
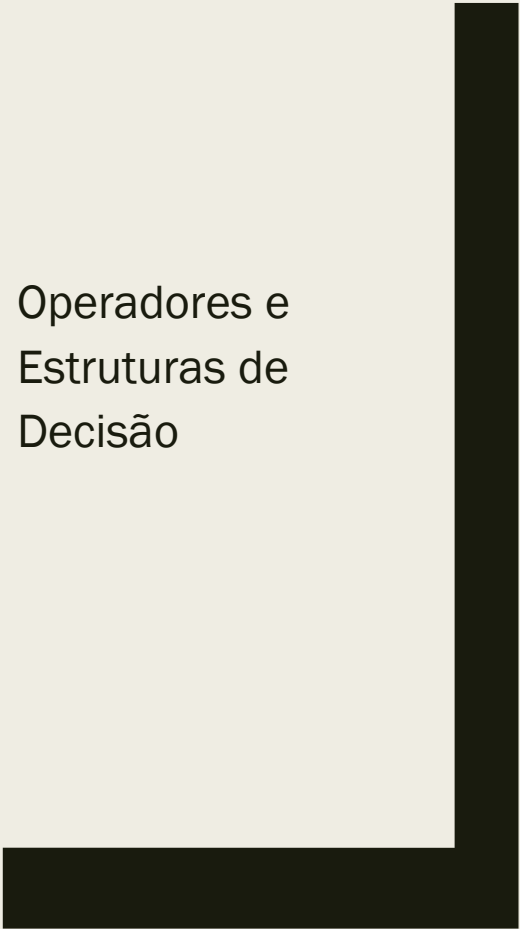




FUNDAMENTOS DE C# – AULA 2



Operadores e
Estruturas de
Decisão



Sobre mim

Nathan Ferreira

- 8º Período Ciências da Computação
- 1 ano e meio de experiencia profissional em C# .NET
- Instagram, X, LinkedIn, Gmail:
nathanf10994
- Whatsapp:
(31) 993-512-934



Processo de Compilação

- C# é uma linguagem **compilada** - o código é transformado em IL (Intermediate Language) e executado pelo CLR (Common Language Runtime)
- Vantagens:
 - Desempenho otimizado
 - Verificação de erros em tempo de compilação
- Outras linguagens são interpretadas (ex.: Python, TypeScript) → o código é lido e executado linha por linha pelos interpretadores (ex.: Navegadores de internet).

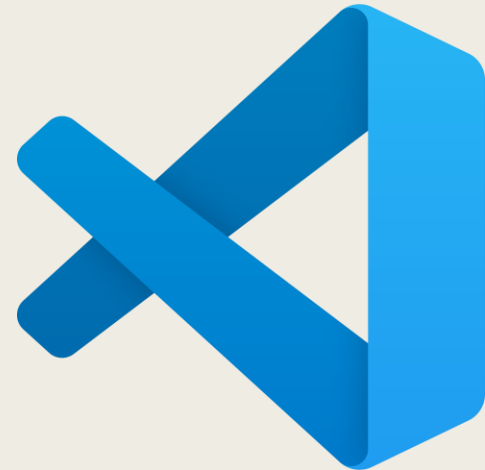


*Ambas as linguagens pertencem a Microsoft e foram criadas pelo mesmo engenheiro - **Anders Hejlsberg**



Criando um Projeto C# no VS Code

- Pré-requisitos:
 - *Instalar .NET SDK*
 - *Instalar extensão C# no VS Code*
- Comandos básicos:
 - **dotnet** new console -n MeuProjeto
**Criar o projeto*
 - **dotnet** run
**Compilar e executar o projeto*
- Estrutura inicial do projeto:
 - *Program.cs* → *ponto de entrada do programa*
 - *bin/* e *obj/* → *pastas de compilação*



Operadores Aritméticos

- + → soma
- - → subtração
- * → multiplicação
- / → divisão
- % → resto da divisão (módulo)

```
int primeiroNumero = 10;
int segundoNumero = 3;
int resultado;

// SOMA
resultado = primeiroNumero + segundoNumero; // 13

// Subtração
resultado = primeiroNumero - segundoNumero; // 7

// Multiplicação
resultado = primeiroNumero * segundoNumero; // 30

// Divisão
resultado = primeiroNumero / segundoNumero; // 3

// Módulo
resultado = primeiroNumero % segundoNumero; // 1
```

Atribuição, Incremento, e Decremento

- `=` → atribui valor
- `+=`, `-=`, `*=`, `/=`, `%=` → operações acumuladas
- `++` → Incremento
- `--` → Decremento

```
int ValueA = 15;
int ValueB = 5;
int ValueC = 0;

ValueC = ValueA + ValueB; //20
ValueC = ValueA; // 15

ValueC += 10; // 25
ValueC -= 3; // 22
ValueC *= 4; // 88
ValueC /= 8; // 11
ValueC %= 3; // 2

ValueC++; // 3
ValueC--; // 2
```

Operadores de Comparação

- == → igual
- != → diferente
- > → maior
- < → menor
- >= → maior ou igual
- <= → menor ou igual

```
int valueA = 10;
int valueB = 10;
int valueC = 9;

// IGUAL
Console.WriteLine(valueA == valueB); // true
Console.WriteLine(valueA == valueC); // false

// DIFERENTE
Console.WriteLine(valueA != valueB); // false
Console.WriteLine(valueA != valueC); // true

// MAIOR
Console.WriteLine(valueA > valueB); // false
Console.WriteLine(valueA > valueC); // true

// MENOR
Console.WriteLine(valueA < valueB); // false
Console.WriteLine(valueC < valueA); // true

// MAIOR OU IGUAL A
Console.WriteLine(valueA >= valueB); // true

// MENOR OU IGUAL A
Console.WriteLine(valueA <= valueB); // true
```

Operadores Lógicos

■ && → E lógico (AND)

- *Todas as condições devem ser verdadeiras para a operação o ser*

■ | | → OU lógico (OR)

- *Ao menos uma das condições deve ser verdadeira*

■ ! → Negação

- *Inverte o valor da operação booleana*

```
int valueA = 10;
int valueB = 20;
int valueC = -30;
int valueD = -40;

// OPERADOR AND &&
Console.WriteLine(valueA > 0 && valueB > 0); // true
Console.WriteLine(valueA > 0 && valueC > 0); // false

// OPERADOR OR ||
Console.WriteLine(valueA > 0 || valueB > 0); // true
Console.WriteLine(valueD > 0 || valueC > 0); // false

// OPERADOR NOT !
Console.WriteLine(!(valueA > 0 || valueB > 0)); // false
Console.WriteLine(valueA > 0 && !(valueC > 0)); // true
```


Estruturas de Decisão

- As **estruturas de decisão** em programação são recursos que permitem ao programa tomar decisões e executar diferentes blocos de código de acordo com determinadas condições.
- Funcionam como “caminhos alternativos”: o programa verifica uma condição (verdadeira ou falsa) e, com base nisso, escolhe qual instrução executar.
 - *if / else* → usados para verificar condições simples ou múltiplas.
 - *switch-case* → usado quando há várias opções possíveis, facilitando a escolha entre diferentes valores.

Estrutura de Decisão IF/ELSE

- Usado para executar blocos de código diferentes com base em condições lógicas.

```
if(condição)
{
    Escopo executado se a
    condição for verdadeira
}
else
{
    Escopo executado se a
    condição for falsa
}
```

```
int valueA = 50;
int ValueB = 100;

if (valueA >= ValueB)
{
    Console.WriteLine("ValueA é maior ou " +
        "igual que ValueB");
}
else
{
    Console.WriteLine("ValueB é maior " +
        "que ValueA");
}
```

Estrutura de Decisão IF/ELSE

```
// IF Simples
if (valueA == 50)
{
    Console.WriteLine("ValueA maior que 50");
}

// IF com mais de uma condição
if(ValueB < 50)
{
    Console.WriteLine("ValueB é menor que 50");
}
else if(ValueB > 100) // segunda condição
{
    Console.WriteLine("ValueB é maior que 100");
}
else
{
    Console.WriteLine("ValueB é maior do que 100");
}
```

Estrutura de Decisão IF/ELSE

```
// IF aninhado
if(ValueB > 100)
{
    if(ValueB == 150)
    {
        Console.WriteLine("ValueB é igual a 150");
    }
    else
    {
        Console.WriteLine("ValueB é diferente de 150");
    }
}
else
{
    Console.WriteLine("ValueB é menor que 100");
}
```

Estrutura de Decisão Switch/Case

- Usado para selecionar a execução de um bloco de código entre várias alternativas possíveis. Múltiplas condições, porém simples

```
switch(variavel)
{
    case valor-condição 1:
        escopo de execução
        break;
    case valor-condição 2:
        escopo de execução
        break;

    // múltiplos cases

    default: // opcional
        execução padrão
        break;
}
```

```
string alternativa = "c";

switch (alternativa)
{
    case "a":
        Console.WriteLine("Operação a");
        break;
    case "b":
        Console.WriteLine("Operação b");
        break;
    case "c":
        Console.WriteLine("Operação c");
        break;
    case "d":
        Console.WriteLine("Operação d");
        break;
    default:
        Console.WriteLine("Operação padrão");
        break;
}
```

Operadores Ternários

- O operador ternário é usado para realizar verificações simples, permitindo escolher entre dois valores de forma compacta em uma única linha.

condição? Valor se verdadeiro : valor se falso

```
int notaDaProva = 59;  
  
string resultado = notaDaProva >= 60 ? "Aprovado!" : "Reprovado";  
  
Console.WriteLine(resultado);
```

Escopo de variáveis

- O escopo de uma variável define onde ela pode ser acessada no código. Em C#, o escopo depende de onde a variável foi declarada:
- **Escopo Local:** a variável é criada dentro de um método, bloco ou estrutura de decisão (if, for, etc.) e só pode ser usada ali.
- **Escopo de Classe (ou Global dentro da classe):** quando a variável é declarada dentro de uma classe, mas fora dos métodos. Pode ser acessada em toda a classe.
- Após o fim do bloco onde foi declarada, a variável local deixa de existir e não pode mais ser utilizada.

Escopo de variáveis

```
class Program
{
    int globalValue = 10; // Escopo de classe

    0 referências
    void Exemplo()
    {
        int localValue = 5; // Escopo local

        Console.WriteLine(globalValue); // OK
        Console.WriteLine(localValue); // OK
    }

    0 referências
    void OutroExemplo()
    {
        // Console.WriteLine(numeroLocal); // ERRO: fora do escopo
        Console.WriteLine(globalValue); // OK
    }
}
```


Exercícios

- **1 – Aprovado e Reprovado** - O programa deve receber 3 notas, calcular a média $((n1 + n2 + n3) / 3)$, e verificar: se maior ou igual a 7, retornar “Aprovado”; se menor que 7 e maior que 5, retornar “Recuperação”; se menor que 5, retornar “Reprovado”. **(IF com múltiplas condições)**
- **2 – Faixa Etária** – O programa deve receber uma idade e verificar a maioridade. Se menor de 18 anos, retornar “Menor de Idade”. Se maior, então deverá fazer uma segunda verificação e retornar se maior que 60 anos “Idoso”, e se menor “Adulto.” **(IF aninhado)**
- **3 - Par ou Impar** – Receber um número e retornar se o valor é “Par” ou “Impar”. **(Operador Ternário)**
- **4 – Calculadora simples** – Receber dois valores e uma operação matemática. Calcular conforme a operação informada. Na opção Dividir verificar se ambos os valores são maiores que 0. **(SWITCH/CASE)**

Exercícios

- **5 – Numero primo** – Receba um número e verifique por qual número é divisível, ou se é primo. (SEM CHATGPT OU SEMELHANTES)

Possíveis temas da próxima aula:

- Estruturas de Repetição (For, Foreach, While, Do While)
- Break, Continue.