Windows Forms

Introdução

Já abordamos a sintaxe da linguagem C# e vimos como desenvolver aplicações que interagem com a entrada e saída padrão, ou seja, aplicações console.

O .NET Framework possui uma biblioteca (namespace) que contém todos os controles necessários para desenvolver aplicações Windows. Essa biblioteca de controles é conhecida pelo nome de Windows Forms (e daí o título deste capítulo) que está armazenada no namespace System.Windows.Forms. O .NET Framework ainda oferece uma outra biblioteca especializada para desenvolver aplicações com uma interface gráfica (e não apenas de janelas) mais rica conhecida como GDI+. Esse assunto está fora do escopo desta matéria e nos limitaremos a falar sobre o desenvolvimento de aplicações Windows tradicionais.

Em princípio, a forma de desenvolver aplicações Windows não mudou, e essa é uma boa notícia porque você não terá de aprender tudo de novo. Alguns controles possuem mais propriedades; é possível trabalhar com Error Providers quando usamos o controle TextBox através dos quais podemos implementar controle de erros de uma forma mais elegante.

De uma forma geral, o que podemos dizer é que os arquitetos da .NET tiveram como alvo principal nos poupar de fazer uso frequente da API do Windows, mesmo porque em .NET isso implica em fazer chamadas a código não-gerenciado.

Mais adiante veremos como podemos desenvolver os nossos próprios componentes, que poderão ser usados como todos os objetos que fazem parte da biblioteca .NET, e melhor ainda, estes poderão ser reusados através de qualquer linguagem que dê suporte a .NET.

Diferença entre Projeto e Solução
O Projeto é constituído por um ou vários grupos de arquivos relacionados (códigos, formulários, imagens, classes). A Solução é constituída por um ou vários projetos, e representa o produto final a ser entregue ao cliente.

Teclas de Atalho
F4- exibe a Janela de Propriedades
F7- exibe o código do formulário atual
F5- executa o projeto
F10 e F11 – usados para debugar.

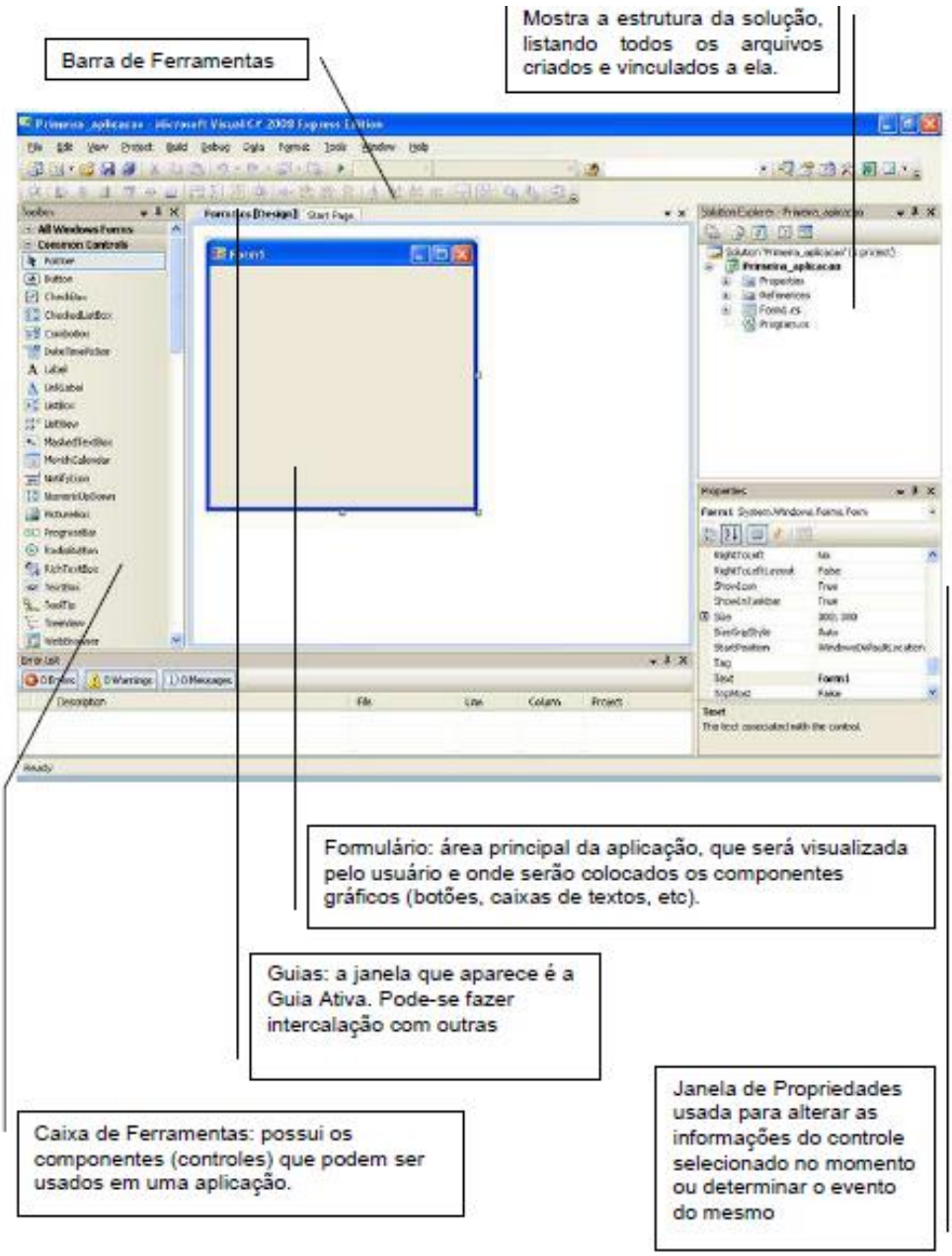
II- Desenvolvimento de um Projeto

Ao iniciar o Microsoft Visual C#, irá aparecer a página inicial onde deve se escolher no menu a opção FILE- New Project. Quando aparecer a caixa de diálogo (figura abaixo) deve-se selecionar o tipo de projeto que deseja fazer, como Windows Forms Application, Console Application, Class Library, entre outros.
O Windows Forms Application é usado para desenvolver aplicações Windows, que possuem elementos gráficos (botões, menus, caixas de texto) com os quais o usuário se

interage. Essa opção é a padrão ao iniciar um novo projeto, devendo apenas mudar o nome e clicar no botão OK.

O arquivo que será gerado é o Primeira_aplicacao.csproj (extensão csproj – projeto C#).

III- Ambiente de Desenvolvimento do C#

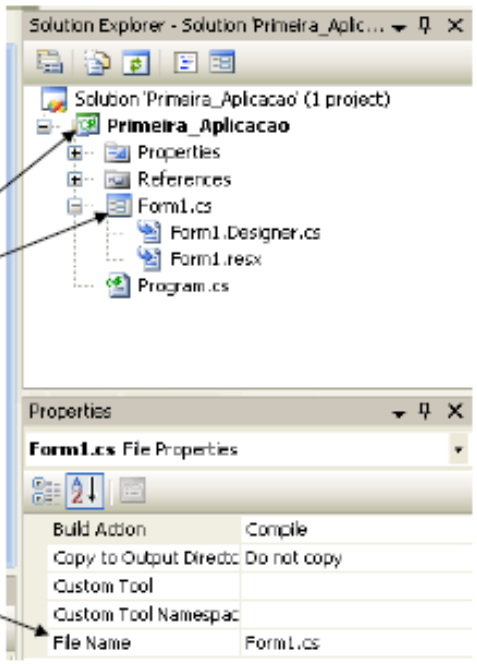


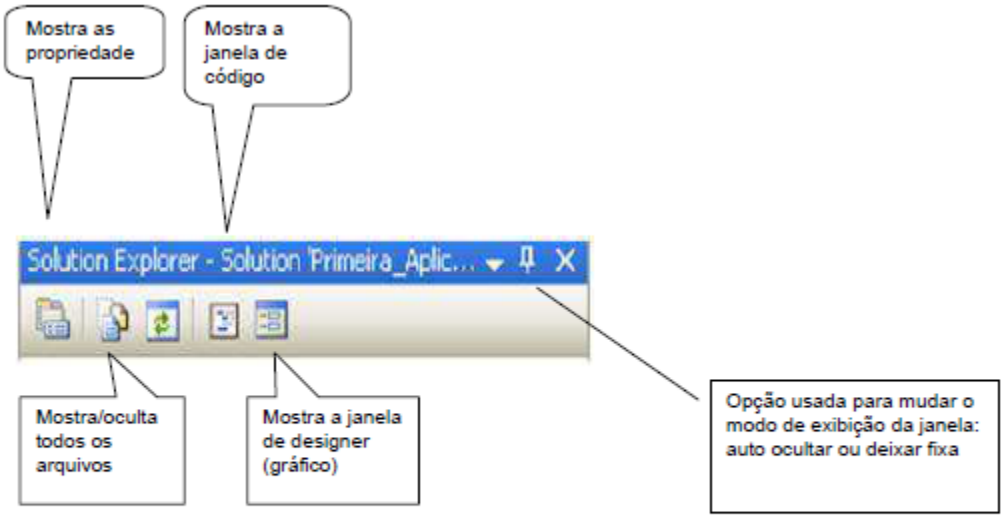
Solution Explorer

No Solution Explorer se gerencia todos os projetos, classes e formulários da solução (aplicação) desenvolvida.

É possível ver o nome dado para o primeiro projeto (que automaticamente ficou para a solução)

O nome inicial do formulário é Form1, que pode ser alterado diretamente clicando sobre o nome Form1.cs e alterando na Janela de Propriedades.





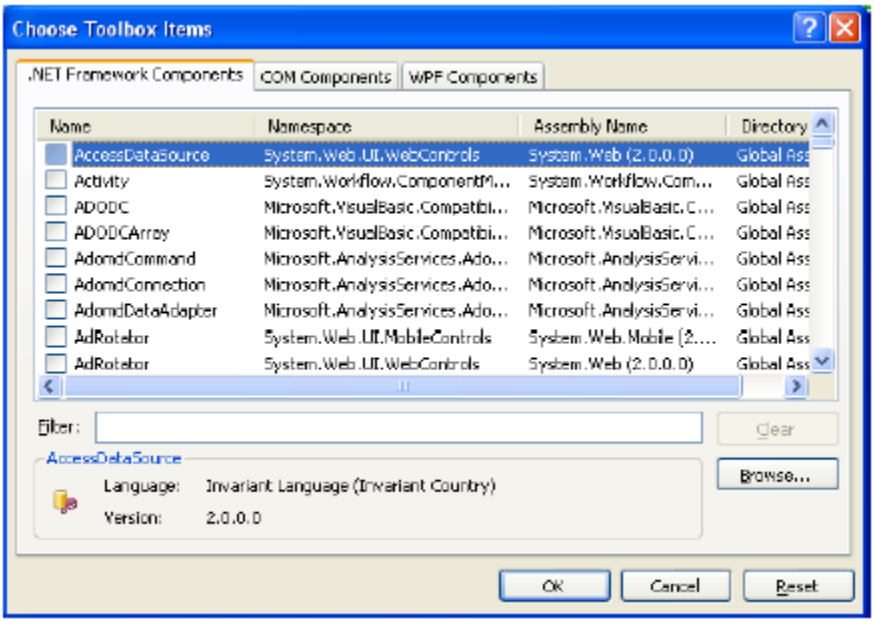
Dica:

Quando as janelas forem alteradas de maneira que não se consiga voltar ao normal, clique na opção Windows do menu, item Reset Windows Layout.

Curiosidade:
O sinal de + que oculta uma lista se chama colapsado(expandir) e o sinal de – é descolapsado.

Caixa de Ferramentas
Na Caixa de Ferramentas são mostrados vários controles (componentes) que poderão ser usados. Esses controles estão relacionados em grupos.

Se for necessário pode-se incluir novos componentes, clicando com o botão da direita do mouse sobre a caixa de Ferramentas e escolhendo a opção Choose Toolbox Item. Quando for acessar essa opção pela primeira vez é normal demorar um pouco para carregar todas as opções.



Dica:
Quando for selecionar vários componentes do mesmo tipo pressione e segure a tecla CTRL antes de selecionar o componente.

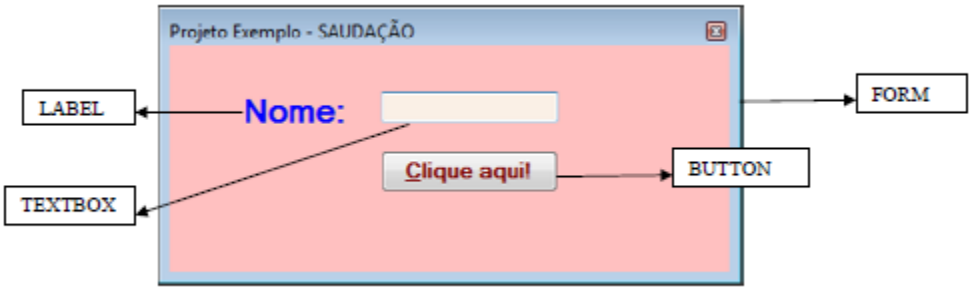
Janela de Propriedades

A Janela de Propriedades pode ser usada para alterar uma informação do controle selecionado (como o nome, cor) ou associar uma ação (evento) do usuário ao mesmo (como ao clicar o mouse).

Sempre deixe a exibição das propriedades em ordem alfabética para facilitar a sua procura.

Primeiro Projeto

Objetivo do Projeto: o usuário irá digitar o nome na caixa de texto e ao pressionar o botão “Clique aqui!” irá aparecer uma mensagem personalizada
“Bom dia <nome do usuário> !“.



Propriedades
As propriedades são usadas para alterar as características (informações) do formulário, de seus componentes e controles, como, por exemplo, mudar o texto, a cor, a posição, incluir uma figura.

Formulário O formulário é um elemento gráfico que aparece na tela. Ele é um contêiner para os componentes e controles		
Propriedade	**Valor**	**Significado**
FormBorderStyle	FixedToolWindow	Faz aparecer apenas a opção Fechar (X) na barra de título
Name	Form_saudacao	Altera o nome do formulário
Text	Projeto exemplo – saudação	Altera o texto que aparece na barra de título
BackColor	Red	Altera a cor do fundo do formulário
Opacity	70%	Modifica a porcentagem de transparência do formulário
StartPosition	CenterScreen	Faz com que o formulário ao ser exibido já fique no centro da tela

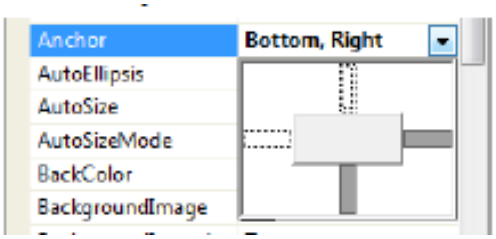
Controle Label O Rótulo, texto que aparece na tela e não pode ser alterado (editado) pelo usuário		
Propriedade	**Valor**	**Significado**
Name	textBox_nome	Altera o nome da caixa de texto
Text		Contém o conteúdo a ser digitado pelo usuário ou informado diretamente.
Font	....	Altera o tipo, estilo e tamanho da fonte
ForeColor	Blue	Altera a cor da fonte

Controle TextBox Caixa de texto que permite o usuário entrar com dados (string)		
Propriedade	**Valor**	**Significado**
Name	label_ nome	Altera o nome do label
Text	Nome:	Altera o texto que aparece na tela
Font	....	Altera o tipo, estilo e tamanho da fonte
ForeColor	Blue	Altera a cor da fonte
BackColor	Linen	Altera a cor do fundo

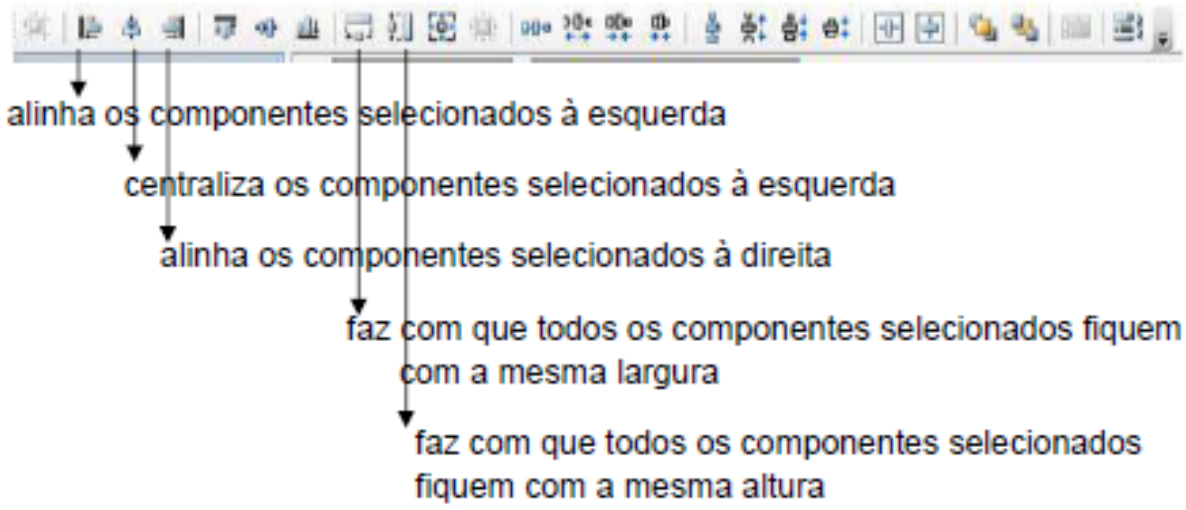
Controle Button Botão, é usado para chamar um evento quando clicado		
Propriedade	**Valor**	**Significado**
Name	button_exibe	Altera o nome
Text	&Clique aqui! Aparece Clique aqui! O operador & é usado para gerar uma tecla de atalho para o botão, ou seja, não precisa pressionar o botão com o mouse, basta pressionar o ALT+C	Altera o texto do botão
Font	....	Altera o tipo, estilo e tamanho da fonte
ForeColor	Maroon	Altera a cor da fonte
BackColor	Linen	Altera a cor do fundo

Dicas:

1ª) A propriedade Anchor determina o ajuste da posição do componente em relação ao formulário, ou seja, quando for redimensionado o formulário como será o comportamento do componente.
A opção Bottom e Rigth é usada para ajustar à direita na parte de baixo.



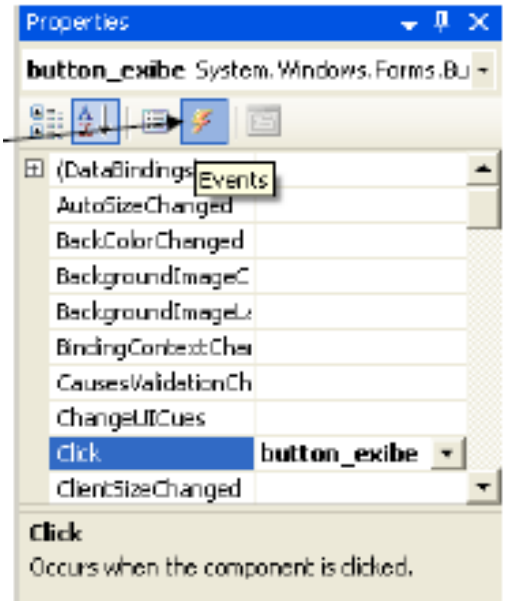
2ª) A barra de ferramenta Layout deve ser usada para agilizar a construção do formulário, onde pode-se selecionar vários componentes e alterar seus tamanhos numa única vez, modificar seus alinhamentos.



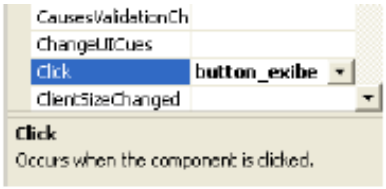
3ª) Para selecionar mais de um mesmo componente, selecione o componente e segure a tecla CTRL. Vá clicando no formulário e adicionando os componentes. Após adicionar os componentes pressione a tecla ESC (desativando a seleção do componente).

Eventos

Os eventos estão associados às interações (ações) realizadas pelo usuário, como o clicar do mouse sobre um botão, mover do mouse sobre uma figura, entre outros.
Nesse primeiro projeto irá ocorrer um evento quando o usuário clicar no botão, onde irá aparecer uma mensagem. Para definir um evento é necessário primeiro selecionar o componente/controle, depois ir janela de propriedades e clicar na opção Events.



O código que irá aparecer ao dar duplo clique no botão e acionar o seu evento está dado por:




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

//namespace raiz é o nome do projeto definido pelo usuário
namespace Projeto_Saudacao
{
/* a classe Form_saudacao que você desenvolve é partial, ou seja, parte dela já foi
escrita pelo C# e a outra parte você escreve*/
   public partial class Form_saudacao : Form
   {      //declaração dos métodos da classe
      public Form_saudacao()
      {
         InitializeComponent();
      }

                              nome do botão          tipo de evento
      private void button_exibe_Click(object sender, EventArgs e)
      {
      }
   }
}
```

Você deverá incluir seu código no evento selecionado (Click)

```
private void button_exibe_Click(object sender, EventArgs e)
{
        /*
        Para exibir uma mensagem usa-se a classe MESSAGEBOX e o método
        dela chamado SHOW
        Existem 21 formas diferentes de se usar o método show (sobrecarga do
        método show)*/
            MessageBox.Show(
            1 of 21  DialogResult MessageBox.Show(string text)
            text: The text to display in the message box.
        }
```

Você pode digitar o MessageBox das seguintes formas:

1ª) MessageBox.Show("Bom dia!");

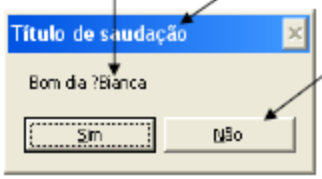
Texto que aparece no centro da Caixa de Diálogo

2ª) MessageBox.Show("Bom dia! "+ textBox_nome.Text);

Faz um concatenação (une) a string "Bom dia!" com o conteúdo digitado na caixa de texto.

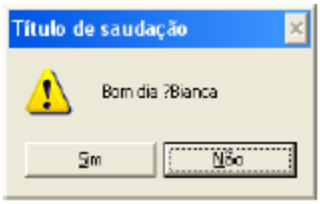
3ª) MessageBox.Show("Bom dia? "+textBox_nome.Text, "Título de saudação", MessageBoxButtons.YesNo

Faz aparecer um texto central um título na caixa de diálogo, e botões de escolha.



Ficará o primeiro botão destacado (que tem o foco), se você não definir o botão padrão.

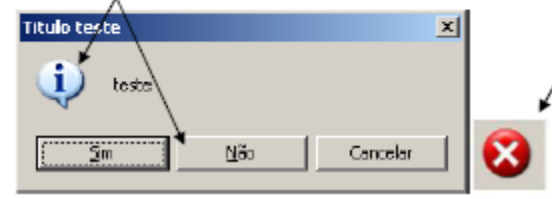
4ª)




```
MessageBox.Show( "Bom dia ?"+textBox_nome.Text,
"Título de saudação",MessageBoxButtons.YesNo,
MessageBoxIcon.Exclamation,MessageBoxDefaultButton.Button2);
```

Tipos de botões e ícones do MessageBox

Botões	Ícones
AbortRetryIgnore OK OKCancel RetryCancel YesNo YesNoCancel	Asterisk - Mostra um balão com a letra i Error ou Stop - Mostra um balão vermelho com um X Exclamation ou Warning - Mostra um triângulo amarelo com um ponto de exclamação. Information - Mostra um círculo com a letra i None - Não é mostrado nenhum ícone Question - Mostra um balão com um ponto de interrogação



Variáveis

Em C# existem vários tipos de dados (variáveis) como: inteiros, ponto flutuante, lógico e caracter. Todas as regras quanto à forma de usar um tipo estão definidos na CTS (Common Type System), ou seja, para haver essa integração e compatibilidade entre as linguagens .Net, existem os tipos comuns compartilhados entre elas (internos).

Na plataforma .Net todas as suas linguagens de programação compartilham o mesmo sistema de CTS, ou seja, o tipo inteiro definido na linguagem C# é o mesmo tipo inteiro definido em VB.Net ou J#.Net. Abaixo segue uma tabela com os tipos de dados, sua referência na CTS, a quantidade de bytes que ocupa.

Tipo de dado	Tipo .Net	Descrição	Bytes
Boll	Boolean	V ou F	1
Byte	Byte	Sem sinal (0 até 255)	1
Sbyte	Sbyte	Com sinal (-128 até 127)	1
Short	Int16	Com sinal (-32.768 até 32.767)	2
Ushort	Uint16	Sem sinal (0 até 65.535)	2
Int	Int32	Com sinal (-2.147.483.648 até 2.147.483.647)	4
Uint	Uint32	Sem sinal (0 até 4.294.967.295)	4
Long	Int64	Com sinal (-9.223.372.036.854.775.808 até 9.223.372.036.854.775.807)	8
Ulong	Uint64	Sem sinal (0 até 18.446.744.073.709.551.615)	8
Float	Single	Ponto flutuante. (+/- $1,5*10^{-45}$ até +/- $3,4*10^{1038}$) 7 dígitos significativos. Exige o sufixo "f" ou "F".	4
Double	Double	Ponto flutuante de precisão dupla. (+/-$5,0*10^{-324}$ até+/-1,8*10308) - 15 a 16 dígitos significativos	8
Decimal	Decimal	Precisão fixa de 28 dígitos e a posição do ponto decimal. Exige o sufixo "m" ou "M".	16
Char	Char	Um único caracter	2
String	String	Armazena cerca de 2 bilhões de caracteres	2 para cada caracter
Datetime	Datetime	Tipo data e hora (vai de 1/1/1 até 31/12/9999 e horas de 0:00 até 23:59:59	8

Sintaxe:

Tipo da variável nome(s) da(s) variável (veis) [= valor inicial];

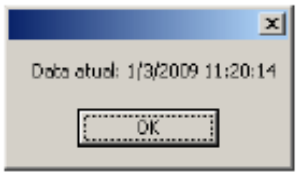
A inicialização da variável não é obrigatória

Exemplo de declaração e atribuição

```
byte idade;
byte idade = 33;
char sexo = “F”;
string nome = “Julia”;
boolean Sair = False;
Datetime  Data_nasc;
Datetime Data_hoje = Datetime.Now;
float nota = 7.5f;
//internamente se define o separador decimal como ponto (.) e coloca o sufixo f de float.
```

Exemplos práticos:

1º)
```
private void button1_Click(object sender, EventArgs e)
{
    DateTime data_atual = DateTime.Now;
    MessageBox.Show("Data atual: " + data_atual.ToString());
}
```

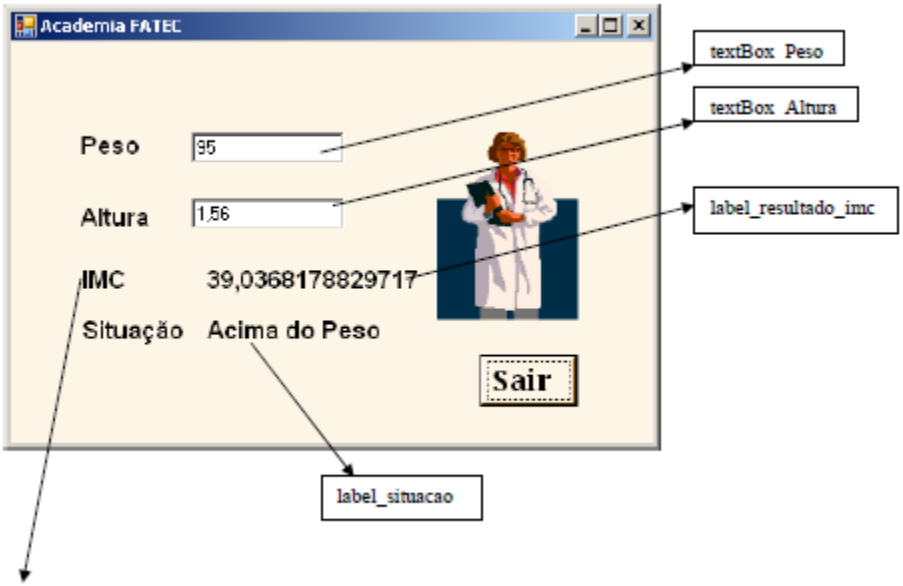


2º)
```
float nota;
nota = 7.5f;
MessageBox.Show("Nota: " + nota.ToString());
```

Alerta:
Na digitação do usuário sempre é vírgula (,) o separador decimal e internamente no código para o programador é ponto(.).

Segundo Projeto

Objetivo do Projeto: Neste segundo projeto serão feitas duas entradas o peso e a altura. Quando o usuário clicar sobre a palavra IMC será calculado o Índice de Massa Corporal do usuário, verificada e mostrada a sua situação de peso (acima, abaixo ou com peso ideal)



Código do evento Click do Label de nome label_imc

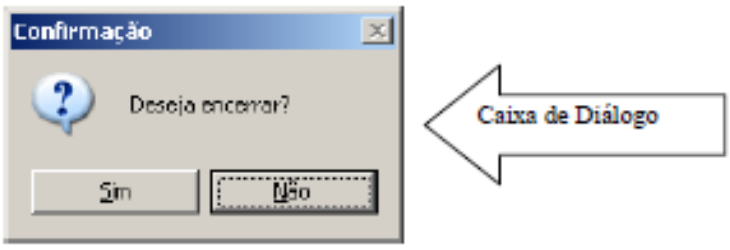
```
private void label_imc_Click(object sender, EventArgs e)
{
   double altura, peso, imc;
   altura = Convert.ToDouble(textBox_Altura.Text);
   peso = double.Parse(textBox_Peso.Text);
   // imc = peso / altura * altura;   ou
   imc = peso / Math.Pow(altura, 2);
   label_resultado_imc.Text = imc.ToString();
   if (imc < 19)
      label_situacao.Text = "Abaixo do Peso";
   else if (imc < 25)
      label_situacao.Text = "Peso Ideal";
   else
      label_situacao.Text = "Acima do Peso";
}
```

Alerta:

- Foi usado o método Pow() da classe Math para calcular a altura^2. A Math está no namespace System.
- Foi usado o Parse e o Convert para mostrar que ambos podem ser usados para conversão de string para Double. Poderia ser usado apenas o Convert.

Código do evento Click do Button de nome button_Sair

```
private void button_Sair_Click(object sender, EventArgs e)
{
        if (MessageBox.Show("Deseja encerrar? ", "Confirmação",
        MessageBoxButtons.YesNo, MessageBoxIcon.Question,
        MessageBoxDefaultButton.Button2) == DialogResult.Yes)
    this.Close();
}
```



Quando foi usada a classe MessageBox e o método Show foram passados alguns argumentos, como:

"Deseja encerrar? " → mensagem que aparece no centro da caixa de diálogo

Confirmação" → mensagem que aparece na barra de título

MessageBoxButtons.YesNo → tipos de botões que vão aparecer

MessageBoxIcon.Question → tipo de ícone que vai aparecer

MessageBoxDefaultButton.Button2 → qual dos botões que tem inicialmente o foco (destaque)

Para fazer a verificação de qual botão o usuário pressionou da caixa de diálogo deve ser usado o DialogResult e a opção a ser testada.

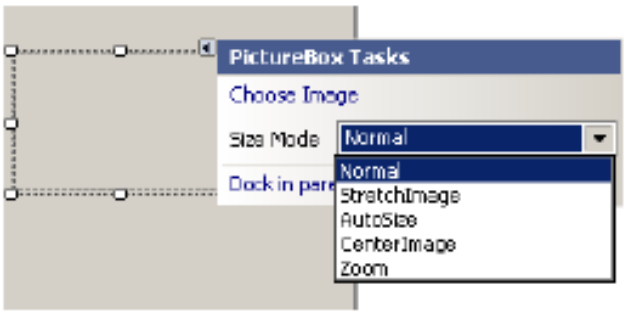
== DialogResult.Yes

Dica:

Quando for usar o controle PictureBox, não precisa ir na propriedade para escolher a figura que deve aparecer e a forma como ela vai aparecer.

Quando for inserido PictureBox e colocado o foco sobre esse controle deve ser clicado na seta superior da caixa e clicado sobre a opção Choose Image para escolher a imagem.

Na mesma seta existem as opções de visualização da imagem, através da opção Size Mode.



Terceiro Projeto

Objetivo do Projeto: Neste terceiro projeto serão feitas entradas de dados para cálculo das despesas de viagem, no final serão mostrados os valores calculados. Quando não for digitada uma informação necessária, será mostrada mensagem de alerta e retornado para o campo não preenchido.

Algumas Propriedades alteradas		
Formulário	Icon	É usada para alterar o ícone que aparece na barra de título do formulário
Botões	Image	É usada para adicionar um imagem ao botão
	ImageAlign	Para definir o alinhamento da imagem em relação ao botão
	TextImageRelation	Modifica o tamanho da imagem em relação ao texto
Groupbox	Text	Meio de transporte
Groupbox	Text	Valores do Pacote
ComboBox	Itens	SP
		RJ SC
TextBox	Enabled	False

- Nesse projeto foi colocado um GrouBox usado para agrupar os dois RadioButton

- Para a inclusão dos itens do ComboBox pode se clicar sobre a seta e ir na opção Edit Items

Código do evento Click do Botão Limpar Dados

O botão limpar terá a ação de limpar o conteúdo selecionado ou digitado

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

  textBox_nome.Text = "";
  radioButton_aviao.Checked= false;
   radioButton_onibus.Checked = false;
}
```

Para limpar uma caixa de texto pode-se usar a propriedade text usar o método Clear().

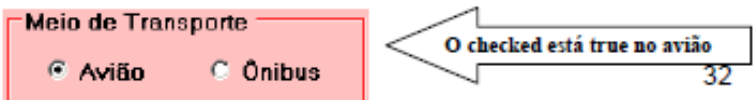
```
textBox_nome.Clear();
```

Ou

```
textBox_nome.Text = string.Empty;
```

Para tirar a opção de checado(escolhido) do radio button é usado a propriedade checked. Quando ela está false não aparece a bolinha, quando está true aparece.

```
radioButton_aviao.Checked= false;
radioButton_onibus.Checked = false;
```



Código do evento Click do Botão Calculdora
A ação desse botão será executar o processo Calculadora do Windows. Para isso é necessário saber a localiação do arquivo .exe

```
private void button_Calculadora_Click(object sender, EventArgs e)
  {
     Process.Start(@"C:\windows\system32\calc.exe");
  }
```

Foi incluído o namespace da classe Process

```
using System.Diagnostics;
```

Código do evento Click do Botão Confirmar

Em todas as verificações se o valor testado for inválido deve ser exibida uma mensagem de alerta e depois desviado o foco para o campo.

```
private void button_confirmar_Click(object sender, EventArgs e)
 {
    if (textBox_nome.Text.Length == 0)
    {
       MessageBox.Show("Digite o nome");
       textBox_nome.Focus();
     }

    if (comboBox_destino.Text == "")
    {
       MessageBox.Show("Destino não escolhido");
       comboBox_destino.Focus();
    }
```

```
if (radioButton_aviao.Checked == false && radioButton_onibus.Checked == false)
        {
           MessageBox.Show("Meio de transporte não escolhido");
           radioButton_onibus.Focus();
        }
```

Código do evento Load do formulário
É possível definir ações que deverão ser executadas ao iniciar o formulário, através do evento Load.
No projeto atual foi redimensionado o formulário para não aparecer o GroupBox com os resultados, já que no começo não existem resultados.

```
private void Form_agencia_Load(object sender, EventArgs e)
{
   this.ClientSize = new System.Drawing.Size(533, 309);
}
```

Código do evento Click do Botão Calcular Valor da Viagem
Ao pressionar o botão calcular serão verificadas as opções selecionadas e de acordo com elas feito o calculo.

- Primeiro: é verificado qual destino o usuário escolheu através do ComboBox.
- Segundo: é verificado qual meio de transporte o usuário escolheu através dos RadioButton.
- Terceiro: é feita a soma total de gastos
- Quarto: o formulário é redimensionado para aparecer aparecer(tornar visível) o GroupBox que tem os valores calculados.
- Quinto: os valores são jogados(atribuídos) aos devidos componentes de exibição.

```
float gasto_destino, gasto_transporte, gasto_total;

switch (comboBox_destino.Text.ToUpper())
     {
   case "SP": gasto_destino = 1000f;
      break;
   case "RJ": gasto_destino = 1500f;
      break;
   case "SC": gasto_destino = 2000f;
      break;
            default:   gasto_destino = 0.0f;
      break;
    }

if (radioButton_aviao.Checked == true)
   gasto_transporte = 100f;
else
   gasto_transporte = 30f;

gasto_total = gasto_transporte + gasto_destino;

this.ClientSize = new System.Drawing.Size(533, 441);
groupBox_Resultado.Visible = true;

label_GastoDestino.Text = gasto_destino.ToString("C2");
label_GastoTransporte.Text = gasto_transporte.ToString("C2");
textBox_ValorTotal.Text = gasto_total.ToString("C2");
```

1 2 3 4 5

Código do evento Leave do TextBoxNome

O evento Leave é usados quando se deseja executar uma ação no componente após ele sair de foco.
No projeto após a pessoa digitar o nome, esse mesmo não importa como foi digitado, será transformado em maiúsculo.

```
private void textBox_nome_Leave(object sender, EventArgs e)
{
   textBox_nome.Text= textBox_nome.Text.ToUpper();
}
```

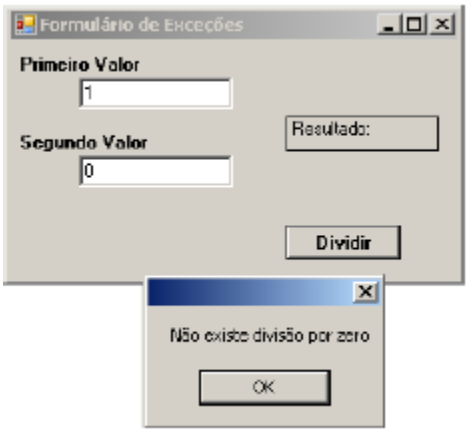
Dica: Fazer o C# trazer a estrutura pronta para uso.

1ª) Ao pressionar o botão da direita do mouse, você deve escolher a opção Insert Snippet..., depois dá uma duplo clique na opção Visual C#. Após isso é só escolher a estrutura e ele a trará pronta na tela. Por exemplo, na aplicação atual você pode utilizar para trazer a do switch().

2ª) Todas as configurações iniciais do formulário e seus componentes podem ser vistas através do arquivo **Form_agencia.Designer.cs**

Quarto Projeto

Objetivo do Projeto: Nesse projeto será mostrado o tratamento de exceções.
De forma que quando for digitado um valor inválido não será interrompida a execução da aplicação.



Um erro ou uma exceção pode ser: não digitação de um valor (causando um erro de conversão), digitação de um valor muito longo, erro quanto ao formato, divisão por zero, definição de um caminho (diretório) ou arquivo inválido, falha do hardware, erro do sistema operacional, erros de dispositivos.

O bloco de comandos que pode causar uma exceção deve estar dentro **do try..catch**, de forma que se possa capturar a exceção e manipulá-la. Somente o código dentro do try..cath está protegido.
Somente um catch será executado caso ocorra uma das possíveis exceções do código definido dentro do try. A estrutura do try..catch pode ser:

```
try
{
        // Código que pode estar sujeito a exceção (erro)
}
catch( TipoExcecao1 )
{
        // Tratamento para exceção tipo 1
}
catch( TipoExcecao2 )
{
        // Tratamento para exceção tipo 2
}
catch
{
        // Tratamento para qualquer tipo de exceção
}
finally
{
        // Trecho que deve sempre ser executado, havendo ou não exceção
}
```

As exceções podem ser tratadas individualmente ou de forma genérica.

Tratadas de forma genérica: Dessa forma não é tratado de forma individual o erro, a exceção.

```
private void dividirbutton_Click(object sender, EventArgs e)
{
    try
    {
      decimal valor1, valor2, resultado;
      valor1 = Convert.ToDecimal(valor1textBox.Text);
      valor2 = Convert.ToDecimal(valor2textBox.Text);
      resultado = valor1 / valor2;
      resultadolabel.Text = Convert.ToString(resultado);
    }
    catch (Exception)
    {
      MessageBox.Show("Não pode ser feita a divisão - Ocorreu uma
exceção");
    }
}
```

Pode ser gerada uma instância da classe Exception

Pode ser feita a instância da classe Exception. No código abaixo o nome dado é **ex**, mas poderia ser qualquer nome. Essa classe tem a propriedade **Message** que mostra a descrição da exceção.

```
private void dividirbutton_Click(object sender, EventArgs e)
{
    try
    {
      decimal valor1, valor2, resultado;
      valor1 = Convert.ToDecimal(valor1textBox.Text);
      valor2 = Convert.ToDecimal(valor2textBox.Text);
      resultado = valor1 / valor2;
      resultadolabel.Text = Convert.ToString(resultado);
    }
    catch (Exception ex)
    {
      MessageBox.Show(ex.Message, "Erro Genérico");
    }
}
```

Tratadas de forma específica

Pode haver a necessidade de tratar cada exceção de forma individual. Para cada exceção tem um catch e uma classe. Todo objeto que representa uma exceção é derivada da classe Exception que está na namespace System.

```
private void dividirbutton_Click(object sender, EventArgs e)
{
    try
    {
      decimal valor1, valor2, resultado;
      valor1 = Convert.ToDecimal(valor1textBox.Text);
      valor2 = Convert.ToDecimal(valor2textBox.Text);
      resultado = valor1 / valor2;
      resultadolabel.Text = Convert.ToString(resultado);
    }

    catch (DivideByZeroException)
    {
      MessageBox.Show("Não existe divisão por zero");
    }
      //trata da exceção de divisão por zero

     catch (OverflowException)
     {
      MessageBox.Show("Erro de estouro");
     }
      //trata da digitação de um número muito grande, não permitido
para o tipo definido ou para o cálculo a ser feito.

      catch (FormatException)
      {
        MessageBox.Show("Formato Inválido de dados");
      }
      //trata da exceção quanto a formato inválido, como exemplo
podemos destacar: quando há a exigência de um número e é digitado um
texto ou um espaço em branco. Principalmente para conversão de valores
}
```

Bloco finally

Tudo definido dentro do finally sempre será executado, existindo ou não a ocorrência de exceções (executando ou não o bloco catch).

```
try
{
      ...
}
catch ()
{
      ...
}
finally
{
      MessageBox.Show("Sempre é executado dando erro ou não");
      this.close();
}
```

Exceção gerada em tempo de execução.

Uma exceção pode criar suas próprias exceções. De forma que podem ocorrer novos erros durante o tempo de execução e para isso deve ser usado o throw (que gera exceção). O throw deve estar dentro do bloco do try.

```
if (valor2 % 2 == 0)
{
      resultado = valor1 / valor2;
      resultadolabel.Text = Convert.ToString(resultado);
}
else
{
     throw new Exception("O segundo valor deve ser par");
}
```

Outro exemplo:

```
if ( CPF.Length != 11 )
{
      throw new Exception( "CPF Inválido" );
}
```

Cuidado:

A definição e tratamento das exceções deve ser feita da mais específica para a mais genérica, nessa ordem, senão ocorrerá erro de código.

Dicas:

Outras Exception:

InvalidcastException (erro de conversão direta- cast)

IndexOutOfrangeException: índice do array fora do limite

FileNotrFoundException: arquivo não encontrado

SqlException: erro na operação do Sql Server.