**Operadores Aritméticos**

Usados em cálculos matemáticos.

|  |  |
| --- | --- |
| **Operador Aritmético** | **Descrição** |
| **+** | Adição |
| **-** | Subtração |
| **\*** | Multiplicação |
| **/** | Divisão |
| **%** | Resto da divisão |

### Operadores Relacionais

Os operadores relacionais comparam dois valores e retornam um valor booleano (**true** ou **false**).

**Operador Relacional Descrição**

**==** Igual a

**!=** Diferente de

**>** Maior que

**<** Menor que

**>=** Maior do que ou igual a

**<=** Menor do que ou igual a

### Operadores Lógicos

Os operadores lógicos trabalham como operandos booleanos e seu resultado também será booleano (true ou false).

**Operador Lógico Descrição**

&& AND = E

**||** OR = Ou

**!** NOT = Não

### Operadores incrementais e decrementais

Os operadores incrementais e decrementais têm a função de aumentar ou diminuir exatamente em **1** o valor de uma variável. Eles podem ser pré ou pós **incremental** e pré ou pós **decremental**. Veja os conceitos de cada um deles e um exemplo prático a seguir:

* Incremental (++):
  + **Pré incremental ou prefixo**– Significa que, se o sinal for colocado **antes** da variável, primeiramente será somado o valor **1** para esta variável, continuando em seguida a resolução da expressão.
  + **Int = x**
  + **++x**
  + **Pós incremental ou sufixo**– Significa que, se o sinal for colocado **após** a variável, é resolvido primeiro a expressão, seja ela adição, subtração, multiplicação ou qualquer outra, para em seguida ser adicionado o valor **1** à variável.
  + **Int = x**
  + **x++**
* **Decremental (--)**
  + **Pré incremental ou prefixo**– Significa que, se o sinal for colocado **antes** da variável, primeiramente será subtraído o valor **1** para esta variável, continuando em seguida a resolução da expressão.
  + **Int = x**
  + **--x**
  + **Pós incremental ou sufixo**– Significa que, se o sinal for colocado **após** a variável, é resolvido primeiro a expressão, seja ela adição, subtração, multiplicação ou qualquer outra, para em seguida ser subtraído o valor **1** à variável.
  + **Int = x**
  + **x—**

## **Operador de igualdade ==**

O operador de igualdade == retornará true se seus operandos forem iguais; caso contrário, false.

## **Operador de desigualdade !=**

O operador de desigualdade != retornará true se seus operandos não forem iguais; caso contrário, false.

Concatenando Strings

|  |
| --- |
| **1- Usando o operador +** |

namespace CShp\_ConcatenandoStrings

{

class Program

{

static void Main(string[] args)

{

string s1 = "informática " + "é " + " muito " + "legal " + " cara";

WriteLine(s1);

ReadKey();

}

}

}

**2- Usando o método Concact**

namespace CShp\_ConcatenandoStrings

{

class Program

{

static void Main(string[] args)

{

IEnumerable<string> lista = new List<string>() { "ola ", "tudo ", "bem", " ir ", "ali" };

string s2 = string.Concat(lista);

WriteLine(s2);

ReadKey();

}

}

}

**3- Usando o método Join**

namespace CShp\_ConcatenandoStrings

{

class Program

{

static void Main(string[] args)

{

string s3 = string.Join(" ", "tenho", "quase", "tudo", "para", "fazer.");

WriteLine(s3);

ReadKey();

}

}

}

**4- Usando a classe StringBuilder**

namespace CShp\_ConcatenandoStrings

{

class Program

{

static void Main(string[] args)

{

StringBuilder sb = new StringBuilder();

sb.Append("Não").Append("sei").Append("mais").Append("o que ").Append(" escrever.");

string s4 = sb.ToString();

WriteLine(s4);

ReadKey();

}

}

}

**5- Usando o método String.Format**

namespace CShp\_ConcatenandoStrings

{

class Program

{

static void Main(string[] args)

{

string s5 = string.Format("{0} {1} {2} {3} {4}", "matheus ", "andreoli ", " trabalho", " para ", "TPV");

WriteLine(s5);

ReadKey();

}

}

}

**6- Usando o operador $ (C# 6.0)**

namespace CShp\_ConcatenandoStrings

{

class Program

{

static void Main(string[] args)

{

string p1 = "oi";

string p2 = "está";

string p3 = "tudo";

string p4 = "bem";

string p5 = "aí";

**string s6 = $"{p1} {p2} {p3} {p4} {p5}";**

WriteLine(s5);

ReadKey();

}

}

}

# Objetos

Uma definição de classe ou struct é como um esquema que especifica o que o tipo pode fazer. Um objeto é basicamente um bloco de memória que foi alocado e configurado de acordo com o esquema. Um programa pode criar vários objetos da mesma classe. Objetos também são chamados de instâncias e podem ser armazenados em uma variável nomeada ou em uma matriz ou coleção. O código de cliente é o código que usa essas variáveis para chamar os métodos e acessar as propriedades públicas do objeto. Em uma linguagem orientada a objetos, como o C#, um programa típico consiste em vários objetos que interagem dinamicamente.

# **Principais Funções Matemáticas em C# – Classe Math**

## **π em C#**

* Math.Pi – Retorna o valor de pi

## **Seno em C#**

* Math.Sin(x) – Retorna o seno de x

## **Cosseno em C#**

* Math.Cos(x) – Retorna o cosseno de x

## **Tangente em C#**

* Math.Tan(x) – Retorna a tangente de x

## **Potência em C#**

* Math.Exp(x) – Retorna o neperiano e elevado a potência x
* Math.Pow(x,y) – Exponenciação, retorna o valor de x elevado a y

## **Logaritmo em C#**

* Math.Log(x) – Retorna o logarítimo na base natural e de x
* Math.Log10(x) – Retorna o logarítimo na base 10 de x

## **Raíz Quadrada em C#**

* Math.Sqrt(x) – Retorna a raiz quadrada de x

## **Arcocosseno em C#**

* Math.Acos(x) – Retorna o arcocosseno de x

## **Arcoseno em C#**

* Math.Asin(x) – Retorna o arcoseno de x

## **Arcotangente em C#**

* Math.Atan(x) – Retorna o arcotangente de x

## **Outras funções matemáticas úteis em C#**

* Math.Min(x,y) – Retorna o menor de dois númeroes
* Math.Max(x,y) – Retorna o maior de dois números
* Math.Abs(x) – Retorna o valor absoluto de um número decimal x
* Math.Round(x,y) – Arredonda o número x para que ele fique com y casas decimais. Na omissão do y, arredonda-se para um valor inteiro.