Centro Paula Souza

ETEC Dr Geraldo José Rodrigues Alckmin

# Apostila de Estruturas de Repetição em C#

## Versão 1.0

**Componente Curricular:** Desenvolvimento de Sistemas
**Série atendida:** 2ª

**Autores**: Gilberto Abud Junior

Reginaldo Luiz Gonçalves

HOME EDITORA

2023

**Diagramação**

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

**Design da capa e projeto gráfico**

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

**Imagem da capa**

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

**Revisão de texto**

Gilberto Abud Junior
Reginaldo Luiz Gonçalves

Home Editora
CNPJ: 39.242.488/0002-80
www.homeeditora.com
contato@homeeditora.com
9198473-5110
Av. Augusto Montenegro, 4120 - Parque Verde, Belém - PA, 66635-110

**Editor-Chefe**
Prof. Dr. Ednilson Souza
**Bibliotecária**
Janaina Karina Alves Trigo Ramos
**Produtor editorial**
Nazareno Da Luz

DOI: 10.46898/home.46d6c8e0-03ef-4b3c-b45d-139365656f01

**Catalogação na publicação**
**Elaborada por Bibliotecária Janaina Ramos – CRB-8/9166**

A165a

Abud Junior, Gilberto

Apostila de estruturas de repetição em C# - Versão 1.0 / Gilberto Abud Junior, Reginaldo Luiz Gonçalves. – Belém: Home, 2023.

44 p., fotos.; 14,8 X 21 cm

ISBN 978-65-84897-85-4

1. C (Linguagem de programação de computador). 2. Algoritmos. 3. Estruturas de dados (Computação). I. Abud Junior, Gilberto. II. Gonçalves, Reginaldo Luiz. III. Título.

CDD 005.133

Índice para catálogo sistemático

I. C (Linguagem de programação de computador) : Algoritmos

# Ficha de Identificação de Material Didático e Autor

1. Título: **Apostila de Estruturas de Repetição em C# – Versão 1.0**.
2. Assunto: **Técnicas e Linguagem de Programação.**
3. Resumo: **Além da explicação teórica rápida, a apostila avança na utilização das Estruturas de Repetição com vários exemplos práticos desenvolvidos em sala de aula com os alunos, utilizando Visual Studio C#.**
4. Área / habilitação a que se destina:
   **Área/Eixo – Informática e Comunicação.**
   **Habilitações: Desenvolvimento de Sistemas.**
5. Componentes Curriculares que atinge:
   **Desenvolvimento de Sistemas.**
6. Séries atendidas: **2ª.**
7. Nome do ( s ) autor ( es ): **Gilberto Abud Junior**
   **Reginaldo Luiz Gonçalves**
8. Unidade Escolar dos autores:
   **Etec Dr. Geraldo José Rodrigues Alckmin – Etec Taubaté**.
9. Telefones de contato dos autores: **(12) 99104-9082**
   **(12) 99108-3878**
10. E-mail dos autores:
    gilberto.junior101@etec.sp.gov.br
    reginaldo.goncalves@etec.sp.gov.br
11. Número de páginas: 43, caracteres com espaço: 19239, laudas: 14.

# Sumário

# Os autores

**Gilberto Abud Junior**

Brasileiro, nascido em Taubaté – SP, é graduado Tecnólogo em Processamento de Dados, Pedagogia e Matemática. Possui Formação Pedagógica para Educação Profissional em Nível Médio. É Especialista em Informática em Educação e Administração Escolar. Atua como professor no curso Técnico em Informática desde 1998 no Colégio UNITAU da Universidade de Taubaté e desde 2000 nas unidades das Etecs do Centro Paula Souza, atualmente na unidade de Taubaté – SP.

**Reginaldo Luiz Gonçalves**

Brasileiro, nascido em Taubaté – SP, é graduado em Computação Científica e Pedagogia. Possui Formação Pedagógica para Educação Profissional em Nível Médio. É Especialista em Informática em Educação, Administração Escolar, Educação a Distância e em Currículo, Didática e Metodologias Ativas. Atua como professor no curso Técnico em Informática desde 1994 no Colégio UNITAU da Universidade de Taubaté e desde 2003 nas unidades das Etecs do Centro Paula Souza, atualmente na unidade de Taubaté – SP.

# Agradecimentos

Aos nossos familiares, pela paciência, pelo incentivo e pelo apoio incondicional nos momentos difíceis, principalmente nos momentos de nossa ausência para a dedicação a esse trabalho.

# Apresentação

Olá,

Caros estudantes, a programação é uma ciência que utiliza a lógica de nosso pensamento para resolvermos problemas dos mais variados níveis. Com ela conseguimos transmitir e "ensinar" ao computador o que ele deve fazer para a solução do problema proposto. Aprendendo a lógica com certeza você estará habilitado a programar em qualquer linguagem que se interessar.

Esta apostila tem o objetivo de atender uma demanda com mais exemplos práticos no que diz respeito a Linguagem de Programação que precisam ser bem entendidos. Além da explicação teórica rápida, acompanha alguns exemplos práticos desenvolvidos em sala de aula com os alunos dos Cursos Técnicos em Informática, Informática para Internet e Desenvolvimento de Sistemas. A metodologia de resolução de problemas é aplicada e cada atividade aqui resolvida foi estudada, explicada e construída pelo professor e pelos alunos dos respectivos cursos.

Bons estudos aos leitores!

# Introdução

Analisar nossa maneira de pensar nos leva a melhorar o pensamento lógico. As linguagens de programação servem para isso. Ao determinarmos o que desejamos que o computador faça, estamos "conversando" com ele. Os computadores, só fazem aquilo que mandamos, e não necessariamente o que desejamos que eles façam. Não deve haver nenhuma ambiguidade nas instruções dos programas que fornecemos ao computador, nem a possibilidade de interpretações alternativas.

O computador sempre tomará algum caminho em suas ações; muito cuidado é necessário para assegurar que o computador siga pelo único caminho correto possível que leve aos resultados desejados. Quando o aluno está interagindo com o computador ele está manipulando conceitos e isso contribui para seu desenvolvimento mental. Ele está adquirindo conceitos da mesma maneira que ele adquire conceitos quando interage com objetos do mundo.

Os alunos aprendem, porque essa interação com o computador propicia um ambiente riquíssimo e bastante efetivo do ponto de vista de construção do conhecimento.

Desejamos aos leitores um bom estudo e sucesso na programação e na construção do conhecimento.

# Capítulo 01

## Estruturas de Repetição

Para uma maior facilidade em muitos momentos da programação, temos um poderoso recurso que nos auxiliará em muitas ocasiões. Quando houver necessidade de se repetir um determinado trecho de nossos programas para que não tenhamos que escrever linhas repetidas, usaremos o recurso das estruturas de repetição que fará o serviço.

Quando houver necessidade de se testar uma condição no início antes de executarmos algum procedimento utilizaremos o ***while.***

Veja a sintaxe da instrução e um exemplo prático logo abaixo:

while (condição)

{

Instruções a serem executadas caso a condição testada seja verdadeira;

Esse grupo de comandos serão repetidos até que essa condição não seja mais verdadeira.

Incremento de variável

}

## Outras Estruturas de Repetição

A linguagem nos oferece mais duas estruturas de repetição. Quando se precisa fazer um teste ao final, usamos a estrutura do/while. Sua sintaxe é muito parecida com a anterior, mas faz o teste no final, após a execução dos comandos que desejamos repetir.

```
do
   {
    Instruções a serem executadas caso a condição seja
    verdadeira

    Esse grupo de comandos serão repetidos até que essa
    condição não seja mais verdadeira

    Incremento de variável

    }
while (condição);
```

A estrutura "for" também é bastante utilizada nos programas. Para estruturas que tenhamos já o conhecimento do número finito de vezes que iremos repetir o procedimento a estrutura acima será aplicada. Muito fácil e utilizaremos os mesmos programas acima para testá-la, já que é só trocarmos a

estrutura utilizada anteriormente. Veja bem a seguir a sintaxe dessa estrutura.

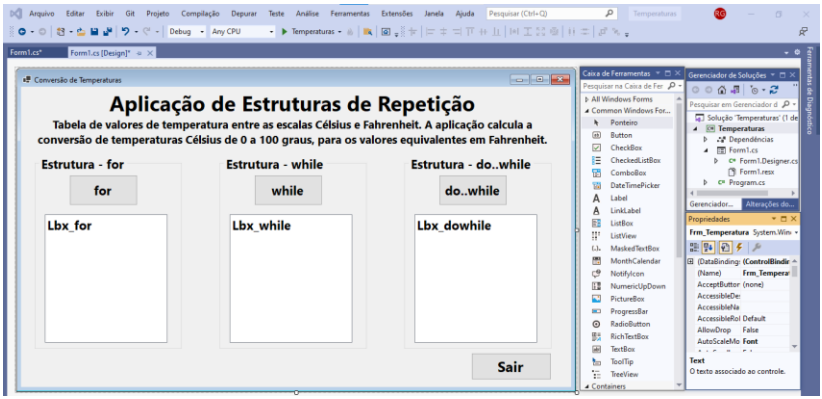
```
for (início; fim; incremento)
{
Instruções a serem executadas;
}
```

# Capítulo 02

# Aplicação de Estruturas de Repetição

## Conversão de Temperaturas

O próximo exemplo de Estrutura de Repetição será uma tabela de valores de temperatura entre as escalas de Celsius e Fahrenheit. A aplicação calcula a conversão de temperaturas Celsius de 0 a 100 graus, para os valores equivalentes em Fahrenheit. São utilizados 3 componentes **ListBox's** para a apresentação da tabela de valores. As 3 estruturas de repetição for, while e do..while, são apresentadas no código dos componentes **Buttons**.



Código do botão **Sair**:

```
1 referência
private void Btn_Sair_Click(object sender, EventArgs e)
{
    Application.Exit();
}
```

Código do botão **for**:

```
1 referência
private void Btn_for_Click(object sender, EventArgs e)
{
    Single TC, // Variável para Temperaturas em Celsius
           TF; // Variável para Temperaturas em Fahrenheit

    for (TC = 0; TC <= 100; TC++) // Estrutura de Repetição
    {
        TF = 1.8f * TC + 32; // Fórmula para Conversão
        Lbx_for.Items.Add("C = " + Convert.ToString(TC) + " *** "
                          + "F = " + Convert.ToString(TF));
    }
}
```

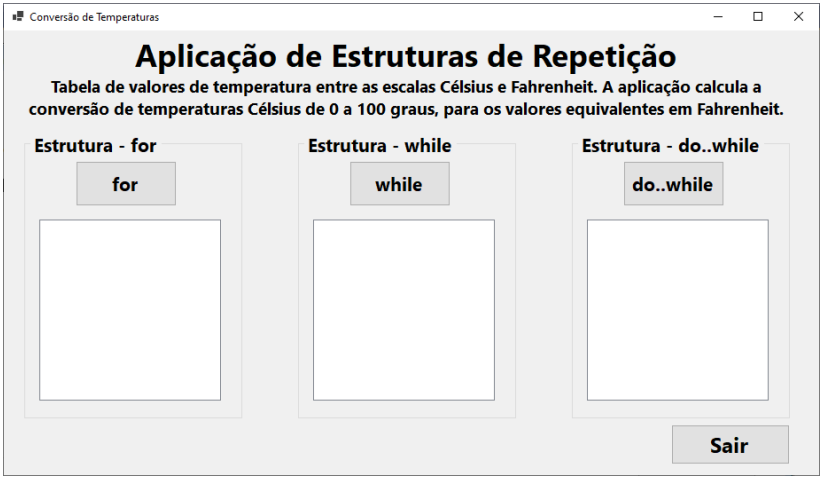
Código do botão **while**:

```
1 referência
private void Btn_while_Click(object sender, EventArgs e)
{
    Single TC, // Variável para Temperaturas em Celsius
           TF; // Variável para Temperaturas em Fahrenheit

    TC = 0;
    while (TC <= 100) // Estrutura de Repetição
    {
        TF = 1.8f * TC + 32; // Fórmula para Conversão
        Lbx_while.Items.Add("C = " + Convert.ToString(TC) + " *** "
                          + "F = " + Convert.ToString(TF));
        TC++;
    }
}
```

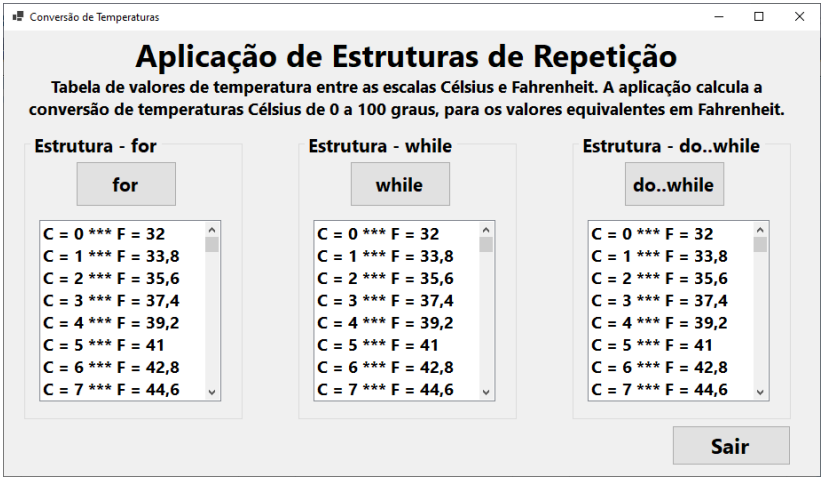
Código do botão **do..while**:

```
1 referência
private void Btn_dowhile_Click(object sender, EventArgs e)
{
    Single TC, // Variável para Temperaturas em Celsius
           TF; // Variável para Temperaturas em Fahrenheit

    TC = 0;
    do // Estrutura de Repetição
    {
        TF = 1.8f * TC + 32; // Fórmula para Conversão
        Lbx_dowhile.Items.Add("C = " + Convert.ToString(TC) + " *** "
                              + "F = " + Convert.ToString(TF));
        TC++;
    } while (TC <= 100);
}
```

Executando a Solução temos:




Após a execução e ao clicar em cada um dos botões, **for**, **while** e **do..while**, temos a próxima imagem com a apresentação das conversões das Temperaturas.


Conversão de Temperaturas
Aplicação de Estruturas de Repetição
Tabela de valores de temperatura entre as escalas Célsius e Fahrenheit. A aplicação calcula a conversão de temperaturas Célsius de 0 a 100 graus, para os valores equivalentes em Fahrenheit.
Estrutura - for
for
C = 0 *** F = 32
C = 1 *** F = 33,8
C = 2 *** F = 35,6
C = 3 *** F = 37,4
C = 4 *** F = 39,2
C = 5 *** F = 41
C = 6 *** F = 42,8
C = 7 *** F = 44,6
Estrutura - while
while
C = 0 *** F = 32
C = 1 *** F = 33,8
C = 2 *** F = 35,6
C = 3 *** F = 37,4
C = 4 *** F = 39,2
C = 5 *** F = 41
C = 6 *** F = 42,8
C = 7 *** F = 44,6
Estrutura - do..while
do..while
C = 0 *** F = 32
C = 1 *** F = 33,8
C = 2 *** F = 35,6
C = 3 *** F = 37,4
C = 4 *** F = 39,2
C = 5 *** F = 41
C = 6 *** F = 42,8
C = 7 *** F = 44,6
Sair

# Capítulo 03

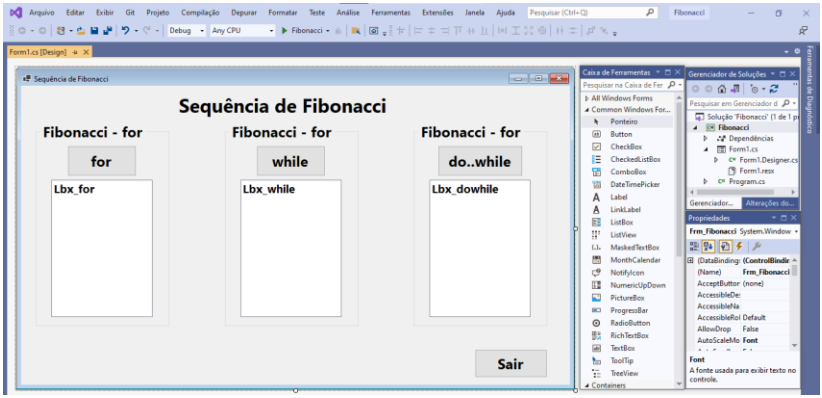
## Mais exemplos importantes

## Sequência de Fibonacci

A **Sequência de Fibonacci** é uma sequência de números, onde o número 1 é o primeiro e segundo termo da ordem e os demais são originados pela soma de seus antecessores. Abaixo um exemplo dos 9 primeiros termos da sequência:

**1 1 2 3 5 8 13 21 54 ...**

O próximo exemplo de Estrutura de Repetição será a apresentação da **Sequência de Fibonacci** para os 15 primeiros termos. São utilizados 3 componentes **ListBox's** para a apresentação da sequência.



O código fonte de cada uma das 3 estruturas de repetição **for**, **while** e **do..while**, para gerar a sequência é apresentado abaixo:

Código do botão **for**:

```
1 referência
private void Btn_for_Click(object sender, EventArgs e)
{
    int T1, T2, T3, i;

    T1 = 1; // Primeiro termo da sequência
    Lbx_for.Items.Add(Convert.ToString(T1));
    T2 = 1; // Segundo termo da sequência
    Lbx_for.Items.Add(Convert.ToString(T2));
    for ( i = 3 ; i <= 15 ; i++) // Estrutura de Repetição
    {
        T3 = T1 + T2; // Cálculo do próximo termo
        Lbx_for.Items.Add(Convert.ToString(T3));
        T1 = T2;
        T2 = T3;
    }
}
```

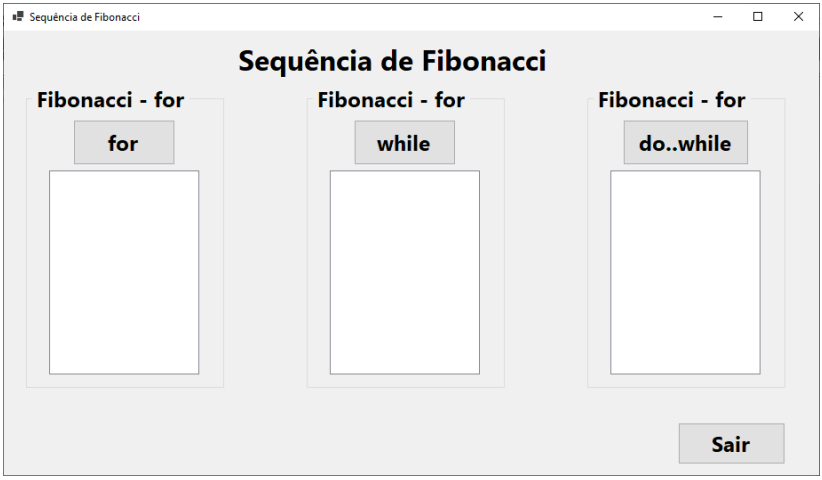
Código do botão **while**:

```
1 referência
private void Btn_while_Click(object sender, EventArgs e)
{
    int T1, T2, T3, i;

    T1 = 1; // Primeiro termo da sequência
    Lbx_while.Items.Add(Convert.ToString(T1));
    T2 = 1; // Segundo termo da sequência
    Lbx_while.Items.Add(Convert.ToString(T2));
    i = 3; // Variável de controle da repetição - contador
    while( i <= 15) // Estrutura de Repetição com teste condicional
    {
        T3 = T1 + T2; // Cálculo do próximo termo
        Lbx_while.Items.Add(Convert.ToString(T3));
        T1 = T2;
        T2 = T3;
        i++; // incremento da variável de controle
    }
}
```

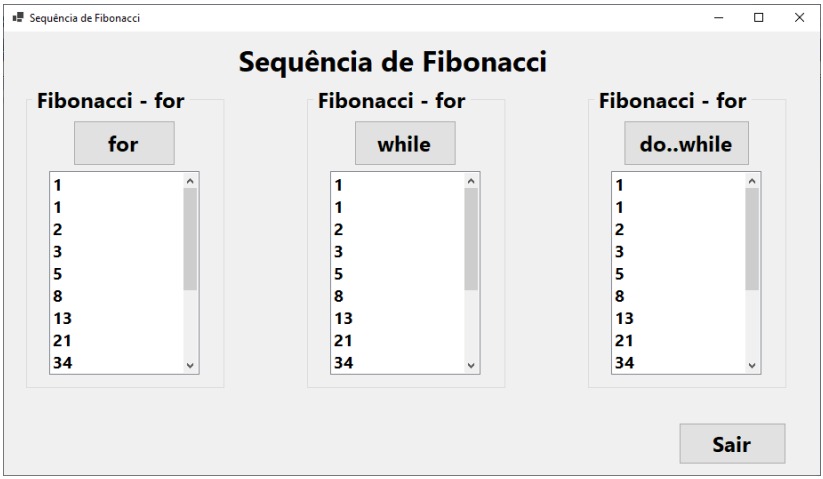
Código do botão **do..while**:

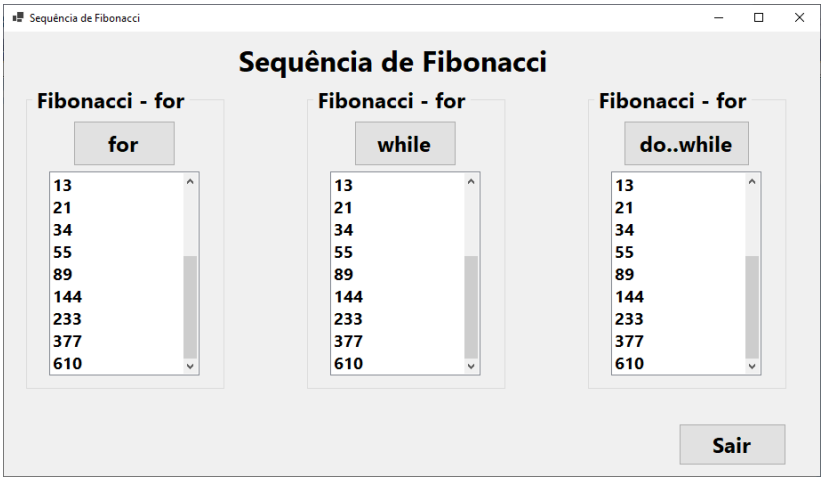
```
1 referência
private void Btn_dowhile_Click(object sender, EventArgs e)
{
    int T1, T2, T3, i;

    T1 = 1; // Primeiro termo da sequência
    Lbx_dowhile.Items.Add(Convert.ToString(T1));
    T2 = 1; // Segundo termo da sequência
    Lbx_dowhile.Items.Add(Convert.ToString(T2));
    i = 3; // Variável de controle da repetição - contador
    do  // Estrutura de Repetição com teste condicional
    {
        T3 = T1 + T2; // Cálculo do próximo termo
        Lbx_dowhile.Items.Add(Convert.ToString(T3));
        T1 = T2;
        T2 = T3;
        i++; // incremento da variável de controle
    } while (i <= 15); // Teste condicional no final
}
```

Executando a Solução temos:



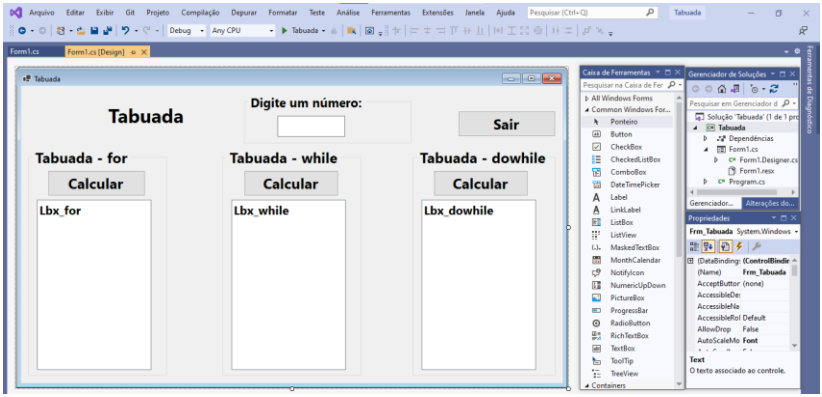
Após a execução e ao clicar em cada um dos botões, **for**, **while** e **do..while**, temos a próxima imagem com a apresentação dos 15 termos da Sequência de Fibonacci.



# Tabuada

O próximo exemplo de Estrutura de Repetição será a apresentação da Tabuada de um número fornecido pelo usuário.




Código do botão **for**:

```
1 referência
private void Btn_Calcular_Click(object sender, EventArgs e)
{
    int i,   // Variável de controle - Contador
        Num, // Número digitado pelo usuário
        Res; // Resultado da operação

    Lbx_for.Items.Clear();
    Num = Convert.ToInt32(txt_Numero.Text); // Conversão do número
    for ( i = 1; i <= 10; i++) // Estrutura de Repetição
    {
        Res = i * Num; // Cálculo da Tabuada
        Lbx_for.Items.Add(  Convert.ToString(i)   + " * " +
                            Convert.ToString(Num) + " = " +
                            Convert.ToString(Res));
    }
    txt_Numero.Focus(); // Retorna o foco da aplicação ao TextBox
}
```

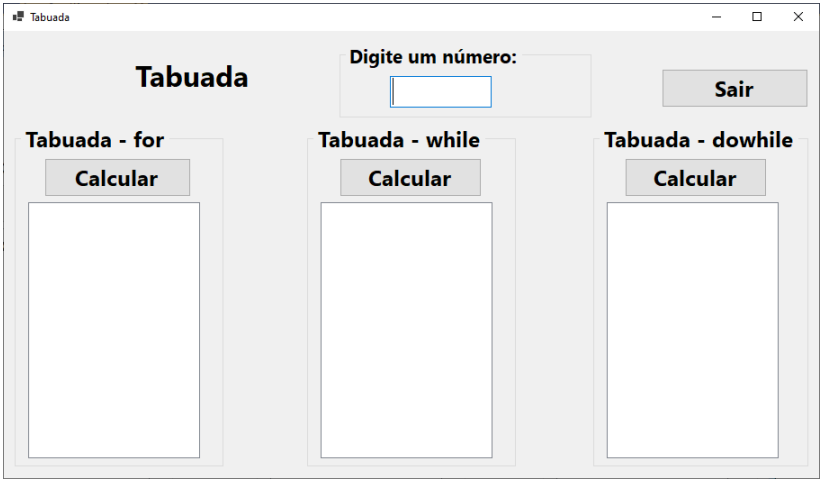
Código do botão **while**:

```
1 referência
private void Btn_Calcular_while_Click(object sender, EventArgs e)
{
    int i,   // Variável de controle - Contador
        Num, // Número digitado pelo usuário
        Res; // Resultado da operação

    Lbx_while.Items.Clear();
    Num = Convert.ToInt32(txt_Numero.Text); // Conversão do número
    i = 1; // Inicializando o contador
    while ( i <= 10) // Estrutura de Repetição com teste condicional
    {
        Res = i * Num; // Cálculo da Tabuada
        Lbx_while.Items.Add(Convert.ToString(i) + " * " +
                            Convert.ToString(Num) + " = " +
                            Convert.ToString(Res));
        i++; // Incrementando o contador
    }
    txt_Numero.Focus(); // Retorna o foco da aplicação ao TextBox
}
```

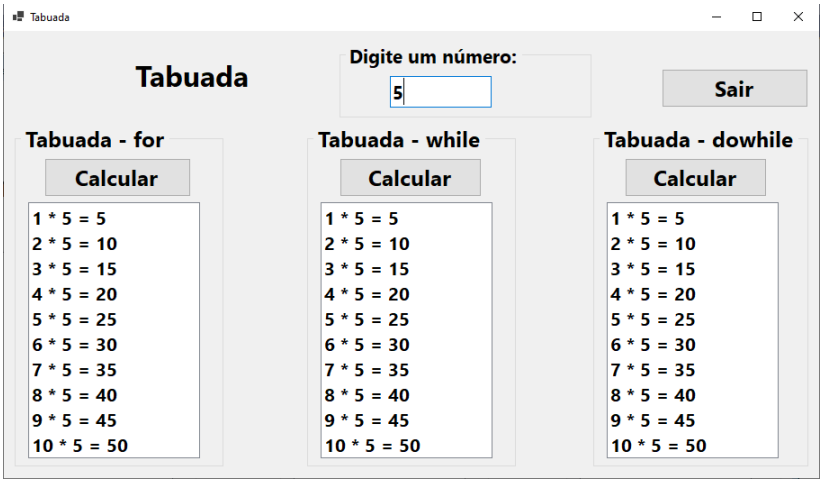
Código do botão **do..while**:

```
1 referência
private void Btn_Calcular_dowhile_Click(object sender, EventArgs e)
{
    int i,   // Variável de controle - Contador
        Num, // Número digitado pelo usuário
        Res; // Resultado da operação

    Lbx_dowhile.Items.Clear();
    Num = Convert.ToInt32(txt_Numero.Text); // Conversão do número
    i = 1; // Inicializando o contador
    do // Estrutura de Repetição com teste condicional
    {
        Res = i * Num; // Cálculo da Tabuada
        Lbx_dowhile.Items.Add(Convert.ToString(i) + " * " +
                            Convert.ToString(Num) + " = " +
                            Convert.ToString(Res));
        i++; // Incrementando o contador
    } while (i <= 10); // Teste condicional no final
    txt_Numero.Focus(); // Retorna o foco da aplicação ao TextBox
}
```

Executando a Solução temos:



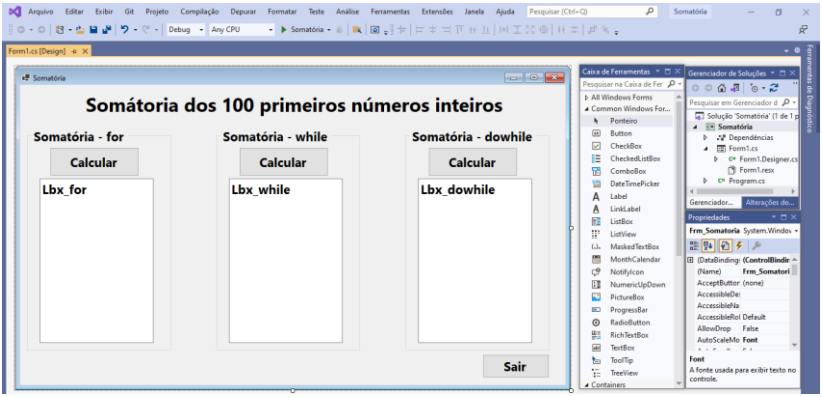

Após a execução, informar um número na caixa TextBox e ao clicar em cada um dos botões **Calcular**, temos a próxima imagem com a apresentação da Tabuada do número fornecido.



# Somatória

O próximo exemplo utilizando Estrutura de Repetição é um exemplo clássico, utilizado por diversos autores, será o cálculo e a apresentação da Somatória dos 100 primeiros números inteiros.

$$1 + 2 + 3 + 4 + 5 + \ldots\ldots\ldots + 98 + 99 + 100 = 5050$$




Código do botão **for**:

```
1 referência
private void Btn_Calcular_for_Click(object sender, EventArgs e)
{
    int i, // Variável de controle - contador
        soma, // Variável para resultado da Somatória
        ant; // Variável auxiliar

    Lbx_for.Items.Clear(); // Apaga todos os itens do ListBox
    soma = 0; // Iniciando com zero
    for( i = 1; i <= 100; i++)
    {
        ant = soma;
        soma = ant + i; // Cálculo da somatória a cada iteração
        Lbx_for.Items.Add(  Convert.ToString(ant) + " + " +
                            Convert.ToString(i) +   " = " +
                            Convert.ToString(soma));
    }
}
```

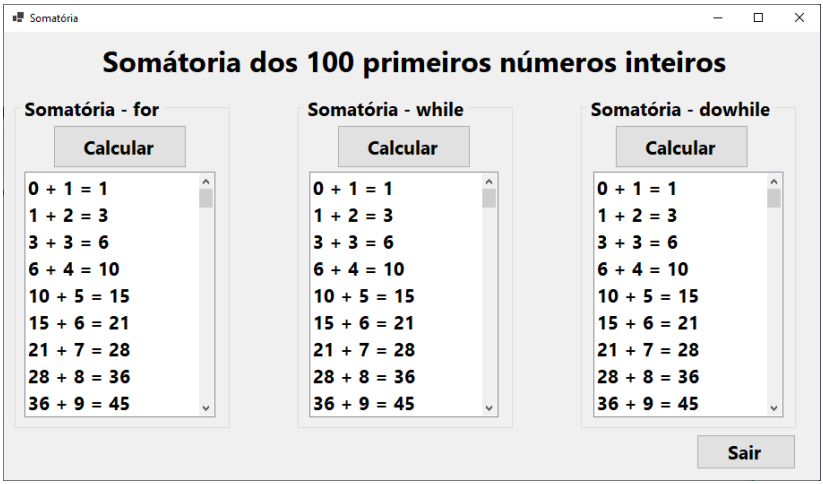
Código do botão **while**:

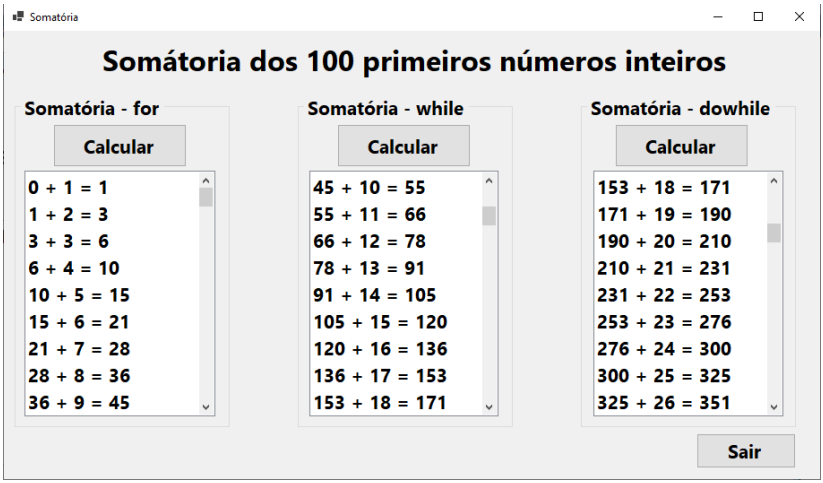
```
1 referência
private void Btn_Calcular_while_Click(object sender, EventArgs e)
{
    int i, // Variável de controle - contador
        soma, // Variável para resultado da Somatória
        ant; // Variável auxiliar

    Lbx_while.Items.Clear(); // Apaga todos os itens do ListBox
    soma = 0; // Iniciando com zero
    i = 1; // Inicializando o contador
    while ( i <= 100) // Estrutura com teste condicional no início
    {
        ant = soma;
        soma = ant + i; // Cálculo da somatória a cada iteração
        Lbx_while.Items.Add(Convert.ToString(ant) + " + " +
                            Convert.ToString(i) + " = " +
                            Convert.ToString(soma));
        i++; // Incrementando o contador
    }
}
```

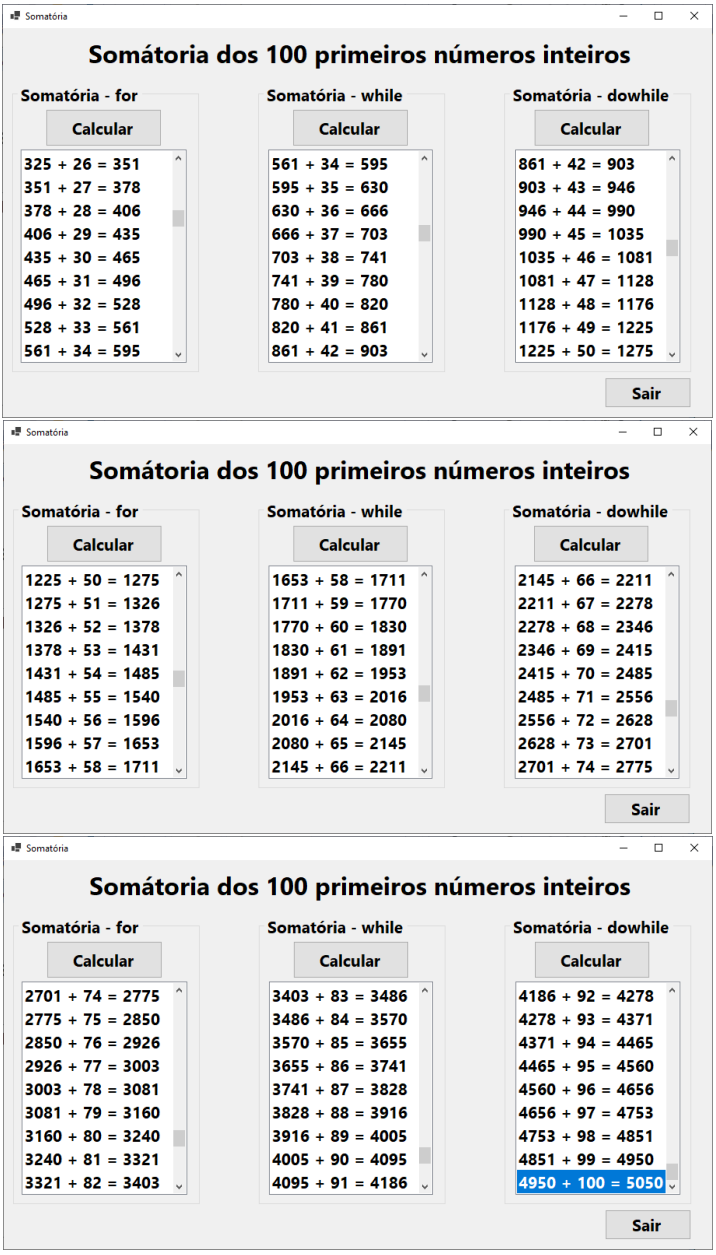
Código do botão **do..while**:

```
1 referência
private void Btn_Calcular_dowhile_Click(object sender, EventArgs e)
{
    int i, // Variável de controle - contador
        soma, // Variável para resultado da Somatória
        ant; // Variável auxiliar

    Lbx_dowhile.Items.Clear(); // Apaga todos os itens do ListBox
    soma = 0; // Iniciando com zero
    i = 1; // Inicializando o contador
    do // Estrutura com teste condicional no final
    {
        ant = soma;
        soma = ant + i; // Cálculo da somatória a cada iteração
        Lbx_dowhile.Items.Add(Convert.ToString(ant) + " + " +
                            Convert.ToString(i) + " = " +
                            Convert.ToString(soma));
        i++; // Incrementando o contador
    } while (i <= 100); // teste condicional no final
}
```

Executando a Solução e ao clicar em cada um dos botões **Calcular**, temos a próxima imagem com a apresentação da Somatória dos 100 primeiros números inteiros, passo a passo, totalizando 5050.



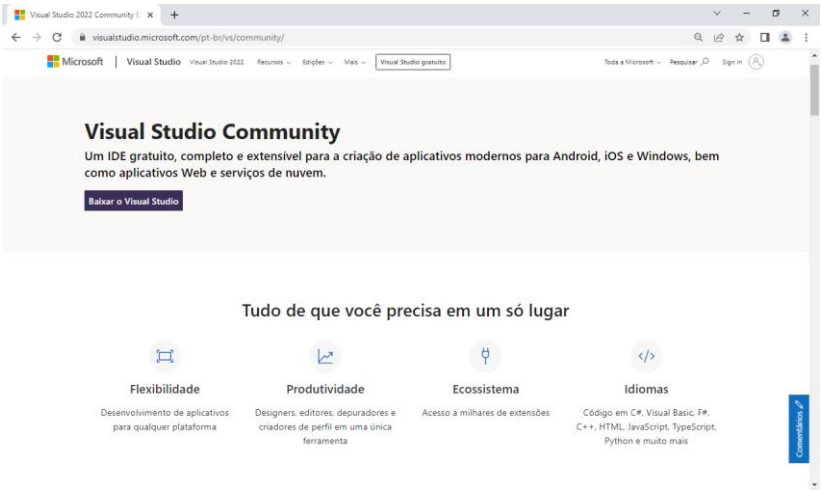




Somatória
Somatória dos 100 primeiros números inteiros
Somatória - for
Calcular
325 + 26 = 351
351 + 27 = 378
378 + 28 = 406
406 + 29 = 435
435 + 30 = 465
465 + 31 = 496
496 + 32 = 528
528 + 33 = 561
561 + 34 = 595
Somatória - while
Calcular
561 + 34 = 595
595 + 35 = 630
630 + 36 = 666
666 + 37 = 703
703 + 38 = 741
741 + 39 = 780
780 + 40 = 820
820 + 41 = 861
861 + 42 = 903
Somatória - dowhile
Calcular
861 + 42 = 903
903 + 43 = 946
946 + 44 = 990
990 + 45 = 1035
1035 + 46 = 1081
1081 + 47 = 1128
1128 + 48 = 1176
1176 + 49 = 1225
1225 + 50 = 1275
Sair
Somatória
Somatória dos 100 primeiros números inteiros
Somatória - for
Calcular
1225 + 50 = 1275
1275 + 51 = 1326
1326 + 52 = 1378
1378 + 53 = 1431
1431 + 54 = 1485
1485 + 55 = 1540
1540 + 56 = 1596
1596 + 57 = 1653
1653 + 58 = 1711
Somatória - while
Calcular
1653 + 58 = 1711
1711 + 59 = 1770
1770 + 60 = 1830
1830 + 61 = 1891
1891 + 62 = 1953
1953 + 63 = 2016
2016 + 64 = 2080
2080 + 65 = 2145
2145 + 66 = 2211
Somatória - dowhile
Calcular
2145 + 66 = 2211
2211 + 67 = 2278
2278 + 68 = 2346
2346 + 69 = 2415
2415 + 70 = 2485
2485 + 71 = 2556
2556 + 72 = 2628
2628 + 73 = 2701
2701 + 74 = 2775
Sair
Somatória
Somatória dos 100 primeiros números inteiros
Somatória - for
Calcular
2701 + 74 = 2775
2775 + 75 = 2850
2850 + 76 = 2926
2926 + 77 = 3003
3003 + 78 = 3081
3081 + 79 = 3160
3160 + 80 = 3240
3240 + 81 = 3321
3321 + 82 = 3403
Somatória - while
Calcular
3403 + 83 = 3486
3486 + 84 = 3570
3570 + 85 = 3655
3655 + 86 = 3741
3741 + 87 = 3828
3828 + 88 = 3916
3916 + 89 = 4005
4005 + 90 = 4095
4095 + 91 = 4186
Somatória - dowhile
Calcular
4186 + 92 = 4278
4278 + 93 = 4371
4371 + 94 = 4465
4465 + 95 = 4560
4560 + 96 = 4656
4656 + 97 = 4753
4753 + 98 = 4851
4851 + 99 = 4950
4950 + 100 = 5050
Sair

# APÊNDICE - 1

# O que é o Visual Studio Community?

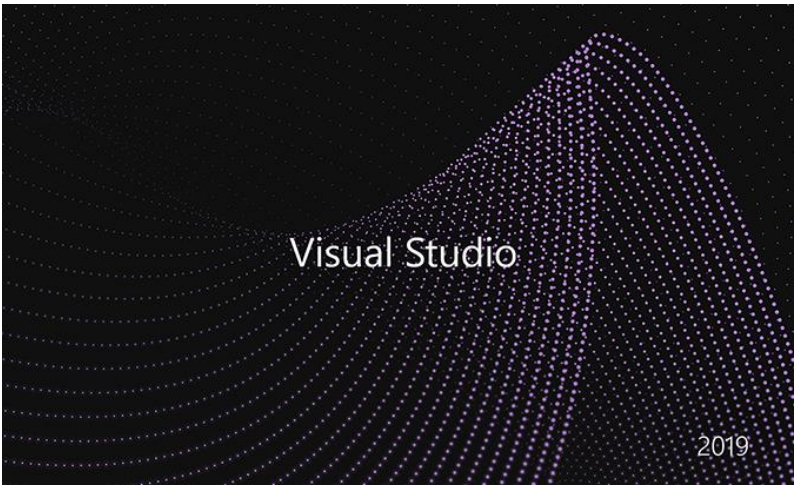
O **Visual Studio Community** é um **IDE** gratuito e completo para alunos, colaboradores de código aberto e desenvolvedores individuais. Para fazer o download, acesse o endereço do site https://visualstudio.microsoft.com/pt-br/vs/community/ e clique em **Baixar o Visual Studio**.




Para desenvolver qualquer tipo de aplicativo, você trabalhará no Ambiente de Desenvolvimento Integrado (IDE) do Visual Studio. Além da edição de código, a IDE do Visual Studio reúne designers gráficos, compiladores, ferramentas de conclusão de código, controle do código-fonte, extensões e muitos outros recursos em um só lugar.

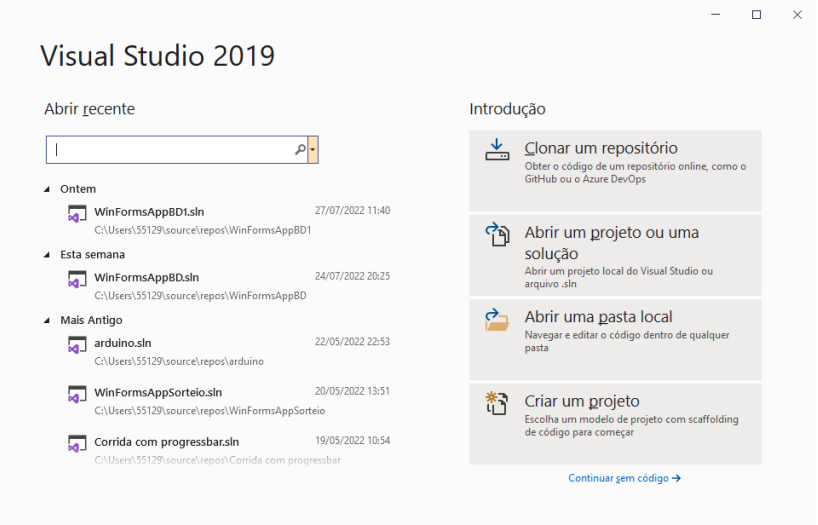
Após a instalação do Visual Studio escolha a opção no menu iniciar 

para execução do aplicativo, a seguinte imagem aparecerá:

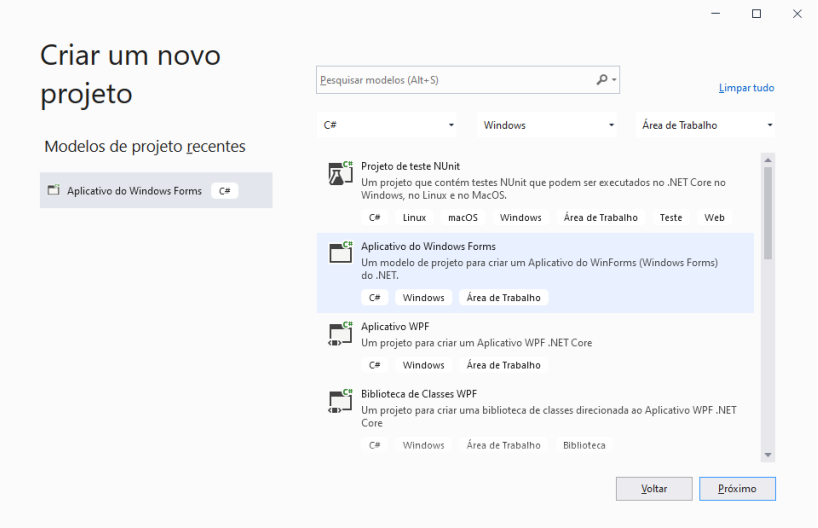



## Windows Form com C#

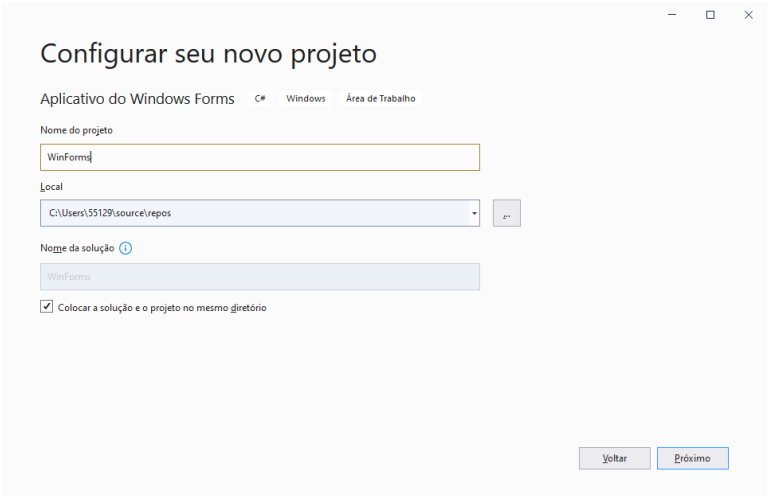
Para desenvolver uma interface gráfica no Visual Studio devemos escolher a opção **Criar um Projeto** na próxima imagem, na sequência será possível escolher o modelo de projeto para começar.



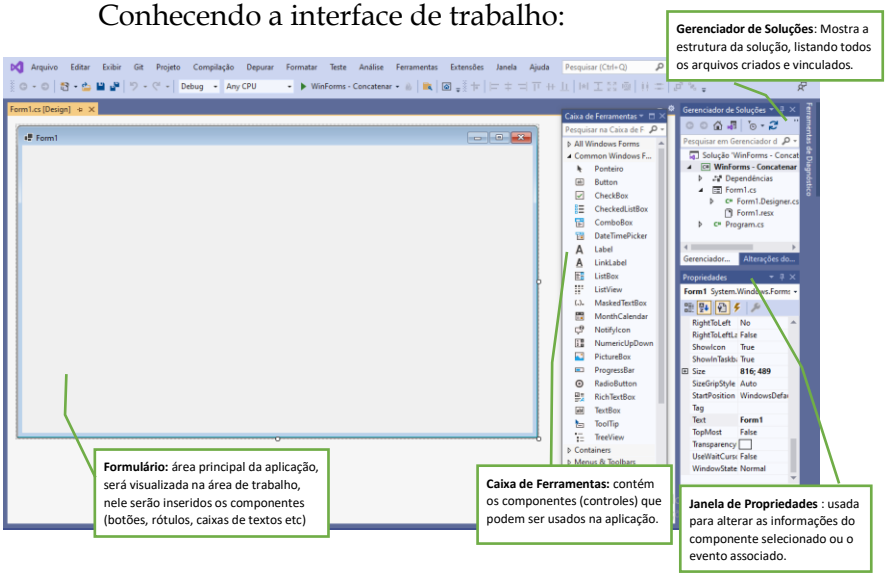

Selecione a linguagem **C#**, a plataforma **Windows**, o destino **Área de Trabalho** e o modelo **Aplicativo do Windows Forms**.



Agora você deverá definir o **Nome do projeto** e o **Local** de armazenamento e logo após clicar no botão **Próximo**:




Conhecendo a interface de trabalho:



# Tipos de Dados

Dados são as informações processadas pelo computador e podem ser de vários tipos. A linguagem **C#** possui um conjunto definido de dados podendo ser eles lógicos, caracteres e numéricos:

- **inteiros**: usamos o tipo **int** que corresponde aos números que pertencem aos conjuntos N e Z. Não possuem casas decimais e podem ser negativos.

  Ex.: 25, -57 e 0;

- **real**: usamos o tipo **single** que corresponde aos números que pertencem aos conjuntos R e Q. Podem possuir casas decimais e podem ser positivos ou negativos.

  Ex.: 0.01, 12.1 e -37.57;

- **caractere**: usamos o tipo **string** que são dados formados por uma sequência de caracteres, que podem ser letras, números ou símbolos especiais. Também são chamados de alfanuméricos, caracteres ou literais.

  Ex.: "Rua 02, nº 123";

- **lógico**: define variáveis do tipo booleano, ou seja, com valor VERDADEIRO ou FALSO. A Linguagem **C#** utilizará 0 e 1 para considerar essa informação. Quando o valor for igual a 0, será considerado um valor falso e quando o valor for 1, será considerado um valor verdadeiro.

# Variável

A quantidade de memória para armazenar uma informação varia de acordo com o tipo de dado, tipo de computador e tipo de linguagem. Variável é o nome dado a uma entidade destinada a guardar uma informação. Possui três atributos: **Nome, Tipo de Dado e Informação**.

**Nome**: diferencia uma variável de outra. Cada linguagem tem sua regra de criação de variáveis. Para a linguagem C# vamos considerar o seguinte:

1. Uma variável pode ter um ou mais caracteres na formação de seu nome;
2. O nome deve começar com uma letra e, de nenhuma maneira poderá se iniciar com um número. Veja isso: NUM10 é válido; 10NUM é inválido;
3. Dê um nome a sua variável de forma que não tenha espaços em branco e nem utilize caracteres que não sejam letras e números. Porém, podemos utilizar o caractere underscore. Veja como podemos utilizá-lo corretamente: VALOR_TAXA. Assim evitamos o espaço em branco entre VALOR e TAXA, utilizando um caractere permitido;
4. Um outro detalhe muito importante em **C#** é que ele diferencia caracteres maiúsculos dos minúsculos.

Assim sendo, quando ele "olhar" para a variável com nome VALOR, ele irá diferenciar da variável Valor por exemplo.

O conceito de variável facilita a programação, permitindo acessar dados por meio de um nome, em vez do endereço de uma célula. As variáveis devem ser definidas antes de serem usadas. Isto permite que se possa reservar um espaço na memória para elas.

## Operadores Aritméticos

São aqueles que o resultado da avaliação é do tipo numérico. Somente é permitido o uso com variáveis e constantes numéricas.

| | |
|---|---|
| **+**<br>**-** | Operadores unários, isto é, são aplicados a um único operando. São os operadores aritméticos de maior precedência. Exemplos: -3, +x. Enquanto o operador unário **-** inverte o sinal do seu operando, o operador **+** não altera o valor em nada o seu valor. |
| \ | Operador de divisão inteira. Por exemplo, 5 \ 2 = 2. Tem a mesma precedência do operador de divisão tradicional. |

| | |
|---|---|
| **+**<br>**-**<br>*<br>**/** | Operadores aritméticos tradicionais de adição, subtração, multiplicação e divisão. Por convenção, * e **/** têm precedência sobre **+** e **-**. Para modificar a ordem de avaliação das operações, é necessário usar parênteses como em qualquer expressão aritmética. |
| % | Operador de módulo (isto é, resto da divisão inteira). Por exemplo, 5 % 2 = 1. Tem a mesma precedência do operador de divisão tradicional. |

## Operadores Relacionais

São aqueles utilizados para efetuar uma comparação. Fornece como resultado um valor lógico. São eles:

| Operador | Comparação |
|---|---|
| == | Igual |
| != | diferente |
| < | menor |
| > | maior |
| < = | menor igual a |
| > = | maior igual a |

# Operadores Lógicos

São aqueles cujo resultado da avaliação é um valor lógico.

| | |
|---|---|
| **!** | Operador de negação **não** VERDADEIRO = FALSO, e **não** FALSO = VERDADEIRO. Esse operador inverte o estado lógico de uma condição |
| **\|\|** | Operador que resulta VERDADEIRO quando um dos seus relacionamentos lógicos for verdadeiro. Esse símbolo é obtido apertando o shift e a tecla de barra invertida ao lado da letra Z. |
| **&&** | Operador que resulta VERDADEIRO somente se seus dois ou mais relacionamentos lógicos forem simultaneamente verdadeiros. |

**Exemplo**: Dadas as variáveis **A** e **B** sendo lógicas, abaixo temos todas as possibilidades de avaliação:

| **A** | **B** | **!A** | **! B** | **A && B** | **A \|\| B** |
|---|---|---|---|---|---|
| **V** | **V** | F | F | V | V |
| **V** | **F** | F | V | F | V |
| **F** | **V** | V | F | F | V |
| **F** | **F** | V | V | F | F |

A tabela anterior é conhecida como **Tabela Verdade**. Podemos concluir, que:

- O operador **!** inverte o valor do operando.
- O operador **||** tem como resultado **F** somente se todos os operandos avaliados forem também **F**.
- O operador **&&** tem como resultado **V** somente se todos os operandos avaliados forem também **V**.

# APÊNDICE - 2

## A classe Math

Fornece constantes e métodos estáticos para trigonométricas, logarítmicas e outras funções matemáticas comuns. (Acesso em https://learn.microsoft.com/pt-br/dotnet/api/system.math?view=net-7.0)

A tabela a seguir (Acesso em https://www.macoratti.net/13/03/c_mat1.htm) contém alguns exemplos de funções matemáticas e constantes existentes na classe **System.Math**:

| Função/ Constantes | Resultado | Exemplo |
|---|---|---|
| **Abs(x)** | Fornece Valor absoluto de x | **Abs(4.5) = 4.5;**<br>**Abs(-4.5) = 4.5;** |
| **Ceiling(x)** | Arredonda x para cima | **Ceiling(4.1) = 5;**<br>**Ceiling(-4.1) = -4;** |
| **Cos(x)** | Obtém o cosseno de x | **Cos(2.0) = -0.4161.....** |
| **Sin(x)** | Obtém o seno de x | **Sin(2) = 0.909297...** |
| **Tan(x)** | Obtém o valor da Tangente de x | **Tan(1.5) = 14.1014...** |

| | | |
|---|---|---|
| **PI** | Obtém o valor de PI | **PI = 3.141516171819...** |
| **E** | Obtém o valor da constante E | **E = 2.7182818284590451** |
| **Exp(x)** | Obtém o exponencial (*e* elevado na x) | **Exp(5.0) = 54.59...** |
| **Floor(x)** | Arredonda o valor de x para baixo | **Floor(2.9) = 2;**<br>**Floor(-2.9)) = -3;** |
| **Log(x)** | Calcula o logaritmo de x na base natural *e* | **Log(3.0) = 1.098612...** |
| **Log10(x)** | Calcula o logaritmo de x na base 10 | **Log10(3.0) = 0.47712...** |
| **Max(x,y)** | Obtém o maior valor entre dois números | **Max(2.46,2.56) = 2.56;**<br>**Max(-2.46,-2.56) = 2.46;** |
| **Min(x,y)** | Obtém o menor valor entre dois números | **Min(1.92,1.89) = 1.89;**<br>**Min(-1.92,-1.89) = 1.92;** |
| **Pow(x,y)** | Obtém o valor de x elevado na y | **Pow(2,4) = 16** |
| **Round(x,y)** | Arredonda x para y casas decimais | **Round(7.6758549,4) = 7.6759** |
| **Sqrt(x)** | Obtém a raiz quadrada de x | **Sqrt(169) = 13** |

Acesso em https://www.macoratti.net/13/03/c_mat1.htm

# Referências

**FORBELLONE**, André Luiz; Henri Eberspacher. **Lógica de Programação:** A Construção de Algoritmos e Estruturas de Dados. 3ª Edição. Editora Pearson. 2005.

**GONÇALVES**, Reginaldo Luiz; Abud, Gilberto J. **Algoritmos e Programação:** Exemplos Práticos. 1ª Edição. Volume 1. Worges Editoração. Belém-Pará. 2022.

**GONÇALVES**, Reginaldo Luiz; Abud, Gilberto J**. Programação em Linguagem C:** Exemplos Práticos. 1ª Edição. Volume 2. Worges Editoração. Belém-Pará. 2022.

**GONÇALVES**, Reginaldo Luiz; Abud, Gilberto J. **Programando em JavaScript:** Exemplos Práticos. 1ª Edição. Volume 3. Worges Editoração. Belém-Pará. 2022.

**GONÇALVES**, Reginaldo Luiz; Abud, Gilberto J. **Programando em PHP:** Exemplos Práticos. 1ª Edição. Volume 4. Worges Editoração. Belém-Pará. 2022.

**GONÇALVES**, Reginaldo Luiz; Abud, Gilberto J. **Apostila de C#: Conceitos Básicos – Versão 1.0**. 1ª Edição. Worges Editoração. Belém-Pará. 2022.

**JANSA**, KRIS; KLANDER, LARS. **Programando em C/C++ - A Bíblia**. Makron Books do Brasil Editora Ltda. 1999. São Paulo.

**KOBAYACHI**, Cintia. Érica Luciane Beu. **Webdesigner**: Estrutura e Programação. Editora Érica. 2001.

**MANZANO**, J.A.N.G.: **Linguagem C:** Estudo Dirigido. Editora Érica, 1997. São Paulo.

**SANTOS**, Jaime Evaristo dos. **Aprendendo a Programar, Programando em C**. Book Express. 2001.

**Sites**

**Acesso em 08/04/2023**. https://learn.microsoft.com/pt-br/dotnet/api/system.math?view=net-7.0

**Acesso em 08/04/2023**. https://www.macoratti.net/13/03/c_mat1.htm

# Comentários finais

Convido a todos a lerem e estudarem também os nossos livros da **Coleção Programando: Aprenda Rápido**, Algoritmos e Programação: Exemplos Práticos, Programação em Linguagem C: Exemplos Práticos , Programando em JavaScript: Exemplos Práticos e Programando em PHP: Exemplos Práticos. Os recursos do PHP unido ao JavaScript, HTML e CSS são inúmeros e merecem ser estudados.

Leiam também as Apostilas de C#: Conceitos Básicos – Versão 1.0 e a Apostila de SQL: Conceitos Básicos – Versão 1.0.

Bons estudos aos leitores!

Home Editora
CNPJ: 39.242.488/0002-80
www.homeeditora.com
contato@homeeditora.com
9198473-5110
Av. Augusto Montenegro, 4120 - Parque Verde, Belém - PA, 66635-110