**Operadores Aritméticos**

**O que são?**

**C# (C-Sharp)**

Operadores Aritméticos são os operadores matemáticos para realização de operações de soma, subtração, multiplicação e divisão. A linguagem C# proporciona a execução dessas operações utilizando os mesmos operadores matemáticos que já conhecemos.

Na tabela a seguir, temos a lista de operadores e o nome da função:

|  |  |
| --- | --- |
| Operação | Operador |
| adição | + |
| subtração | - |
| multiplicação | \* |
| divisão | / |
| módulo | % |

O operador de resto % calcula o resto após dividir o operando à esquerda pelo à direita. Exemplo:

Console.WriteLine(30 % 4); //Output: 2

**Operadores Relacionais em C#**

Os operadores relacionais estabelecem a relação entre dois valores, geralmente contidos em variáveis, e retorna sempre um valor lógico **True** ou **False**, de acordo com essa relação.

A tabela a seguir mostra os operadores relacionais em C# e seus significados. Na coluna de Exemplo vamos supor que existam duas variáveis, A e B, com os valores 10 e 20 respectivamente:

|  |  |  |
| --- | --- | --- |
| **Operador** | **Significado** | **Exemplo** |
| **==** | Verifica se dois valores são iguais; se forem, retorna true, caso contrário, retorna false | (A == B) Retorna falso |
| **!=** | Verifica se dois valores são diferentes; se forem, retorna true, caso contrário, retorna false | (A! = B) Retorna true |
| **>** | Verifica se o primeiro valor é maior que o segundo, retornando verdadeiro em caso afirmativo | (A > B) Retorna false |
| **<** | Verifica se o primeiro valor é menor que o segundo, retornando verdadeiro em caso afirmativo | (A < B) Retorna true |
| **>=** | Verifica se o primeiro valor é maior ou igual ao segundo, retornando verdadeiro em caso afirmativo | (A >= B) Retorna false |
| **<=** | Verifica se o primeiro valor é menor ou igual ao segundo, retornando verdadeiro em caso afirmativo | (A <= B) Retorna true |

O código a seguir mostra os operadores relacionais em uso:

int a = 10;  
int b = 20;

|  |
| --- |
| Console.WriteLine(“a é igual a b? {0}”, (a == b)); Console.WriteLine(“a é diferente de b? {0}”, (a! = b)); Console.WriteLine(“a é maior do que b? {0}”, (a > b)); Console.WriteLine(“a é menor do que b? {0}”, (a < b)); Console.WriteLine(“a é maior ou igual a b? {0}”, (a >= b)); Console.WriteLine(“a é menor ou igual a b? {0}”, (a <= b)); |

Veja o resultado da execução desse código a seguir:

![Texto

Descrição gerada automaticamente]()

**Operadores Lógicos em C#**

Os operadores lógicos trabalham como operandos booleanos e seu resultado também será booleano (True ou False). Eles são usados **somente** em expressões lógicas.

|  |  |
| --- | --- |
| **Operador Lógico** | **Descrição** |
| **&&** | AND = E |
| **||** | OR = Ou |
| **!** | NOT = Não |

Assim, em um teste lógico usando o operador && (**AND**), o resultado somente será verdadeiro (**true**) se **todas** as expressões lógicas forem avaliadas como verdadeiras.

Já, se o operador usado for o II (**OR**), basta que apenas uma das expressões lógicas seja verdadeira para que o resultado também seja verdadeiro.

Completando, o operador lógico ! (**NOT**) é usado para gerar uma negação. Desta forma, é invertida toda a lógica da expressão.

Veja o exemplo na **Listagem 8** e o resultado em seguida.

Texto

Descrição gerada automaticamente

No código acima fazemos as avaliações:

* se a variável **a** é igual a 5 **E** se a variável **d** é igual a 10 (a ==5 && d == 10).
* se a variável **c** é menor que a variável **b** **OU** se a variável **d**é igual a 20 (c < b II d == 20).
* nega o resultado da avaliação se a variável **b** é maior que a variável **a** (! (b>a)).

e a cada avaliação o resultado será do tipo lógico, true ou false que será exibido no prompt de comando.

O resultado pode ser visto a seguir:

![Texto

Descrição gerada automaticamente]()

**Operações Aritméticas em Javascript**

**Operadores aritméticos** tem valores numéricos (literais ou variáveis) como seus operadores e retornam um valor numérico único. Os operadores aritméticos padrões são adição (+), subtração (-), multiplicação (\*), e divisão (/).

|  |  |
| --- | --- |
| **Operador** | **Descrição** |
| **+** | **Adição** |
| **-** | **Subtração** |
| **\*** | **Multiplicação** |
| **\*\*** | **Exponenciação** |
| **/** | **Divisão** |
| **%** | **Módulo (Restante)** |
| **++** e **--** | **Incremento e Decremento** |

**Exponenciação (\*\*)**

O operador de exponenciação retorna o resultado do primeiro operando elevado ao segundo operando. É o mesmo que var1var2, onde var1 e var2 são variáveis. O operador de exponenciação é associativo à direita, ou seja, a \*\* b \*\* c é igual a a \*\* (b \*\* c).

**Sintaxe**:

**Operador**: var1 \*\* var 2

**Operadores Lógicos em Javascript**

Operadores lógicos são tipicamente usados com valores Booleanos (lógicos). Quando eles o são, retornam um valor booleano. Porém, os operadores && e || de fato retornam o valor de um dos operandos específicos, então se esses operadores são usados com valores não booleanos, eles podem retornar um valor não booleano.

Os operadores lógicos são descritos na tabela abaixo:

|  |  |  |
| --- | --- | --- |
| **Operador** | **Utilização** | **Descrição** |
| Lógica **AND** (&&) | && | Retorna expr1 se essa pode ser convertido para falso; senão, retorna expr2. Dessa forma, quando usado para valores Booleanos, && retorna verdadeiro se ambos os operandos forem verdadeiros; senão, retorna falso. |
| Lógica **OR** (II) | **II** | Retorna expr1 se essa pode ser convertido para verdadeiro; senão, retorna expr2. Dessa forma, quando usado para valores Booleanos, || retorna verdadeiro se qualquer dos operandos for verdadeiro; se ambos são falsos, retorna falso. |
| Lógica **NOT** (!) | ! | Retorna falso se o seu operando pode ser convertido para verdadeiro; senão, retorna verdadeiro. |

Se um valor pode ser convertido para verdadeiro, este valor é chamado de truthy. Se um valor pode ser convertido para falso, este valor é chamado de falsy.

Exemplos de expressões que podem ser convertidas para falso são:

* null;
* NaN;
* 0;
* string vazia ("");
* undefined.

Mesmo que os operadores && and || possam ser utilizados com operandos que não são valores Booleanos, eles ainda podem ser considerados como operadores booleanos visto que seus valores de saída sempre podem ser convertidos em valores booleanos.

**Operadores Relacionais em Javascript**

Cada um desses operadores chamará a função valueOf () em cada operando antes que uma comparação seja feita.

**Operador Maior (>)**

O operador de Maior retorna true se o operando da esquerda for maior que o operando da direita.

**Sintaxe**: x > y

Exemplo: 4 > 3 //verdadeiro

**Operador Menor (<)**

O operador menor retorna true (verdadeiro) se o operando da esquerda for menor que o operando da direita.

**Sintaxe** x < y

Exemplo: 3 < 4 //verdadeiro

**Operador Menor ou Igual (<=)**

O operador menor ou igual retorna true (verdadeiro) se o operando da esquerda for menor ou igual ao operando da direita.

**Sintaxe**: x <= y

Exemplo: 3 <=4

**Operadores de Igualdade**

Os operadores de igualdade padrão (== e! =) Usam o Algoritmo de Comparação de Igualdade Abstrata.

Para comparar dois operandos. Se os operandos forem de tipos diferentes, ele tentará convertê-los para o mesmo tipo antes de fazer a comparação. Por exemplo, na expressão 5 == '5', a sequência à direita é convertida em Number antes da comparação ser feita.

Os operadores de igualdade estrita (=== e! ==) usam o Algoritmo de comparação estrita de igualdade e se destinam a executar comparações de igualdade em operandos do mesmo tipo. Se os operandos são de tipos diferentes, o resultado é sempre falso, então 5! == '5'.

Use operadores de igualdade estrita se os operandos precisarem ser de um tipo específico e também de valor ou se o tipo exato dos operandos for importante. Caso contrário, use os operadores de igualdade padrão, que permitem comparar a identidade de dois operandos, mesmo que não sejam do mesmo tipo.

Quando a conversão de tipos está envolvida na comparação (por exemplo, comparação não estrita), o JavaScript converte os tipos String, Number, booleano ou) operandos da seguinte forma:

* Ao comparar um número e uma string, a string é convertida em um valor numérico. JavaScript tenta converter o literal numérico de string em um valor de tipo Number. Primeiro, um valor matemático é derivado do literal numérico da string. Em seguida, esse valor é arredondado para o valor de tipo de número mais próximo.
* Se um dos operandos for booleano, o operando booleano é convertido em 1 se for verdadeiro e +0 se for falso.
* Se um objeto é comparado com um número ou string, o JavaScript tenta retornar o valor padrão para o objeto. Os operadores tentam converter o objeto em um valor primitivo, um valor String ou Number, usando os métodos valueOf e toString dos objetos. Se essa tentativa de converter o objeto falhar, será gerado um erro de tempo de execução.
* Observe que um objeto é convertido em primitivo se, e somente se, seu comparando for um primitivo. Se os dois operandos forem objetos, eles serão comparados como objetos, e o teste de igualdade será verdadeiro apenas se ambos fizerem referência ao mesmo objeto.

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

**Fontes:**

Operador Aritmético em C#:

<http://excript.com/csharp/operador-aritmetico-csharp.html#:~:text=Operadores%20Aritm%C3%A9ticos%20s%C3%A3o%20os%20operadores,e%20o%20nome%20da%20fun%C3%A7%C3%A3o>.

<https://docs.microsoft.com/pt-br/dotnet/csharp/language-reference/operators/arithmetic-operators#code-try-9>

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/Arithmetic_Operators>

<https://www.devmedia.com.br/tipos-de-operadores-do-csharp/18873>

Operadores Relacionais e Lógicos em C#

<http://www.bosontreinamentos.com.br/csharp/programacao-em-c-operadores-relacionais-e-logicos-07/#:~:text=Operadores%20Relacionais%20em%20C%23,de%20acordo%20com%20essa%20rela%C3%A7%C3%A3o>.

<http://excript.com/csharp/operador-relacional-csharp.html>

Operador Aritmético, Relacional e Lógico em Javascript:

<http://www.bosontreinamentos.com.br/programacao-em-javascript/curso-basico-de-javascript-operadores-aritmeticos/#:~:text=Operadores%20Aritm%C3%A9ticos%20em%20JavaScript,o%20m%C3%B3dulo%20e%20incremento%20%2F%20decremento>.

<https://www.w3schools.com/js/js_operators.asp>

<https://www.w3schools.com/js/js_comparisons.asp>

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Guide/Expressions_and_Operators#operador_virgula>

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/Arithmetic_Operators>

<https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Operators/Operadores_de_compara%C3%A7%C3%A3o>