

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

Introdução

C# (lê-se CSharp) - Programação para Windows

Diferenças entre aplicações console e aplicações Windows.

- **Aplicações Console**
 - Programa é responsável pelo fluxo do processamento
 - Temos que definir que instruções serão executadas e em que ordem serão executadas
- **Aplicações Windows**
 - O programa não controla o fluxo do processamento, ele responde e trata eventos que ocorrem no sistema.
 - Mais ou Menos assim – o usuário clica com o mouse e o Windows verifica que objeto estava debaixo do mouse no momento que foi clicado. Em seguida manda uma mensagem para a aplicação informando que ocorreu um clique em determinada coordenada. A aplicação então responde à mensagem executando alguma funcionalidade implementada.

Vamos criar nosso primeiro projeto. Você já deve ter feito esse clássico exercício usando uma Aplicação Console. Trata-se do exemplo do Cálculo do IMC (Índice de Massa Corporal) de uma pessoa e sua classificação (abaixo do peso, peso ideal e obeso), dentro de uma faixa de valores fornecida através da tabela.

Definição do nosso programa:

- O usuário deve informar o peso e a altura da pessoa
- Cálculo do IMC $\rightarrow \text{IMC} = \text{Peso} / (\text{Altura}^2)$
- Classificar a pessoa de acordo com o índice do IMC

IMC	Classificação
$\text{Imc} < 20$	Abaixo do Peso
$20 > \text{Imc} < 25$	Peso Ideal
$\text{Imc} > 25$	Obeso

Iremos utilizar este projeto para iniciar nosso conhecimento em particularidades de programação de aplicações Windows. Veremos alguns conceitos de componentes, objetos, propriedades, métodos, etc.

Bom, vamos por a mão na massa.

Primeiramente inicie o Microsoft Visual Studio.
No Menu File, selecione a opção New Project.

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

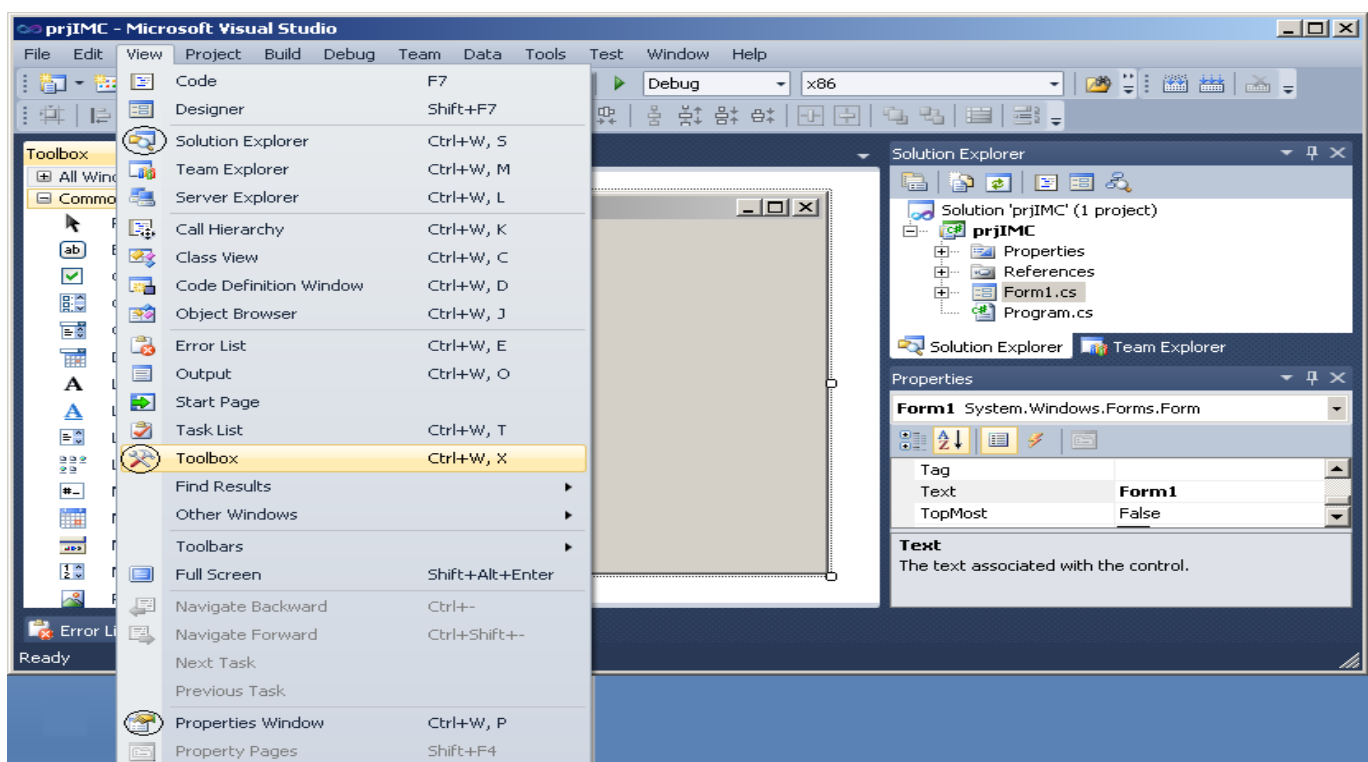
Selecione a linguagem C# e opção Windows Forms Application e em seguida troque o nome da aplicação.

Vamos por enquanto nos conter ao básico. Em Name: vamos colocar prjIMC (prj=projeto), na Location: escolha ou crie a sua pasta de trabalho para o projeto clicando no botão Browse...

Veja que automaticamente o Solution name: foi alterado para prjIMC. Deixe checado a opção Create directory for solution, assim será criado uma pasta com o nome prjIMC.

Feito isso, deverá surgir na tela o ambiente de desenvolvimento do projeto conforme abaixo:
Nesta tela temos as áreas do Form (design), a Toolbox, Solution Explorer e Properties.

Se você não estiver visualizando é só ir no Menu em View e selecionar.



Na Toolbox temos vários controles que iremos utilizar, os mesmos estão agrupados por tipo.

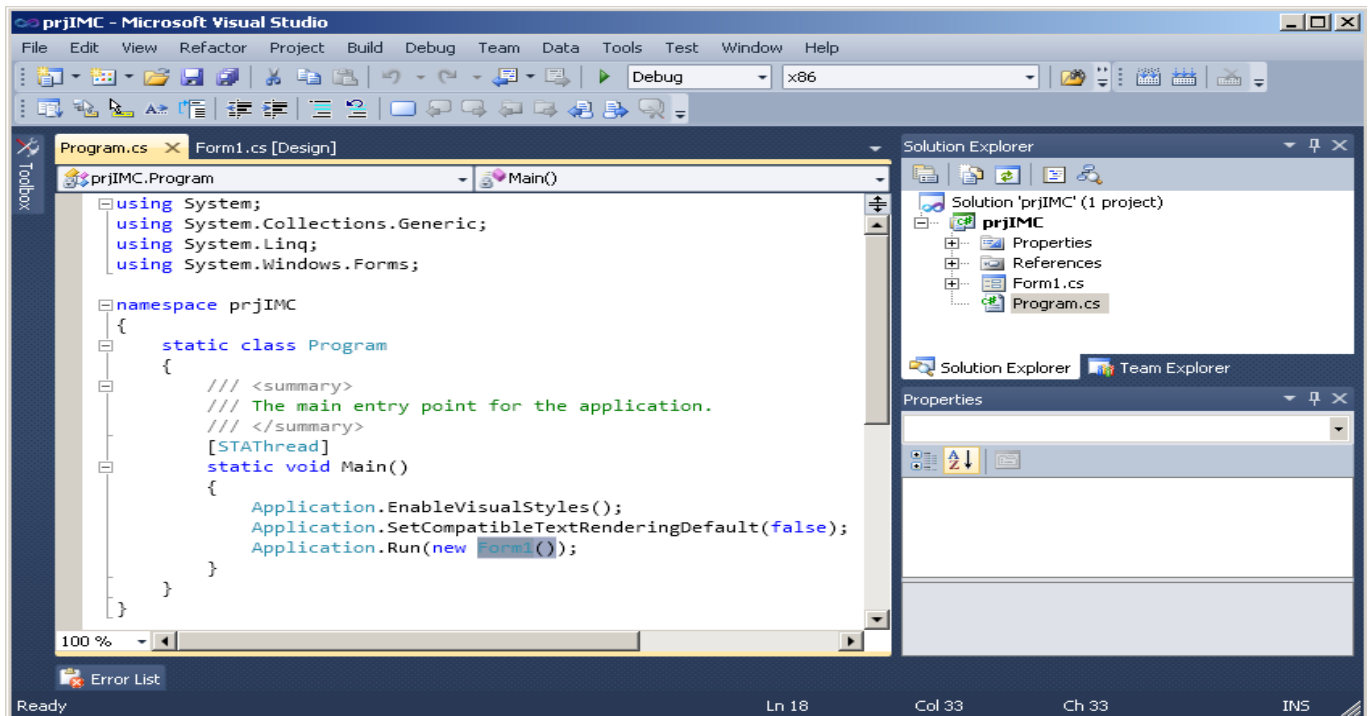
Na Solution Explorer, temos tudo referente a sua solução, quantidades de Forms, Pastas, Classes, etc...

Veremos mais adiante alguns destes itens. Veja que temos um arquivo Program.cs, dando um duplo clique no mesmo ou na ícone View Code no Solution Explorer, você verá o código, este mostra por onde o projeto será iniciado.

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

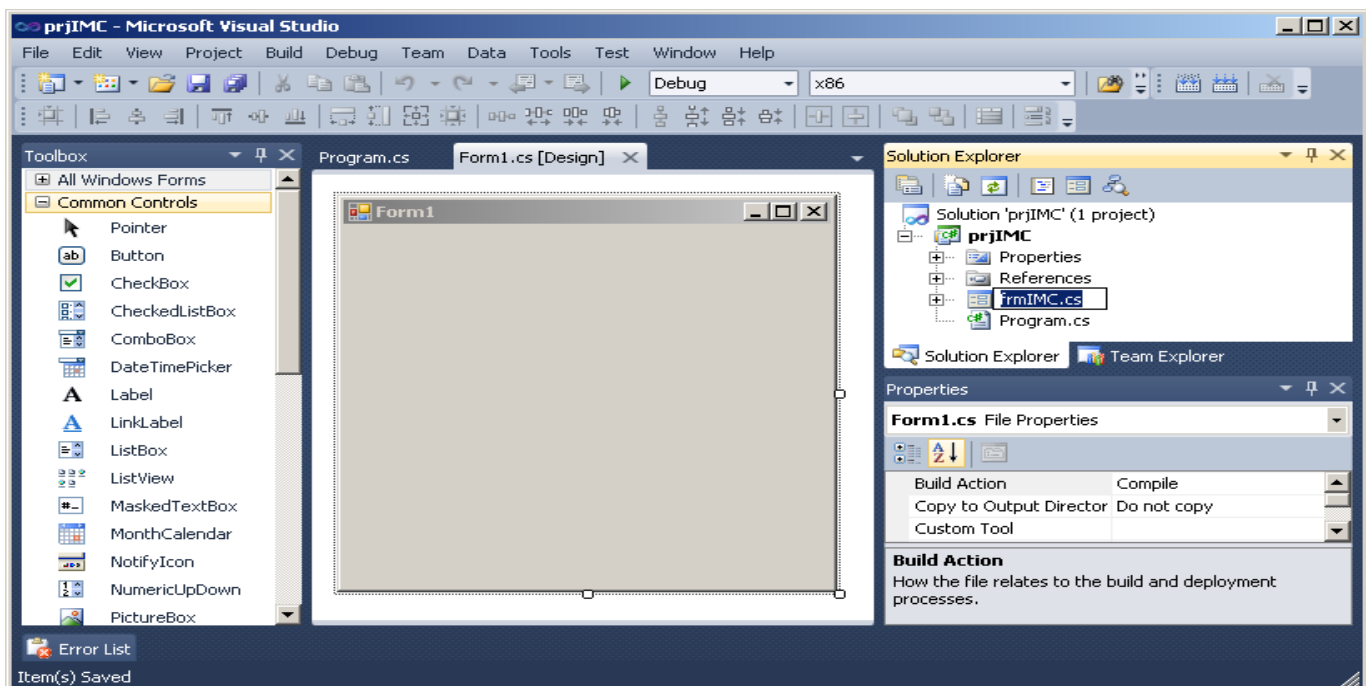
Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo



Em primeiro lugar vamos alterar o nome do Form1 (nome do arquivo) para frmIMC. Veja que quando se altera este nome o mesmo pedirá se quer colocar o nome do objeto, de Form1 como frmIMC.

De um clique no nome Form1 ou clique com o botão direito do mouse e escolha Rename.

Não se esquecer de deixar no nome que será dado e o tipo do arquivo .cs

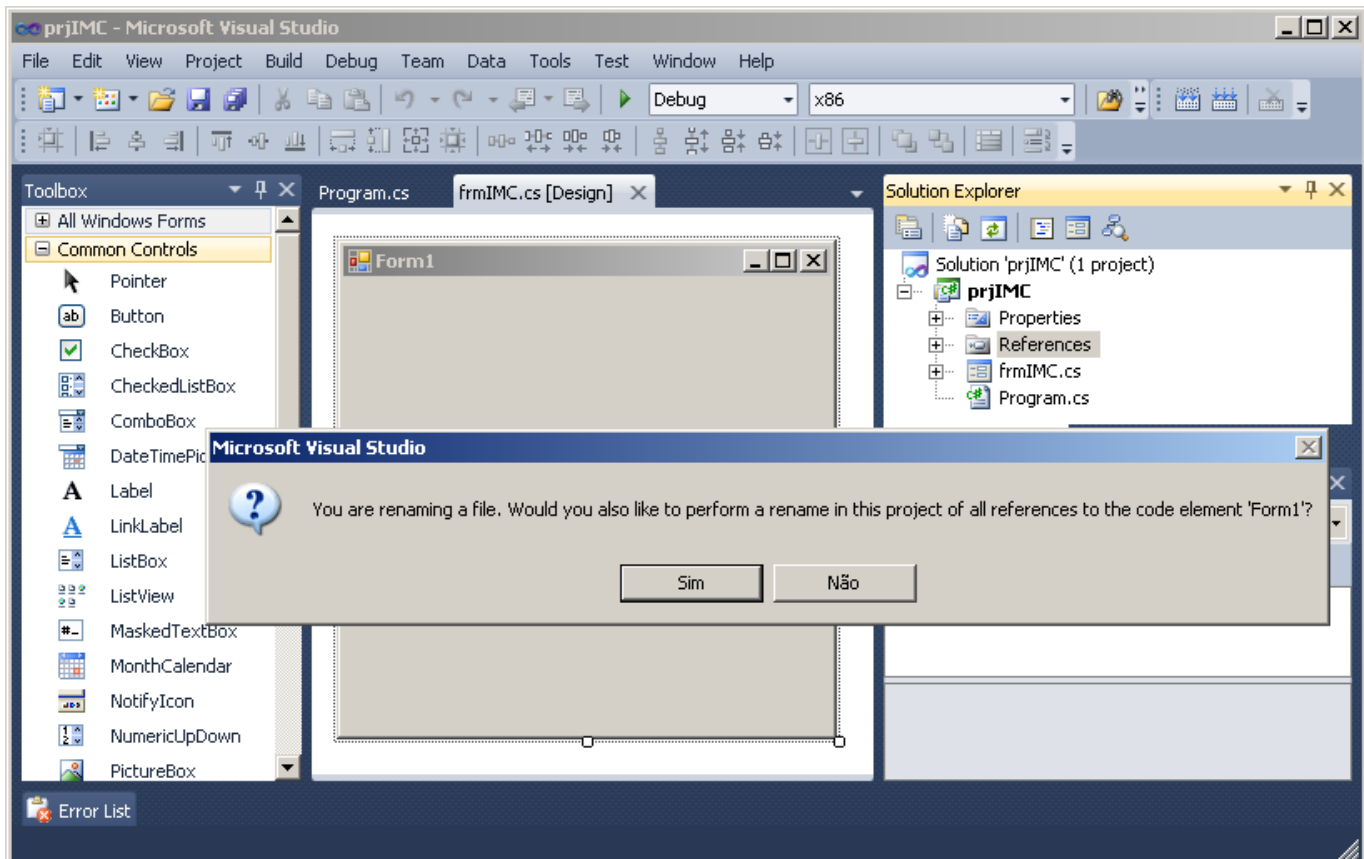


Depois de renomear será mostrado a tela abaixo: Clique em Sim.

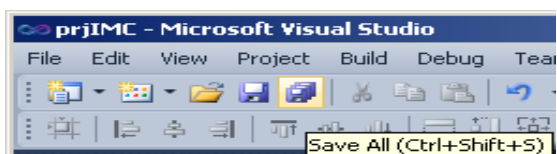
Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

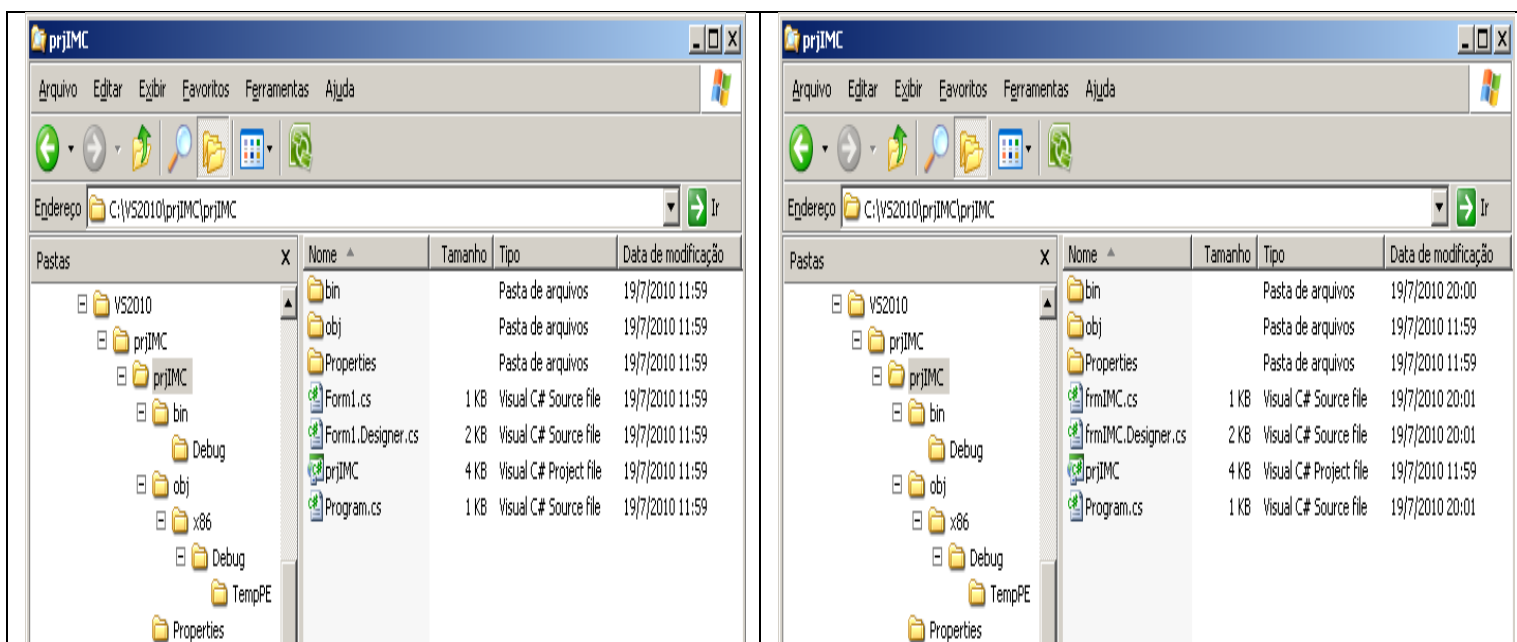
Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo



Clique na ícone Save All.



Desta forma teremos o mesmo nome do arquivo físico e do form. Veja as imagens abaixo:

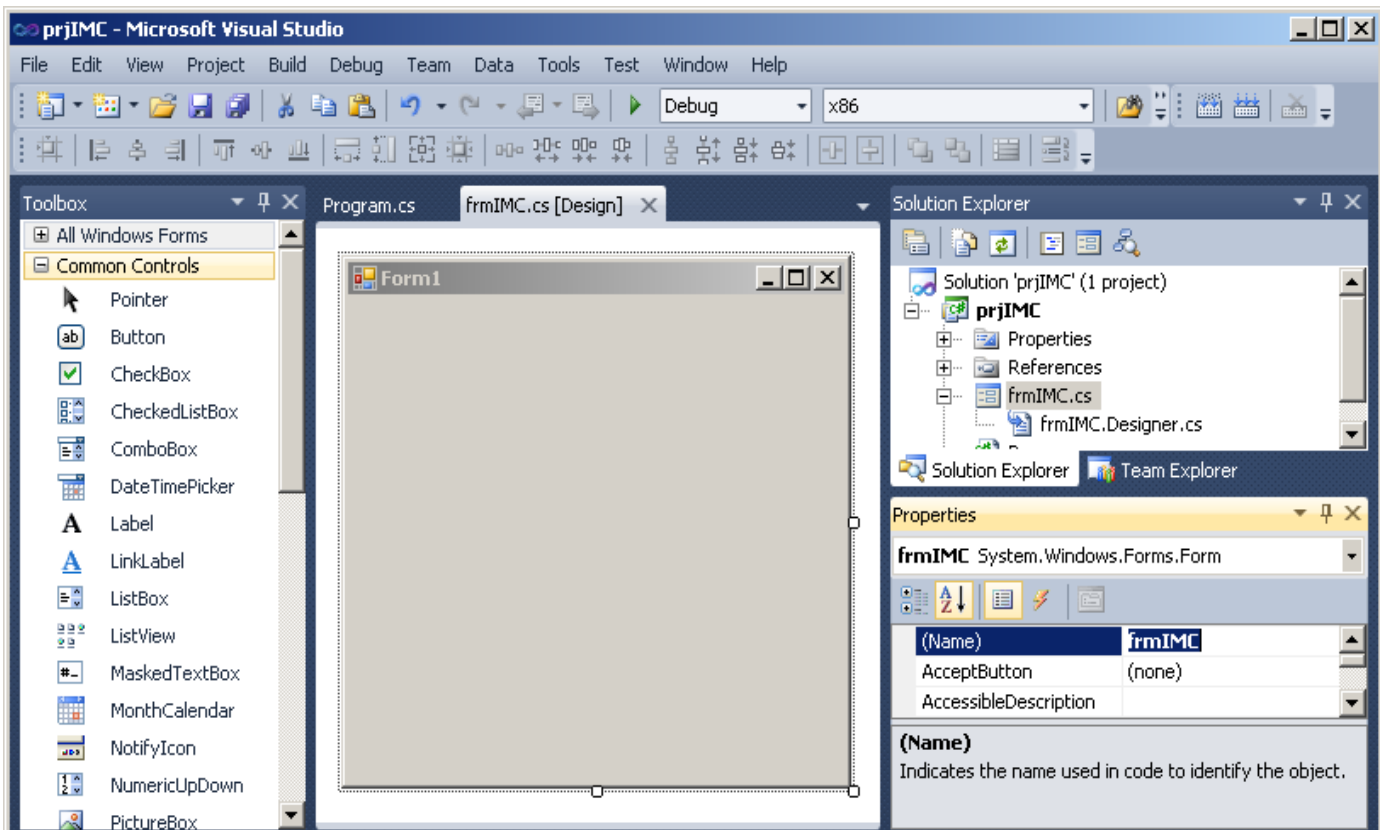


O nome do objeto form foi alterado para frmIMC.

Apostila - Microsoft Visual Studio – C#

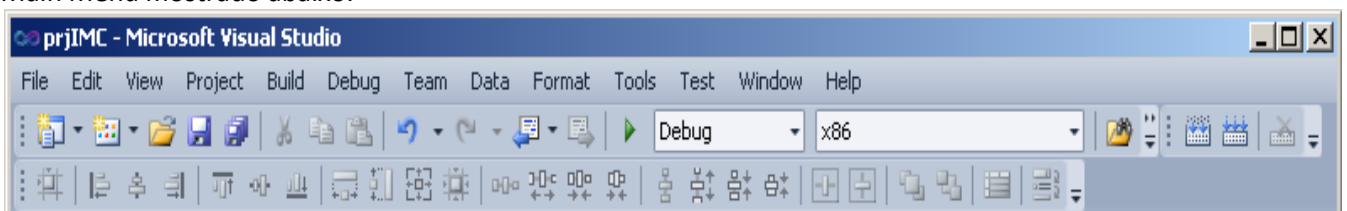
C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo



a) Vamos identificar nesta área as seguintes regiões.

- Formulário no centro (ou frmIMC.cs [Design]) – container onde inserimos os componentes que irão dar aparência e funcionalidade ao projeto.
- Toolbox no lado esquerdo – onde ficam os componentes disponíveis para utilização no projeto
- Solution Explorer do lado direito acima – Exibe a estrutura da solução / projeto
- Properties do lado direito abaixo – Exibe / permite editar as propriedades do componente atualmente selecionado.
- Main Menu mostrado abaixo.



Antes de começarmos o nosso projeto vamos ver a janela **Properties**, onde temos várias ícones e as principais são a de **Properties** (onde será mostrado todas as propriedades do objeto selecionado) e a de **Events** (serão os eventos que você poderá fazer, note que como estamos no objeto **Form** ele está posicionado no evento **Load**, se fosse um objeto botão estaria no evento **Click** e assim por diante.

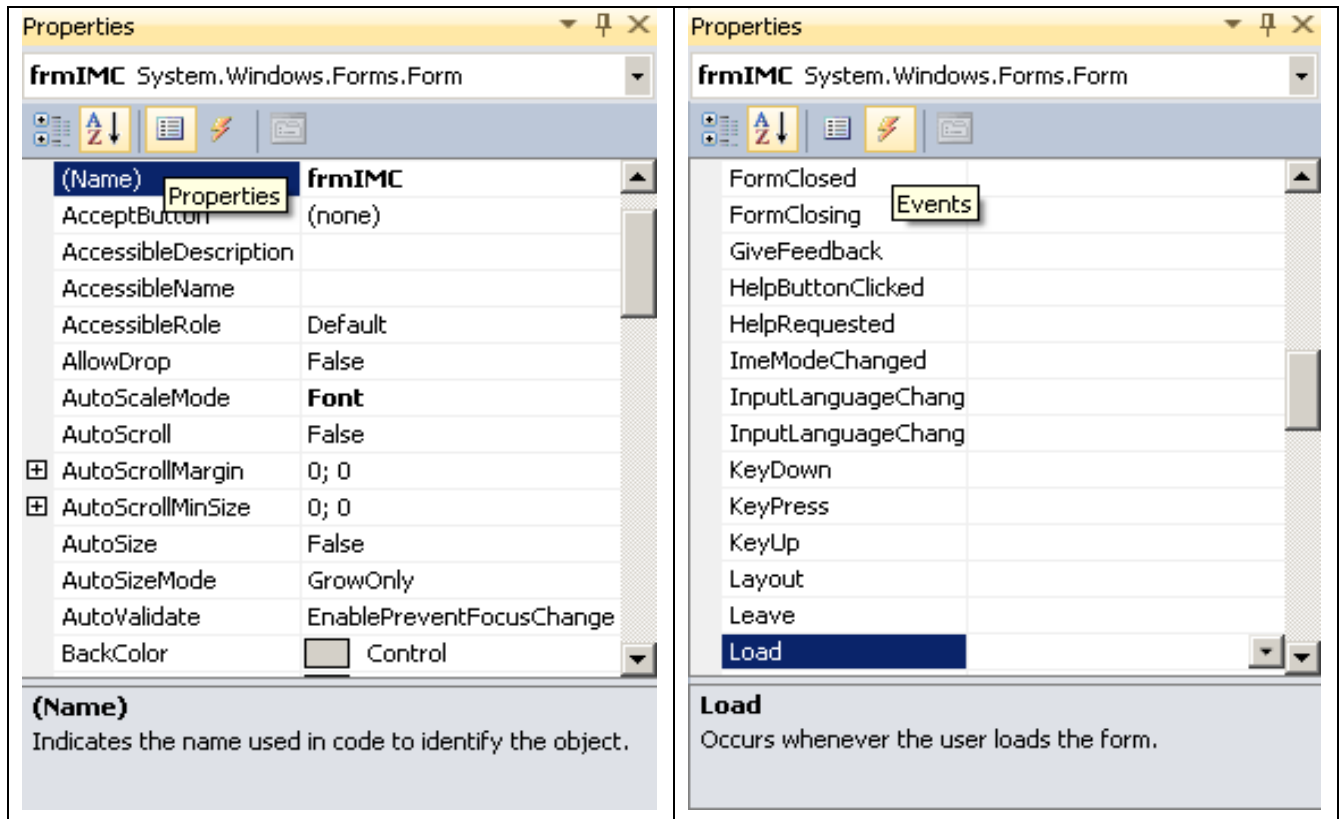
Cada objeto terá um evento default quando for dado um duplo clique no mesmo.

Veremos vários eventos onde selecionaremos o mesmo para elaboração do código.

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo



- No nosso primeiro projeto usaremos apenas os seguintes componentes: **Button**, **Label**, **TextBox**, além, é claro, do próprio **Form** e os eventos **Click** dos botões.

Nestas imagens mostraremos os objetos e suas propriedades e eventos mais usados.

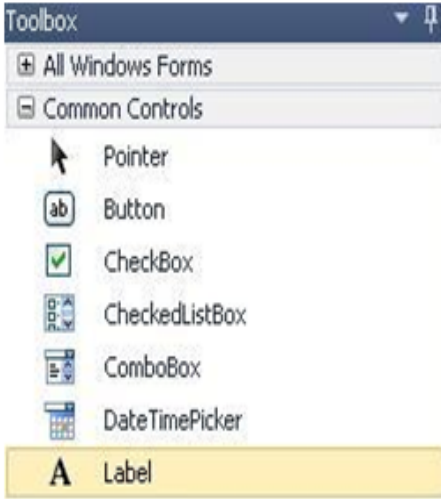
Formulário	Conteúdo Padrão		
	Propriedades	Conteúdo	Descrição
	Name	frmIMC	Nome do controle (frm)
	BackColor	Control	Alterar a cor de fundo do controle
	BackgroundImage	none	Inserir uma imagem de fundo no controle
	BackgroundImageLayout	tile	Layout de como mostrar a imagem
	ControlBox	True	retirar (False) botões de Minimizar, Maximizar e Fechar
	Enabled	True	Habilitado (True) ou Desabilitado (False)
	Font	Microsoft Sans Serif; 8,25pt	Fonte do texto, quando inserir controles
	ForeColor	ControlText	Cor do texto
	FormBorderStyle	Sizable	Tipos de bordas
	Icon	Icon	Ícone no canto superior esquerdo
	MaximizeBox	True	Botão de Maximizar o Form (False retira)
	MinimizeBox	True	Botão de Minimizar o Form (False retira)
	StartPosition	WindowsDefaultLocation	Local onde o form aparecerá quando executado
	Text	IMC - Índice de Massa Corporal	Texto que fica na parte superior do form (caption)
	WindowState	Normal	Na execução, o form vem Normal, Maximizado, Minimizado
Eventos			
	Load	Se dá na inicialização do form, antes de mostrar a sua execução.	

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

Botão		Conteúdo Padrão	
	Propriedades	Conteúdo	Descrição
	Nome BackColor BackgroundImage BackgroundImageLayout Enabled Font ForeColor Image ImageAlign TabIndex TabStop Text TextAlign	btnCalcular Control none tile True Microsoft Sans Serif; 8,25pt ControlText none MiddleCenter 0 True &Calcular MiddleCenter	Nome do controle (btn) Alterar a cor de fundo do controle Inserir uma imagem de fundo no controle Layout de como mostrar a imagem Habilitado (True) ou Desabilitado (False) Fonte do texto Cor do texto Inserir uma imagem no controle Alinhamento da imagem no controle Ordem de tabulação dos controles Pressionando o Tab, para ou não no controle (True) para Texto que fica dentro do controle, atalho (Alt+Letra) colocando o & na frente da letra. Alinhamento do texto
Eventos		Click	Executado quando em execução for dado um clique no mesmo

Label		Conteúdo Padrão	
	Propriedades	Conteúdo	Descrição
	Nome AutoSize BackColor BorderStyle Enabled Font ForeColor Image ImageAlign Text TextAlign	label1 True Control none True Microsoft Sans Serif; 8,25pt ControlText none MiddleCenter Peso: TopLeft	Nome do controle (btn) Tamanho definido pelo Text Alterar a cor de fundo do controle Inserir uma imagem de fundo no controle Habilitado (True) ou Desabilitado (False) Fonte do texto Cor do texto Inserir uma imagem no controle Alinhamento da imagem no controle Texto que fica dentro do controle. Alinhamento do texto
Eventos			

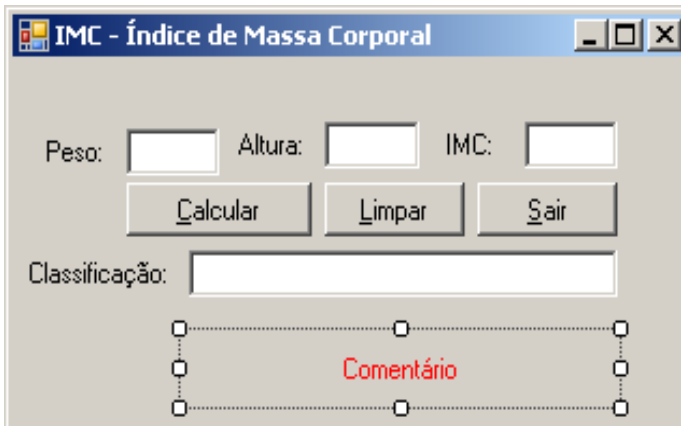
Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

textBox		Conteúdo Padrão	
	Propriedades	Conteúdo	Descrição
	Name	txtPeso	Nome do controle (btn)
	BackColor	Window	Alterar a cor de fundo do controle
	BorderStyle	Fixed3D	Estilo da borda
	CharacterCasing	Normal	Texto em caixa alta ou caixa baixa
	Enabled	True	Habilitado (True) ou Desabilitado (False)
	Font	Microsoft Sans Serif; 8,25pt	Fonte do texto
	ForeColor	WindowText	Cor do texto
	MaxLength	32767	Quantidade de caracteres que poderá ser digitado
	MultiLine	False	Uma ou várias linhas
	PassWordChar		Insira o caracter se for uma caixa para senha, ex: "*"
	ReadOnly	False	Se for True, tem acesso para copiar o conteúdo
	RightToLeft	No	Alinhamento do texto no controle
	ScrollBars	none	Sendo várias linhas poderá ter a barra de rolagem
	TabIndex	1	Ordem de tabulação dos controles
	TabStop	True	Pressionando o Tab, para ou não no controle (True) para
	Text		Texto que fica dentro do controle, aquilo que você irá digitar. Todo conteúdo desta caixa é uma string
	TextAlign	Left	Alinhamento do texto
	Eventos	Enter	Ocorre quando o objeto recebe o foco
		Leave	Ocorre quando o foco sai do objeto
		TextChanged	Executado quando for feita qualquer alteração na propriedade text

Vamos inserir então os componentes necessários para que nossa aplicação fique de acordo com o seguinte layout:



Controle	Nome	Propriedade	Atribuição
Label	label1	Text	Peso:
Label	label2	Text	Altura:
Label	label3	Text	IMC:
Label	label4	Text	Classificação:
Label	lblComentario	Text	Comentário

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

		AutoSize	False
		ForeColor	Red
		TextAlign	MiddleCenter
TextBox	txtPeso	Text	
TextBox	txtAltura	Text	
TextBox	txtIMC	Text	
TextBox	txtClassificacao	Text	
		TextAlign	Center
Button	btnCalcular	Text	&Calcular
Button	btnLimpar	Text	&Limpar
Button	btnSair	Text	&Sair

Até aqui, posicionamos os controles no **form** alteramos a **propriedade** Text de cada um deles, de modo a ficar com a aparência acima. **Propriedades** são características dos objetos e podem ser alteradas tanto em **design** como fizemos ou então via **código**.

Após termos alterado as propriedades dos componentes de nossa aplicação, podemos até executá-la, inserir dados nos controles onde são lidos o peso e a altura, pressionar os botões, etc. Porém nada acontecerá, pois não escrevemos código para efetuar qualquer tipo de processamento.

A partir de agora vamos ver como fazer nossa aplicação funcionar realmente. Para isso devemos ter uma noção básica do que são eventos e como tratá-los.

Eventos

- Cada objeto possui uma lista de eventos como já mostramos nas imagens acima os mais comuns.
- Para visualizar os eventos de determinado objeto → Selecione o objeto, na janela **Properties** clique na ícone **Events**.
- Não é necessário tratar todos os eventos, aliás, tratamos muito pouco, de acordo com o comportamento desejado da aplicação.

Como determinar que eventos trataremos?

Devemos primeiramente pensar: - Como a aplicação irá funcionar? Nesse layout simples que fizemos não temos muitas opções. Ele provavelmente funcionará da seguinte maneira:

1. O programa é iniciado
2. É exibida a tela principal para o usuário
3. O usuário informa o peso e a altura
4. Pressiona o botão Calcular
5. O sistema calcula e exibe o valor do IMC e a respectiva classificação
6. O usuário pode querer informar novos dados, limpando as informações anteriores através do pressionamento do botão limpar; ou
7. O usuário pode sair do sistema.

Vamos agora escrever o código para que nossa aplicação funcione.

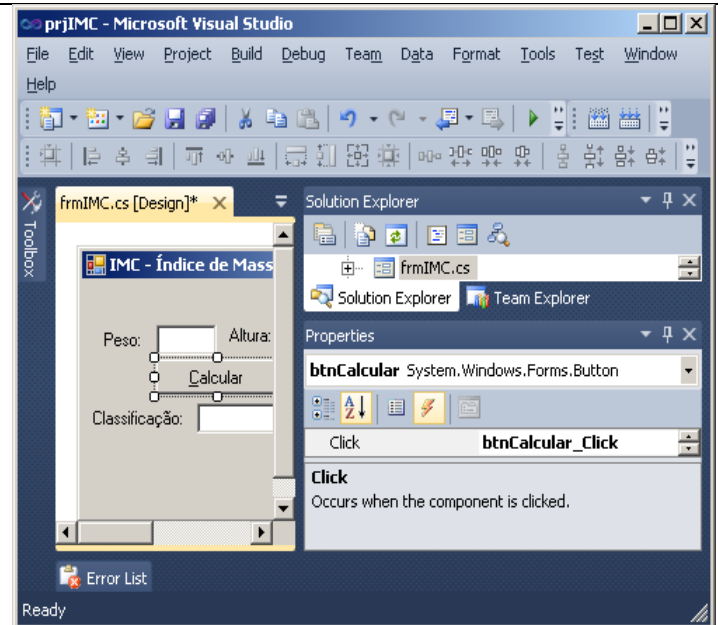
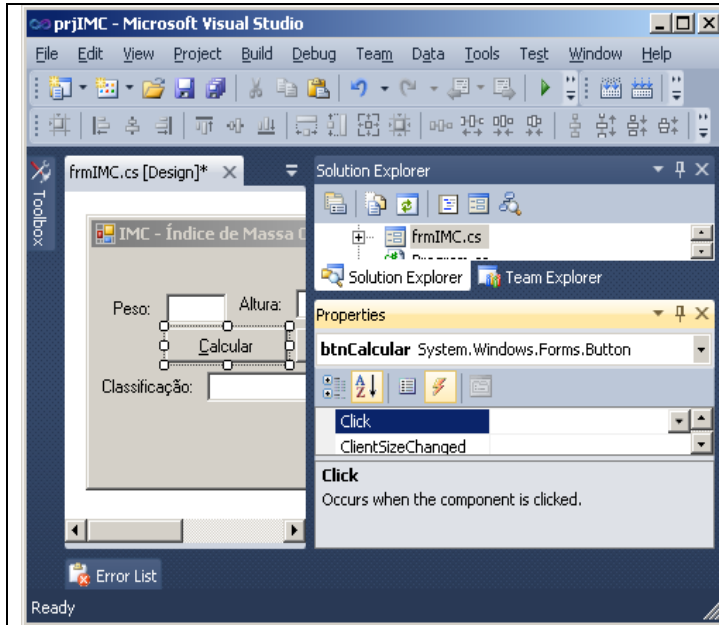
Dê um duplo clique no botão Calcular ou na janela **Properties** clique em **Events**.

Veja que será mostrado o evento default **Click**, do lado direito dê um duplo clique e veremos a área de código.

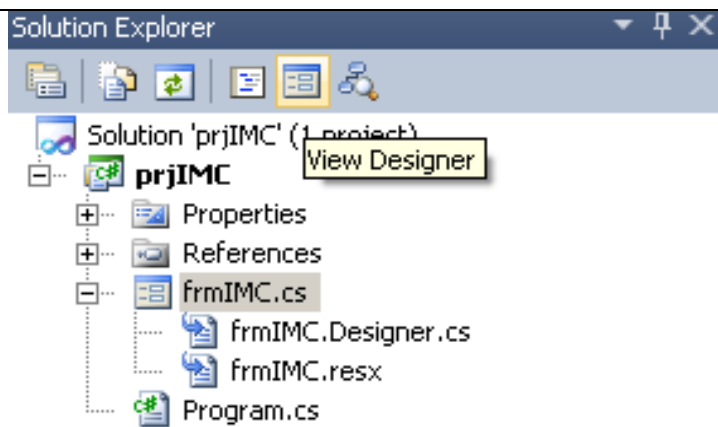
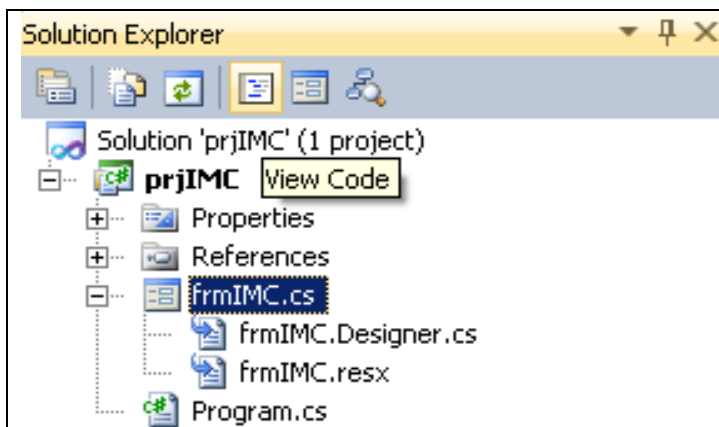
Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo



Na janela **Solution Explorer** podemos alternar entre a área de código e a área de design clicando nas ícones mostradas abaixo:



Abaixo temos a nossa área de código. Digitaremos o que fazer entre a chaves do evento **btn_Calcular_Click**.

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

```
frmIMC.cs [Design]*   frmIMC.cs*   Program.cs
prjIMC.frmIMC
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 prjIMC
{
    public partial class frmIMC : Form
    {
        public frmIMC()
        {
            InitializeComponent();
        }

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

Para fazermos o cálculo do IMC, temos que saber como é a formula e as restrições para classificação.

$IMC = \text{Peso} / (\text{Altura}^2)$

IMC	Classificação
$Imc < 20$	Abaixo do Peso
$20 > Imc < 25$	Peso Ideal
$Imc > 25$	Obeso

- Vamos definir algumas variáveis para melhor trabalhar na programação.
- As variáveis podem ser do tipo Local ou Global, depende de onde iremos declarar. Se declararmos dentro do evento, ela será do tipo local e se declararmos fora de qualquer evento, ela será do tipo global. Na imagem abaixo mostramos alguns tipos de dados.

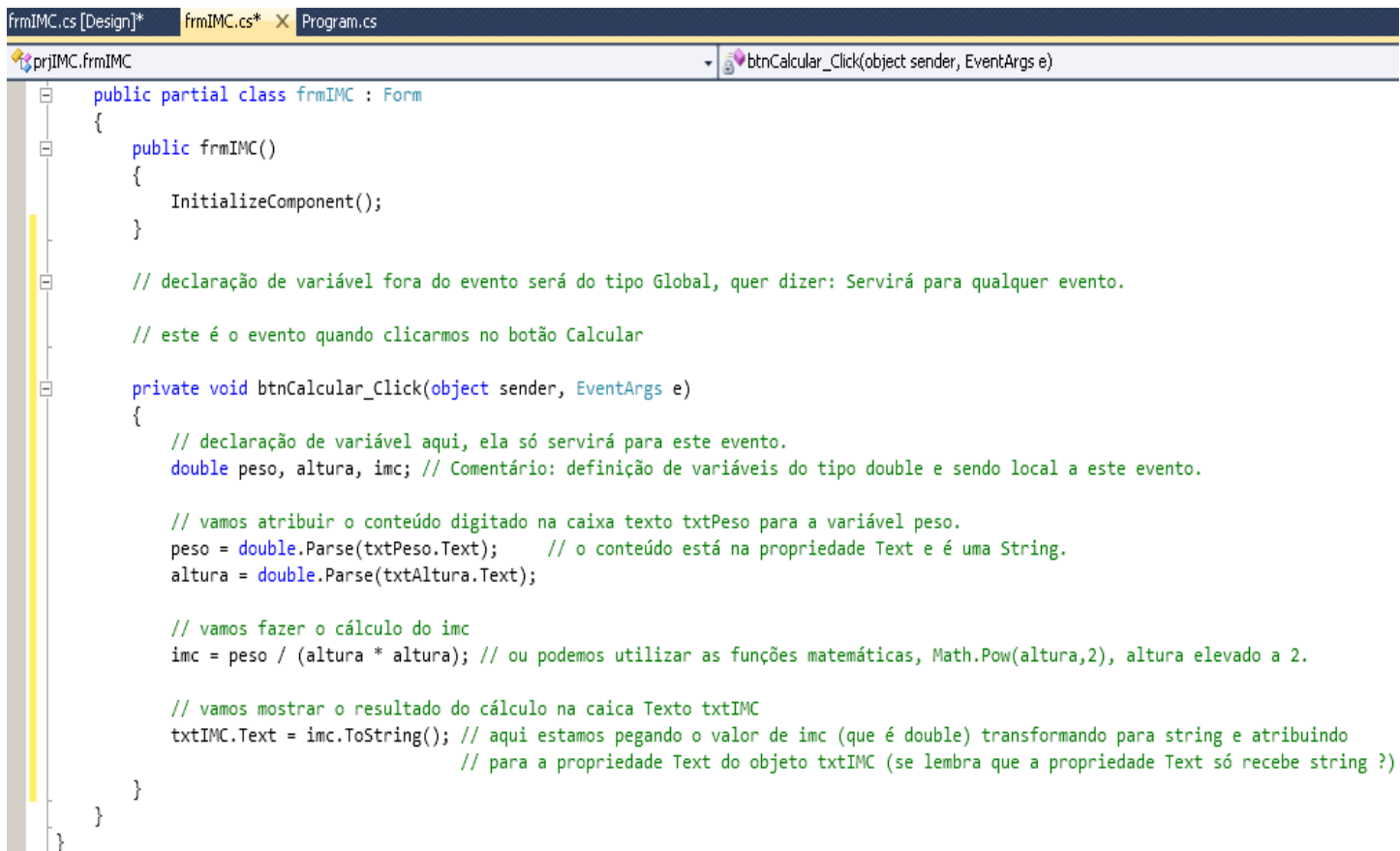
Tipo de Dado	Descrição	Tamanho em bits	Intervalo	Exemplo
int	número inteiro	32	-2147483648 até 2147483647	int idade; Idade = 50;
long	número inteiro	64	-9223372036854775808 até 9223372036854775807	long x; x=54L;
float	números reais	32	$\pm 3,4 * 10^{38}$	float nota; nota=5.5F
double	números reais	64	$\pm 1,7 * 10^{308}$	double x; x=0.125;
decimal	valores monetários	128	28 números significativos	decimal v; v=0.42M
string	seqüência de caracteres	16 bits por caracter		string n; n="Navio";
char	único caracter	16	0 a 32767	char Letra; Letra='A';
bool	booleano	8	true ou false	bool Resp; Resp=true;

- Podemos fazer comentários em nosso código colocando // antes do conteúdo. Veja imagens abaixo:

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo



```
frmIMC.cs [Design]*   frmIMC.cs*   Program.cs
prjIMC.frmIMC
btnCalcular_Click(object sender, EventArgs e)

public partial class frmIMC : Form
{
    public frmIMC()
    {
        InitializeComponent();
    }

    // declaração de variável fora do evento será do tipo Global, quer dizer: Servirá para qualquer evento.

    // este é o evento quando clicarmos no botão Calcular

    private void btnCalcular_Click(object sender, EventArgs e)
    {
        // declaração de variável aqui, ela só servirá para este evento.
        double peso, altura, imc; // Comentário: definição de variáveis do tipo double e sendo local a este evento.

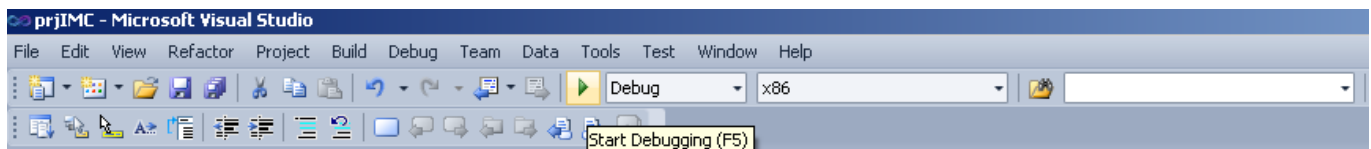
        // vamos atribuir o conteúdo digitado na caixa texto txtPeso para a variável peso.
        peso = double.Parse(txtPeso.Text); // o conteúdo está na propriedade Text e é uma String.
        altura = double.Parse(txtAltura.Text);

        // vamos fazer o cálculo do imc
        imc = peso / (altura * altura); // ou podemos utilizar as funções matemáticas, Math.Pow(altura,2), altura elevado a 2.

        // vamos mostrar o resultado do cálculo na caica Texto txtIMC
        txtIMC.Text = imc.ToString(); // aqui estamos pegando o valor de imc (que é double) transformando para string e atribuindo
                                     // para a propriedade Text do objeto txtIMC (se lembra que a propriedade Text só recebe string ?)
    }
}
```

Agora já podemos executar o nosso projeto e verificar algum resultado.

No Menu, clique na ícone ou pressione a tecla F5, será compilado e executado o projeto.



Digite algum valor na caixa Peso e Altura, em seguida clique no botão Calcular e veja o resultado.



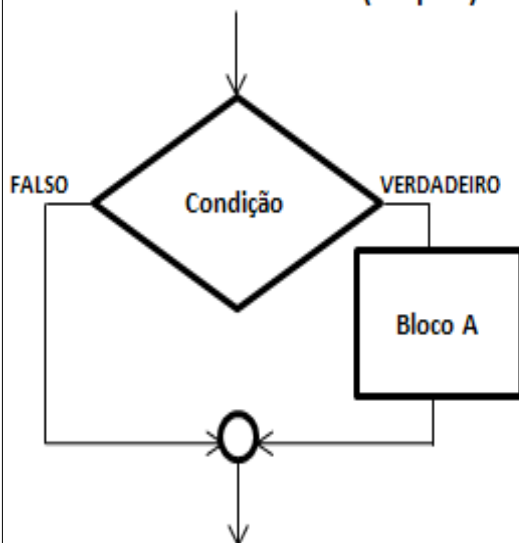
Vamos fazer a classificação a partir do resultado do IMC. Para isso temos que utilizar uma estrutura de decisão o (IF). Vamos ver como ele é estruturado.

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

Estrutura de Decisão if (simples)

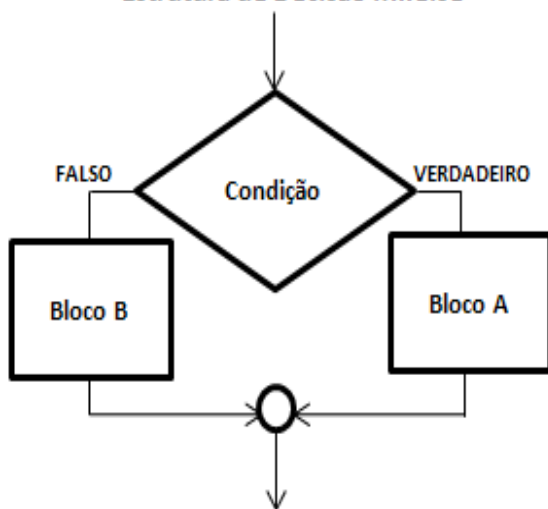


```
if (media >=7.0)                                uma instrução
    MessageBox.Show("Aluno Aprovado");           } executa o Bloco A
    MessageBox.Show("Média = " + media.ToString()); } sequência do programa
```

```
if (media >=7.0)
{
    MessageBox.Show("Aluno Aprovado");           } mais de uma instrução
    MessageBox.Show("Parabéns!");                 } executa o Bloco A
                                                    } entre chaves
}
    MessageBox.Show("Média = " + media.ToString()); } sequência do programa
```

Neste caso, se a Condição for VERDADEIRO executa os procedimentos do Bloco A se for FALSO, o programa segue sua execução linear, ignorando o Bloco A

Estrutura de Decisão if...else



```
if (media >=7.0)
    MessageBox.Show("Aluno Aprovado");           } uma instrução, Bloco A
else
    MessageBox.Show("Aluno Reprovado");           } uma instrução, Bloco B
    MessageBox.Show("Média = " + media.ToString()); } sequência do programa
```

```
if (media >=7.0)
{
    MessageBox.Show("Aluno Aprovado");           } mais de uma instrução
    MessageBox.Show("Parabéns!");                 } Bloco A
                                                    } entre chaves
}
else
{
    MessageBox.Show("Aluno Reprovado");           } mais de uma instrução
    MessageBox.Show("Fazer Novamente");           } Bloco B
                                                    } entre chaves
}
    MessageBox.Show("Média = " + media.ToString()); } sequência do programa
```

Neste caso, se a Condição for VERDADEIRO executa os procedimentos do Bloco A em seguida o programa segue sua execução linear. se for FALSO, executa os procedimentos do Bloco B e em seguida o programa segue sua execução linear.

Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

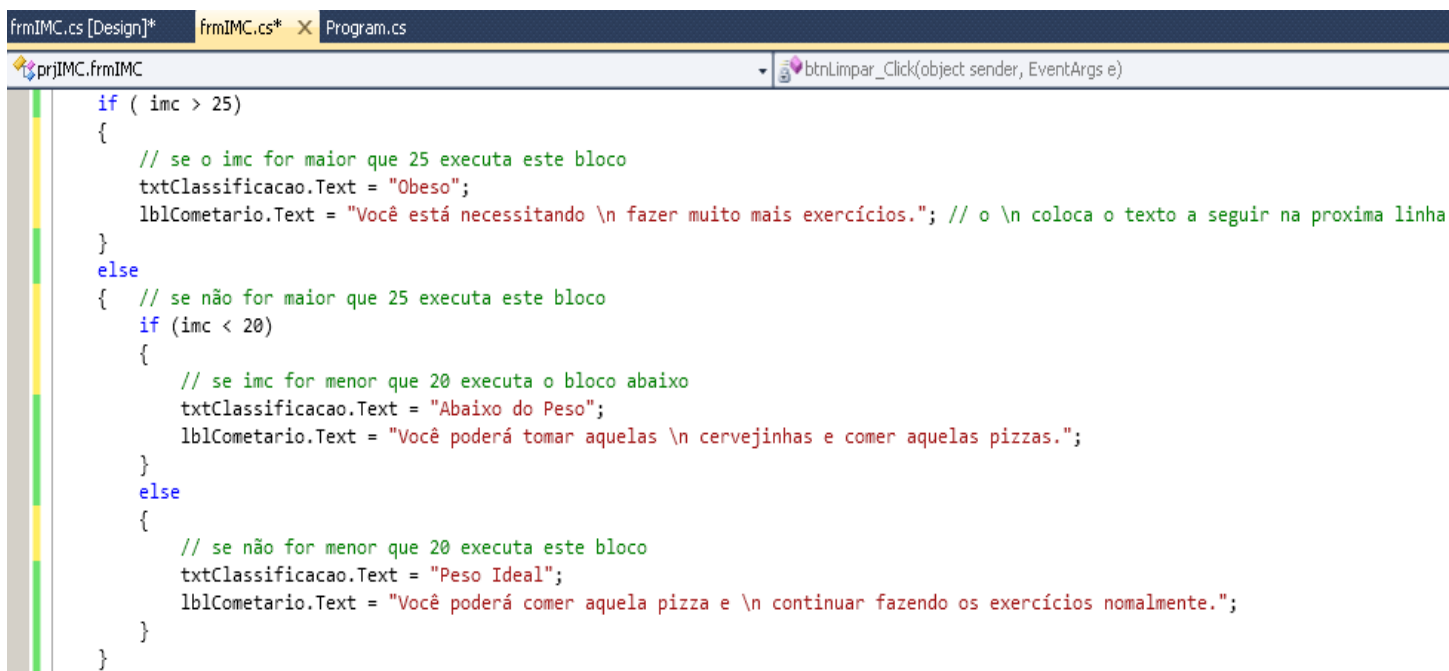
Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

```
// Exemplo: com uma instrução
if (media >= 7.0)
    MessageBox.Show("Aluno Aprovado"); // uma instrução
else
    MessageBox.Show("Aluno Reprovado"); // uma instrução

MessageBox.Show("Média = " + media.ToString()); // sequência do programa

// Exemplo: com mais de uma instrução
if (media >= 7.0)
{
    MessageBox.Show("Aluno Aprovado"); // mais de uma instrução, tem que estar entre chaves
    MessageBox.Show("Parabéns");
}
else
{
    MessageBox.Show("Aluno Reprovado"); // mais de uma instrução, tem que estar entre chaves
    MessageBox.Show("Ano que vem recupera");
}

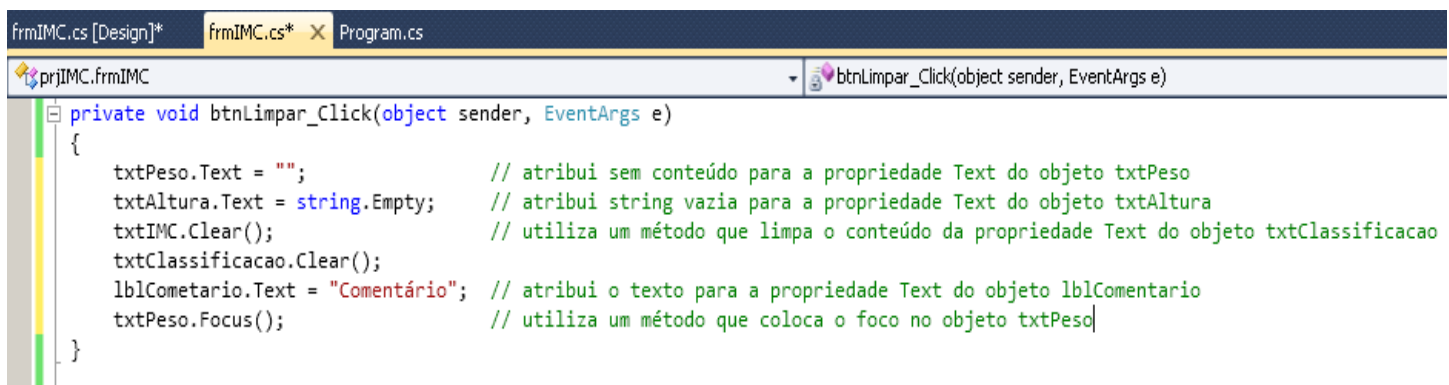
MessageBox.Show("Média = " + media.ToString()); // sequência do programa
```

The screenshot shows the Visual Studio IDE with three tabs: 'frmIMC.cs [Design]*', 'frmIMC.cs*', and 'Program.cs'. The 'Program.cs' tab is active, showing a C# code file named 'prjIMC.frmIMC'. The code implements a BMI calculation logic. It starts with an 'if' statement checking if 'imc' is greater than 25. If true, it sets 'txtClassificacao.Text' to 'Obeso' and 'lblComentario.Text' to 'Você está necessitando \n fazer muito mais exercícios.'. If false, it enters an 'else' block where it checks if 'imc' is less than 20. If true, it sets 'txtClassificacao.Text' to 'Abaixo do Peso' and 'lblComentario.Text' to 'Você poderá tomar aquelas \n cervejinhas e comer aquelas pizzas.'. If false, it sets 'txtClassificacao.Text' to 'Peso Ideal' and 'lblComentario.Text' to 'Você poderá comer aquela pizza e \n continuar fazendo os exercícios nomalmente.'. The code is color-coded with green for comments and red for strings.

```
frmIMC.cs [Design]*   frmIMC.cs*   Program.cs
prjIMC.frmIMC        btnLimpar_Click(object sender, EventArgs e)

if ( imc > 25)
{
    // se o imc for maior que 25 executa este bloco
    txtClassificacao.Text = "Obeso";
    lblComentario.Text = "Você está necessitando \n fazer muito mais exercícios."; // o \n coloca o texto a seguir na proxima linha
}
else
{
    // se não for maior que 25 executa este bloco
    if (imc < 20)
    {
        // se imc for menor que 20 executa o bloco abaixo
        txtClassificacao.Text = "Abaixo do Peso";
        lblComentario.Text = "Você poderá tomar aquelas \n cervejinhas e comer aquelas pizzas.";
    }
    else
    {
        // se não for menor que 20 executa este bloco
        txtClassificacao.Text = "Peso Ideal";
        lblComentario.Text = "Você poderá comer aquela pizza e \n continuar fazendo os exercícios nomalmente.";
    }
}
```

Agora vamos fazer o evento no botão Limpar. Dê um duplo clique no mesmo e vamos fazer o código.

The screenshot shows the Visual Studio IDE with the same three tabs as before. The 'Program.cs' tab is active, showing the 'prjIMC.frmIMC' code file. The code defines a private void method 'btnLimpar_Click' that takes 'object sender' and 'EventArgs e' as parameters. Inside the method, it resets the form by setting 'txtPeso.Text' to an empty string, 'txtAltura.Text' to 'string.Empty', clearing 'txtIMC' and 'txtClassificacao', setting 'lblComentario.Text' to 'Comentário', and calling 'txtPeso.Focus()' to place the focus on the weight input field. The code is color-coded with green for comments and red for strings.

```
frmIMC.cs [Design]*   frmIMC.cs*   Program.cs
prjIMC.frmIMC        btnLimpar_Click(object sender, EventArgs e)

private void btnLimpar_Click(object sender, EventArgs e)
{
    txtPeso.Text = ""; // atribui sem conteúdo para a propriedade Text do objeto txtPeso
    txtAltura.Text = string.Empty; // atribui string vazia para a propriedade Text do objeto txtAltura
    txtIMC.Clear(); // utiliza um método que limpa o conteúdo da propriedade Text do objeto txtClassificacao
    txtClassificacao.Clear();
    lblComentario.Text = "Comentário"; // atribui o texto para a propriedade Text do objeto lblComentario
    txtPeso.Focus(); // utiliza um método que coloca o foco no objeto txtPeso
}
```

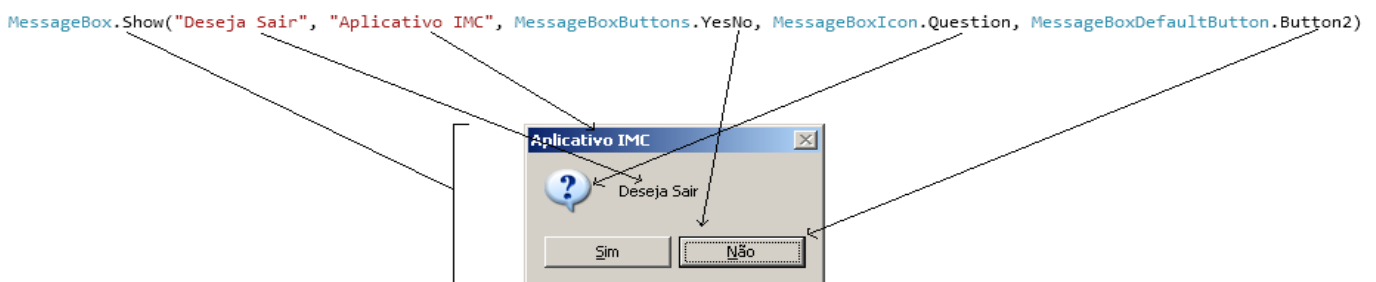
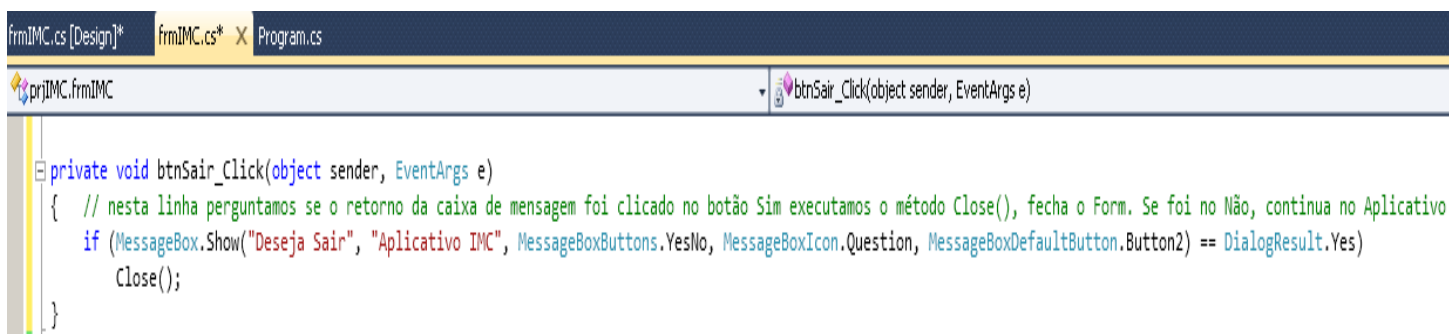
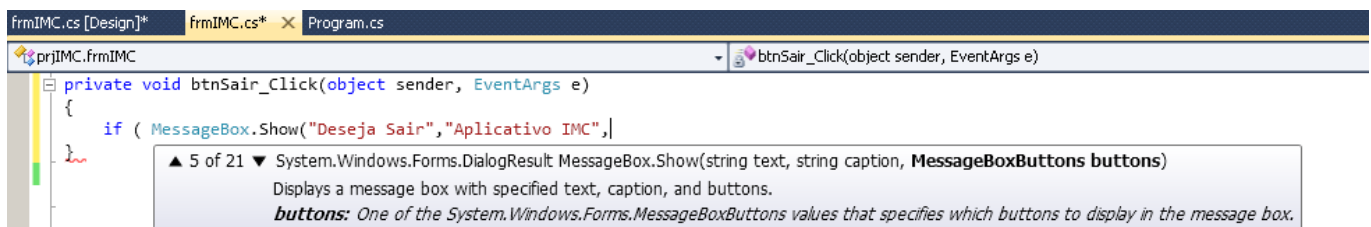
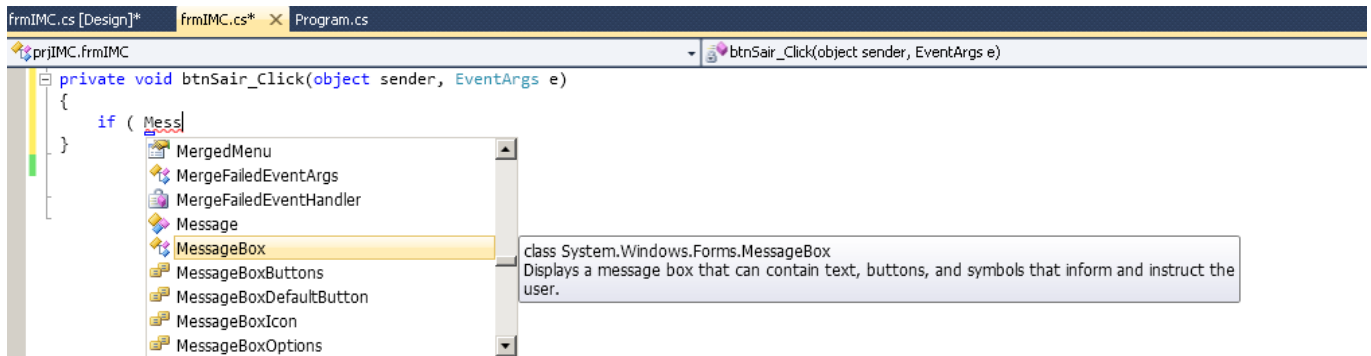
Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

No botão Sair também vai programar no evento Click. Dê um duplo clique no botão para elaborar o evento.

A seguir enquanto iremos codificando o evento do botão Sair, iremos explicar sobre o **Intellisense**, isto quer dizer que, quando digitamos algo na área de código ele mostra os objetos, propriedades, métodos, ou como podemos utilizar o método e quais parâmetros temos que informar. Se o mesmo não aparecer, é só segurar a tecla Ctrl e pressionar Barra de Espaço.



Mais algumas dicas úteis para sua área de código:

```
// comentário na linha
/* bloco de comentário
   bloco de comentário
*/
```

```
Math.Pow(variável,2) // variável elevado a potência 2
```

```
Math.Sqrt( variável ) // variável, raiz quadrada
```

```
Math.PI // valor do PI=3,14159265358979
```


Apostila - Microsoft Visual Studio – C#

C# - Aplicações para Windows

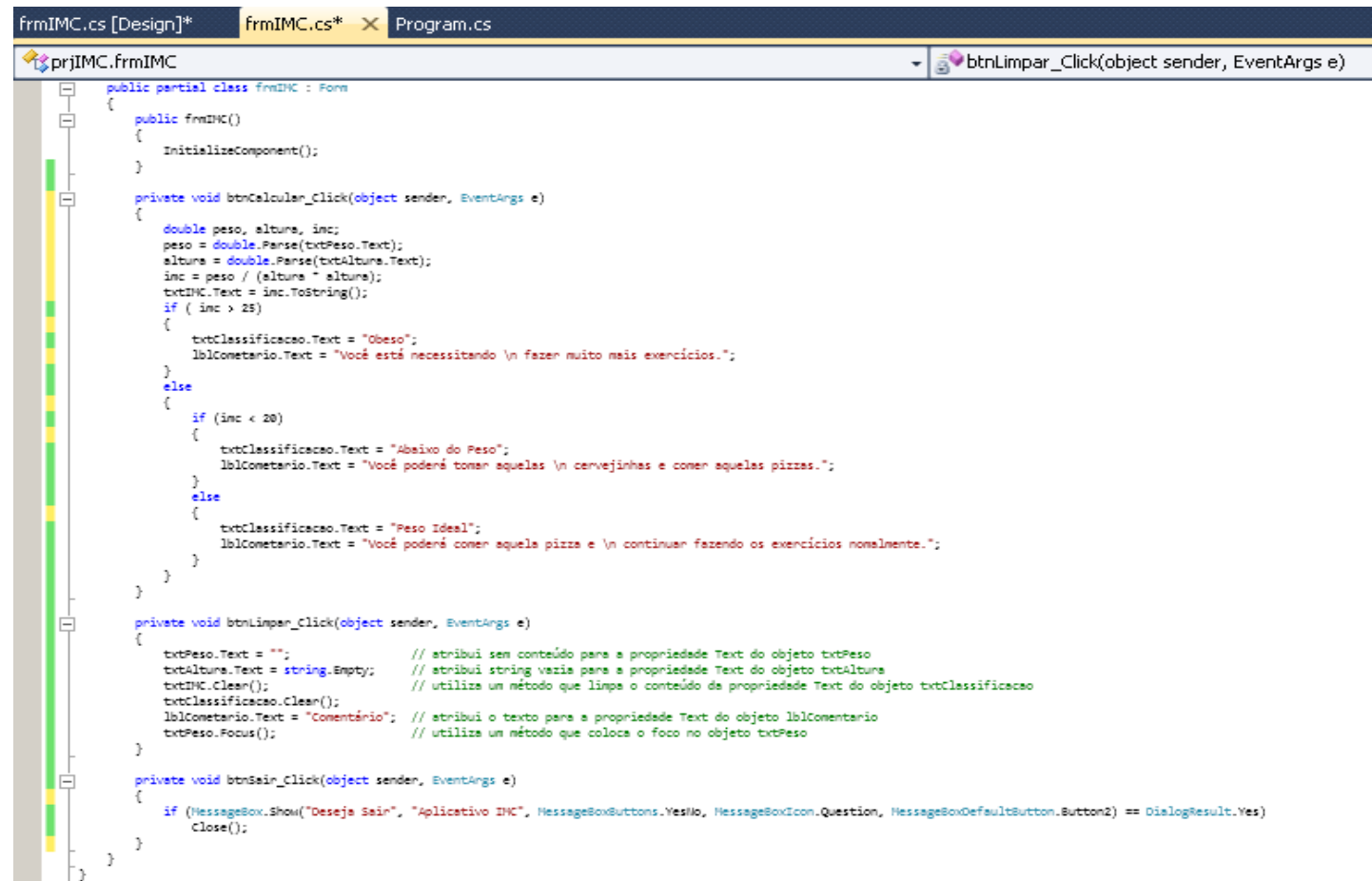
Prof. Edson Aparecido Leguth Prof. Carlos Jair Coletto Prof. Marcelo Pereira Bergamaschi Prof. Maurício Neves Asenjo

#region Evento do botão Calcular

.. evento, código, texto, comentários...

#endregion

Veja como ficou o nosso código: Fizemos de propósito e reduzimos o tamanho da font para ser mostrado em uma só imagem.



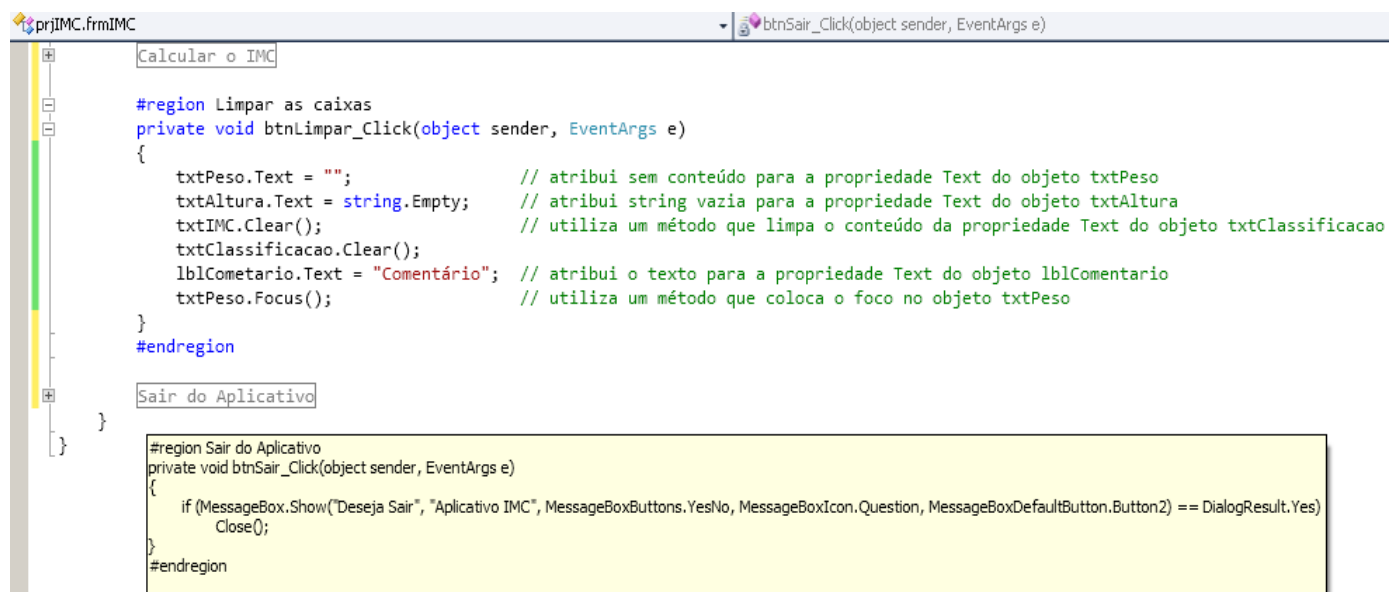
```
public partial class frmIMC : Form
{
    public frmIMC()
    {
        InitializeComponent();
    }

    private void btnCalcular_Click(object sender, EventArgs e)
    {
        double peso, altura, imc;
        peso = double.Parse(txtPeso.Text);
        altura = double.Parse(txtAltura.Text);
        imc = peso / (altura * altura);
        txtIMC.Text = imc.ToString();
        if (imc > 25)
        {
            txtClassificacao.Text = "Obeso";
            lblComentario.Text = "Você está necessitando \n fazer muito mais exercícios.";
        }
        else
        {
            if (imc < 20)
            {
                txtClassificacao.Text = "Abaixo do Peso";
                lblComentario.Text = "Você poderá tomar aquelas \n cervejinhas e comer aquelas pizzas.";
            }
            else
            {
                txtClassificacao.Text = "Peso Ideal";
                lblComentario.Text = "Você poderá comer aquela pizza e \n continuar fazendo os exercícios normalmente.";
            }
        }
    }

    private void btnLimpar_Click(object sender, EventArgs e)
    {
        txtPeso.Text = "";
        txtAltura.Text = string.Empty;
        txtIMC.Clear();
        txtClassificacao.Clear();
        lblComentario.Text = "Comentário";
        txtPeso.Focus();
    }

    private void btnSair_Click(object sender, EventArgs e)
    {
        if (MessageBox.Show("Deseja Sair", "Aplicativo IMC", MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2) == DialogResult.Yes)
        {
            Close();
        }
    }
}
```

Agora vamos utilizar **#region** e **#endregion** para melhor podermos organizar e visualizar nosso código.



```
#region Limpar as caixas
private void btnLimpar_Click(object sender, EventArgs e)
{
    txtPeso.Text = "";
    txtAltura.Text = string.Empty;
    txtIMC.Clear();
    txtClassificacao.Clear();
    lblComentario.Text = "Comentário";
    txtPeso.Focus();
}
#endregion

Sair do Aplicativo

#region Sair do Aplicativo
private void btnSair_Click(object sender, EventArgs e)
{
    if (MessageBox.Show("Deseja Sair", "Aplicativo IMC", MessageBoxButtons.YesNo, MessageBoxIcon.Question, MessageBoxDefaultButton.Button2) == DialogResult.Yes)
    {
        Close();
    }
}
#endregion

Calcular o IMC

private void btnCalcular_Click(object sender, EventArgs e)
{
    double peso, altura, imc;
    peso = double.Parse(txtPeso.Text);
    altura = double.Parse(txtAltura.Text);
    imc = peso / (altura * altura);
    txtIMC.Text = imc.ToString();
    if (imc > 25)
    {
        txtClassificacao.Text = "Obeso";
        lblComentario.Text = "Você está necessitando \n fazer muito mais exercícios.";
    }
    else
    {
        if (imc < 20)
        {
            txtClassificacao.Text = "Abaixo do Peso";
            lblComentario.Text = "Você poderá tomar aquelas \n cervejinhas e comer aquelas pizzas.";
        }
        else
        {
            txtClassificacao.Text = "Peso Ideal";
            lblComentario.Text = "Você poderá comer aquela pizza e \n continuar fazendo os exercícios normalmente.";
        }
    }
}
```