





## Alex Sander Resende de Deus

A 25 anos ensinando programação a jovens e adultos.

Apaixonado por tecnologia é atualmente coordenador de cursos na ETEC Albert Einstein. Na FIAP atua como professor na FIAP School, lecionando C#, SQLServer e Desenvolvimento Mobile


## Aula 04

☒ One

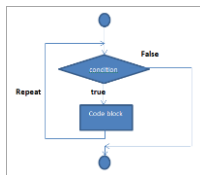
☐ Two

☐ Three

Radio Button



ComboBox



Laços de Repetição

# RADIO BUTTON

---



— □ ●

• • • + • □

• ● +

- Um botão de opção ou botão de rádio é um elemento de interface gráfica com **dois estados: selecionado e não-selecionado**, quando o usuário clica com o mouse ou pressiona a tecla espaço. Botões de opção são usados num grupo para apresentar um conjunto limitado de escolhas que são exclusivas

•

□

+ + •

• • • •

□ • • ● ●

Exemplo: Elabore um programa que identifique qual radiobutton foi selecionada: Masculino ou Feminino

MainWindow

Selecione o sexo:

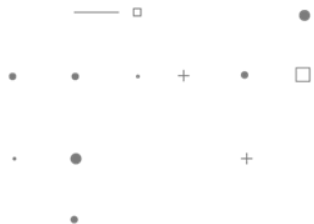
☐ Masculino

☐ Feminino

```
private void btnChecar_Click(object sender, RoutedEventArgs e)
{
    if (rdbMasc.IsChecked == true)
    {
        MessageBox.Show("Você selecionou o sexo masculino", "Atenção",
            MessageBoxButton.OK, MessageBoxImage.Exclamation);
    }
    else
    {
        MessageBox.Show("Você selecionou o sexo feminino", "Atenção",
            MessageBoxButton.OK, MessageBoxImage.Exclamation);
    }
}
```

# ComboBox

---





- ComboBox é usado para exibir dados **em uma caixa de combinação listada**. Por padrão, o controle aparece em duas partes: a parte superior é uma caixa de texto que permite que o usuário digite um item de lista. A segunda parte é uma caixa de listagem que exibe uma lista de itens na qual o usuário pode selecionar item.

Para descobrir qual o item foi selecionado normalmente utilizamos a posição do item na lista através de um **switch case**.





Vamos refazer o programa anterior utilizando a ComboBox

# PROGRAMANDO EM C#

---

Laços de Repetição  
**WHILE e FOR**

# LOOP WHILE

---

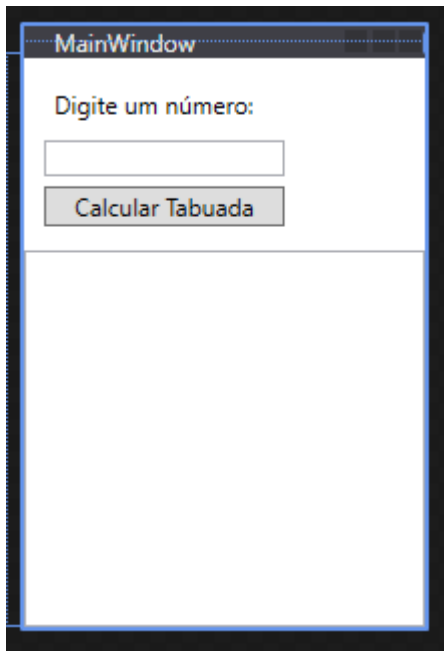


Uma das situações mais recorrentes na vida de um desenvolvedor de sistemas é precisar que um determinado trecho do código seja repetido diversas vezes.

Quem resolve essa questão são os loops. Loops são estruturas de programação capazes de repetir a execução de um determinado trecho do código.

Está difícil de imaginar a utilidade? Então vamos ver um caso prático.

Imagine um programa que leia um valor qualquer e calcule a tabuada deste número.



```
private void btnTabuada_Click(object sender, RoutedEventArgs e)
{
    int numero, res;
    numero = Convert.ToInt32(txtNumero.Text);

    res = numero * 1;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 2;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 3;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 4;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 5;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 6;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 7;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 8;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 9;
    lstTabuada.Items.Add(res.ToString());
    res = numero * 10;
    lstTabuada.Items.Add(res.ToString());
}
```



Agora que identificamos em nosso programa uma situação na qual há a repetição de diversas instruções, podemos utilizar um loop para nos ajudar.

O loop While é um loop baseado em condição, ou seja, ele continua executando enquanto uma determinada condição é verdadeira.

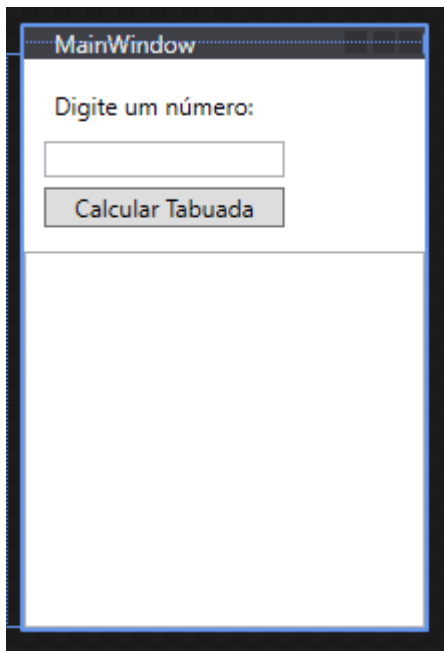




```
while (condição) {  
    //instruções que serão repetidas enquanto a condição for  
    verdadeira  
}
```

**A sintaxe do loop While permite que indiquemos uma condição e um conjunto de instruções que devem ser repetidas enquanto a condição for verdadeira.**

Com um laço de repetição veja a diferença de código



```
private void btnTabuada_Click(object sender, RoutedEventArgs e)
{
    int numero, res, cont;
    numero = Convert.ToInt32(txtNumero.Text);
    cont = 1;
    while (cont <= 10)
    {
        res = numero * cont;
        lstTabuada.Items.Add(res.ToString());
        cont++;
    }
}
```



# LOOP FOR

---

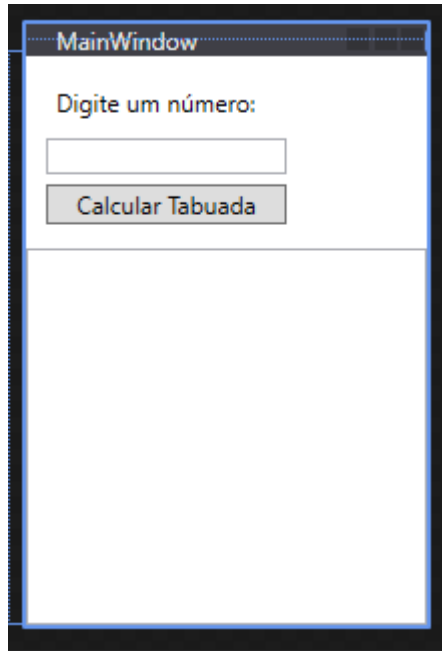


Já sabemos o que são Loops: estruturas de programação capazes de repetir a execução de um determinado trecho do código.

Enquanto um loop **While é baseado em uma condição** e é executado enquanto a condição é satisfeita, temos diferentes tipos de loops para diferentes cenários (alguns que só exploraremos no curso de Orientação a Objetos, por exemplo).

Hoje conheceremos o **loop For, que é baseado em um contador**.

Vamos refazer o mesmo programa (tabuada) utilizando o **looping FOR**



```
private void btnTabuada_Click(object sender, RoutedEventArgs e)
{
    int numero, res;
    numero = Convert.ToInt32(txtNumero.Text);

    for (int cont=1; cont <= 10; cont++)
    {
        res = numero * cont;
        lstTabuada.Items.Add(res.ToString());
    }
}
```

```
for (variavel contadora; condição de parada; incremento){
    //instruções que serão repetidas
}
```

Essa é a sintaxe do loop For. Com ele o programador não precisa se preocupar em criar antecipadamente a variável contadora e nem corre o risco de esquecer o incremento e acabar criando um loop infinito.

```
for (variavel contadora; condição de parada; incremento){
    //instruções que serão repetidas
}
```

No primeiro parâmetro desse loop devemos indicar qual é a variável contadora e qual é seu valor inicial.

Se indicarmos algo como *i=0*, a variável *i* será nossa contadora e iniciará valendo zero, porém terá que ser criada anteriormente pelo programador.

Se indicarmos *int i=0*, a variável *i* será criada automaticamente no início do loop e destruída após ele, economizando espaço em memória.

```
for (variavel contadora; condição de parada; incremento){
    //instruções que serão repetidas
}
```

A condição é similar à que escreveríamos em um loop while.

A lógica é: enquanto a condição for verdadeira, o loop continua sendo executado.

```
for (variavel contadora; condição de parada; incremento){
    //instruções que serão repetidas
}
```

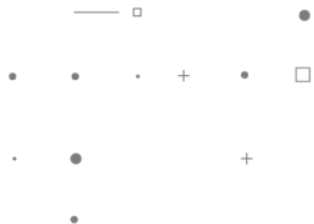
O incremento, por sua vez, deve apresentar quanto será somado ou subtraído da variável contadora a cada volta do loop.

Se escrevermos `i++`, `i=i+1`, `i=i+2` ou qualquer outra adição, a variável será incrementada a cada volta do loop.

Se escrevermos `i--`, `i=i-1`, `i=i-2`, por outro lado, a variável será decrementada a cada volta do loop.

# LOOP DO WHILE

---



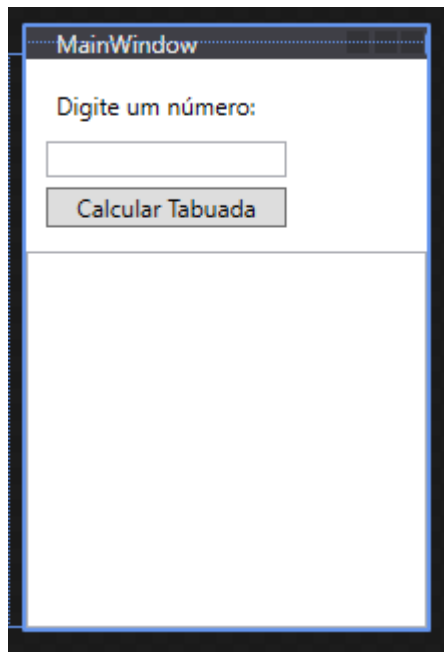




- Um **loop do while** é uma instrução de fluxo de controle que executa um bloco de código pelo **menos uma vez** e depois executa o bloco repetidamente ou para de executá-lo, dependendo de uma determinada condição booleana no final do bloco .



Vamos agora analisar o programa tabuada utilizando o **looping DO WHILE**



```
private void btnTabuada_Click(object sender, RoutedEventArgs e)
{
    int numero, res, cont;
    numero = Convert.ToInt32(txtNumero.Text);
    cont = 1;
    do
    {
        res = numero * cont;
        lstTabuada.Items.Add(res.ToString());
        cont++;
    } while (cont <= 10);
}
```



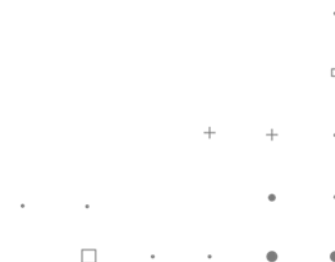
# Momento Hands On





- Utilizando o **looping while**, crie um programa que leia um número qualquer fornecido pelo usuário e calcule o **fatorial deste número**. Exemplo:

$$5! = 5 \times 4 \times 3 \times 2 \times 1 = 120$$





- Utilizando o **looping for**, elabore um programa que leia dois números quaisquer e exiba o somatório de todos os números que estão entre os dois digitados. Lembre-se que o primeiro número deve ser sempre menor que o segundo.





- Utilizando o **looping do while**, elabore um programa que leia dois números quaisquer e exiba a quantidade de números pares entre os dois valores digitados. Lembre-se que o primeiro número deve ser sempre menor que o segundo.



# Lembre-se de versionar seu projeto

---



# OBRIGADO



profalex.deus@fiap.com.br



[linkedin.com/in/alexanderresende](https://www.linkedin.com/in/alexanderresende)

FIAP MBA<sup>+</sup>

Copyright © 2019 | Professor (a) Nome do Professor

Todos os direitos reservados. Reprodução ou divulgação total ou parcial deste documento, é expressamente proibido sem consentimento formal, por escrito, do professor/autor.



FIAP