# Desenvolvimento de Jogos

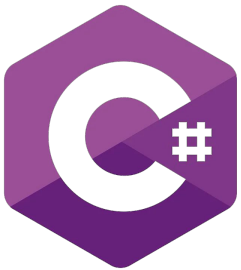
## C#

Profa. Thaiana Pereira dos Anjos Reis, Dra. Eng.
thaiana.anjos@ifsc.edu.br

Prof. Roberval Silva Bett, Me. Eng.
roberval.bett@ifsc.edu.br

# Agenda

- Escrita
- Operadores Lógicas e Matemáticas
- Estrutura de Dados
- Conversão
- Estruturas Condicionais



INSTITUTO FEDERAL
Santa Catarina

# Primeiro programa em C#

```
using System;
class Hello
{
    public static void Main ()
    {
        //Mostra a mensagem Hello World!!! na tela
        Console.WriteLine("Hello World!!!");
    }
}
```

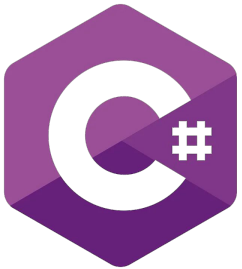
# Biblioteca

**using System;**

Comando utilizado para importar a biblioteca System.

INSTITUTO FEDERAL
Santa Catarina

# Programação Orientada a Objetos



# Classe

- **Declaração da classe:**

```
class Hello
{
...
}
```

# Classe

- **Declaração do método Main:**

```
public static void Main ()
{
…
}
```

Dentro da nossa definição de classe temos um método chamado ‘Main’. Todo programa em C# tem um método Main, que é o ponto de partida do programa, ou seja, a execução do aplicativo se inicia aqui.

# Classe

- **Escrevendo na tela:**

```
Console.WriteLine("Hello World!!!");
```

O método ‘WriteLine’ pertence a classe ‘Console’, que por sua vez está na localizada na biblioteca ‘System’.

**Atente-se que no final da instrução temos o ponto e virgula (;).**

# Métodos

- **Rotina:** Executa alguma ação e não retorna nada.

<modificador de acesso> **void** <nome da rotina>()

```
public void MinhaRotina() {
    x = 1;
}
```

*Neste caso temos uma rotina, pois void (vazio) significa que não há nenhum retorno simplesmente uma ação. Neste caso atribuindo o valor 1 a variável x.*

- **Função:** Executa alguma ação e retorna algum resultado sempre.

<modificador de acesso> **<tipo de dado>** <nome da função>()

```
public string MinhaFuncao(){
    return “Isto é uma função pois retorna algo.”;
}
```

*Neste caso temos uma função, pois ao invés de void (vazio) temos um valor de retorno nesse caso uma string.*

INSTITUTO FEDERAL
Santa Catarina

# Tipos de Dados

## Tipos de Dados primitivos mais usados no C#

| Data type | Description | Size (bits) | Range | Sample usage |
|---|---|---|---|---|
| *int* | Whole numbers | 32 | $-2^{31}$ through $2^{31}-1$ | `int count;`<br>`count = 42;` |
| *long* | Whole numbers (bigger range) | 64 | $-2^{63}$ through $2^{63}-1$ | `long wait;`<br>`wait = 42L;` |
| *float* | Floating-point numbers | 32 | $\pm 1.5 \times 10^{45}$ through $\pm 3.4 \times 10^{38}$ | `float away;`<br>`away = 0.42F;` |
| *double* | Double-precision (more accurate) floating-point numbers | 64 | $\pm 5.0 \times 10^{-324}$ through $\pm 1.7 \times 10^{308}$ | `double trouble;`<br>`trouble = 0.42;` |
| *decimal* | Monetary values | 128 | 28 significant figures | `decimal coin;`<br>`coin = 0.42M;` |
| *string* | Sequence of characters | 16 bits per character | Not applicable | `string vest;`<br>`vest = "fortytwo";` |
| *char* | Single character | 16 | 0 through $2^{16}-1$ | `char grill;`<br>`grill = 'x';` |
| *bool* | Boolean | 8 | True or false | `bool teeth;`<br>`teeth = false;` |

INSTITUTO FEDERAL
Santa Catarina

# Operadores Aritméticos

| Operador Aritmético | Descrição |
|---|---|
| + | Adição |
| - | Subtração |
| * | Multiplicação |
| / | Divisão |
| % | Módulo (resto da divisão) |

INSTITUTO FEDERAL
Santa Catarina

# Operadores Aritméticos de Atribuição

| Operador Aritmético | Descrição |
|---|---|
| + = | mais igual |
| - = | menos igual |
| * = | vezes igual |
| / = | dividido igual |
| % = | módulo igual |

INSTITUTO FEDERAL
Santa Catarina

# Operadores Relacionais

| Operador Relacional | Descrição |
|---|---|
| == | Igual a |
| != | Diferente de |
| > | Maior que |
| < | Menor que |
| >= | Maior do que ou igual a |
| <= | Menor do que ou igual a |

INSTITUTO FEDERAL
Santa Catarina

# Operadores Ternários

O operador ternário é composto por três operandos separados pelos sinais ? e : e tem o objetivo de atribuir o valor a uma variável de acordo com o resultado de um teste lógico.

Sintaxe: teste lógico ? valor se verdadeiro : valor se falso;

```
1  int x = 5, y = 10; // declaradas duas variáveis de tipo int
2
3  Console.WriteLine(x < y ? "sim" : "não"); // expressão x < y é avaliad
4  // se for verdadeira exibe "sim"
5  // se não for verdadeira exibe "não"
6  Console.ReadKey();
```

INSTITUTO FEDERAL
Santa Catarina

# Variáveis

- **Declaração de Variáveis:**

<tipo de dado> <variavel>;

int i;

- **Inicializando as variáveis:**

int i = 5;

double d1, d2;

bool fechado = false, aberto = true;

# Variáveis

- **Declaração de Variáveis:**

## Atenção!

O C# é Case-Sensitive, ou seja, diferencia maiúsculas de minúsculas.
“int num1;” é diferente de “int Num1;”

double d1, d2;

bool fechado = false, aberto = true;

INSTITUTO FEDERAL
Santa Catarina

# Constantes

- **Declaração de constantes:**

const <tipo de dados> <nome> = <valor>;

Exemplo:

const double PI = 3.14;

Uma constante sempre deve ser inicializada em sua declaração caso contrário será gerado um erro.

# Conversões de Dados

- **Conversão explícita**: quando uma variável pode ser mais de um tipo.

```
float x;

double y = 10.5;

x = (float) y;
```

Estamos atribuindo um valor a variável x do tipo float, e esse valor é a variável y que por sua vez é do tipo double. Neste caso fazendo uma conversão explícita (cast) de um tipo para o outro.

# Conversões de Dados

- **Conversão implícita**: acontece de forma transparente.

```
long x;

int y = 5;

x = y;
```

Neste caso estamos pegando o valor de y e simplesmente atribuindo a variável x do tipo long.

INSTITUTO FEDERAL
Santa Catarina

# Estruturas Condicionais

# IF

- **IF**

```
int x = 0;
int y = 5;

if (x > y) return;
```

- **IF…ELSE**

```
if (x > y) {
MessageBox.Show("X > Y");
}
else {
MessageBox.Show("X < Y");
}
```

INSTITUTO FEDERAL
Santa Catarina

# Dúvidas?

Profa. Thaiana Pereira dos Anjos Reis, Dra. Eng.
thaiana.anjos@ifsc.edu.br

Prof. Roberval Silva Bett, Me. Eng.
roberval.bett@ifsc.edu.br

INSTITUTO FEDERAL
Santa Catarina
Câmpus Garopaba