

**ÍNDICE**

1. Visual Studio.....	1
1.1 C#.NET .....	1
1.2 A Interface do Visual Studio .....	1
1.2.1 Tela inicial para criação de uma solução ou projeto.....	1
1.2.2 Apresentação da Interface do C#.NET .....	2
1.3 Principais Componentes do C#.Net.....	8
1.3.1 Componente Formulário.....	8
1.3.2 Componentes – Seleção de Opções.....	9
1.3.3 Componente de Informação .....	10
1.3.4 FOCO .....	11
1.3.5 PROPRIEDADES TABINDEX E TABSTOP .....	11
1.4 Orientação a Eventos.....	11
1.4.1 EVENTO CLICK.....	12
1.4.2 EVENTOS ENTER E LEAVE .....	12
1.4.3 EVENTOS VALIDATED E VALIDATING.....	12
1.4.4 EVENTO KEY PRESS .....	12
1.4.5 EVENTO LOAD DO FORM .....	12
1.5 MESSAGEBOX.....	12
1.6 COMENTÁRIOS .....	13
1.7 OPERADOR: + .....	14
1.8 Estrutura Básica de um aplicativo .NET .....	14
2. EXERCÍCIO .....	15
3. REFERÊNCIAS .....	16

**1. Visual Studio**

O Visual Studio 2022 é um Ambiente de Desenvolvimento Integrado (Integrated Development Environment - IDE), com recursos novos e aprimorados que simplificam e aumentam a produtividade de desenvolvimento, desde o design (projeto) até a implantação.

**1.1 C#.NET**

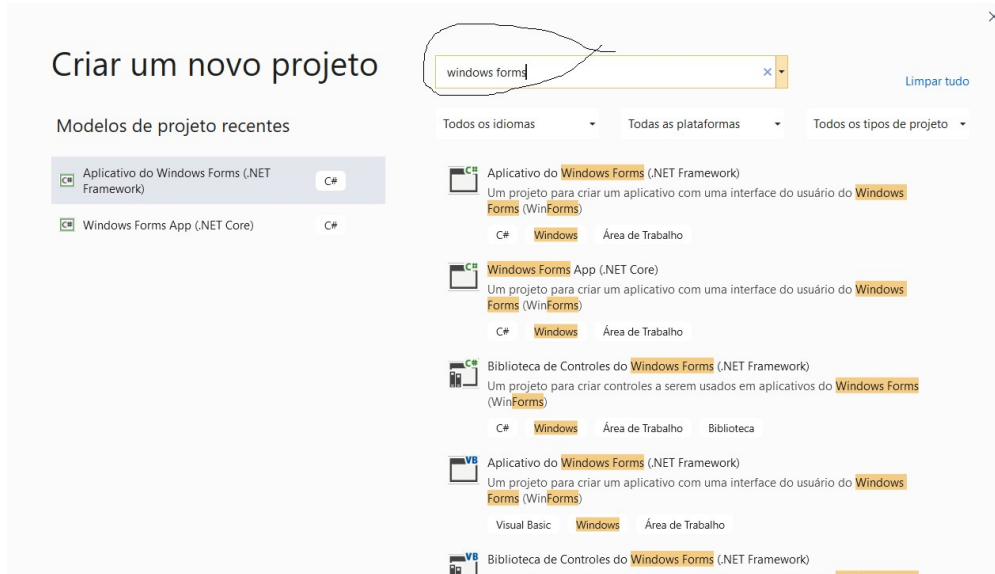
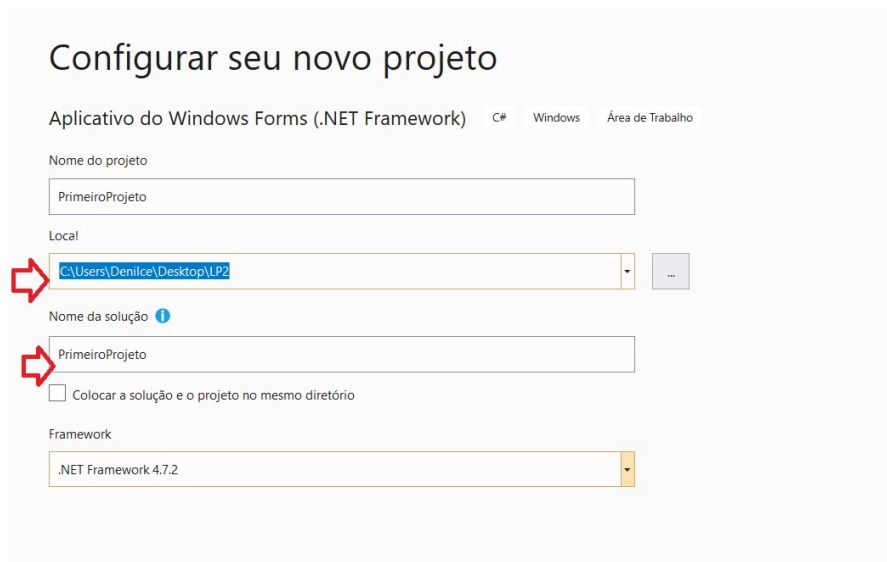
O C# é uma das linguagens que se pode programar no Visual Studio. Trata-se de uma hierarquia de classes que estão incluídas no .NET Framework sendo uma linguagem orientada a objetos.

**1.2 A Interface do Visual Studio**

O ambiente do Visual Studio também é chamado de IDE (Integrated Development Environment ou Ambiente Integrado de Desenvolvimento) do Visual Studio é parecido com a tela seguinte:

**1.2.1 Tela inicial para criação de uma solução ou projeto**

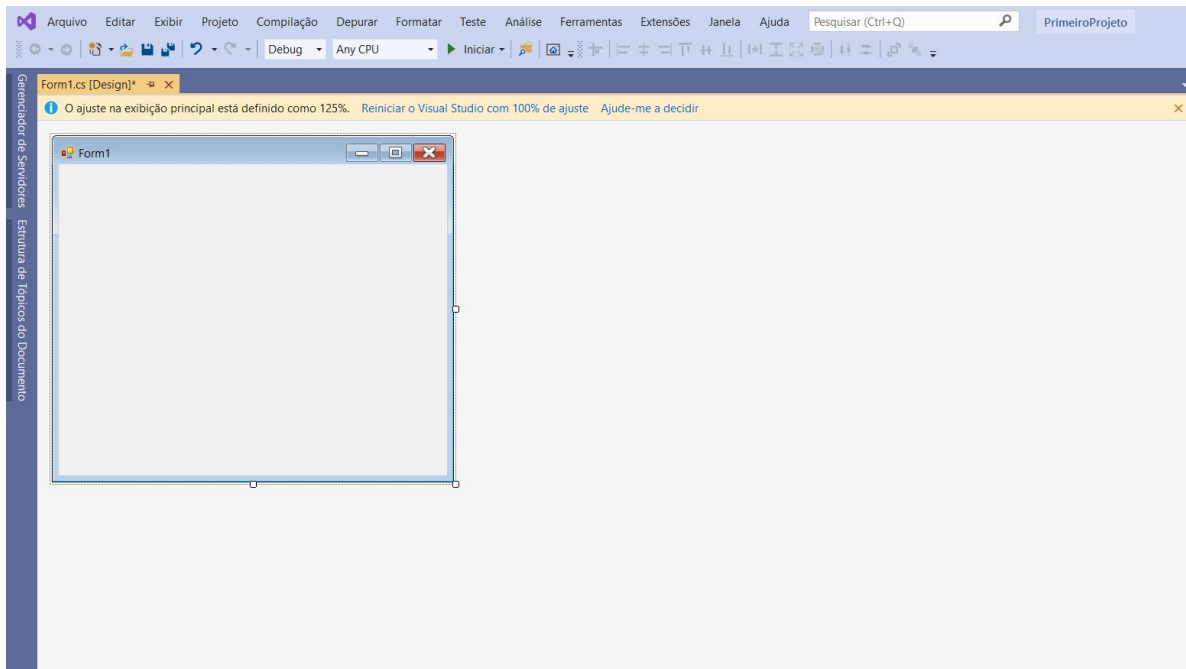
A figura abaixo mostra a tela para criação de uma solução ou projeto. Atentar para escolher o tipo de projeto Windows Forms (.NET Framework) e a linguagem (Idioma).

**Tela para Criação do Projeto (1)****Tela para Criação do Projeto (2)**

Um *projeto* está contido dentro de uma *solução*. Ela é apenas um contêiner de **um ou mais projetos** relacionados, juntamente com informações de build, configurações de janela do Visual Studio e arquivos diversos que não estão associados a nenhum projeto específico. Uma solução é descrita por um arquivo de texto (extensão *.sln*) com seu próprio formato exclusivo; não se destina à edição manual. Um projeto contém todos os arquivos que são compilados em um executável, biblioteca ou site.

### 1.2.2 Apresentação da Interface do C#.NET

A figura abaixo mostra a janela principal (padrão) C#.NET.



***Interface Padrão do C#.NET***

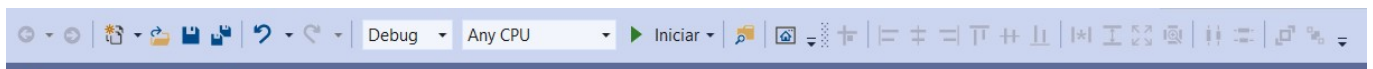
Os principais elementos da Interface do C#.NET com o desenvolvedor são:

- 1.2.2.1 - A Barra de Ferramentas(Toolbar)
- 1.2.2.2 - A Paleta de Componentes(Toolbox)
- 1.2.2.3 - Gerenciador de Soluções (Solution Explorer)
- 1.2.2.4 – Propriedades (Properties)
- 1.2.2.5 - Editor Gráfico/Editor de Código

Quando algum desses elementos não estiver sendo mostrado procurar o item desejado no menu Exibir, ou utilizar algum atalho, por exemplo para o item 1.2.2.2 o atalho é CTRL + ALT + X.

### 1.2.2.1 – Barra de Menu e Barra de Ferramentas (Toolbars)

A Toolbars permite a utilização de controles como *rodar a aplicação*, *executar passo a passo*, salvar, abrir, entre outros. Esses comandos (e outros) estão incluídos também na barra de menu.



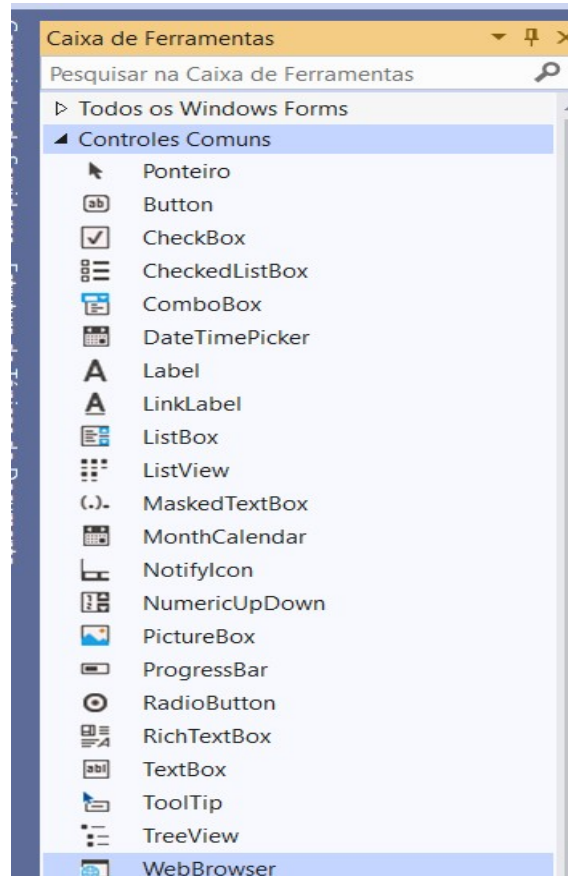
#### **Barra de Ferramentas(Toolbar)**

Também é possível adicionar ou remover itens na Barra de Ferramentas, clicar com o botão direito do mouse sobre qualquer posição da barra de ferramentas, irá surgir a janela abaixo, selecionar um grupo:



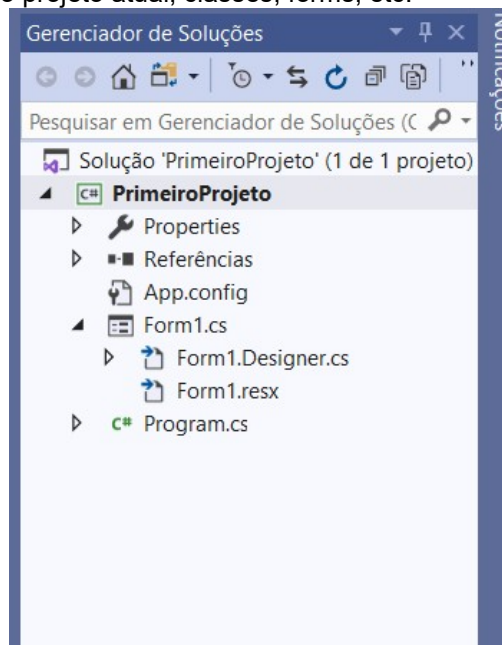
#### **Barra de Ferramentas (adição ou exclusão de itens)**

**1.2.2.2 – Paleta de Componentes (Toolbox):** exibe uma lista de componentes (controles) que poderão ser utilizados nas aplicações(programas). A paleta é dividida categorias. A categoria All... mostra todos em ordem alfabética. Ao passar o mouse sobre o componente aparecerá uma breve descrição sobre o componente.



**Paleta de componentes (Toolbox)**

**1.2.2.3 – Gerenciador de Soluções (Solution Explorer):** exibe uma estrutura de árvore que representam as pastas e objetos correspondentes ao projeto atual, classes, forms, etc.



**Gerenciador de Soluções (Solution Explorer)**

#### 1.2.2.4 – Propriedade (Properties)

Exibe uma lista de atributos/eventos que estão associados ao componente selecionado.

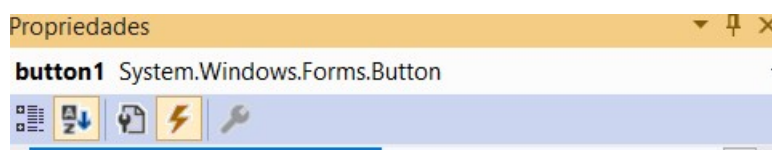
**Form1** System.Windows.Forms.Form

FormBorderStyle	Sizable
HelpButton	False
Icon	(Ícone)
ImeMode	NoControl
IsMdiContainer	False
KeyPreview	False
Language	(Padrão)
Localizable	False
Location	0; 0
Locked	False
MainMenuStrip	(nenhum)
MaximizeBox	True
MaximumSize	0; 0
MinimizeBox	True
MinimumSize	0; 0
Opacity	100%
Padding	0; 0; 0; 0
RightToLeft	No
RightToLeftLayout	False
ShowIcon	True
ShowInTaskbar	True
Size	<b>818; 497</b>
SizeGripStyle	Auto
StartPosition	WindowsDefaultLocation
Tag	
Text	<b>Form1</b>
TopMost	False
TransparencyKey	
UseWaitCursor	False
WindowState	Normal

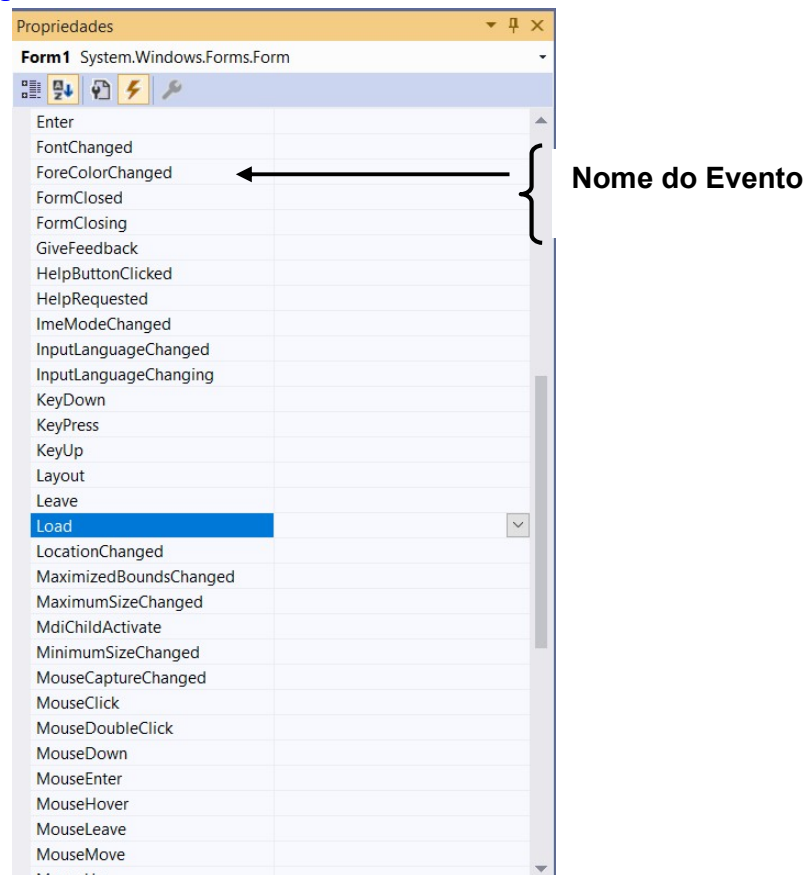
**Locked**  
A propriedade Locked determina se o controle pode ser movido ou

**Propriedades (Properties)**


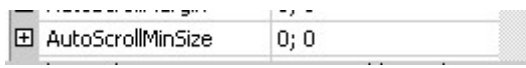


Se clicar no ícone “raio” será trocado para a lista de eventos com relação ao componente selecionado.



**Chamando a tela de Eventos**

**Propriedades (Eventos)****Formas de edição de atributos no Properties**

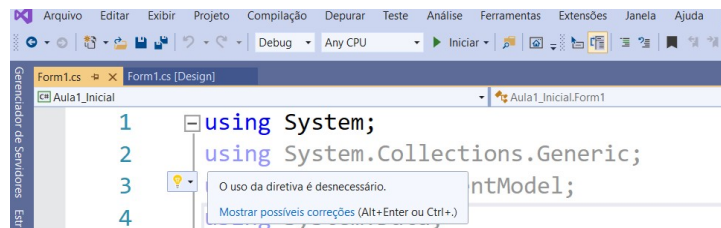
A forma de edição no Properties varia de acordo com o tipo de atributos, conforme exemplo abaixo.

Atributo	Forma de edição no Properties
	O atributo pode ser preenchido livremente com qualquer string de caracteres.
	O atributo traz um sinal de + à esquerda de seu nome, significando que existem sub-atributos, que podem ser vistos após um duplo clique de mouse sobre o nome da propriedade. Seus valores podem ser alterados.
	Botão com reticências (...) na coluna de valor. Os atributos desse tipo são preenchidos por editores especiais que, tipicamente, apresentam-se como caixas de diálogos, que aparecem quando se clica sobre as reticências.
	Clicando na seta para baixo, à direita na coluna de valor, irá aparecer uma lista de possíveis valores do atributo. Um duplo clique em um valor da lista, irá transferir esse valor para coluna de valor do atributo.

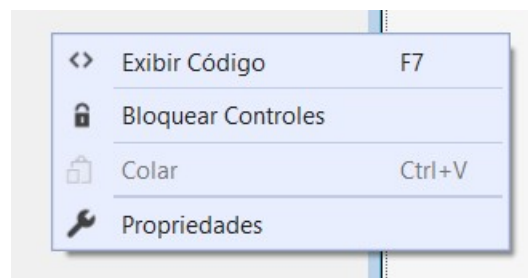
**Formas de Edição das Propriedades****1.2.2.5 - Editor Gráfico / Editor de Código**

**Prof.<sup>a</sup>: Denilce Veloso – Aula 1**

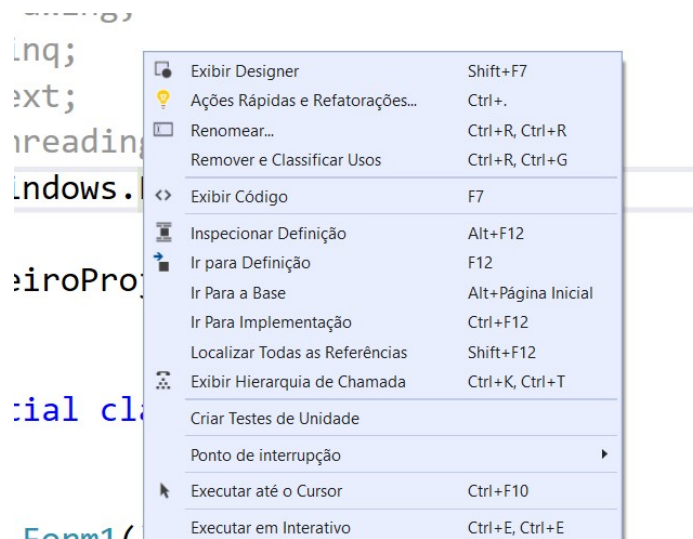
Os códigos e o layout (design) do formulário são divididos em guias.

**Guias Editor de Código e Editor Gráfico**

Para visualizar os códigos relacionados ao formulário, clicar na guia correspondente ou clicar com botão direito do mouse e selecione Exibir Código ou tecla F7.

**Menu de Contexto do Editor Gráfico**

Para visualizar o formulário relacionado ao código clicar na guia correspondente ou clicar com botão direito do mouse e selecione Exibir Designer ou teclas SHIFT + F7.

**Menu de Contexto do Editor de Código****1.3 Principais Componentes do C#.Net**

Os principais controles (componentes) da Caixa de Ferramentas (ToolBox) estão listados a seguir, lembrando que todos os componentes possuem a propriedade Name = nome do componente.

**1.3.1 Componente Formulário**

*Form*

Name - Nome do Formulário



### 1.3.2 Componentes – Seleção de Opções

Existem dois controles padrão do Windows que permitem ao usuário escolher diferentes opções, assim como controles para agrupar conjuntos de opções.

CheckBox e RadioButton, são controles de opções a serem selecionadas. O CheckBox permite múltiplas opções, imagine um texto que pode estar formatado em itálico, negrito ou sublinhado, já o RadioButton permite somente uma opção.

#### CheckBox

Essa caixa de seleção corresponde a uma opção que pode ser selecionada independentemente do status de outras caixas de verificação (observe que existe também o componente CheckListBox).

Configurar a propriedade – Three State – da caixa de seleção permite a apresentação de três estados – *Selecionado, Não Selecionado e Acinzentado* – que se alternam quando o usuário dá um clique na caixa de seleção. Se precisar somente de 2 estados deixar essa propriedade como falso. A Propriedade checked mostra se está checado ou não. Se ativar o 3º estado deverá testar a opção CheckState = `CheckState.Checked` ou `CheckState.Indeterminate` ou `CheckState.Unchecked`.

Text – texto que aparece no checkbox

#### RadioButton

Esse tipo de controle é o *botão de rádio* que corresponde a uma seleção exclusiva. Dois desses componentes dentro do mesmo contêiner não podem ser selecionados simultaneamente. É recomendado que um deles sempre esteja selecionado. (propriedade checked=true)

A propriedade – *Checked* – do componente *RadioButton* define se ele está selecionado (True) ou não (False).

Text – texto que aparece no Radiobutton

### Agrupando Componentes – Seleção de Opções

Para agrupar vários componentes de Seleção de Opções podem ser utilizados os componentes da **ToolBox \ Container** – por exemplo pode ser usado um *GroupBox* para agrupar os componentes *GroupBox* ou *RadioGroup*, que os mantêm juntos tanto funcional como visualmente.

#### GroupBox

Para construir uma Caixa de Grupo – *GroupBox* – com outros componentes por exemplo *RadioButton* ou *CheckBox*, basta colocar o componente *GroupBox* no *form* e depois incluir os componentes de seleção nele.

Os componentes de seleção podem ser manipulados individualmente, porém é mais fácil navegar no *Array de controles pertencentes à Caixa de Grupo – GroupBox* – através das propriedades:

*Controls* - É o Array do componente *GroupBox* – contém todos os controles (por exemplo: componentes de seleção) inseridos no *GroupBox*.

Text – texto que aparece no GroupBox

*Combobox* - Um combo, é uma caixa de seleção de opções. O objeto do combo, ou melhor, dropdown list, trabalha com objetos do tipo LISTITEM (se você for carregá-lo em tempo de execução com comandos manuais ou com dados que vem de uma tabela) que contém um conjunto de propriedades para o item do combo.

*ListBox* - O listbox, apesar de ser outro controle diferente do combo, tem as mesmas características, e renderiza um <SELECT>, da mesma forma.

Combobox e ListBox tem a propriedade Items (collection) - onde são incluídos os dados.

Exemplo: Para adicionar itens em um Combobox em tempo de execução:

```
comboBox1.Items.Add("Novo Item")
comboBox1.Items.Add("Outro Item")
```

Para Excluir o item do Combox  
comboBox1.Items.Remove("Novo Item")

Para Limpar o ComboBox (zerar)

Prof.<sup>a</sup>: Denilce Veloso – Aula 1

```
comboBox1.Items.Clear()
```

Para Incluir vários Itens de uma vez, criar uma matriz:

```
String[] Lista = new String[3];  
Lista[0] = "Primeiríssimo";  
Lista[1] = "Primeiro";  
Lista[2] = "Segundo";  
comboBox1.Items.AddRange(Lista);
```

Para verificar se um item está selecionado no combobox ou no listbox é necessário verificar a propriedade **SELECTEDITEM**.

É possível selecionar mais de um item no LISTBOX (desde que seja alterada a propriedade **SELECTION MODE** por exemplo para **MULTISIMPLE**) - nesse caso verificar a propriedade **SELECTEDITEMS.Count** para verificar o total de itens selecionados e **SELECTEDITEMS(n)** para verificar cada item selecionado.

**Sorted** - ordena os itens em ordem alfabética.

**DropDownStyle** – escolher **DropDownList** para não permitir digitação de dados.

**SelectedIndex** – indica qual é a posição do item selecionado, o primeiro item é 0. Se não for selecionado nenhum, essa propriedade volta -1.

**CheckedListBox** - é uma lista de checkbox onde podem ser escolhidas quantas opções forem desejadas.

Também tem a propriedade **items**.

**checkedListBox1.CheckedItems.Count** - verifica todos total de itens checados

**checkedListBox1.SelectedItems.Count** - verifica total de itens selecionados

**checkedListBox1.SelectedItems(n)** - verifica item específico selecionado

**checkedListBox1.CheckedItems(n)** - verifica item específico checado

### 1.3.3 Componente de Informação

**Label** - é o campo que usamos para inserir textos estáticos. O Label é preenchido ao desenhar o form, ou em tempo de execução.

#### Componentes de Entrada de Dados

**TextBox** - é um campo de texto, qual podemos exibir informações, e ou entrar com dados.

##### **MaskedTextBox**

É semelhante ao textbox porém é utilizado quando desejamos entrar com dados utilizando uma máscara, tipo valores, ceps, etc.

##### Propriedades importantes:

**Mask** – diz respeito à máscara de entrada/saída de dados.

9 – entrada opcional número ou espaço, + e – não permitido

# - entra opcional qualquer caractere inclusive espaço

0 – entrada obrigatório número, + e – não permitidos

L – A a Z – entrada requerida

? – A a Z – entrada opcional

A – letra ou dígito entrada requerida

a – letra ou dígito entrada opcional

& - qualquer caractere ou espaço – entrada requerida

C – qualquer caractere ou espaço – entrada opcional

**Prof.<sup>a</sup>: Denilce Veloso – Aula 1**

,.;-/-/ marcador de posição decimal e separadores de milhares, de data e hora (depende da configuração do Windows)

< faz com que os caracteres sejam convertidos para minúsculos

> faz com que os caracteres sejam convertidos para maiúsculos

\ faz com que o caracter seja exibido literalmente, Exemplo: \A vai ser considerado com A

Exemplo1: (###)###-####

(\_\_\_\_)\_\_\_\_-\_\_\_\_\_  
(062)621-3862

**Exemplo2: \$#,##0.000**

**Resultado:**



PromptChar

Sinal (prompt) que será mostrado na hora da digitação por exemplo \_ (underscore ou underline)

TextMaskFormat

Várias opções para retornar os dados incluindo os prompts (sinais que são mostrados na hora da digitação e as literais (por exemplo virgula, traços e etc).

Branco no controle Masked Edit Box

O controle MS Masked Edit apenas aceita entrada de dados dentro da máscara formatada (mask). Isto impede o programador de limpar a text do controle diretamente (masked1.text = ""), pois, o caracter espaço (ou nulo) pode não se encaixar no formato da máscara. Por exemplo, algumas possuem o formato # (aceitam somente números). Logo, o "" não seria aceito. Este problema é resolvido por este código:

```
vTemp = masked1.mask;
masked1.mask = "";
masked1.text = "";
masked.mask = vTemp;
```

Removendo a máscara é possível limpar o texto. Depois, basta devolver a máscara original ao controle. Uso isto no evento Data1\_Validation\_Error quando adiciono um novo registro.

### 1.3.4 FOCO

É o termo usado para descrever o componente (objeto) que está em destaque no momento. O objeto está sendo utilizado pelo usuário. Um objeto pode perder ou ganhar o foco, quando estamos o manipulando ele ganha o foco e se apertarmos a tecla TAB ou clicarmos com o mouse em outro objeto ele perde o foco.

### 1.3.5 PROPRIEDADES TABINDEX E TABSTOP

A propriedade TABINDEX que aparece na maioria dos componentes possui uma sequência numérica que representa a ordem de tabulação dentro do form. O primeiro componente a ser inserido possui sempre o número 0, e assim que outros são inseridos, este número vai sendo incrementado em mais um.

Essa sequência é utilizada quando teclamos a tecla TAB e vamos passando de componente para componente.

A propriedade TABSTOP (pode ser true ou false) quando está false possui a finalidade de ignorar o controle na tabulação, ou seja, o componente não receberá o foco via teclado somente se estiver sendo utilizado o mouse.

## 1.4 Orientação a Eventos

Responder a eventos (por exemplo um Click do Mouse) na construção de uma interface de usuário usando Windows Forms ou do Windows é muito importante, por isso a programação para Windows é comumente conhecida como programação orientada a eventos. **Chamamos de evento, tudo que ocorre durante o tempo de execução de um programa, ou uma tela.** Se a tela contém um botão, e o usuário clica o botão, ocorreu um

**Prof.<sup>a</sup>: Denilce Veloso – Aula 1**

evento, o clique do botão. Ao carregar uma página, ocorrem eventos. Cada componente tem um evento padrão cujo código é mostrado sempre que é dado um duplo clique no objeto (componente). Por exemplo no Button é click, no Form é load e assim por diante. Porém, pode se escolher o evento, selecionando as propriedades em um componente e clicando no “raio” (ver item 1.2.2.4).

### 1.4.1 EVENTO CLICK

É o evento que ocorre quando se clica com o botão esquerdo do mouse em um objeto (controle) selecionado.

### 1.4.2 EVENTOS ENTER E LEAVE

ENTER – é o evento que ocorre quando um objeto (componente) recebe o foco

LEAVE – é o evento que ocorre quando um objeto (componente) perde o foco

### 1.4.3 EVENTOS VALIDATED E VALIDATING

VALIDATED - Ocorre quando a validação do controle é concluída, saiu do componente. Testa se está correto e se não estiver pode usar o método focus para voltar ao componente.

VALIDATING - Ocorre quando o controle está sendo validado, ainda não saiu do componente. Testa se está correto e se não estiver pode usar e.cancel=true para cancelar o evento.

### 1.4.4 EVENTO KEY PRESS

Acontece sempre que uma tecla é acionada. Exemplo: Por exemplo deseja-se trocar uma tecla por outra, no caso TAB por ENTER.

```
private void comboBox1_KeyPress(object sender,
System.Windows.Forms.KeyPressEventArgs e)
{
    if (e.KeyChar == (Char)13) {
        SendKeys.Send("{TAB}");
        e.Handled = true; //Para remover aquele som...
    }
}
```

**Observação:**

KeyDown: acontece quando a pessoa pressiona uma tecla (quando o teclado detecta um dedo em uma tecla pela primeira vez, isso acontece quando a tecla é pressionada).

KeyPress: acontece quando uma tecla é pressionada e depois solta.

KeyUp: acontece quando a chave é liberada.

A diferença entre keydown() e keypress() está que o segundo não é capturado quando as teclas Ctrl, Alt ou Shift são pressionadas.

### 1.4.5 EVENTO LOAD DO FORM

É o evento que ocorre no carregamento do formulário.

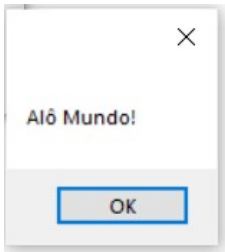
## 1.5 MESSAGEBOX

**Prof.ª: Denilce Veloso – Aula 1**

O MessageBox é o primeiro comando que aprendemos para impressão na tela. Exemplo de uma apresentação padrão:

```
MessageBox.Show("Alô Mundo!");
```

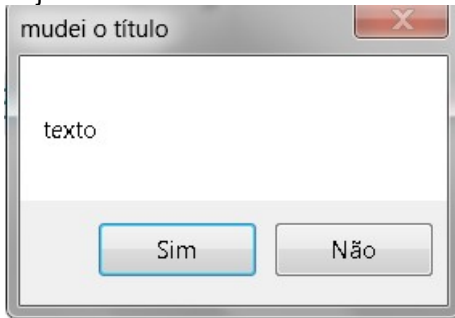
Se digitar mbox e duas vezes a tecla TAB ele já escreve o comando.



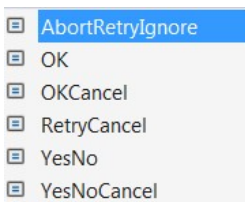
E se desejar mudar o texto na barra de título, é possível basta colocar mais uma vírgula, assim:

```
MessageBox.Show("texto", "mudei o titulo", MessageBoxButtons.YesNo, MessageBoxIcon.Question);
```

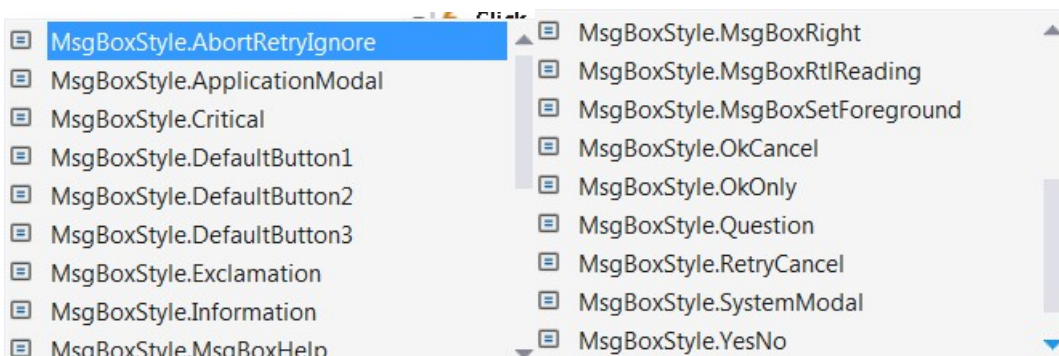
Veja o resultado abaixo:



Os botões podem ser:



Os estilos podem ser:



## 1.6 COMENTÁRIOS

Os comentários são partes de um programa que são ignorados pelo compilador do C#, o que significa pode se escrever o que quiser neles. O que eles deveriam fazer é ajudar o desenvolvedor entender o que cada parte do código está fazendo.

Aqui o comentário é representado pela barra dupla // ou por barra asterisco, que também deve ser fechada, /\* <texto em comentário> \*/.

## 1.7 OPERADOR: +

Quando se usa variáveis no C#, deve-se saber qual tipo de dados deseja armazenar nelas. Será falado mais adiante sobre os tipos de dados e conversões entre eles, todavia quando for necessário imprimir um texto e conectar textos ou números poderão ocorrer várias situações, observe:

```
int X;
```

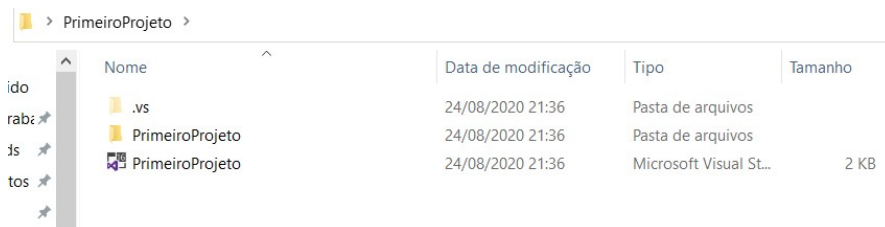
```
MessageBox.Show(Convert.ToString(X)); -> precisa converter para texto
```

```
MessageBox.Show("O número " + X) -> como vai concatenar o número com outro texto e usar operador +, o próprio operador converte o número para string, não precisa de conversão.
```

## 1.8 Estrutura Básica de um aplicativo .NET

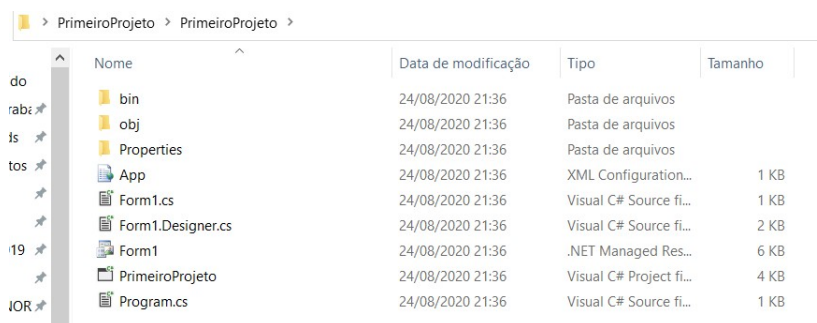
De uma maneira bem geral, para projetos simples utilizando o C#.NET temos que manter os seguintes arquivos:

- .sln (arquivo da solução – conjunto dos projetos)
- a subpasta com o seu conteúdo.



Nome	Data de modificação	Tipo	Tamanho
.vs	24/08/2020 21:36	Pasta de arquivos	
PrimeiroProjeto	24/08/2020 21:36	Pasta de arquivos	
PrimeiroProjeto	24/08/2020 21:36	Microsoft Visual St...	2 KB

### Conteúdo da Pasta Principal do Projeto



Nome	Data de modificação	Tipo	Tamanho
bin	24/08/2020 21:36	Pasta de arquivos	
obj	24/08/2020 21:36	Pasta de arquivos	
Properties	24/08/2020 21:36	Pasta de arquivos	
App	24/08/2020 21:36	XML Configuration...	1 KB
Form1.cs	24/08/2020 21:36	Visual C# Source fi...	1 KB
Form1.Designer.cs	24/08/2020 21:36	Visual C# Source fi...	2 KB
Form1	24/08/2020 21:36	.NET Managed Res...	6 KB
PrimeiroProjeto	24/08/2020 21:36	Visual C# Project fi...	4 KB
Program.cs	24/08/2020 21:36	Visual C# Source fi...	1 KB

### Conteúdo da Sub-Pasta do Projeto

App.xml – informações sobre o framework, versão, etc.

xxxxxx.csproj: arquivo XML que descreve aspectos essenciais para a compilação do projeto chamado 'xxxxxx'. A ausência desse arquivo faz com que o Visual Studio não consiga abrir o projeto.

fffff.cs: é o arquivo texto que contém o código C# referente ao formulário chamado 'fffff'. A lógica escrita para manipular os eventos do formulário, como por exemplo o clique de mouse.

fffff.Designer.cs: arquivo texto que descreve, em notação própria para o Windows Form Designer, a estrutura do formulário chamado 'fffff'. Apesar de poder ser aberto por meio de editores de texto comuns, é importante que este arquivo não seja modificado fora do Windows Form Designer.

**Prof.<sup>a</sup>: Denilce Veloso – Aula 1**

fffff.resx: arquivo XML que descreve recursos requeridos pelo formulário chamado 'fffff'. É útil para definir a utilização de imagens, idiomas e aspectos do *parsing* XML requerido. Pode ser editado em um editor de textos simples, embora isso possa ser perigoso.

Program.cs – arquivo que contém o programa principal (o main) do C#, ele que chama o primeiro formulário. Quando for necessário mudar o primeiro formulário a ser chamado, alterar nesse arquivo.

Além desses arquivos fonte mais básicos, ao salvar o projeto é criada uma subpasta chamada Properties que contém diversos outros fontes essenciais. A ausência dessa pasta ou de seu conteúdo impede a compilação do projeto.

**Atenção:**

**Não é recomendado que esses arquivos sejam editados fora da IDE do Visual Studio.**

**As subpastas .obj e .bin podem ser apagadas juntamente com o seu conteúdo, pois são geradas durante a etapa de compilação do programa.**

\* Toda aplicação .NET deve ser constituída de um ou mais assembly.

\* O programa Microsoft Intermediate Language Disassembler (ildasm.exe) permite visualizar toda a estrutura de um assembly incluindo os seus metadados.. Esse aplicativo pode ser encontrado em *C:\Program Files (x86)\Microsoft SDKs\Windows\v10.0A\bin\NETFX 4.8 Tools*

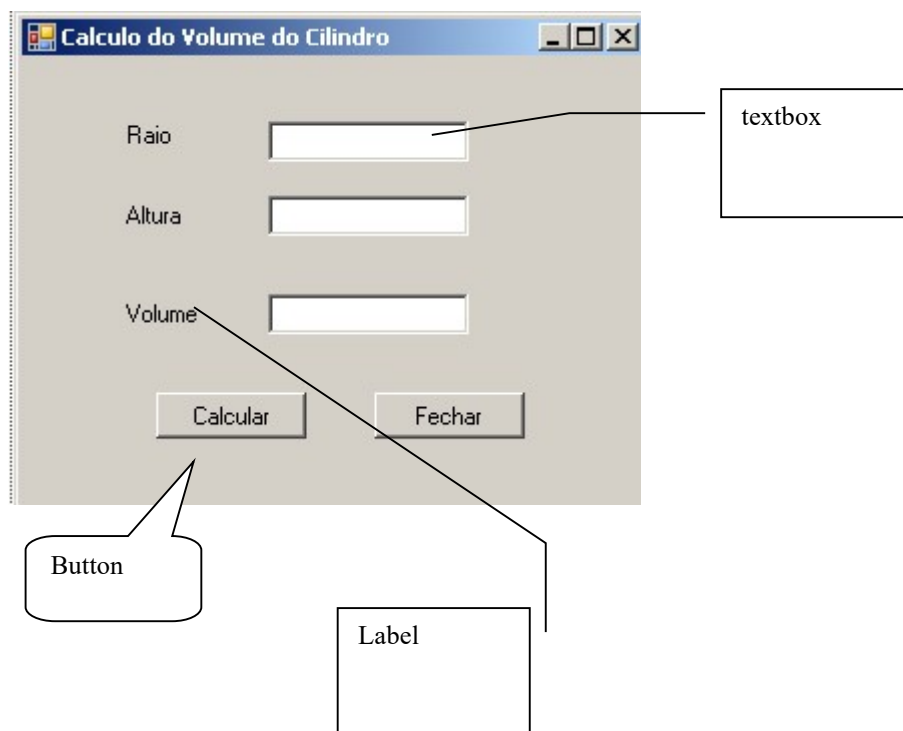
**Atenção,** após a compilação do programa, o arquivo executável pode ser encontrado na pasta .\bin\Debug.

**2. EXERCÍCIO**

- 1) Construir um programa (PVolume) em C# para calcular o Volume de um Cilindro. Solicitar raio e altura (testar se são números).

$$\text{Volume} = \pi \times r^2 \times \text{altura}$$

Lembrete:  $V = \pi \times R \times R \times H$



### 3. REFERÊNCIAS

- CSharp. Disponível em: <<https://docs.microsoft.com/pt-br/dotnet/csharp/programming-guide/>> Acesso:01.Fev.2018
- Manzano, José Augusto N. G. - Estudo Dirigido - Microsoft Visual C# Community 2015 - 1ª Ed. – Érica
- Stellman, Andrew / Greene, Jennifer - Use a Cabeça C# - 2ª Ed. - 2010 - Alta Books
- Deitel, Harvey. Deitel, Paul. C# Como Programar 1ª Ed. – Pearson
- CSHARP2. Disponível em: <<http://www.macoratti.net/pageview.aspx?catid=18>> Acesso:01.Fev.2018
- VS. Disponível em: <<https://www.visualstudio.com>> Acesso:23.Fev.2024
- Soluções e Projetos no VS. Disponível em: <<https://docs.microsoft.com/pt-br/visualstudio/ide/solutions-and-projects-in-visual-studio?view=vs-2019>> Acesso:24.08.2020