

Microsoft Visual Studio 2010 – C#

Apostila desenvolvida pelos Professores **Ricardo Santos de Jesus** e **Rovilson de Freitas**, para as Disciplinas de Desenvolvimento de Software I e II, nas Etecs de Francisco Morato e Dr. Emilio Hernandez Aguilar (Franco da Rocha).

Sumário

Microsoft Visual Studio 2010 C# – Instalação	4
Iniciando o programa:	12
Tela Inicial.....	14
Começando um Novo Projeto:	15
Primeiro Projeto	22
Caixa de Ferramentas	23
Principais Ferramentas	24
TextBox.....	24
Label	25
Principais propriedades da ferramenta Label:.....	25
Ferramenta Button	26
Ferramenta Radiobutton	27
Exemplo de Formulário	28
Como Declarar as variáveis?	29
Como Atribuir:	29
1º Diretamente na declaração:.....	29
2º Após a declaração:	29
Operadores.....	30
Estruturas de Decisão.....	31
Case (múltiplas decisões).....	32
Comando para tratamento de Erro	33
Laços de Repetição	34
Criação de Procedimentos e Funções no C#	35
Procedimentos	35
Funções	35
Calculando a área do triângulo	36
Código final do Exercício:.....	39
Outra Forma de fazer o mesmo código:	40
Exemplo de Código com Tomada de Decisão (IF – ELSE) e Controle de Erros....	41
Exemplo do laço de repetição For, utilizando uma ListBox	42
Exemplo do laço While.....	45
Exemplo do Comando Switch Case	46
Exemplo de Procedimento e Funções.....	49
Procedimento Limpar.	49
Funções para o cálculo	50
Trabalhando com formulários MDI e Menus.....	53
Programando os menus.....	55
Outros comandos para trabalhar com formulários	58
Para arranjo de vários formulários:.....	58
Para fechar vários forms de uma vez:	58
Exibir em um menu os formulários filhos.....	59
Propostas de Exercícios.....	61
Referências Bibliográficas	66

Microsoft Visual Studio 2010 C# - Instalação

Normalmente, ao colocar o CD aparecerá a seguinte tela:



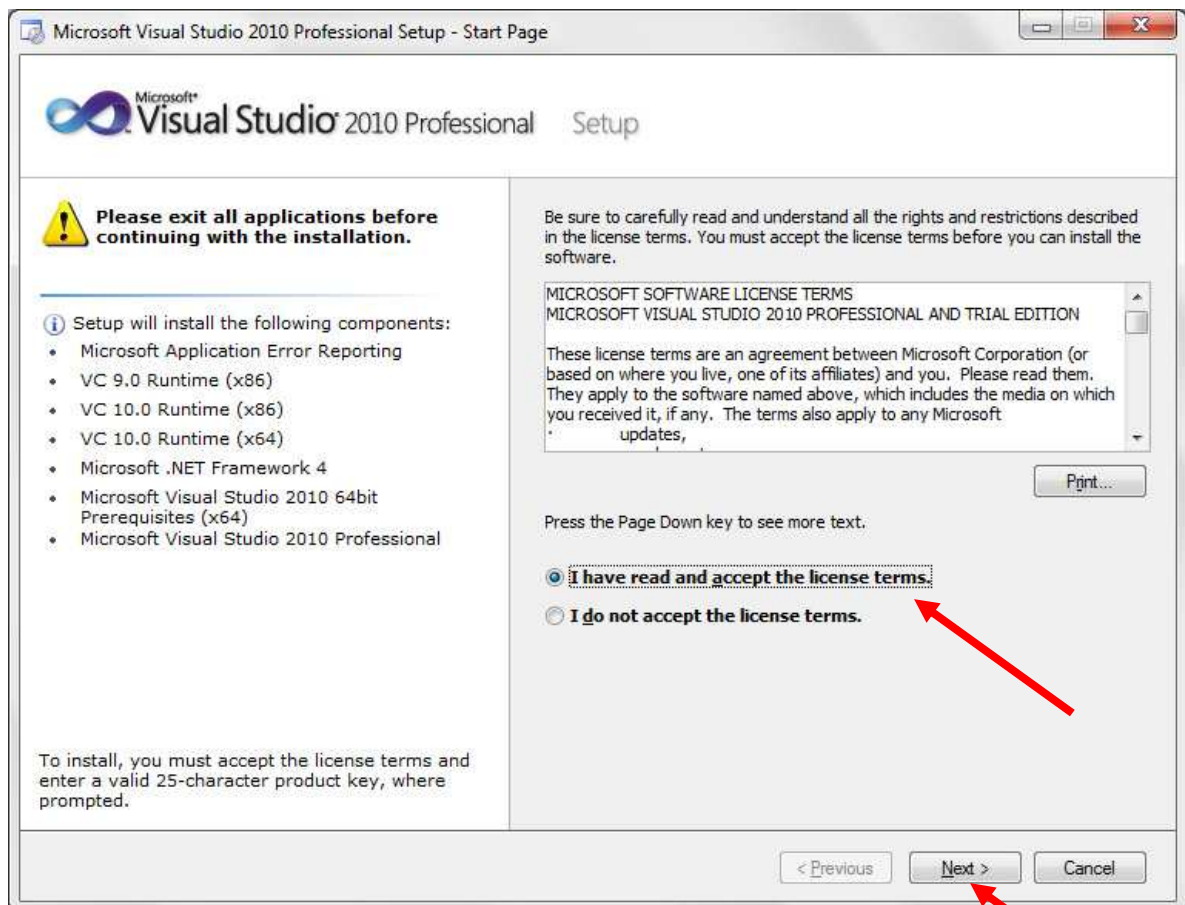
Clique na primeira opção: **Install Microsoft Visual Studio 2010**

Em seguida aparecerão as seguintes telas:

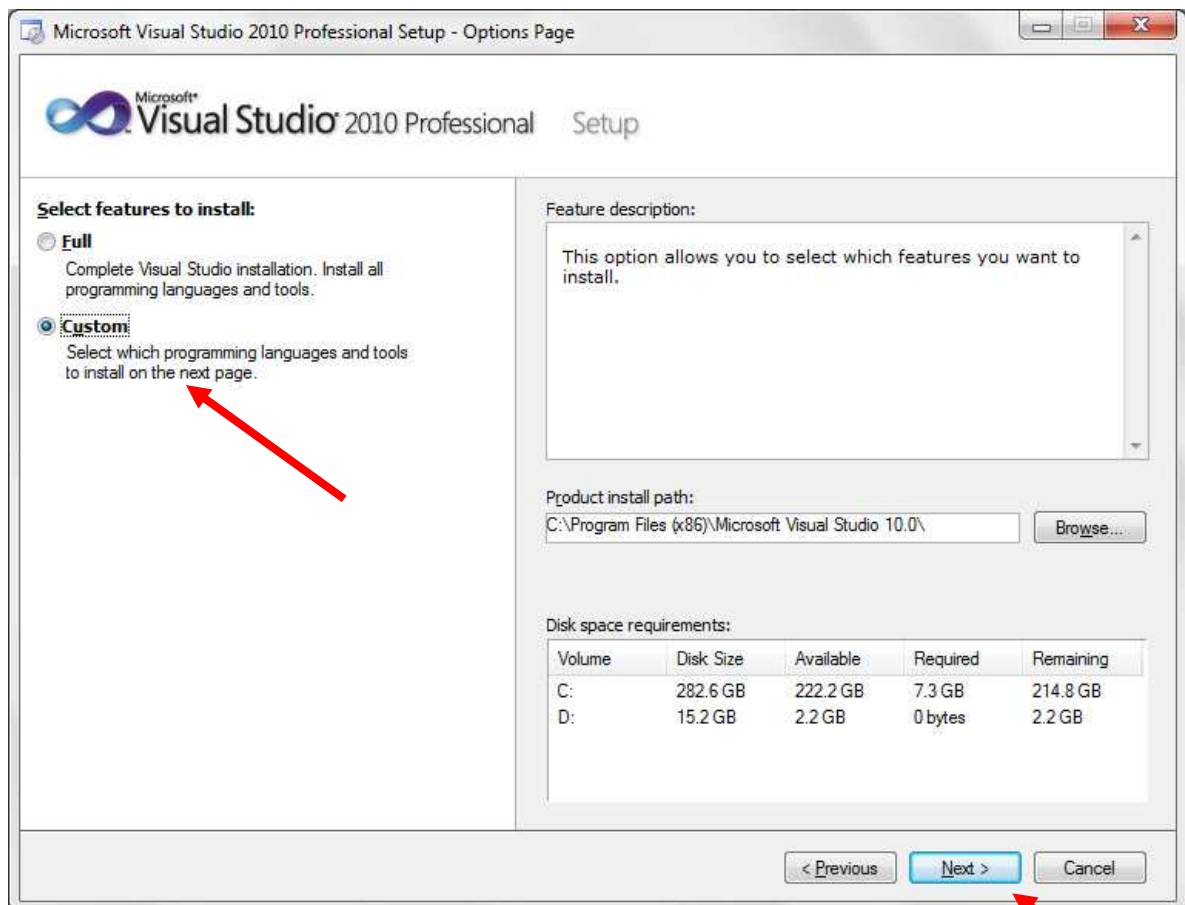




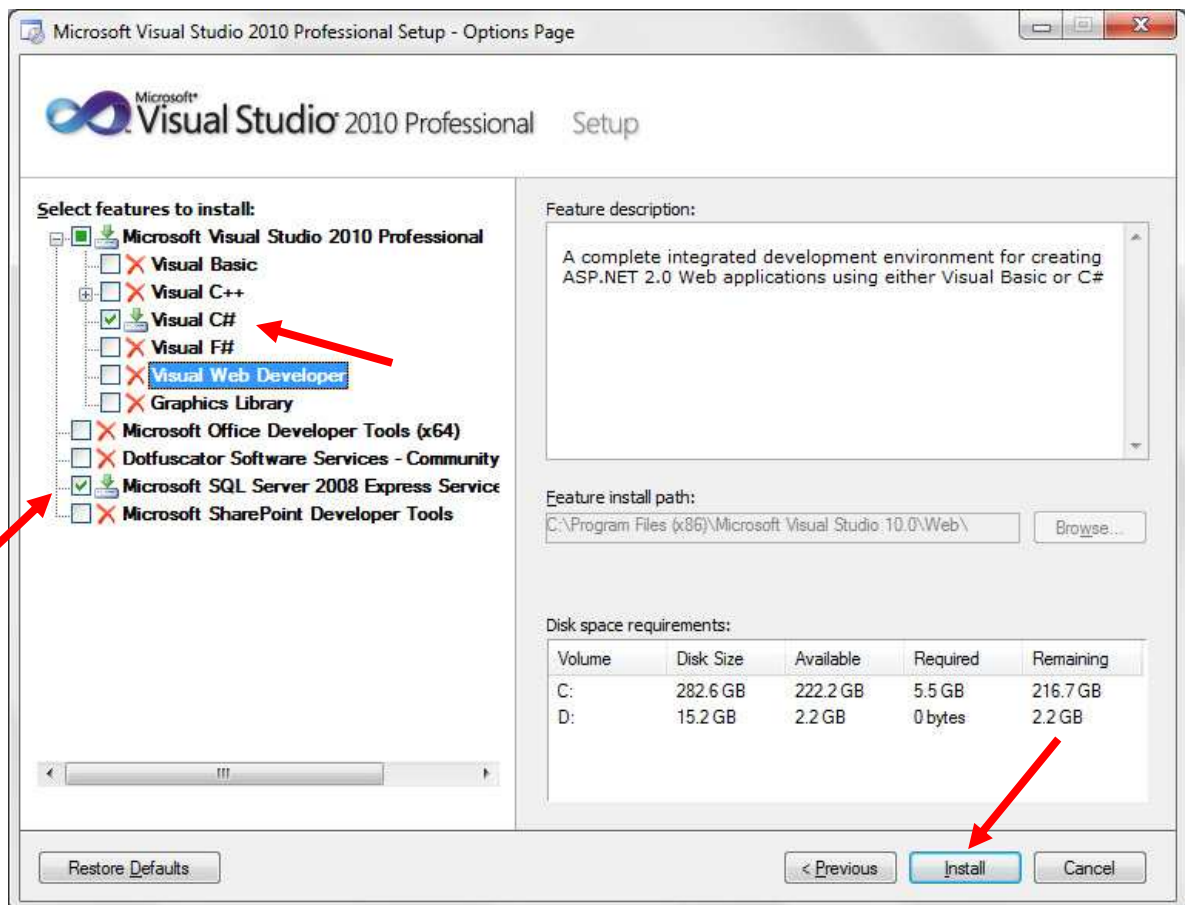
Clique em Next



Nessa tela, marcar a opção I have read and accept the licence terms, opção onde você aceita e concorda com os termos de instalação. Depois, Next

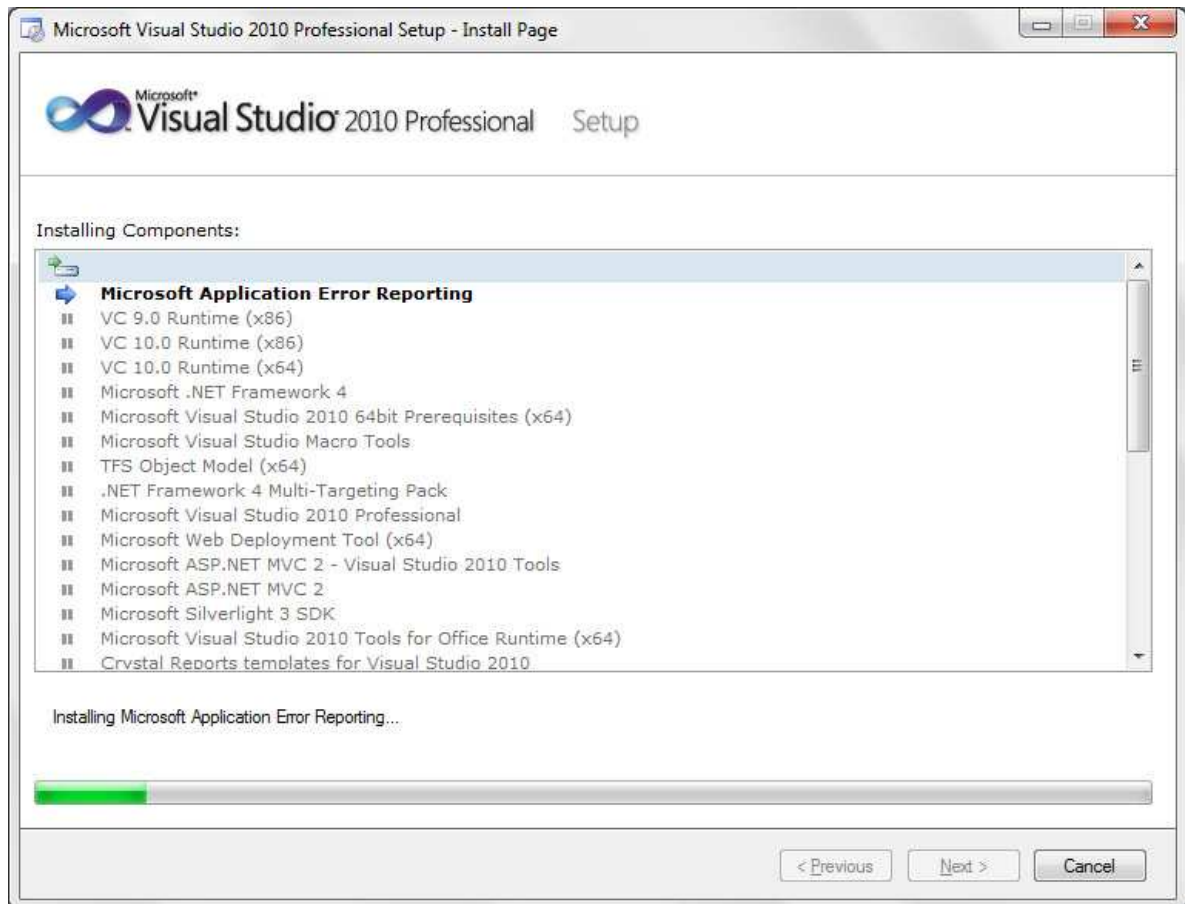


Essa é a tela mais importante. Deverá ser marcada a opção **Custom**, para que possamos escolher os itens a serem instalados. Não escolha a opção Full, pois serão instalados itens desnecessários que podem prejudicar o funcionamento do programa.

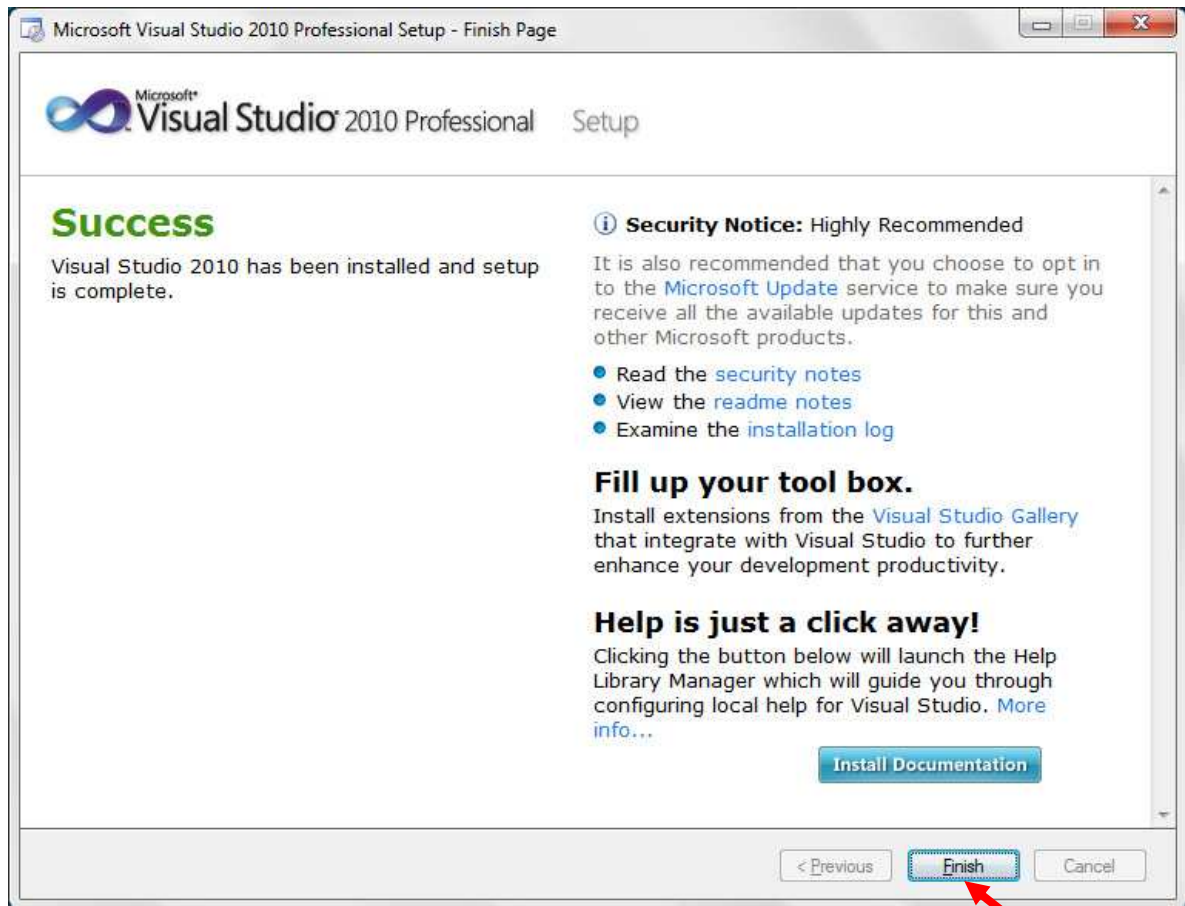


Desmarque todas as opções, marcando apenas Visual C# e Microsoft SQL Server 2008 Express Service. Caso seja necessária a instalação de alguma opção desmarcada, basta colocar o CD e recomeçar o processo, marcando o que se deseja. Depois, só clicar em Instal.

A tela a seguir, mostra o processo de instalação, apenas aguarde o término da instalação:



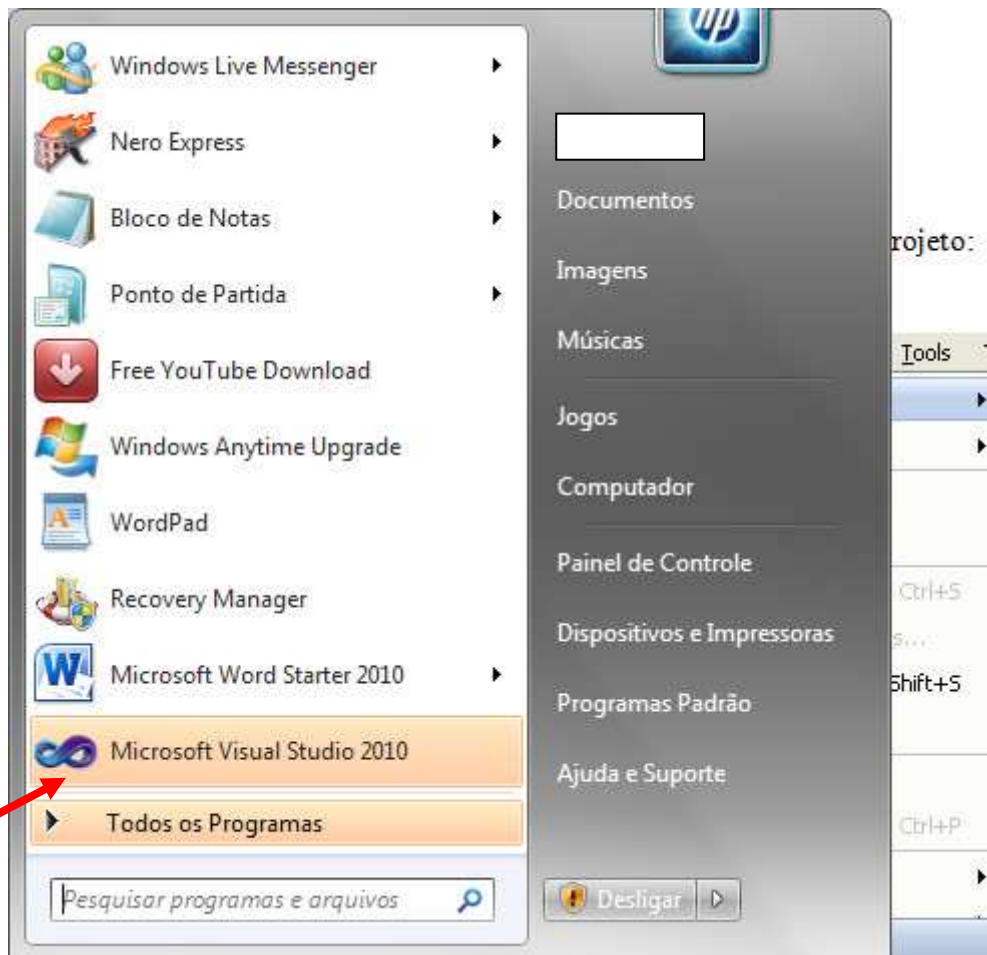
A tela a seguir, mostra que você instalou o programa com sucesso:



Microsoft Visual Studio 2010

C#

Iniciando o programa:

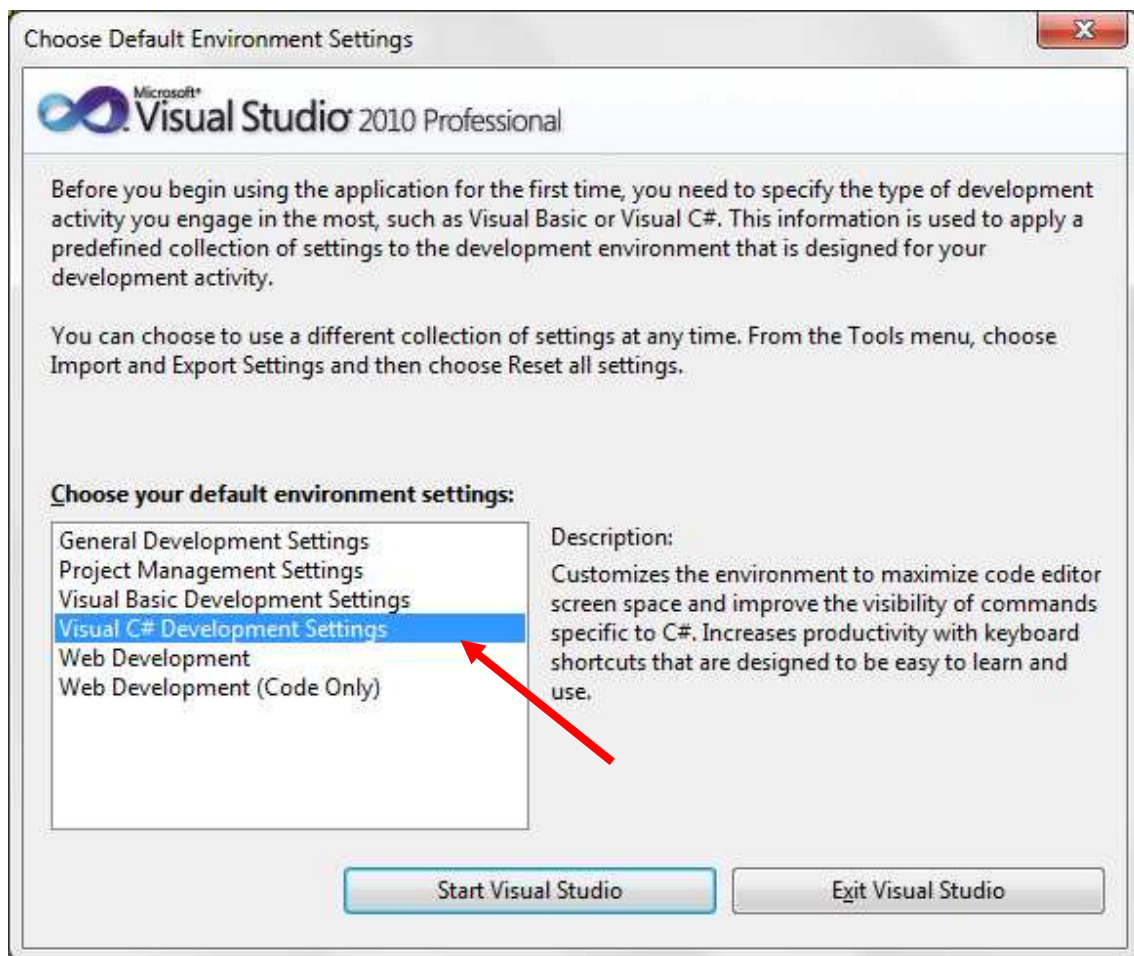


Ou:

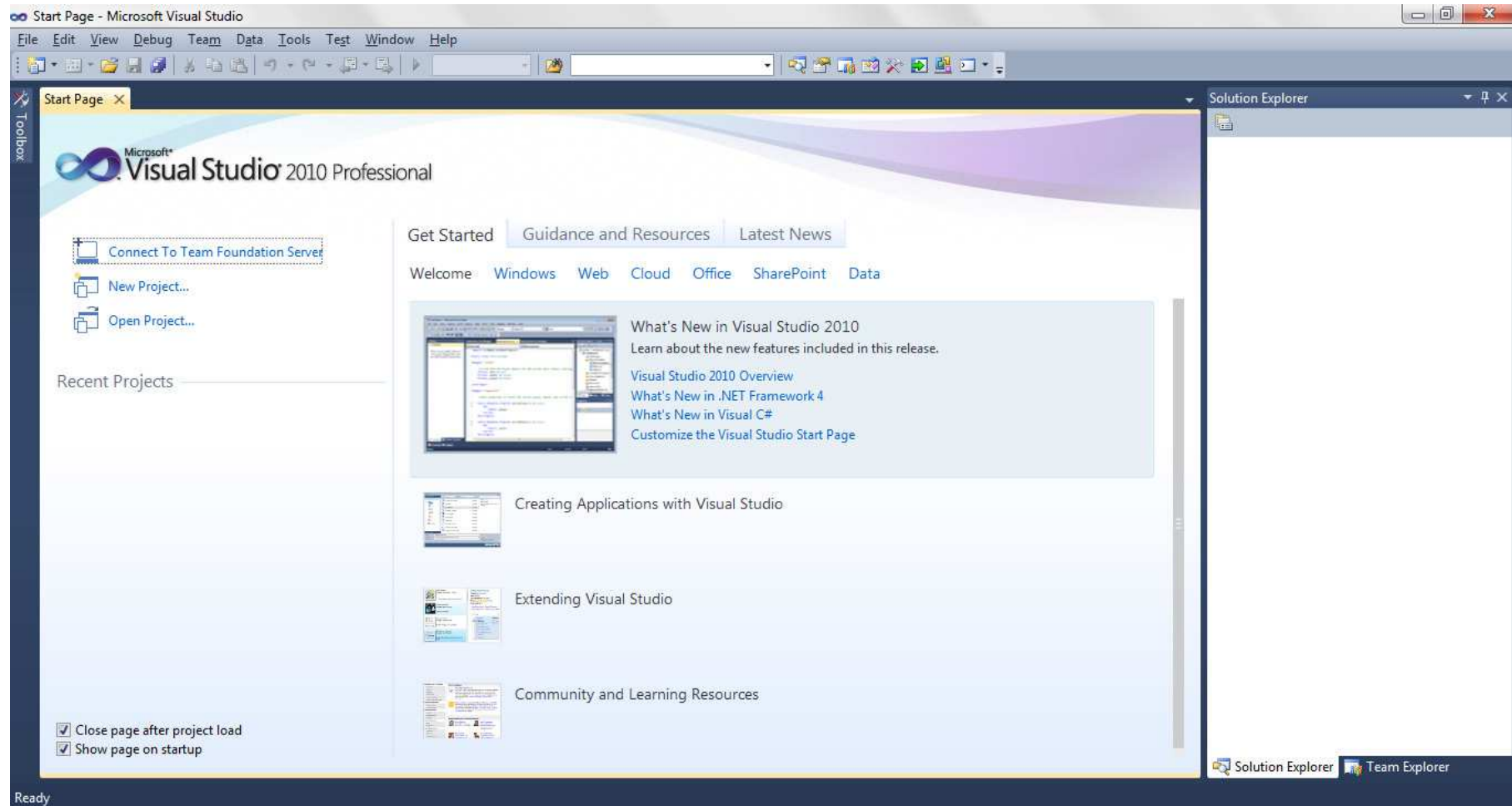
Clique em Todos os Programas e procure:



Caso apareça a seguinte janela, Escolha a opção selecionada e depois Start Visual Studio:

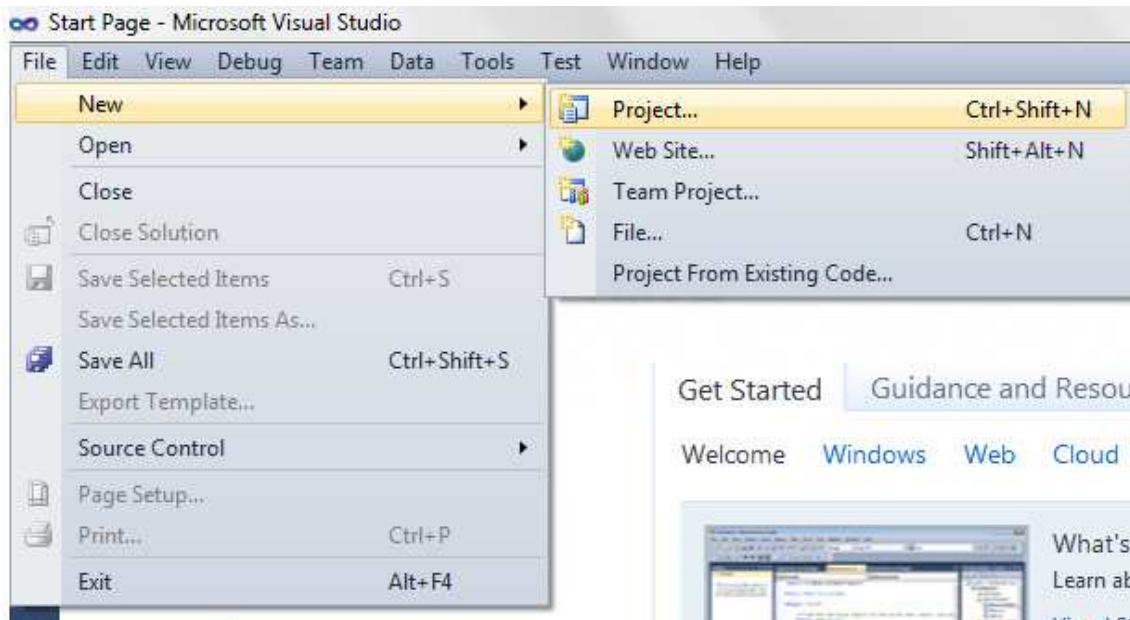


Tela Inicial

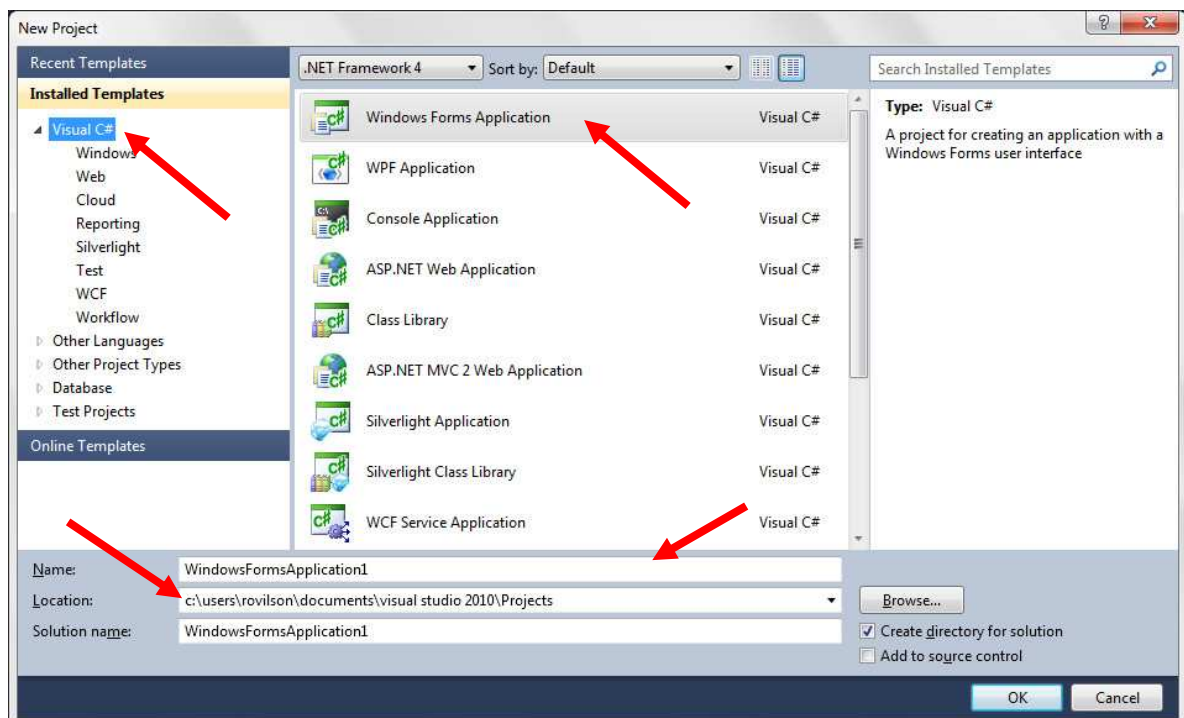


Começando um Novo Projeto:

Menu File, New Project:



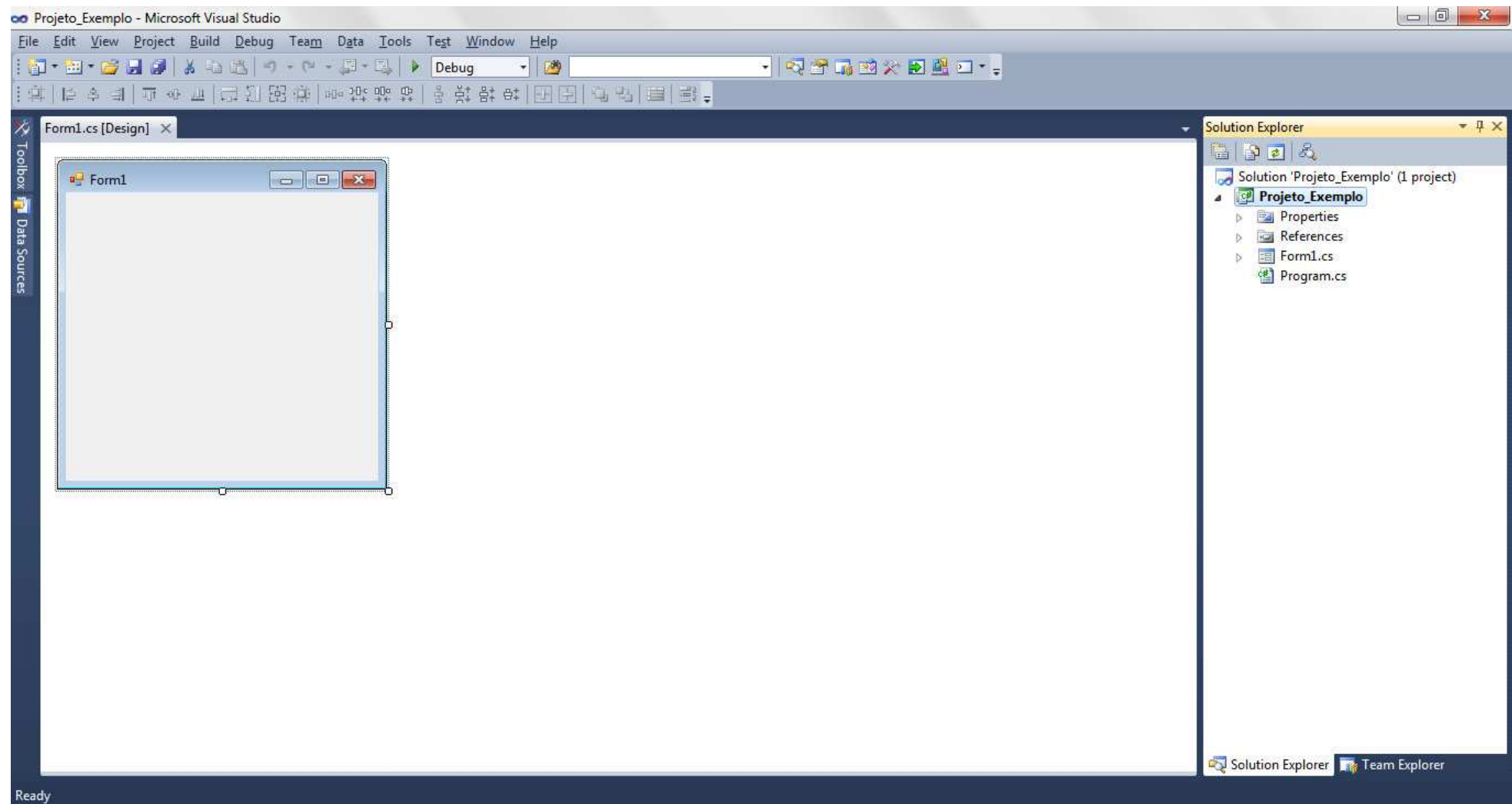
Na lista a esquerda, selecione a opção Visual C#. Depois, escolha na lista central a opção Windows Forms Application



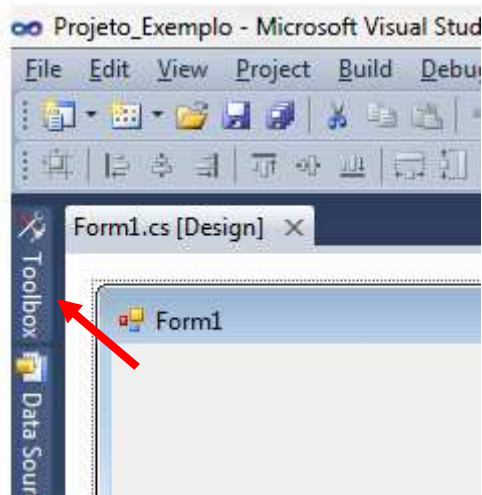
Na opção Name, escolha o nome do projeto. Em Location, escolha a pasta onde seu projeto será salvo. Uma pasta será criada automaticamente no local escolhido.

Depois, clique em OK.

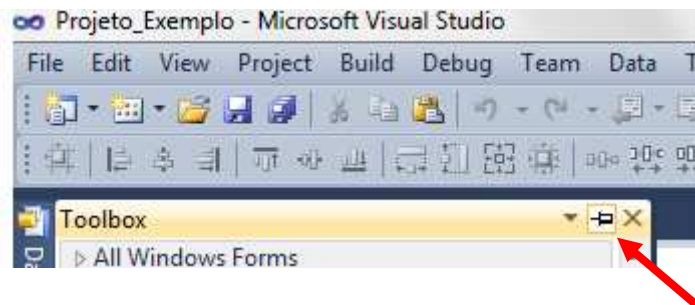
A tela será a seguinte:



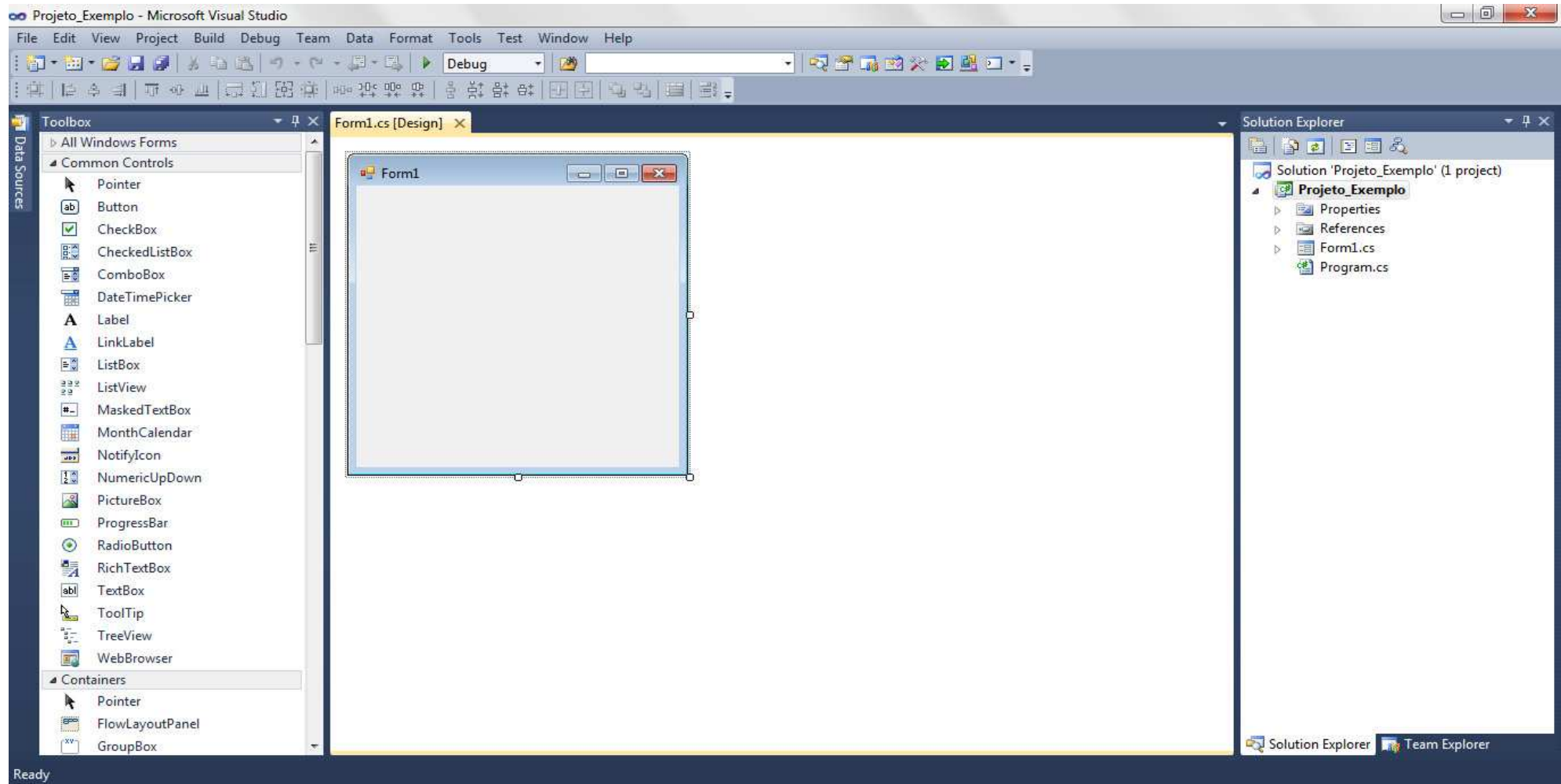
1ºPasso: Visualizar a Caixa de Ferramentas. Para isso, clique em Toolbox:



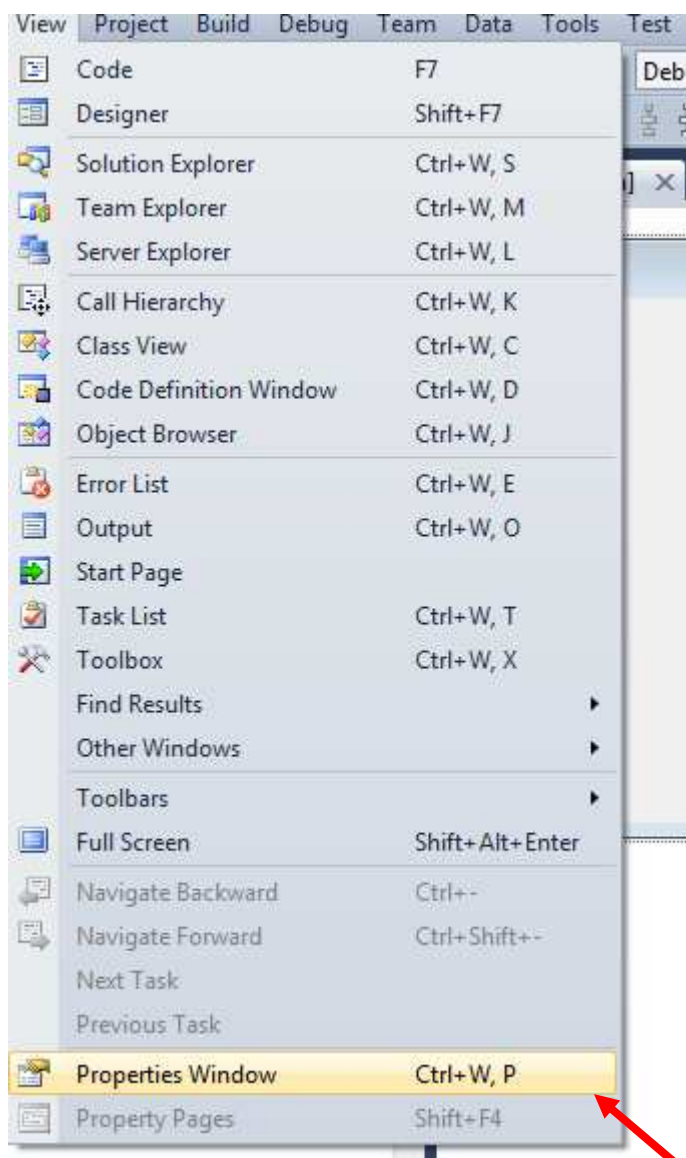
Para que a janela fique disponível sempre, clique no botão Auto Hide :

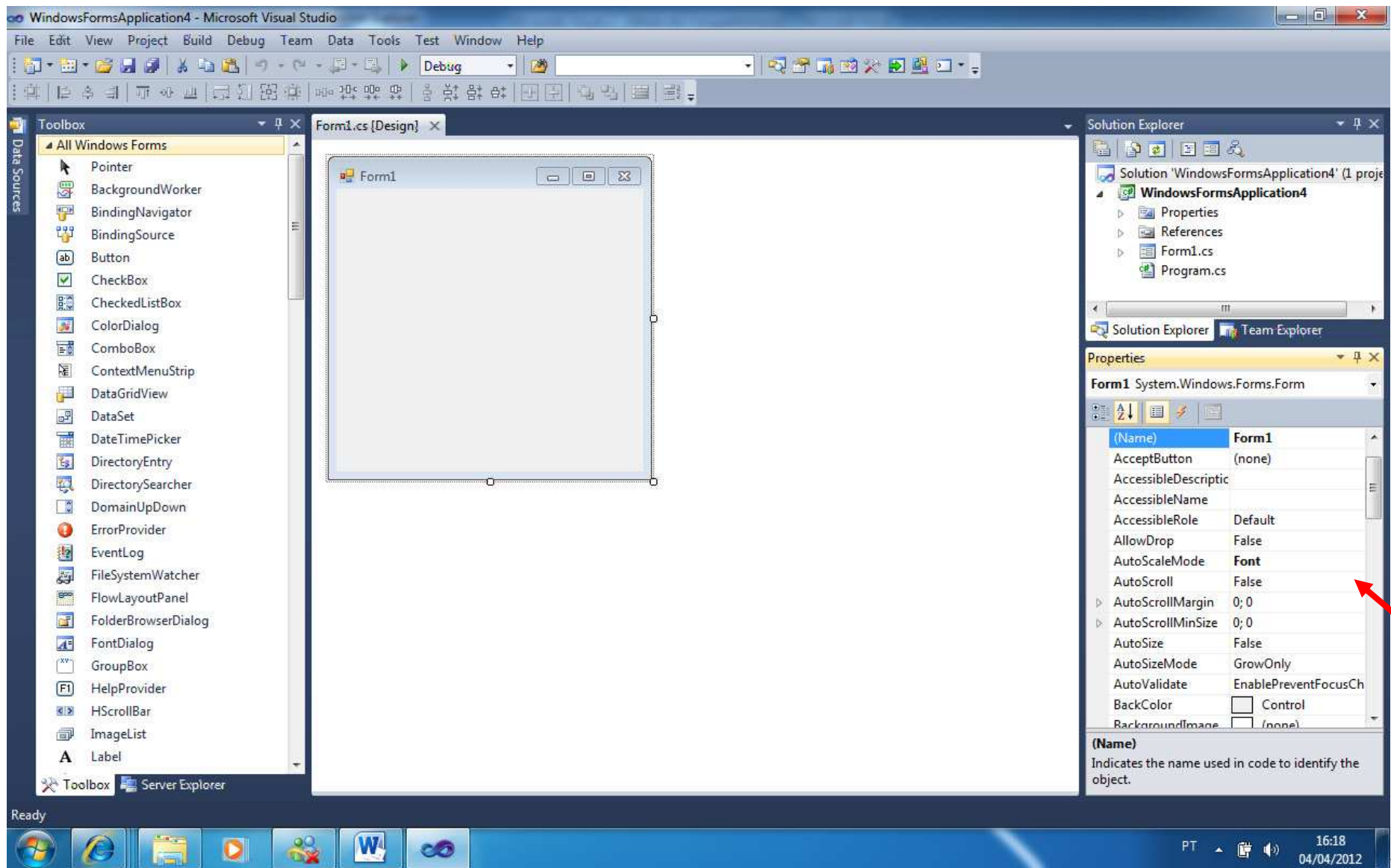


A janela ficará dessa forma:



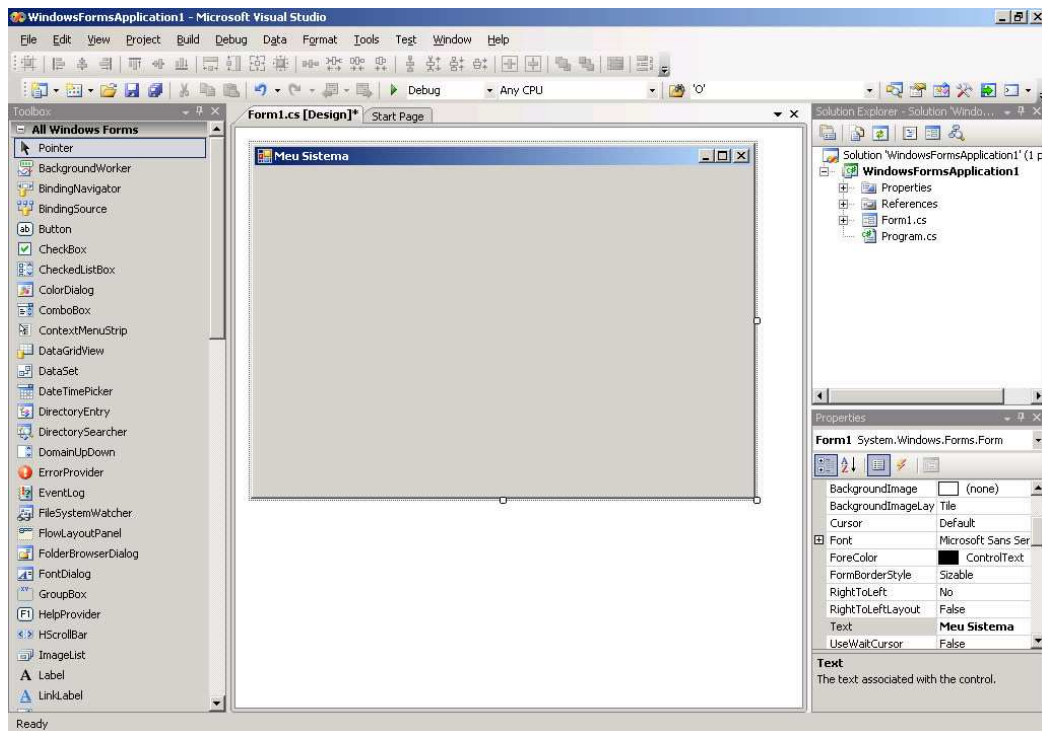
2º Passo: Visualizar a janela de propriedades (Properties Window). Para isso, clique no menu View, Properties Window :





Sempre que for necessário acrescentar uma janela no seu projeto, procure o menu View.

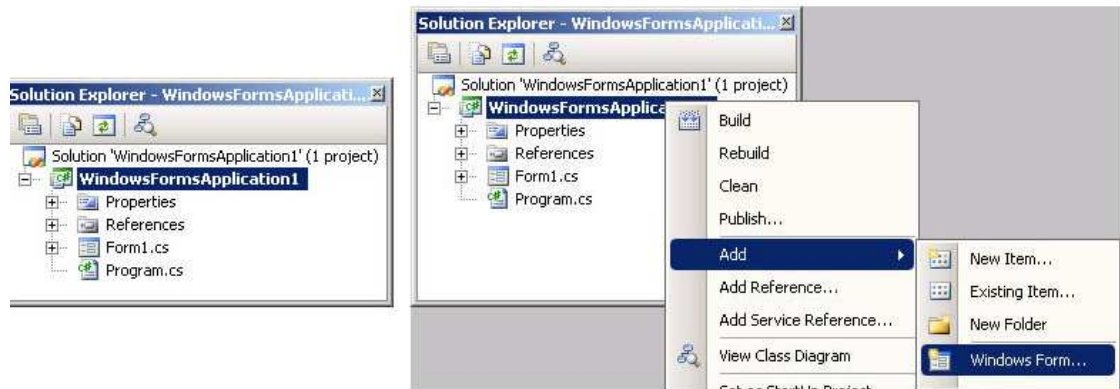
Primeiro Projeto



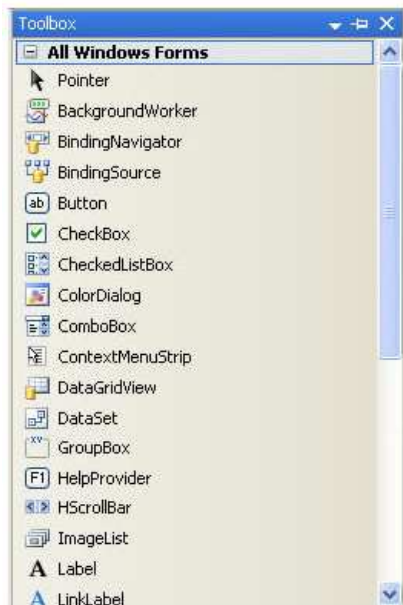
Assim que criamos um novo projeto, um novo form automaticamente é criado.

As aplicações do tipo *Windows Forms*, em geral, são compostas por mais do que um formulário.

Através da barra de ferramentas **Solution Explorer**, pode-se visualizar os formulários que compõe o projeto, como mostra a figura da esquerda, além de permitir que novos formulários sejam adicionados a ele, como mostra a figura da direita.



Caixa de Ferramentas



No canto esquerdo da tela do Visual Studio, encontramos a aba *Toolbox*. Ela nos traz todos os controles que podemos utilizar em nosso aplicativo.

Para adicionar um controle, basta arrastá-lo do menu *Toolbox* para a sua janela. Simples assim!

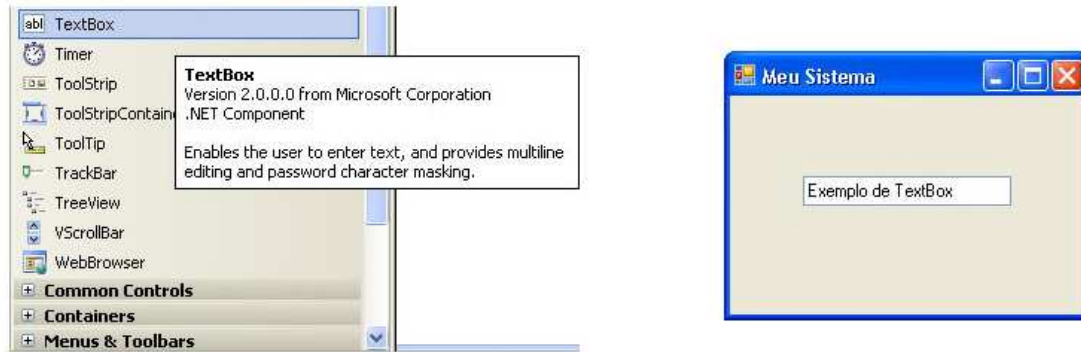
Daqui em diante, iremos discutir cada um dos principais controles utilizados. São eles:

- *Textbox*
- *Label*
- *Button*
- *MessageBox*
- *ComboBox*
- *CheckBox*
- *Panel*
- *RadioButton*
- *DataGridView*
- *Menu*

Principais Ferramentas

TextBox

TextBox, como o próprio nome diz, são caixas de texto. Elas permitem que o usuário digite uma informação textual pertinente ao fluxo do programa, como nome, descrição, senhas etc.



Principais propriedades da ferramenta Text:

As principais propriedades e eventos de um controle do tipo *TextBox* são relacionadas ao texto armazenado no objeto. Entre as propriedades, destacam-se:

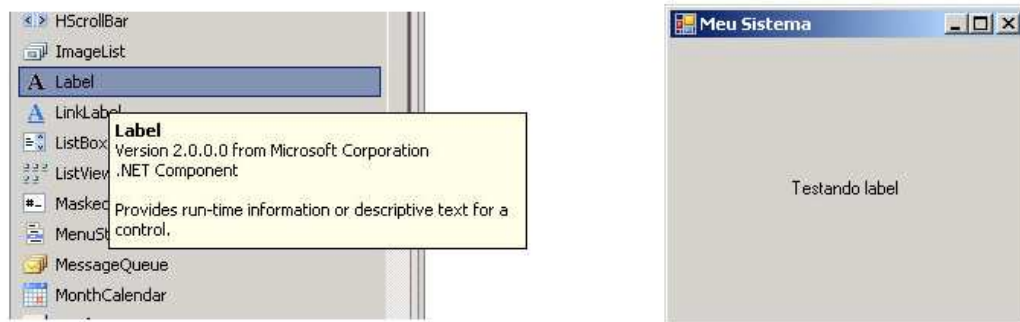
- **ReadOnly**: desabilita a edição do texto.
- **Text**: texto que será exibido por padrão.
- **PasswordChar**: habilita o modo senha, ocultando os caracteres digitados, exibindo somente o caractere definido.
- **MaxLength**: número máximo de caracteres permitidos

Label

O controle **Label** é utilizado para exibir pequenos textos em tela, tais como o rótulo de campos requeridos e outras mensagens ao usuário.

O texto exibido em um controle dessa classe é somente leitura, isto é, não pode ser editado pelo usuário.

Abaixo, na figura à esquerda, observa-se o controle *Label* disponível na barra de ferramentas *ToolBox*. Na figura à direita, é exibido um formulário com uma *Label*.



Principais propriedades da ferramenta Label:

As principais propriedades dos controles do tipo *Label* são relacionadas ao texto exibido e a sua formatação e tamanho. Algumas delas são:

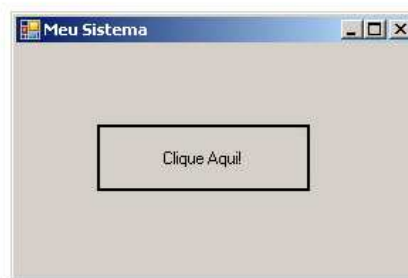
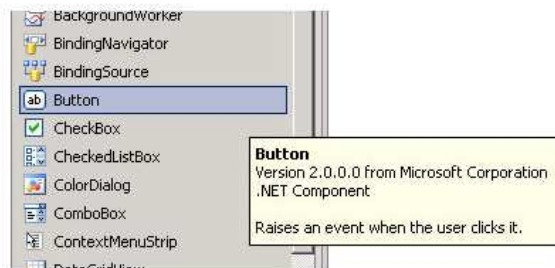
- **Text:** texto a ser exibido
- **ForeColor:** define a cor do texto

Esse controle não realiza interação com o usuário, devido ao seu caráter somente leitura. Assim, os eventos de um *Label* dificilmente são explorados a fundo, dessa forma, os eventos mais utilizados em objetos deste tipo já foram citados junto aos eventos básicos dos controles.

Ferramenta Button

O controle **Button** é utilizado para executar uma ação que deve ser disparada voluntariamente pelo usuário. Botões são utilizados para muitas finalidades, tais como enviar os dados de um formulário de cadastro, fechar ou abrir uma janela e muitas outras ações.

Na figura ao lado, vemos uma formulário com um botão.



A figura à esquerda exibe o controle *Button* disponível na barra de ferramentas *Toolbox*.

Ferramenta Radiobutton

O controle **RadioButton** é utilizado para exibir uma lista de opções, geralmente com poucos itens onde apenas um item pode ser selecionado.

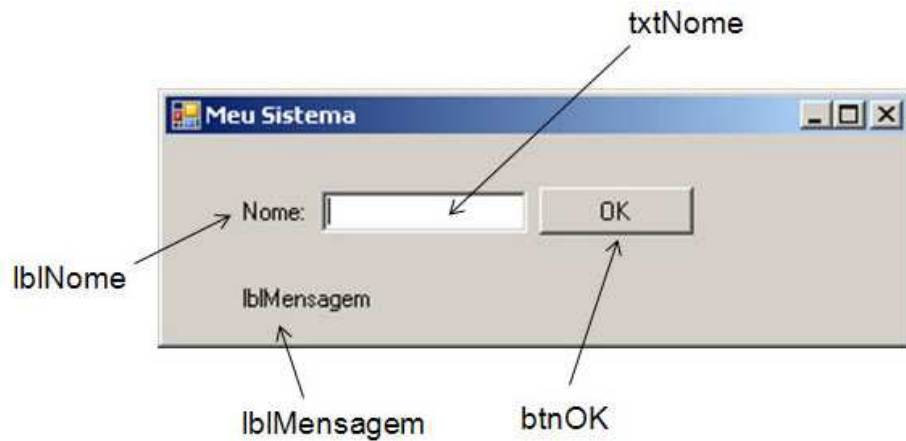
Na tela abaixo, um conjunto com 3 *RadioButtons* é exibido:



Repare que não faz sentido um *RadioButton* existir sozinho. Ao menos 2 desses controles são necessários para permitir que o usuário possa escolher uma opção. Assim, ao colocar 2 ou mais *RadioButtons* em um formulário, somente um deles poderá ser selecionado.

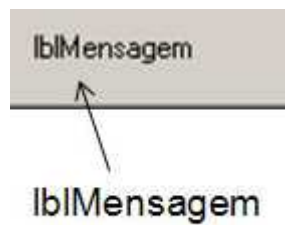
Mas como fazer para criar 2 grupos de seletores, um com o sexo (masculino/feminino) e um com o tipo de pessoa (física/jurídica), por exemplo, em um mesmo formulário? É o que veremos a seguir.

Exemplo de Formulário



Por padrão, a propriedade Name das ferramentas deve seguir a seguinte regra:

Primeiro nome minúsculo, segundo nome com a primeira letra maiúscula.



Como Declarar as variáveis?

Tipo Nome da Variável;

Int c;

Como Atribuir:

Podemos atribuir as variáveis de 2 formas diferentes:

1º Diretamente na declaração:

Int c=10;

Double Altura = Convert.ToDouble(TxtAltura.Text);

Int Altura = int.Parse(TxtAltura.Text);

2º Após a declaração:

Int c;

Double Altura;

C=10;

Altura= Convert.ToDouble(TxtAltura.Text);

Operadores

OPERADOR	OPERAÇÃO	EXEMPLO
=	ATRIBUIÇÃO	A = B ;
==	IGUALDADE	(A==B)
+	SOMA	X = A + B;
-	DIFERENÇA	X = A - B;
*	MULTIPLICAÇÃO	X = A * B;
/	DIVISÃO	X = A / B;
&&	AND - CONDICIONAL	(X==Y) && (Y==Z)
	OR - CONDICIONAL	(X==Y) (Y==Z)
>	MAIOR	X > Y
<	MENOR	X < Y
>=	MAIOR IGUAL	X >= Y
<=	MENOR IGUAL	X <= Y
!=	DIFERENTE	0 != 1

Estruturas de Decisão

```
if (cond > 10)
{
//comandos para cláusula verdade
}
else if
{
//comandos para cláusula verdade
}
else
{
//comandos para cláusula verdade
}
```

Case (múltiplas decisões)

```
switch(i)
```

```
{
```

```
case 1:
```

```
    comando se caso for 1;
```

```
    break; -- comando obrigatorio para estrutura
```

```
case 2:
```

```
    comando se caso for 2;
```

```
    break;
```

```
default:
```

```
    significa se nao ocorrer nenhum dos cases. Ele
```

```
executa esta instrução
```

```
    break;
```

```
}
```


Comando para tratamento de Erro

```
try
```

```
{
```

```
    Executara comandos
```

```
}
```

```
catch(Exception e)
```

```
{
```

```
    Se ocorrer alguma exceção acima no try ele capturará  
    e executará este bloco de comando
```

```
}
```

```
finally
```

```
{
```

```
    Mesmo que houver uma exceção no programa esta  
    cláusula obriga o programa a executa-la. Ela não é  
    obrigatória, porém em determinados casos se faz muito  
    útil
```

```
}
```

Laços de Repetição

for (int i=0; i >=10; i ++)

{

//executará este bloco tantas vezes até que a condição não
mais se satisfaça

}

while(cond > 10)

{

//executará até que esta condição não se satisfaça mais

}

do

{

//será executado pelo menos uma vez e será avaliada a
condicional, se verdade será repetido N – vezes até que
condição se faça falsa.

}

while(cond>10);

Criação de Procedimentos e Funções no C#

Procedimentos

Para criação de um método no c# temos que simplesmente informar se ele será vazio (void) e quais são os parâmetros que ele receberá.

Ex.

```
public void Calcula()  
{  
    //comandos para seu procedimento  
}
```

Funções


Na criação de Funções temos que informar o tipo e obrigatoriamente mostrar o retorno.

Ex.

```
public string ToString(Int i)  
{  
    return Convert.ToString.Parse(nome);  
}
```

Calculando a área do triângulo

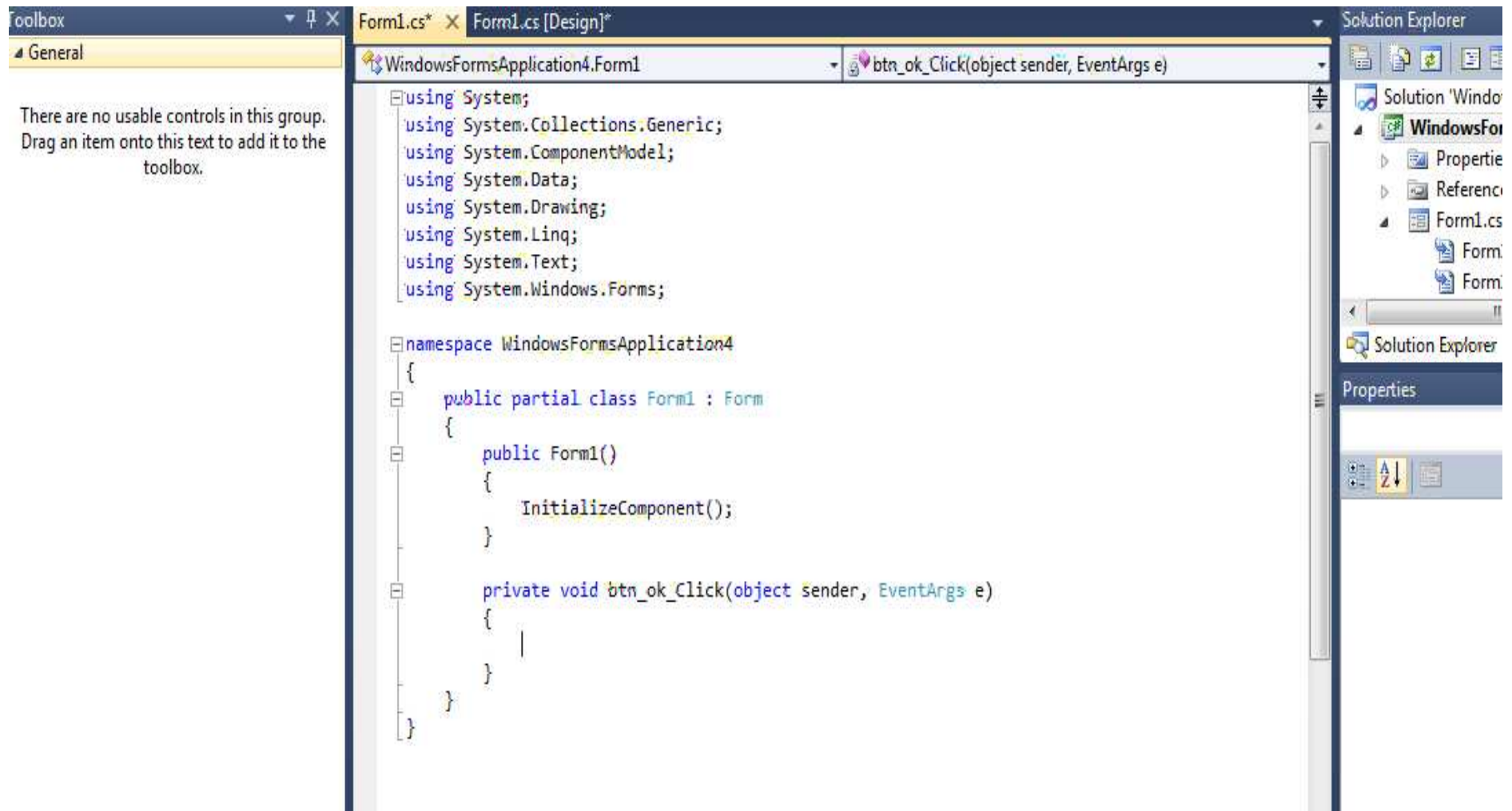
1º Passo: Criar o seguinte form:



The image shows a screenshot of a Windows application window titled "Triângulo". The window has a standard Windows XP-style title bar with minimize, maximize, and close buttons. Inside the window, there is a light gray background. On the left side, there are two text labels: "Altura:." and "Base:.", each followed by a white rectangular text input field. Below these input fields is a blue "OK" button. At the bottom of the window, the word "Resultado" is displayed, indicating where the calculated area will be shown.

Para efetuar a codificação, que será responsável pelo cálculo, dê um duplo clique no botão OK.

A seguinte tela vai surgir:



O código será desenvolvido dentro do procedimento do botão ok entre as chaves {}:

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

Primeiramente, devemos declarar e atribuir os valores às variáveis:

```
double Altura = Convert.ToDouble(TxtAltura.Text);
double Base = Convert.ToDouble(TxtBase.Text);
```

ou

```
double Altura = double.Parse(TxtAltura.Text);
double Base = double.Parse(TxtBase.Text);
```

Tanto o comando `Convert.ToDouble` quanto o `Double.Parse` são comandos utilizados para converter valores de tipos diferentes. No exemplo do triângulo, `TxtAltura.Text` representa um texto, enquanto a variável `Altura` é do tipo numérico. Como são de tipos de dados diferentes, é necessária a conversão.

Depois faremos o cálculo:

```
double Area = (Altura * Base) / 2;
```

E mostramos o resultado na Label

```
lblArea.Text = Convert.ToString(Area);
```

Novamente temos tipos de dados diferentes (Area – Numérico e LblArea – String). Por isso é necessário novamente utilizar o recurso da conversão (.ToString()).

Código final do Exercício:

```
private void OK_Click(object sender, EventArgs e)
{
    double Altura = Convert.ToDouble(TxtAltura.Text);
    double Base = Convert.ToDouble(TxtBase.Text);

    double Area = (Altura * Base) / 2;

    lblArea.Text = Convert.ToString(Area);
}
```

Outra Forma de fazer o mesmo código:

```
private void OK_Click(object sender, EventArgs e)
{
    double Altura;
    double Base;
    double Area;

    Altura = Double.Parse(TxtAltura.Text);
    Base = Double.Parse(TxtBase.Text);

    Area = (Altura * Base) / 2;

    lblArea.Text = Area.ToString();
}
```

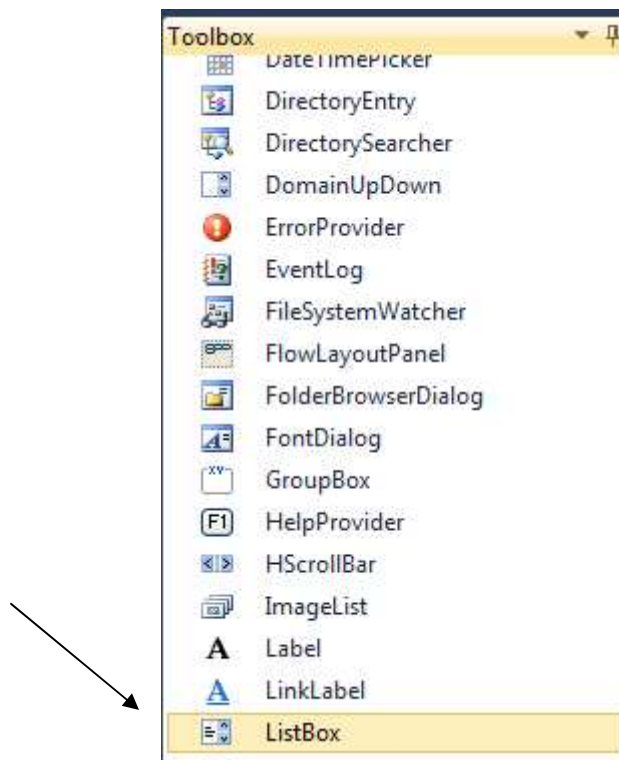

Exemplo de Código com Tomada de Decisão (IF - ELSE) e Controle de Erros

```
private void btnCalcular_Click(object sender, EventArgs e)
{
    try
    {
        double salario = Convert.ToDouble(txtSalario.Text);
        double result;

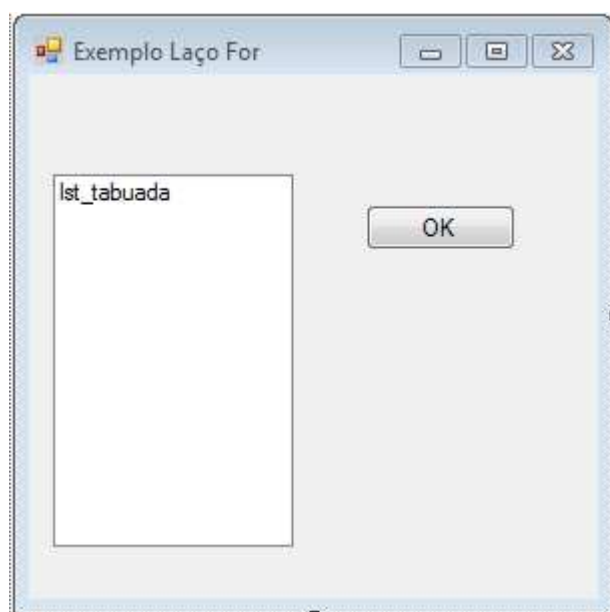
        if (salario < 965.67)
        {
            result = salario * 0.08;
            txtDesconto.Text = Convert.ToString(result);
        }
        else if (salario <= 1609.45)
        {
            result = salario * 0.09;
            txtDesconto.Text = Convert.ToString(result);
        }
        else if (salario <= 3218.90)
        {
            result = salario * 0.11;
            txtDesconto.Text = Convert.ToString(result);
        }
        else if (salario > 3218.90)
        {
            result = 354.08;
            txtDesconto.Text = Convert.ToString(result);
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Insira numeros nos campos!");
    }
}
```

Exemplo do laço de repetição For, utilizando uma ListBox

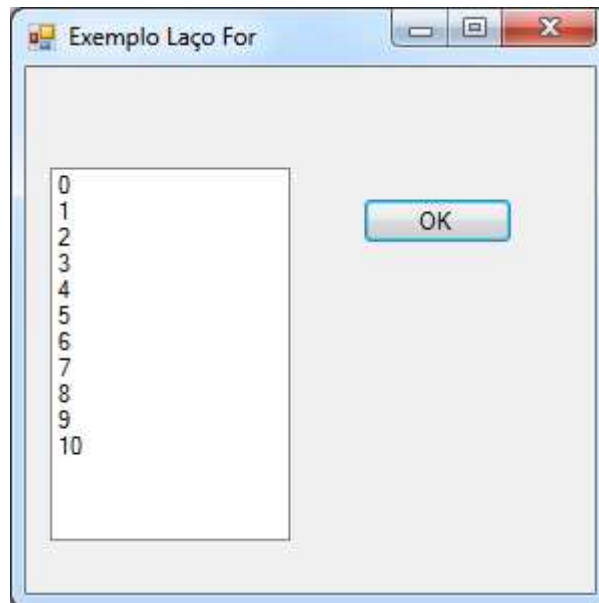
Primeiro, selecionamos e desenhamos no Form a Ferramenta ListBox e Button:



O resultado deve ser esse:



O Objetivo desse programa é, ao clicar no botão, mostrar uma lista de números, de 0 a 10:



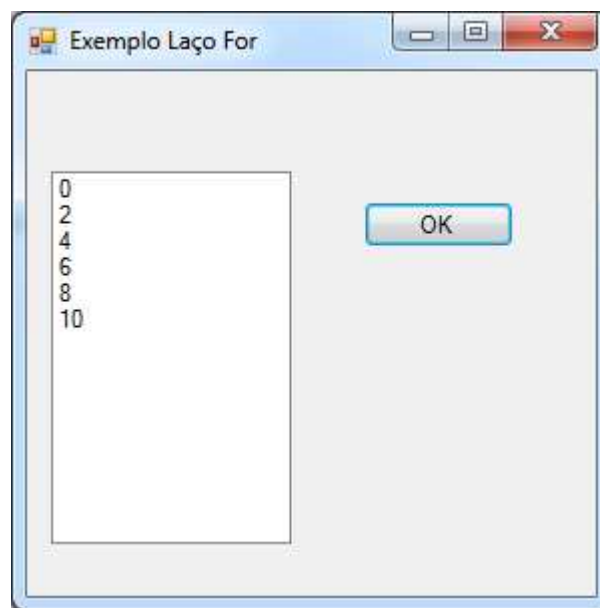
Utilizaremos o laço For, então o código ficará assim:

```
private void btn_ok_Click(object sender, EventArgs e)
{
    for (int i=0; i<=10;i++)
    {
        lst_tabuada.Items.Add(i.ToString());
    }
}
```

Perceba que dentro da cláusula For, temos o comando `i++`, que serve para incrementar a variável `i` em 1. Caso o incremento não seja de 1 em 1, mas de 2 em 2 por exemplo, basta substituir por `i+=2`.

```
private void btn_ok_Click(object sender, EventArgs e)
{
    for (int i=0; i<=10;i+=2)
    {
        lst_tabuada.Items.Add(i.ToString());
    }
}
```

O resultado será:

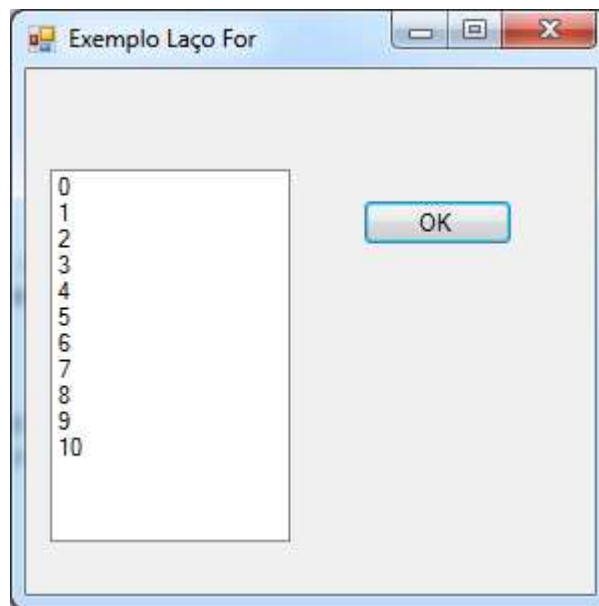


Exemplo do laço While

O mesmo exemplo anterior, só que agora com o comando While:

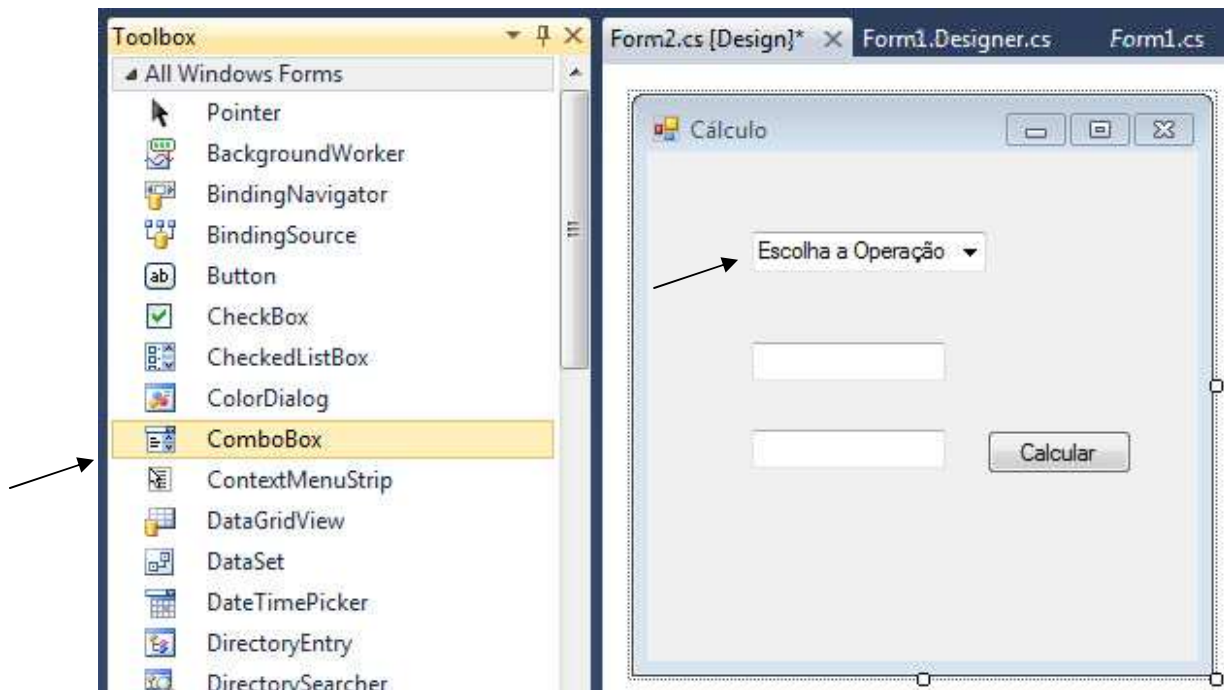
```
int c; //variável que incrementa e determina o final do laço na condição
private void btn_ok_Click(object sender, EventArgs e)
{
    while (c <= 10)
    {
        lst_tabuada.Items.Add(c.ToString());
        c = c + 1; // Variável c sendo incrementada em 1
    }
}
```

O resultado é o mesmo:

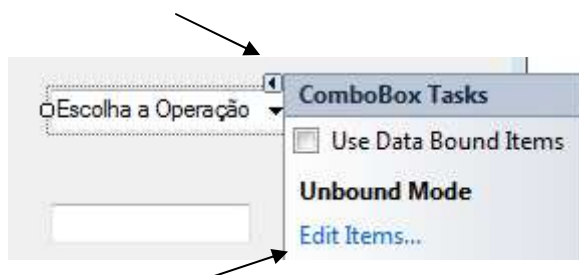


Exemplo do Comando Switch Case

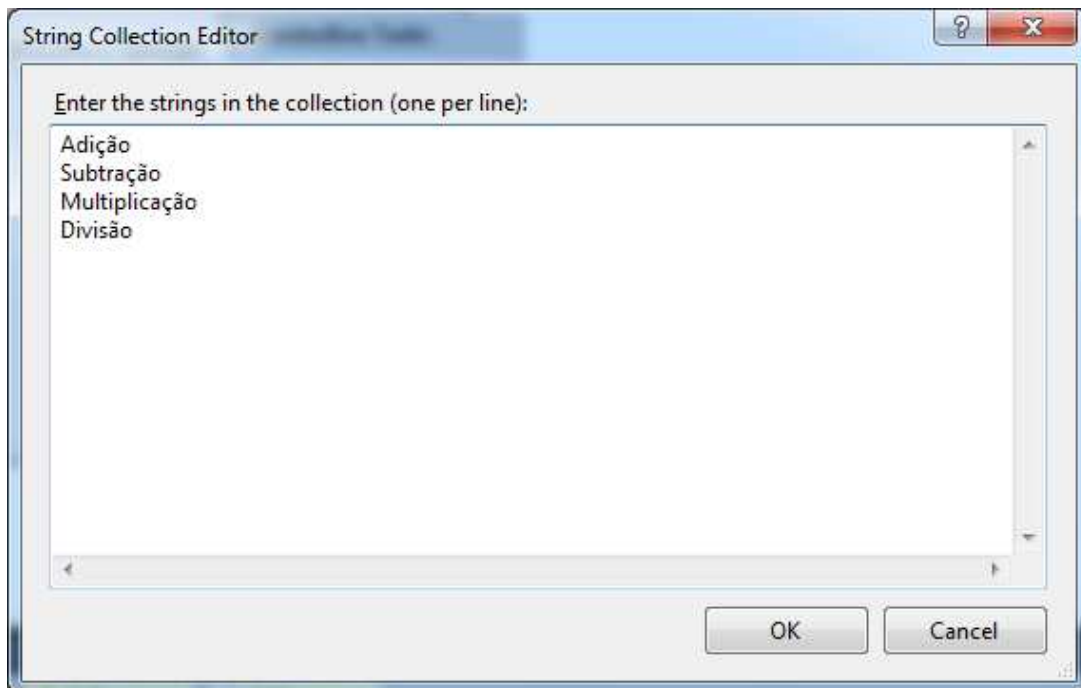
Para esse exemplo, utilizaremos a ferramenta Combobox, criaremos um Form para realizar uma das 4 operações básicas. Além da Combobox, teremos 2 caixas de texto e um botão:



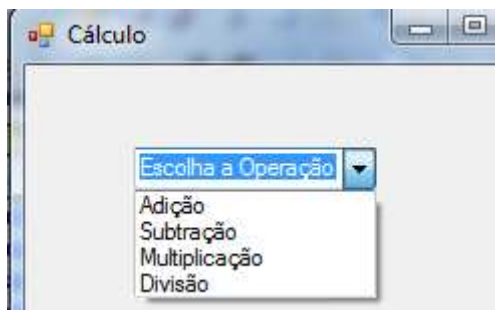
Para acrescentar as operações à Combobox, basta selecionar, e clicar na seta acima da ferramenta e escolher a opção Edit Items:



Em cada linha, adicione uma operação:



O resultado será esse:



Para efetuar o cálculo, é necessário escolher uma das operações. Poderíamos utilizar o IF, mas temos uma alternativa, o Switch Case. A propriedade da Combobox a ser verificada é a propriedade `SelectedIndex`. Cada operação está vinculada a um valor, de acordo com a sequência de digitação (Soma=0, Subtração=1, Multiplicação=2, Divisão=3). Então, teremos que verificar qual delas foi a escolhida (`SelectedIndex=0`, significa que foi escolhida a operação Soma). É importante verificar também, se o usuário não escolheu nenhuma das operações.

```

private void button1_Click(object sender, EventArgs e)
{
    double num1, num2, res; //Declaração das variáveis
    try
    {
        num1 = double.Parse(txtN1.Text); //Atribuição das Variáveis
        num2 = double.Parse(txtN2.Text); //Atribuição das Variáveis

        switch (cboOperacoes.SelectedIndex)
        {
            case 0:
                res = num1 + num2;
                MessageBox.Show(res.ToString());
                break; //comando obrigatorio para estrutura
            case 1:
                res = num1 - num2;
                MessageBox.Show(res.ToString());
                break;
            case 2:
                res = num1 * num2;
                MessageBox.Show(res.ToString());
                break;
            case 3:
                res = num1 / num2;
                MessageBox.Show(res.ToString());
                break;
            default:
                MessageBox.Show("Escolha uma Operação Válida"); // Para garantir que uma operação será escolhida
                break;
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Verifique os valores digitados");
    }
}

```

No código acima, verificamos caso a caso qual das operações foi escolhida, e efetuamos o cálculo correspondente. Nunca se esquecendo do controle de erros (Try/Catch), em caso de digitação incorreta do usuário.

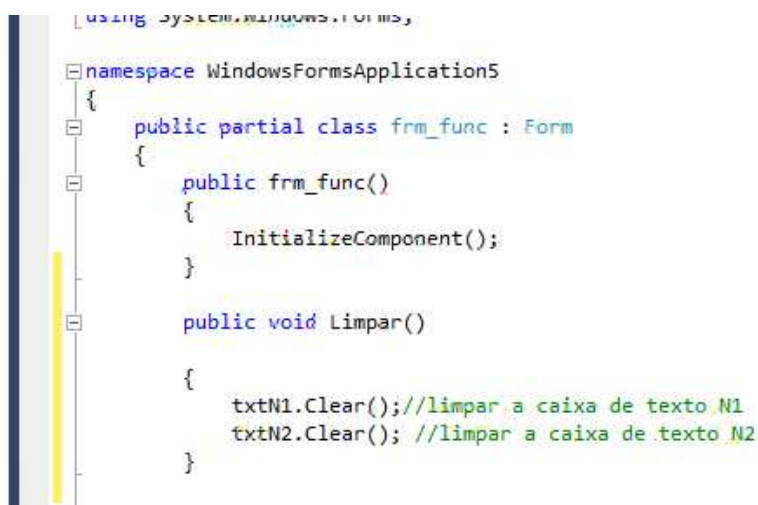
Exemplo de Procedimento e Funções

Faremos o mesmo exemplo anterior, só que agora utilizando Procedimentos e Funções.

Procedimento Limpar.

O procedimento é uma forma de automatizar a programação e evitar a repetição desnecessária de códigos. O procedimento não retorna resultado, tem apenas a execução dos comandos.

Nesse exemplo, criamos um procedimento para limpar as caixas de texto. Veja a seguir onde e como criar o procedimento Limpar:



```
using System.Windows.Forms;

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

        public void Limpar()
        {
            txtN1.Clear(); //limpar a caixa de texto N1
            txtN2.Clear(); //limpar a caixa de texto N2
        }
    }
}
```

Nesse exemplo escolhemos o modificador public, que permite a utilização pública do procedimento em qualquer parte do projeto. Como não há retorno de valor, utilizamos o comando void. Por último escolhemos o nome, nesse exemplo Limpar().

Entre chaves, colocamos os respectivos códigos.

Funções para o cálculo

Logo abaixo do procedimento, já podemos criar as nossas funções. Basicamente com a mesma ideia do procedimento, a função retorna obrigatoriamente um resultado, por isso é necessário definir um tipo, e que parâmetros serão usados para esse resultado.

```
Public Tipo Nome (tipo parâmetro1, tipo parâmetro2,...) tipo parâmetro n)
{
Comandos;
}
```

Faremos uma função para cada operação:

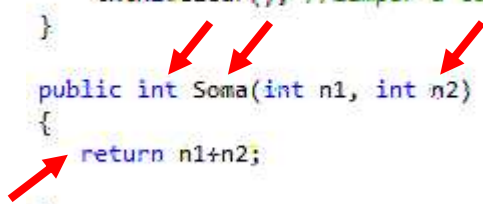
```
public void Limpar()
{
    txtN1.Clear(); //limpar a caixa de texto N1
    txtN2.Clear(); //limpar a caixa de texto N2
}

public int Soma(int n1, int n2)
{
    return n1+n2;
}

public int Sub(int n1, int n2)
{
    return n1-n2;
}

public int Mult(int n1, int n2)
{
    return n1*n2;
}

public double Dividir(int n1, int n2)
{
    return n1/n2;
}
}
```



As setas apontam na ordem: O tipo de retorno, o nome da função, os parâmetros (nesse exemplo, são dois números para o cálculo. Os Parâmetros representam de maneira genérica o que será calculado) e por último, o comando return (obrigatório), responsável pelo retorno do resultado.

Feitas as funções, agora às utilizaremos conforme a escolha da combobox:

```

private void btn_calc_Click(object sender, EventArgs e)
{
    int num1, num2;
    double res; //Declaração das variáveis
    try
    {
        num1 = int.Parse(txtN1.Text); //Atribuição das Variáveis
        num2 = int.Parse(txtN2.Text); //Atribuição das Variáveis

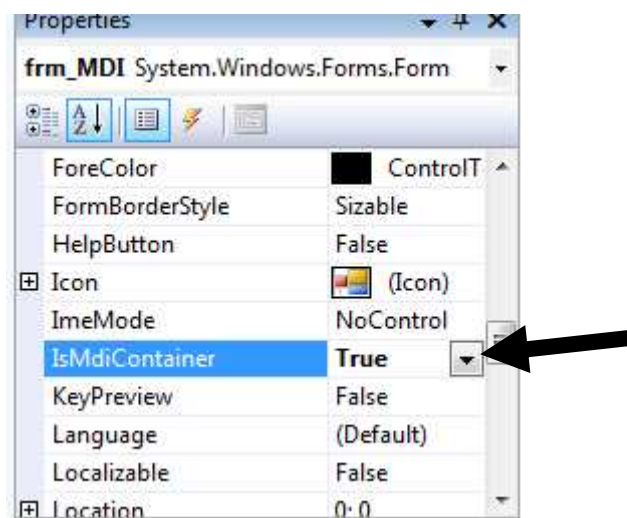
        switch (cboOperacoes.SelectedIndex)
        {
            case 0:
                res = Soma(num1, num2);
                lblRes.Text = res.ToString();
                Limpar();
                break; //comando obrigatorio para estrutura
            case 1:
                res = Sub(num1, num2);
                lblRes.Text = res.ToString();
                Limpar();
                break;
            case 2:
                res = Mult(num1, num2);
                lblRes.Text = res.ToString();
                Limpar();
                break;
            case 3:
                res = Dividir(num1, num2);
                lblRes.Text = res.ToString();
                Limpar();
                break;
            default:
                MessageBox.Show("Escolha uma Operação Válida"); // Para garantir que uma operação será escolhida
                break;
        }
    }
    catch (Exception)
    {
        MessageBox.Show("Verifique os valores digitados");
    }
}
}

```

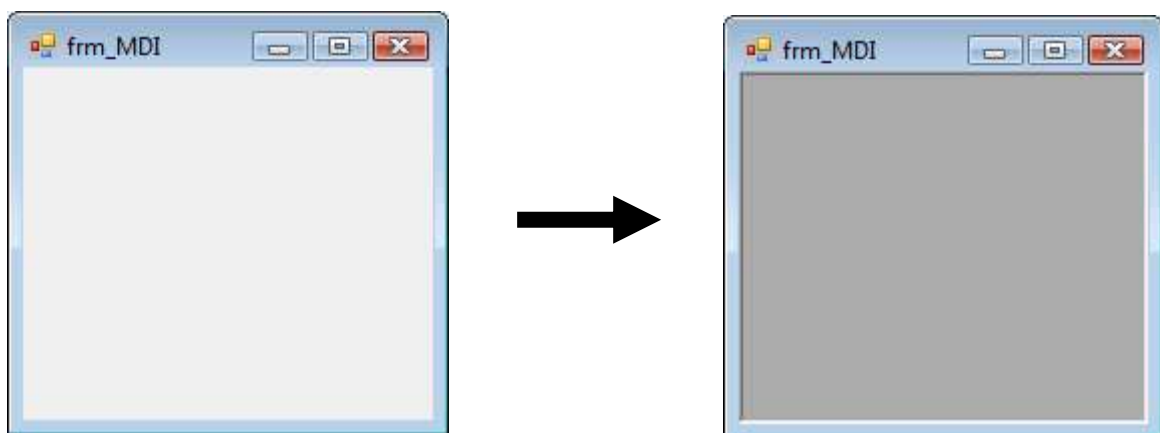
Perceba que depois do cálculo (com as funções) e de mostrar os resultados na lblRes, utilizaremos o procedimento Limpar(), para que as caixas de texto fiquem vazias após o cálculo. Outra diferença do exemplo anterior é a utilização de uma label para exibir o resultado ao invés de uma messagebox.show().

Trabalhando com formulários MDI e Menus

Primeiramente devemos transformar um form comum em MDI. Para isso, selecione o form e na lista de propriedades selecione a opção ISMDICONTAINER, alterando de false para true.

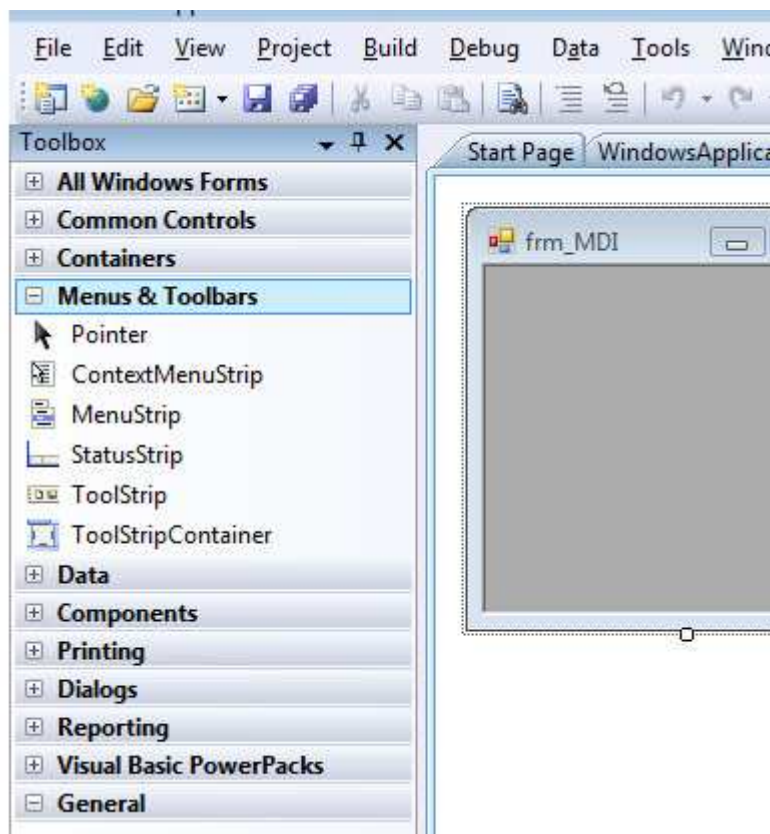


O form mudará, ficará na cor cinza.

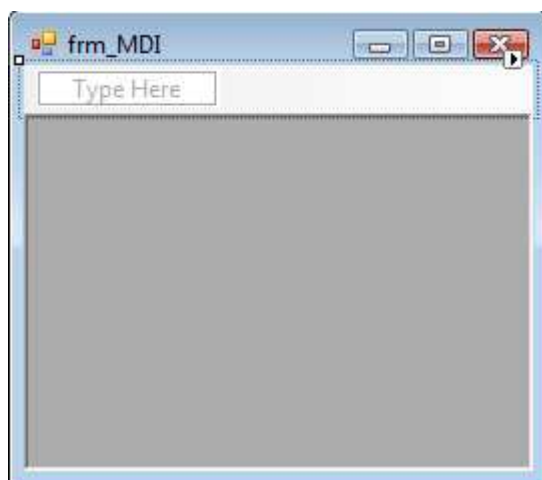


Feito isso e com o form renomeado, deveremos acrescentar a ferramenta para criar a estrutura de menus:

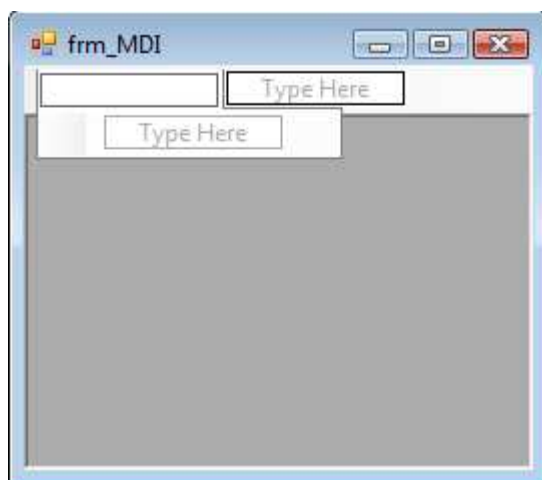
Localize na lista de Ferramentas, na categoria Menus e Toolbars, a ferramenta MENUSTRIP:



Desenhe no Form:

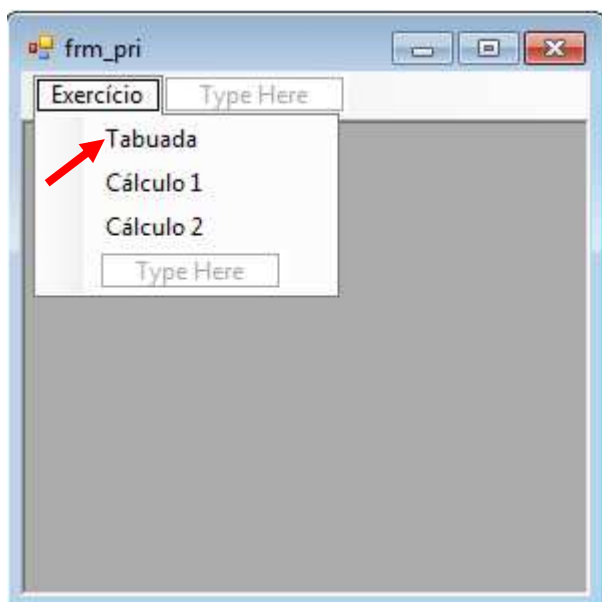


Depois, basta Clicar na Opção **Type Here** e construir o menu, clicando nas opções abaixo ou ao lado.



Programando os menus

Dado o exemplo:



Dê um duplo clique na opção do menu que será programada (nesse exemplo, será a opção Tabuada). Para chamar os forms, primeiro é necessário criar uma instância com o form que será chamado. Depois, forçar que o formulário chamado fique preso (dentro) do formulário Principal. Depois, basta chamar o form correspondente:


```
private void tabuadaToolStripMenuItem_Click(object sender, EventArgs e)
{
    frm_tab frm = new frm_tab();// Instância de referência com o Form a ser chamado, nesse caso, frm está relacionado com o frm_tab
    frm.MdiParent = this; //"Prende" o form ao Formulário Principal
    frm.Show();//Chama o formulário. Também podemos utilizar nesse caso o ShowDialog
}
```

Basta repetir o processo com os outros forms. No caso da utilização do ShowDialog ao invés do Show, o Form chamado deverá ser finalizado para que outro formulário seja utilizado. Não será possível utilizar dois formulários concomitantemente.

```
private void cálculo1ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frm_calc1 frm = new frm_calc1();
    frm.MdiParent = this;
    frm.Show();
}

private void cálculo2ToolStripMenuItem_Click(object sender, EventArgs e)
{
    frm_func frm = new frm_func();
    frm.MdiParent = this;
    frm.Show();
}
```

Outros comandos para trabalhar com formulários

Para arranjo de vários formulários:

```
// Define o Layout para cascade.
```

```
this.LayoutMdi(MdiLayout.Cascade);
```

```
// Define o Layout para tile horizontal.
```

```
this.LayoutMdi(MdiLayout.TileHorizontal);
```

```
// Define o Layout para tile vertical.
```

```
this.LayoutMdi(MdiLayout.TileVertical);
```

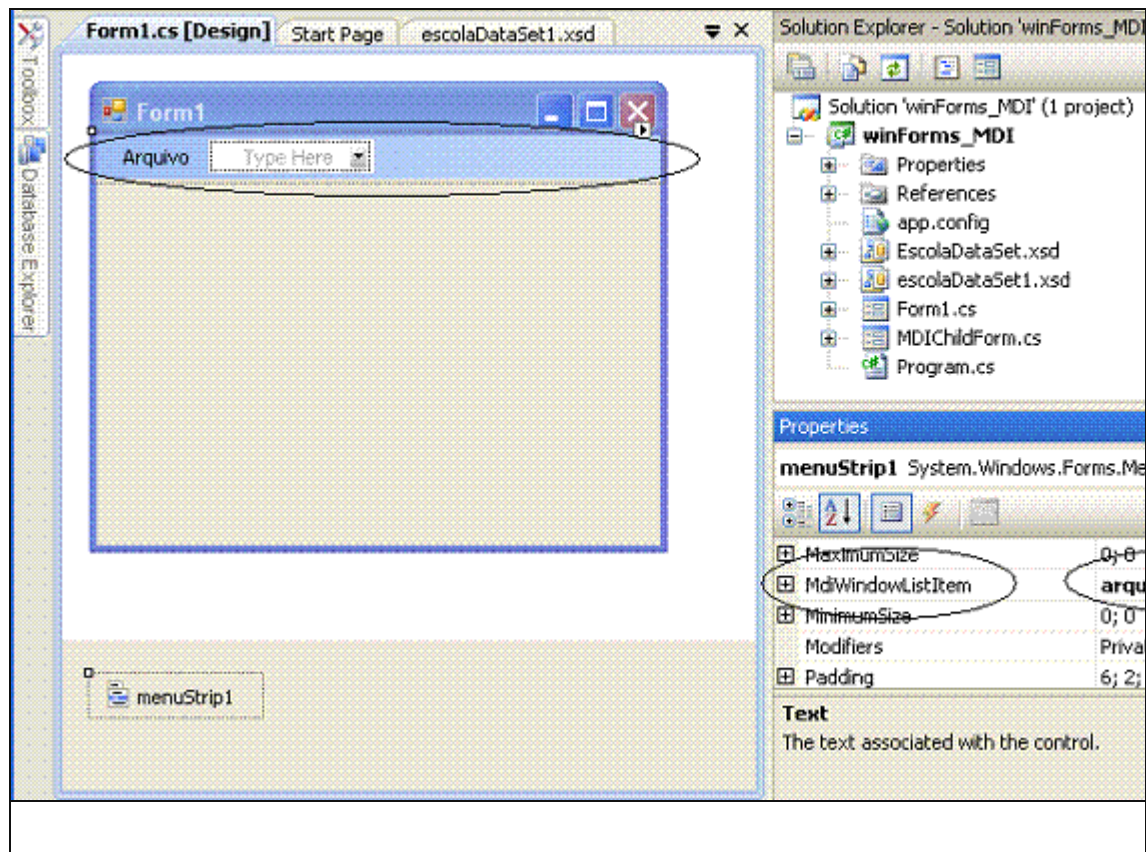
Para fechar vários forms de uma vez:

```
private void fecharTodosToolStripMenuItem_Click(object sender, EventArgs e)
{
    foreach (Form mdiChildForm in MdiChildren)
    {
        mdiChildForm.Close();
    }
}
```

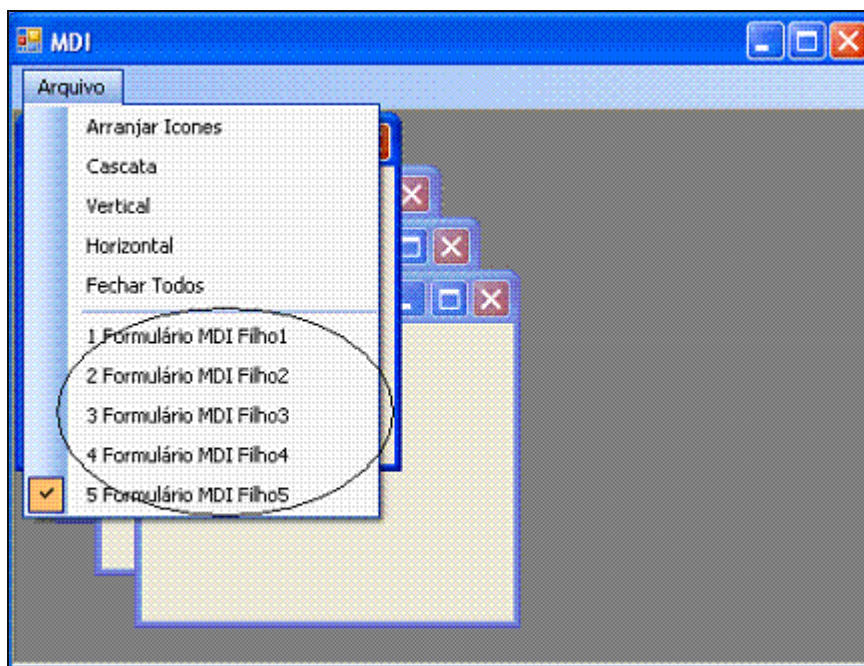
Exibir em um menu os formulários filhos

Selecione o controle MenuStrip e na janela de propriedades selecione a propriedade MdiWindowListItem;

A seguir informe qual item do menu irá exibir os formulários filhos abertos:



O resultado é exibido a seguir:



Propostas de Exercícios

- 1) Faça um programa que calcule o desconto do INSS. O programa deverá receber o salário do funcionário, e de acordo com esse salário, faça o cálculo seguindo a tabela abaixo:

SALÁRIO	Taxa de desconto
Até 965,67	8%
de 965,68 até 1.609,45	9%
de 1.609,46 até 3.218,90	11%
Acima de 3218,90	R\$ 354,08

- 2) Crie um programa que mostre numa label a quantidade de parcelas do seguro desemprego, caso o trabalhador tenha direito. O programa deverá receber a quantidade de meses trabalhados, e o resultado será dado através da tabela abaixo:

Meses trabalhados	Número de parcelas
<6	Não tem direito
>=6 a =11	3 parcelas
>11 a =24	4 parcelas
>24	5 parcelas

3) Crie um programa que mostre numa Label o valor da parcela do seguro desemprego. O usuário deverá digitar o salário, e de acordo com a tabela a seguir faça o cálculo:

Faixas de Salário Médio	Valor da Parcela
< 465,00	Salário Inválido
Até R\$ R\$ 767,60	Salário Médio * 0.8 (80%)
De R\$ 767,61 até R\$ 1.279,46	(Salário-767,60 *0.5) + 614,08.
Acima de R\$ 1.279,46	O valor da parcela será de R\$ 870,01 invariavelmente.

4) Crie um programa que calcule a média ponderada de um aluno. Serão digitadas as notas de um trabalho e uma prova.

Fórmula

$$(\text{Trabalho} * 4 + \text{prova} * 6) / 10$$

Atenção: As notas tanto da prova quanto do trabalho serão de 0 a 10, apenas!

5) Crie um programa que receba a quantidade colhida na ultima safra de cana-de açúcar. Verifique a situação da safra, de acordo com a tabela abaixo:

Quantidade da colheita	Situação
Até 1000 kg	Abaixo da meta
Entre 1000 e 5000 kg	Dentro da meta
Acima de 5000 kg	Acima da meta

6) Calcule o valor da evasão escolar de uma faculdade. Para isso, é necessário digitar a quantidade de alunos matriculados e a quantidade de alunos formados.

Exemplo:

100 **alunos matriculados**

90 **alunos formados**

10% de evasão

Atenção: deverá aparecer a Porcentagem: 10%

Avalie a evasão, de acordo com a tabela abaixo, e apresenta ao usuário:

Evasão	Situação
Menor que 10%	Baixa evasão
10% a 15%	Evasão média
15% a 25%	Evasão Alta
Acima de 25%	Evasão Muito Alta

7) Crie um programa que calcule o valor das despesas de um Vestibular. O usuário deverá digitar o valor arrecadado com as inscrições, a quantidade de fiscais e o valor gasto com alimentação. Sabendo que cada fiscal recebe R\$ 70, 00, calcule o quanto sobrou para a universidade (ARRECADAÇÃO – (FISCAIS *70) – ALIMENTAÇÃO).

8) Faça o cálculo das conversões

Conversão de	para	Fórmula
Celsius	Fahrenheit	$^{\circ}\text{F} = ^{\circ}\text{C} \times 1,8 + 32$
Celsius	kelvin	$\text{K} = ^{\circ}\text{C} + 273,15$
Celsius	Rankine	$^{\circ}\text{R} = ^{\circ}\text{C} \times 1,8 + 32 + 459,67$
Celsius	Réaumur	$^{\circ}\text{Ré} = ^{\circ}\text{C} \times 0,8$
kelvin	Celsius	$^{\circ}\text{C} = \text{K} - 273,15$
kelvin	Fahrenheit	$^{\circ}\text{F} = \text{K} \times 1,8 - 459,67$
kelvin	Rankine	$^{\circ}\text{R} = \text{K} \times 1,8$
kelvin	Réaumur	$^{\circ}\text{Ré} = (\text{K} - 273,15) \times 0,8$
Fahrenheit	Celsius	$^{\circ}\text{C} = (^{\circ}\text{F} - 32) / 1,8$
Fahrenheit	kelvin	$\text{K} = (^{\circ}\text{F} + 459,67) / 1,8$
Fahrenheit	Rankine	$^{\circ}\text{R} = ^{\circ}\text{F} + 459,67$
Fahrenheit	Réaumur	$^{\circ}\text{R} = (^{\circ}\text{F} - 32) / 2,25$

9) Crie um programa que calcule a classificação e a comissão de vendas de um vendedor:

Vendas	Classificação	Taxa de comissão	Bônus
Até =1000	Baixo	3% de Comissão	Sem Bônus
1000 a =5000	Médio	4% de Comissão	Bônus de 50,00
Acima de 5000	Alto	7% de Comissão	Bônus de 75,00

O programa deverá retornar a classificação, a taxa de comissão (calculada) e de quanto é o bônus.

- 10) Crie um programa que calcule o IMC (índice de massa corpórea). A fórmula é a seguinte:**
IMC = peso / (altura)²

Após mostrar o resultado numa Label, faça a avaliação de acordo com a tabela abaixo:

IMC em adultos	Situação
abaixo de 18,5	abaixo do peso ideal
entre 18,5 e 25	no peso ideal
entre 25 e 30	acima do peso ideal
acima de 30	obeso

Referências Bibliográficas

FILHO, Ralfe Della Croce; RIBEIRO, Carlos Eduardo. C Sharp *in* Programação de Computadores – Centro Paula Souza, Volume 4, São Paulo, Fundação Padre Anchieta, 2010, p. 100-114.

Disponível em <http://www.macoratti.net/09/08/c_md11.htm>. Acesso em 18/03/2012

Disponível em <<http://www.macoratti.net>>. Acesso em 18/03/2012

Introdução ao C# - Disponível em <<http://www.ev.org.br> >. Acesso em 10/04/2012