelefone.Text = "";
}

private void mskDtnasc_TypeValidationCompleted(object sender,
TypeValidationEventArgs e)
{
    if (!e.IsValidInput)
        MessageBox.Show("Data inválida");
}
```

FORM Compra (frmCadCompra.cs)



No cadastro de compra coloque os seguintes objetos:

Objeto:	Propriedade:	Valor:
TextBox	Name	txtvalcompra
	Name	txtadicional
	Name	txtdias
	Name	txttotal
MaskedTextBox	Name	Mskpgto
	Mask	00/00/0000
	Name	Mskvenc
	Mask	00/00/0000
Button	Name	BtnPagar
	Text	Pagamento
	Name	Btnlimpar
	Text	Limpar

Eventos:

- Nos MaskedTextBox's (Mskpgto, Mskvenc) será aceito apenas números durante a digitação;
- O usuário vai digitar o valor do pagamento na textbox(txtvalcompra) em seguida entrará com a data de vencimento e data de pagamento(Mskpgto, Mskvenc);
- Quando o usuário clicar no botão calcular será exibido os dias em atraso/adiantamento(txtdias), o valor da multa/desconto(txtadicional) e o total a pagar(txttotal).

Para efetuar os cálculos considere as seguintes condições:

- Pagamentos em atraso serão cobrados uma multa de 2% sobre o valor da compra por cada dia em atraso;

- Pagamentos feitos antes do vencimento terá um desconto de 3% sobre o valor da compra;
 - Pagamentos feitos no dia do vencimento não terá descontos/acréscimos.
- O campo que vai receber o valor da compra (txtvalcompra) não poderá estar em branco, senão ocorrerá um erro no cálculo;
- Os campos de data deverão ser validados para aceitar valores válidos.

Programação:

- Para efetuar o cálculo precisamos saber o número de dias em atraso/adiantamento no pagamento, para isto precisamos calcular a diferença entre as datas de pagamento e vencimento, para isto utilizamos a estrutura Timespan como vimos no tópico anterior.
- Lembrando que o resultado retornado pela estrutura Timespan, neste caso nos interessa apenas os dias, por isso utilizamos o método .Days.
- Em seguida precisamos efetuar os cálculos de acordo com o número de dias (maior que zero- Pgto em atraso/igual a zero – Pgto no dia do vencimento/menor que zero- Pgto adiantado) obedecendo os valores visto anteriormente.
- Antes dos cálculos serem realizados precisamos verificar se o usuário digitou algum valor , para isto faremos a seguinte verificação :

```
if (txtvalcompra.Text == "")  
{  
    MessageBox.Show("Campo não pode ficar em branco");  
    txtvalcompra.Focus();  
    return;  
}
```

- Para ocorrer a validação de datas utilizamos o evento **TypeValidationCompleted** e precisamos fazer a validação de dados de entrada utilizando a estrutura IF, conforme abaixo:

```
if (!e.IsValidInput)
```



<pre>private void btnguardar_Click(object sender, EventArgs e) { DateTime venc, pgto; double adicional, total, valor; int dias; if (txtvalcompra.Text == "") { MessageBox.Show("Campo não pode ficar em branco"); txtvalcompra.Focus(); return; } valor = Convert.ToDouble(txtvalcompra.Text); venc = Convert.ToDateTime(mskvenc.Text);</pre>	
---	--

```
pgto = Convert.ToDateTime(mskpgto.Text);

TimeSpan result = pgto.Subtract(venc);

txtdias.Text = Convert.ToString(result.Days);
dias =(result.Days);
if (dias > 0)
{
    adicional = (valor * 0.02 * dias);
    txtadicional.Text = Convert.ToString(adicional);
    total = (valor + adicional);
    txttotal.Text = Convert.ToString(total);
}
if (dias < 0)
{
    adicional = (valor * 0.03);
    txtadicional.Text = Convert.ToString(adicional);
    total = (valor - adicional);
    txttotal.Text = Convert.ToString(total);
}
if (dias == 0)
{
    adicional = 0;
    txtadicional.Text = Convert.ToString(adicional);
    total = (valor);
    txttotal.Text = Convert.ToString(total);
}

private void txtvalcompra_Validated(object sender, EventArgs e)
{
}

private void txtvalcompra_KeyPress(object sender, KeyPressEventArgs e)
{
    if (!char.IsNumber(e.KeyChar))
        e.Handled = true;
}

private void mskvenc_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
{
    MessageBox.Show("Digite apenas numeros");
}

private void mskpgto_MaskInputRejected(object sender,
MaskInputRejectedEventArgs e)
{
    MessageBox.Show("Digite apenas numeros");
}

private void mskvenc_TypeValidationCompleted(object sender,
TypeValidationEventArgs e)
{
    if (!e.IsValidInput)
        MessageBox.Show("Data inválida");
}

private void mskpgto_TypeValidationCompleted(object sender,
TypeValidationEventArgs e)
{
```

```
        if (!e.IsValidInput)
            MessageBox.Show("Data inválida");
    }

    private void btnlimpar_Click(object sender, EventArgs e)
    {
        mskpgto.Text = "";
        mskvenc.Text = "";
        txtadicional.Text = "";
        txtdias.Text = "";
        txttotal.Text = "";
        txtvalcompra.Text = "";

    }

}
```



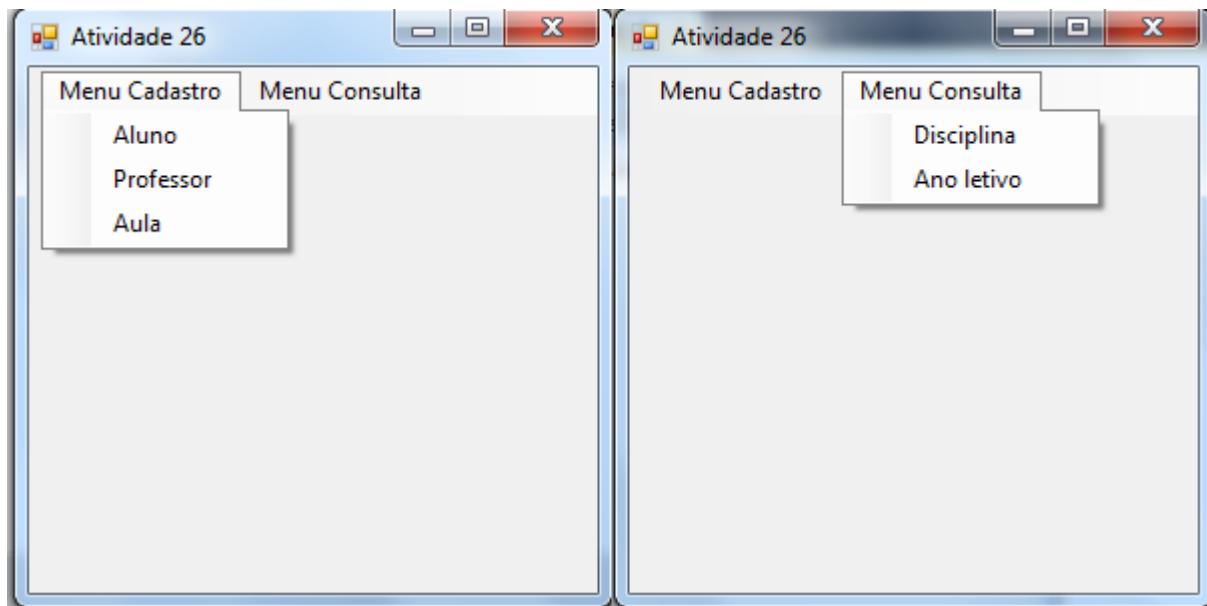
OBJETIVO:	Estruturas de seleção em C#
METODOLOGIA:	Desenvolvimento de projetos
DATA ENTREGA:	DATA: ___/___/___
LISTA_ATV_P.4	

ATIVIDADE 26

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

Um formulário principal com o menu principal conforme layout da imagem abaixo:

Eventos

- Quando o usuário clicar em cada item do menu deverá ser aberto um formulário com a opção correspondente.
- Em cada formulário deverá ter um botão para voltar ao menu inicial;

ATIVIDADE 27

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- 01 Form
- 02 textbox
- 02 maskedtextbox

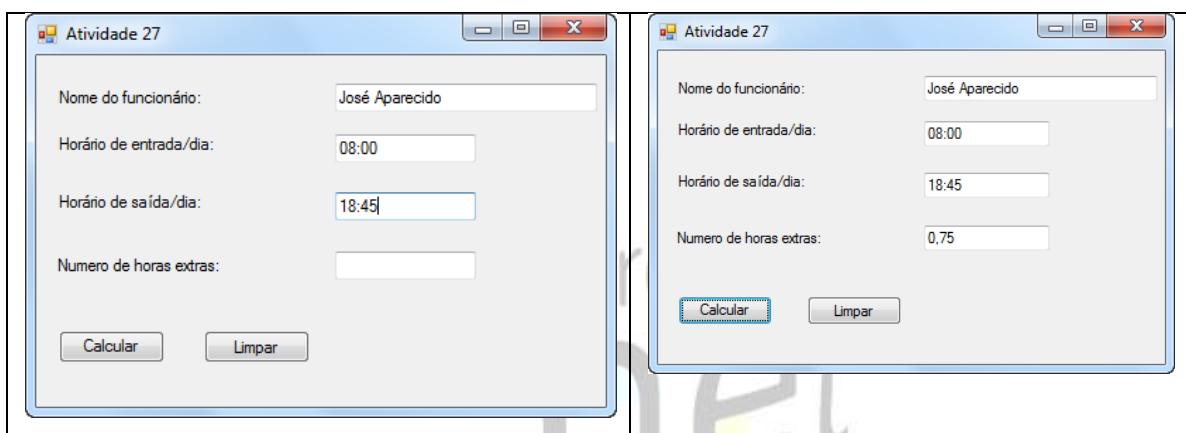
- 04 label
- 02button

Eventos

Construa um aplicativo para o usuário entrar com o nome, e o horário de entrada e saída de um funcionário e calcule o numero de horas extras, considere que a carga diária de trabalho é de 08 horas e que os funcionários possuem 02 horas de almoço.

Obs: Formate os campos de horário para 00:00.

Faça a validação de dados nos campos de horário.



ATIVIDADE 28

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- | | | |
|---------------|------------------|--------------------|
| • 01 Form | • 03 radiobutton | • 01 maskedtextbox |
| • 01 groupbox | • 01 label | |

Eventos

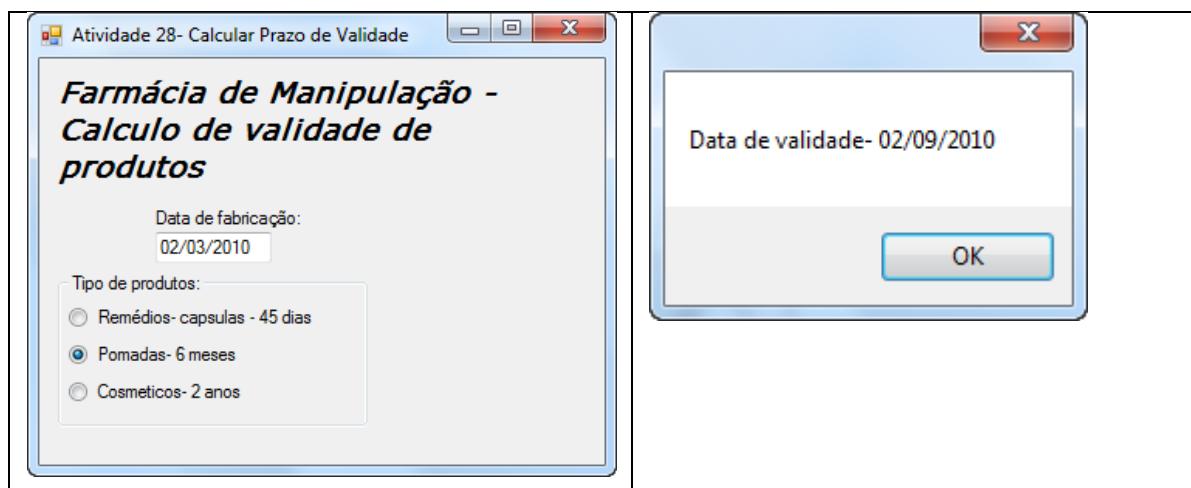
Construa um aplicativo para calcular o prazo de validade de produtos para uma - farmácia de manipulação;

- O usuário deverá entrar com a data de fabricação através de um maskedtextbox produto e selecionar o tipo de produto através de radiobutton's e deverá ser exibido a data de vencimento;

- Considere os seguintes prazos :

Remedios – 25 dias/Pomadas – 06 meses/Cosméticos- 2 anos

Obs: Utilize os métodos AddXXX correspondentes (ex. AddDays, AddYear, AddMinutes).



Microsoft
.net

7

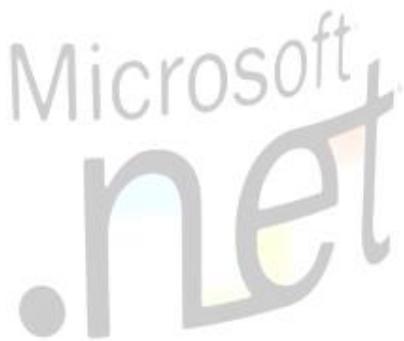
CAPÍTULO

CAPÍTULO 7- Laços de Repetiçao em C#

OBJETIVOS

Laços de repetição em C#

CONTEÚDO



Laço for da linguagem C#

O laço for em C# é usado quando queremos executar uma instrução ou um bloco de instruções um determinado número de vezes. Este laço é composto de três partes:

- **Inicialização:** Laços for são controlados por uma variável de controle. Nesta parte nós definimos o tipo de dados e o valor inicial desta variável.
- **Teste da condição de parada:** Cada iteração do laço acontece mediante uma condição. Esta condição é avaliada e, caso o retorno seja satisfatório, a execução do laço continua.
- **Incremento ou decremento da variável de controle:** Esta parte do laço é executada após cada iteração. É aqui que incrementamos ou decrementamos o valor da variável de controle.

Sintaxe:

```
for(int cont = 0; cont <= 10; cont++)  
{  
    comando 1;  
    comando 2;  
    comando n  
}
```

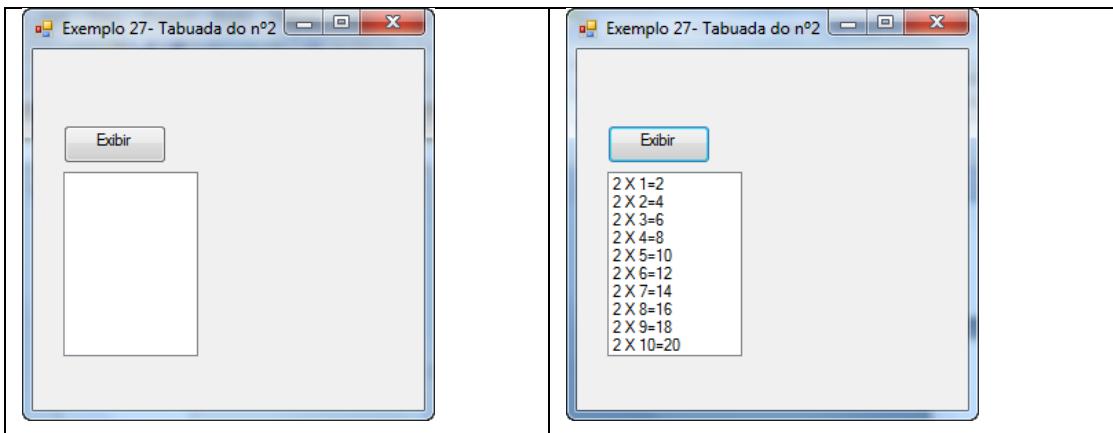
Exemplo 27

Criar uma interface gráfica em C# para o projeto abaixo:

- Um form principal
- Um button (*btnexibir*)
- Um listbox(*lsttabuada*)

Eventos

- Quando o usuário clicar no botão será exibido no listbox a tabuada do numero 2.



```
private void btnexibir_Click(object sender, EventArgs e)
{
    int cont, result;
    for (cont = 1; cont <= 10; cont++)
    {
        lsttabuada.Items.Add("2 X " + cont + "=" + 2 * cont);
    }
}
```

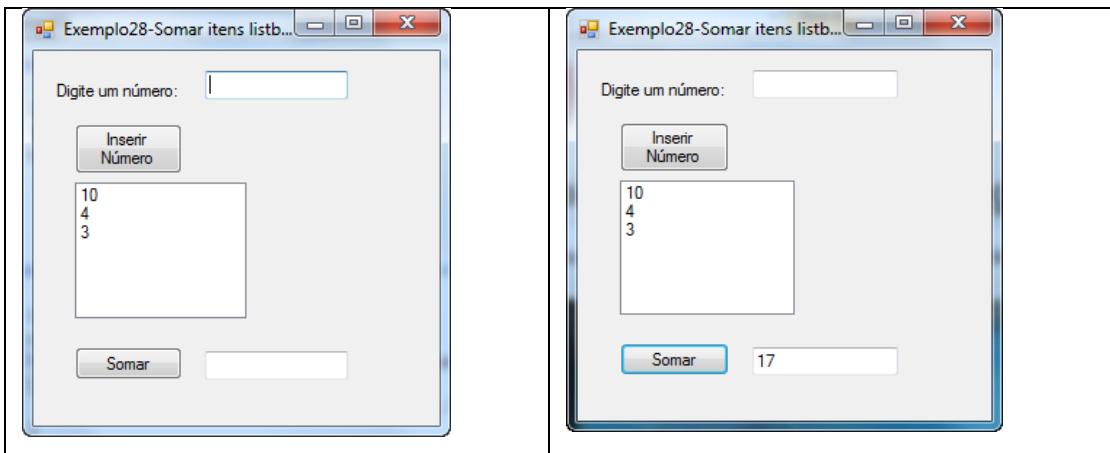
Exemplo 28

Criar uma interface gráfica em C# para o projeto abaixo:

- Um form principal(frmexempl28)
- Um button (btninserir)
- Um button (btnsomar)
- Um textbox(txtnum)
- Um textbox(txttotal)
- Um label
- Um listbox(lstnumeros)
- Uma caixa de texto

Eventos

- O usuário vai digitar um número na textbox(txtnum) e quando clicar no botão (btninserir) o valor vai ser inserido no listbox(lstnumeros) em seguida a caixa de texto vai ser limpa e o foco volta no objeto para o usuário entrar com mais números.
- Quando o usuário clicar no botão (btnsomar) vai ser exibido na textbox(txttotal) a soma dos itens inseridos na textbox.



Programação:

- Para inserir dados do textbox no listbox usamos o método Add do listbox
`lstnumeros.Items.Add(txtnum.Text);`
- Para colocar o foco em um objeto utilizamos o metodo Focus() do objeto textbox;
- Para efetuar a soma dos itens de um listbox, precisamos utilizar um laço de repetição, pois não sabemos quantos números o usuário vai digitar.
- Utilizamos no laço for duas variáveis Soma, Cont. A variável cont é utilizada para contar o numero de vezes que o laço vai ser executado, a variável soma é utilizada para ir somando os valores que é lido cada vez que o laço é executado.
- O método count do listbox efetua a contagem de quantos items existe no listbox.
- Para pegar o valor o conteúdo de um item do listbox utilizamos o método Items

	<pre> private void btninserir_Click(object sender, EventArgs e) { lstnumeros.Items.Add(txtnum.Text); txtnum.Text = ""; txtnum.Focus(); } private void btnsomar_Click(object sender, EventArgs e) { int cont, soma; soma = 0; for (cont = 0; cont < lstnumeros.Items.Count; cont++) { soma = soma + Convert.ToInt32(Convert.ToString(lstnumeros.Items[cont])); } txttotal.Text = Convert.ToString(soma); } </pre>
--	--

Laço While da linguagem C#

O laço while (enquanto) é usado quando queremos repetir uma instrução ou bloco de instruções ENQUANTO uma condição for satisfatória.

Utilizamos este tipo de laço quando não temos conhecimento de quantas vezes o laço vai ser executado.

Sintaxe:

```
int cont;
cont=1

while(cont <= condicao)
{
    comando 1;
    comando 2;
    comando n
    cont++; //incremento
}
```

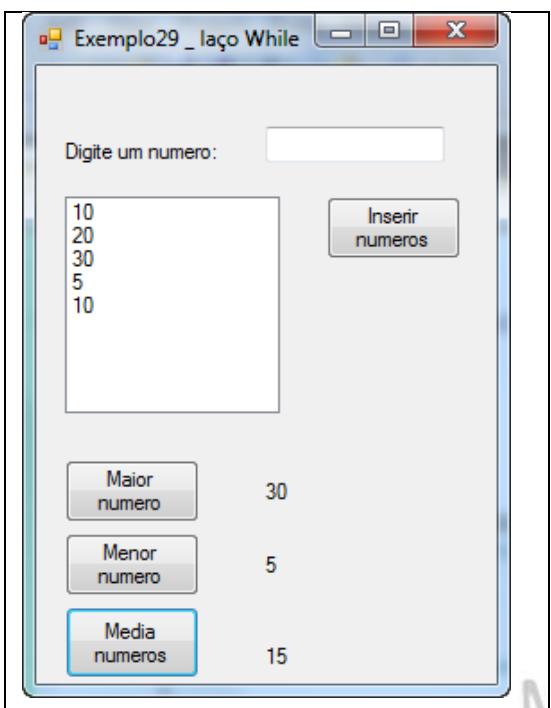
Exemplo 29

Criar uma interface gráfica em C# para o projeto abaixo:

- Um form principal(frmexempl29)
- Quatro button's (btninserir, btnmaior, brnmenor, btnmedia)
- Um textbox(txtnum)
- Quatro label's (lblnum, lblmaiornum, lblmenornum, lblmedianum)
- Um listbox(lstnum)

Eventos

- O usuário vai digitar um número na textbox(txtnum) e quando clicar no botão (btninserir) o valor vai ser inserido no listbox(lstnum) em seguida a caixa de texto vai ser limpa e o foco volta no objeto para o usuário entrar com mais números.
- Quando o usuário clicar no botao (btnmaiornum) vai ser exibido na label(lblmaiornum) o maior número digitado;
- Quando o usuário clicar no botao (btnmenornum) vai ser exibido na label(lblmenornum) o menor número digitado;
- Quando o usuário clicar no botao (btnmedianum) vai ser exibido na label(lblmedianum) a média dos números digitados;



Programação:

- Neste exemplo para exibir o maior número declaramos uma variável (maior) e iniciamos seu valor com 0, e cada vez que o laço While for executado os valores do listbox serão comparados com este valor, caso seja maior que o anterior então a variável irá receber este valor.
- Para exibir o menor valor usamos a mesma lógica do item anterior, armazenando na variável menor um número grande e depois ir comprando seu valor com os demais.
- Para exibir a media dos números, efetuamos a soma dos números e depois efetuamos a divisão pela quantidade de números (cont).

	<pre>private void btninserir_Click(object sender, EventArgs e) { lstnum.Items.Add(txtnum.Text); txtnum.Text= ""; txtnum.Focus(); } private void btnmaior_Click(object sender, EventArgs e) { int cont, maior; maior = 0; cont = 1; while (cont < 5) { if (cont == 1) maior = lstnum.Items[cont].ToString(); else if (Convert.ToInt32(lstnum.Items[cont].ToString()) > Convert.ToInt32(maior)) maior = lstnum.Items[cont].ToString(); cont++; } txtmaior.Text = maior; }</pre>
--	---

```
        if (Convert.ToInt32(lstnum.Items[cont]) > maior)
            maior = Convert.ToInt32(lstnum.Items[cont]);

            cont = cont + 1;
    }
    lblmaiornum.Text=(Convert.ToString(maior));
}

private void btnmenor_Click(object sender, EventArgs e)
{
    int cont, menor;
    menor = 10000;
    cont = 1;
    while (cont < lstnum.Items.Count)
    {
        if (Convert.ToInt32(lstnum.Items[cont]) < menor)
            menor = Convert.ToInt32(lstnum.Items[cont]);

            cont = cont + 1;
    }
    lblmenornum.Text = (Convert.ToString(menor));
}

private void btnmedia_Click(object sender, EventArgs e)
{
    int cont;
    double media;
    media = 0;
    cont = 0;
    while (cont < lstnum.Items.Count)
    {
        media = (media + Convert.ToInt32(lstnum.Items[cont]));
        cont = cont + 1;
    }
    lblmedianum.Text = (Convert.ToString(media / cont));
}
```

OBJETIVO:	Estruturas de seleção em C#
METODOLOGIA:	Desenvolvimento de projetos
DATA ENTREGA:	DATA: ___ / ___ / ___
LISTA_ATV_P.4	

ATIVIDADE 29

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- Um formulário
- Um button
- Uma caixa de texto

Eventos

- O usuário vai digitar com um numero inteiro através da caixa de texto;
- Quando o usuário clicar no botão será exibido em uma caixa de mensagem o fatorial do numero digitado.

Considere que para calcular o fatorial do numero :

$$N! = 1 * 2 * 3 \dots * N$$

ATIVIDADE 30

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

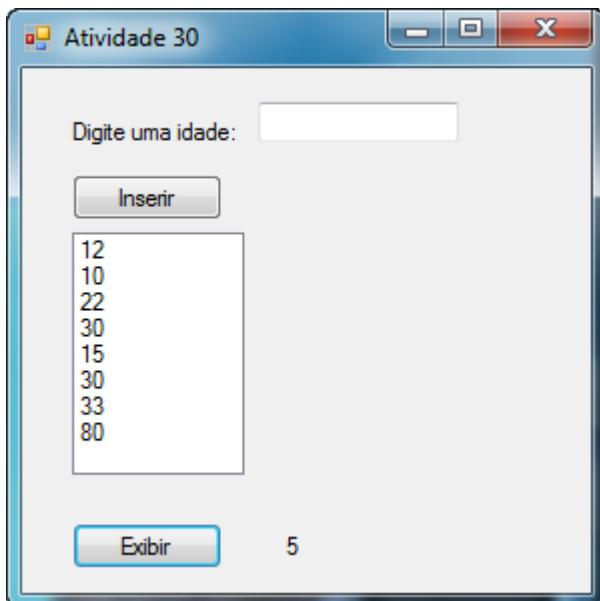
- Um formulário
- Dois button
- Uma textbox
- Uma listbox
- Duas label's

Eventos

- O usuário vai digitar a idade de 8 pessoas através de um textbox;
- Quando o usuário clicar no botao (btninserir) a idade será inserida no listbox;

- Quando o usuário clicar no botão (btnexibir) será exibido em uma label quantas pessoas são maiores de 18 anos.

Dica: Dentro do laço será necessário usar a estrutura If para verificar a idade.



ATIVIDADE 31

Faça uma solução no Visual Studio(C#) de acordo com as seguintes instruções:

Objetos

- Um formulário
- Três button's
- Uma textbox
- Uma listbox
- Duas label's

Eventos

- O usuário vai digitar 6 números inteiros através de um textbox;
- Quando o usuário clicar no botão (btninserir) os números serão inseridos no listbox;
- Quando o usuário clicar no botão (btncpar) será exibido em uma label(lblpar) quantos números são par;
- Quando o usuário clicar no botão (btndimpar) será exibido em uma label(lblimpar) quantos números são ímpar;

Dica: Use o operador % para analisar o resto da divisão para saber se o número é par/ímpar.