

## Introdução a Programação orientada a objetos

Professor Sergio Souza Novak



# Introdução a Programação Orientada a Objetos

- Antes de começar orientação a objetos, precisamos revisar os princípios da programação de computadores.
- Também é necessário conhecer o básico de C#

# Variáveis



# Variáveis

- As variáveis guardam informações de um **tipo** específico.

```
int i = 1;
```

```
Point p = new Point(11, 11);
```

# Variáveis

***Declarar várias variáveis para os alunos utilizando a IDE Visual Studio...***

# Variáveis

- Além do tipo int (para representar inteiros), temos também os tipos double e float (para números reais), string (para textos), entre outros.

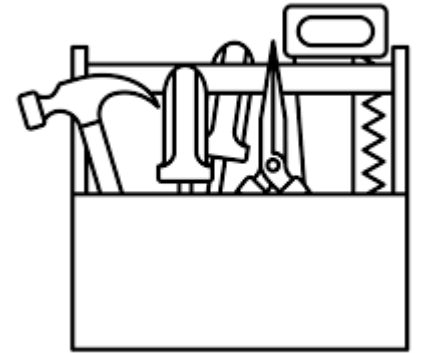
# Variáveis

## ***Discursar a respeito do Código Limpo***

- Evite informações erradas
- Use nomes passíveis de busca
- Use nomes pronunciáveis
- Evite codificações
- Use nomes do domínio do problema



# Tipos Primitivos





# Tipos Primitivos

- C# toda variável possui um tipo, utilizamos o **int** quando queremos armazenar valores inteiros e **double** para números reais. Agora vamos descobrir quais são os outros tipos de variáveis do C#.

Tipo	Tamanho	Valores Possíveis
bool	1 byte	true e false
byte	1 byte	0 a 255
sbyte	1 byte	-128 a 127
short	2 bytes	-32768 a 32767
ushort	2 bytes	0 a 65535
int	4 bytes	-2147483648 a 2147483647
uint	4 bytes	0 to 4294967295
long	8 bytes	−9223372036854775808L to 9223372036854775807L
ulong	8 bytes	0 a 18446744073709551615
float	4 bytes	Números até 10 elevado a 38. Exemplo: 10.0f, 12.5f
double	8 bytes	Números até 10 elevado a 308. Exemplo: 10.0, 12.33
decimal	16 bytes	números com até 28 casas decimais. Exemplo 10.991m, 33.333m
char	2 bytes	Caracteres delimitados por aspas simples. Exemplo: 'a', 'ç', 'o'

Os tipos listados nessa tabela são conhecidos como tipos primitivos ou value types da linguagem C#.

# Tipos primitivos

*Abordar as conversões de tipo  
exemplificando no **Visual Studio...***

```
int valor = 1;  
short valorPequeno = valor;
```

```
int valor = 1;  
long valorGrande = valor;
```

# Comentários



# Comentários

- Muitas vezes precisamos escrever diversas linhas de comentários para, por exemplo, documentar uma lógica complexa da aplicação.
- Nesses casos podemos utilizar o comentário de múltiplas linhas que é inicializado por um `/*` e terminado pelo `*/`. Tudo que estiver entre a abertura e o fechamento do comentário é ignorado pelo compilador da linguagem:

```
/*  
    Isso é um comentário  
    de múltiplas linhas  
*/
```

# Comentários

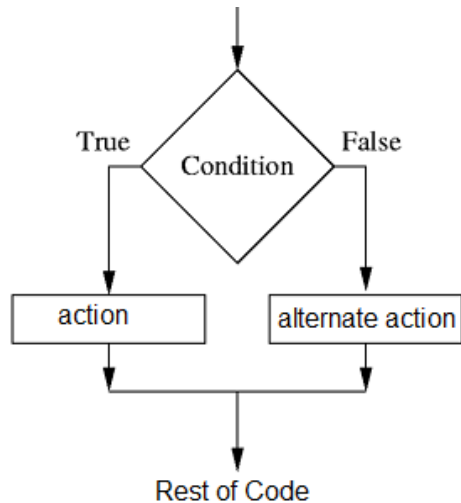
## ***Discursar a respeito do Código Limpo***

- Alerta de consequência
- Comentário TODO
- Comentários Ruins
- Comentários Redundantes



# Estruturas de controle

*Tomando decisões no código*



# Estrutura Condicional

- No C#, podemos executar código condicional utilizando a construção if:

```
if (condicao)
{
    // Esse código será executado somente se a condição for verdadeira
}
```



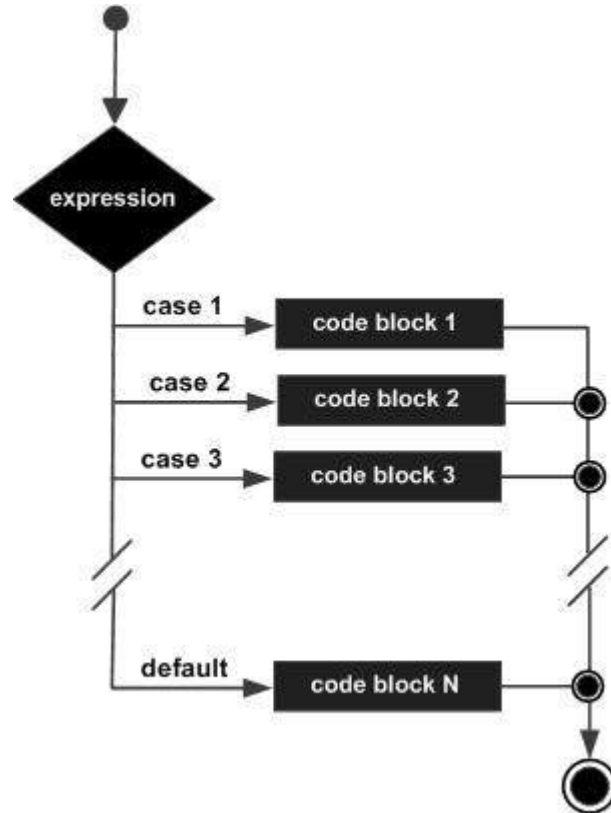
# Estrutura Condicional

```
if (saldo >= valorSaque)
{
    // código do saque
}
else
{
    MessageBox.Show("Saldo Insuficiente");
}
```

# Estrutura Condicional

- Repare na expressão que passamos para o if: saldo >= valorSaque.
- Nele, utilizamos o operador "maior ou igual". Além dele, existem outros operadores de comparação que podemos utilizar:
- maior (>), menor (<), menor ou igual (<=), igual (==) e diferente (!=). Podemos também negar uma condição de um if utilizando o operador ! na frente da condição que será negada.

# Estrutura condicional switch

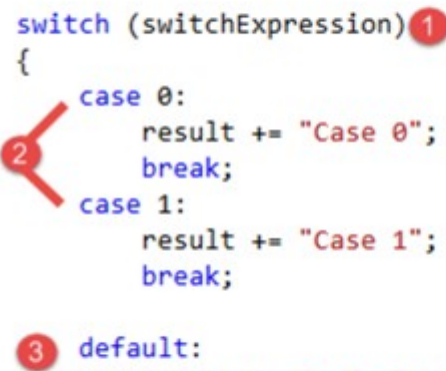


# Estrutura condicional switch

```
public partial class Default : System.Web.UI.Page
{
    protected void Page_Load(object sender, EventArgs e)
    {
        string result = "";
        int switchExpression = 4;

        switch (switchExpression) ❶
        {
            case 0:
                result += "Case 0";
                break;
            case 1:
                result += "Case 1";
                break;
            ❷ default:
                result += "Default (Optional)";
                break;
        }

        resultLabel.Text = result;
    }
}
```



# Estrutura Condicional

***...Exemplificar o uso de estrutura condicional no Visual Studio....***

# Estruturas de repetição

*Repetindo um bloco de código*

# Loop de while

**while** -No C# quando utilizamos o while, a condição do loop é checada antes de todas as voltas (iterações) do laço.

```
double valorInvestido = 1000.0;
int i = 1;
while (i <= 12)
{
    valorInvestido = valorInvestido * 1.01;
    i += 1;
}
MessageBox.Show("Valor investido agora é " + valorInvestido);
```

# Loop de for

**For** – usado quando sabemos exatamente quantas vezes queremos repetir um bloco de instruções.

```
for (valor inicial; teste booleano; incremento)

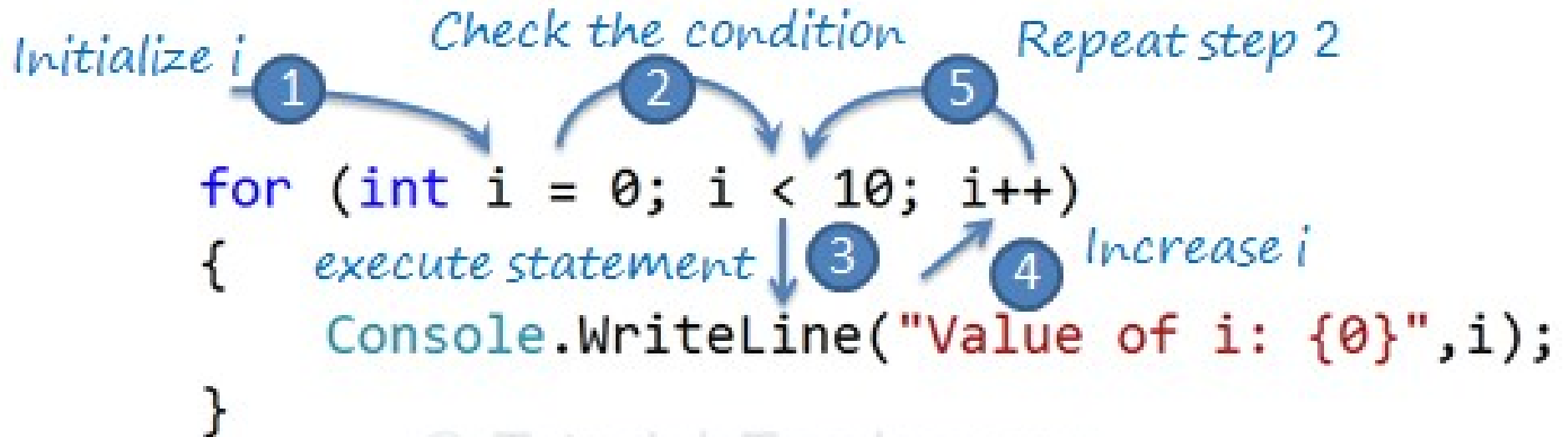
{

    //Lógica

}
```



# Loop de for



# Loop do while

- mas e se quiséssemos garantir que o corpo do laço seja executado **pelo menos uma vez**? Nesse caso, podemos utilizar um outro tipo de laço do C# que é o **do while**:

```
do
{
    // corpo do loop
}
while(condição);
```

Com o do while a condição do loop só é checada no fim da volta, ou seja, o corpo do loop é executado e depois a condição é checada, então o corpo do do...while sempre é executado pelo menos uma vez.

# Exercicios

