

# **C#: estruturas básicas e *console application*.**

Jonathan Vinícius Suter

## **1 O que é o C# (ou C Sharp)?**

É uma linguagem de programação fortemente tipada, criada pela Microsoft, que permite o desenvolvimento de aplicações para praticamente todos os tipos de sistemas operacionais, desde aplicativos para *smartphones* até aplicações *Windows* e até para *Linux*.

## **2 Desenvolvendo em C# com Visual Studio 2019**

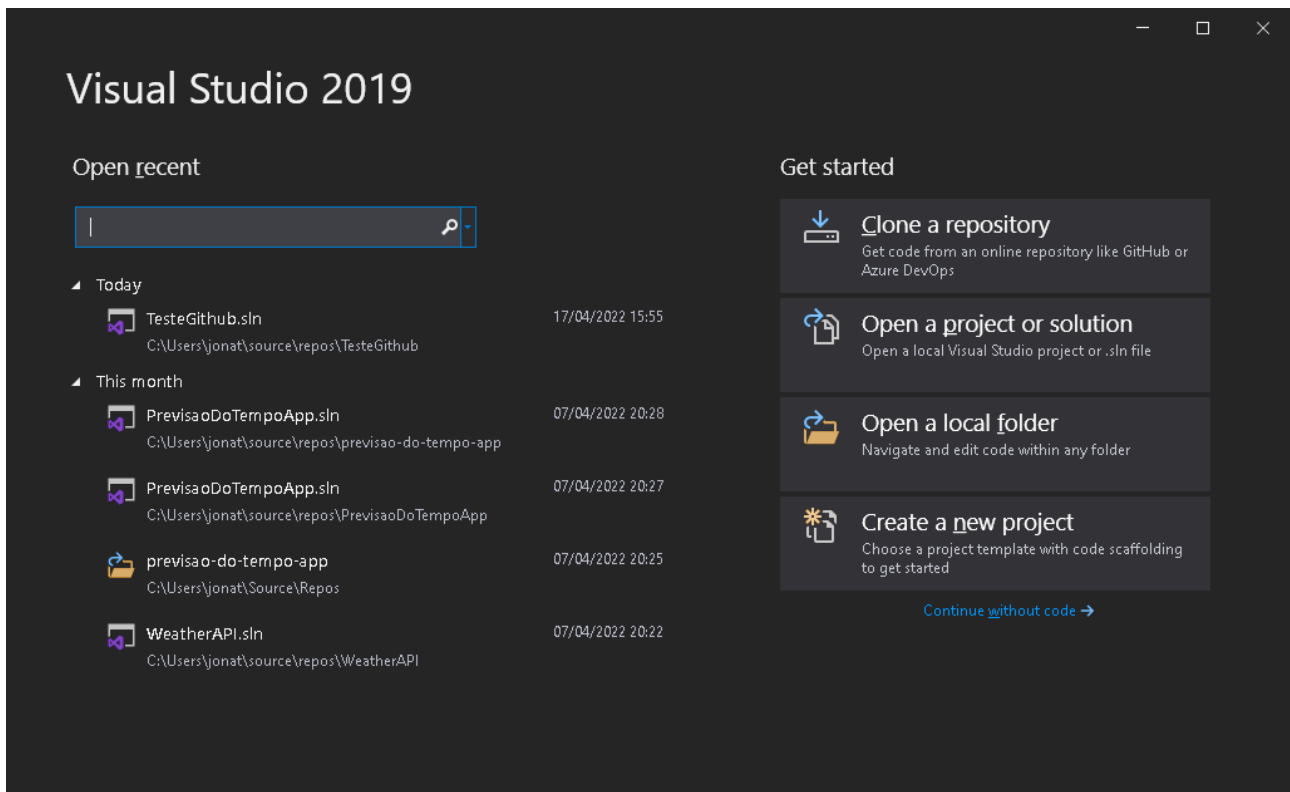
A escolha da versão 2019 do Visual Studio 2019 para este guia se deve às facilidades que a IDE (interface de programação) proporciona ao desenvolvedor. Seja pelas poucas configurações que precisam ser feitas ou as ferramentas que a IDE dispõe.

### **2.1 Instalando o Visual Studio 2019**

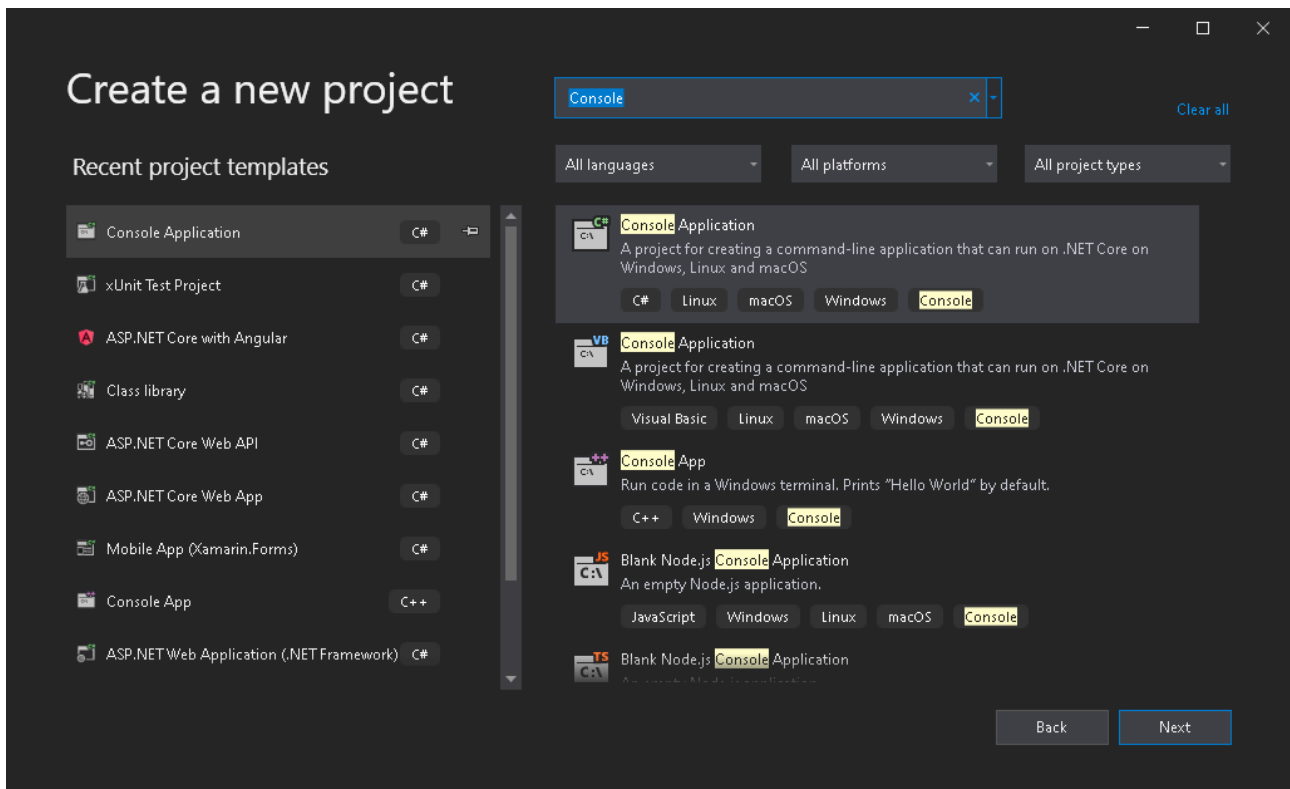
Para instalar esta versão específica do Visual Studio, deverá baixar o instalador através [deste link](#). A instalação básica já será o suficiente para desenvolver as aplicações.

### 3 Criando uma aplicação de console

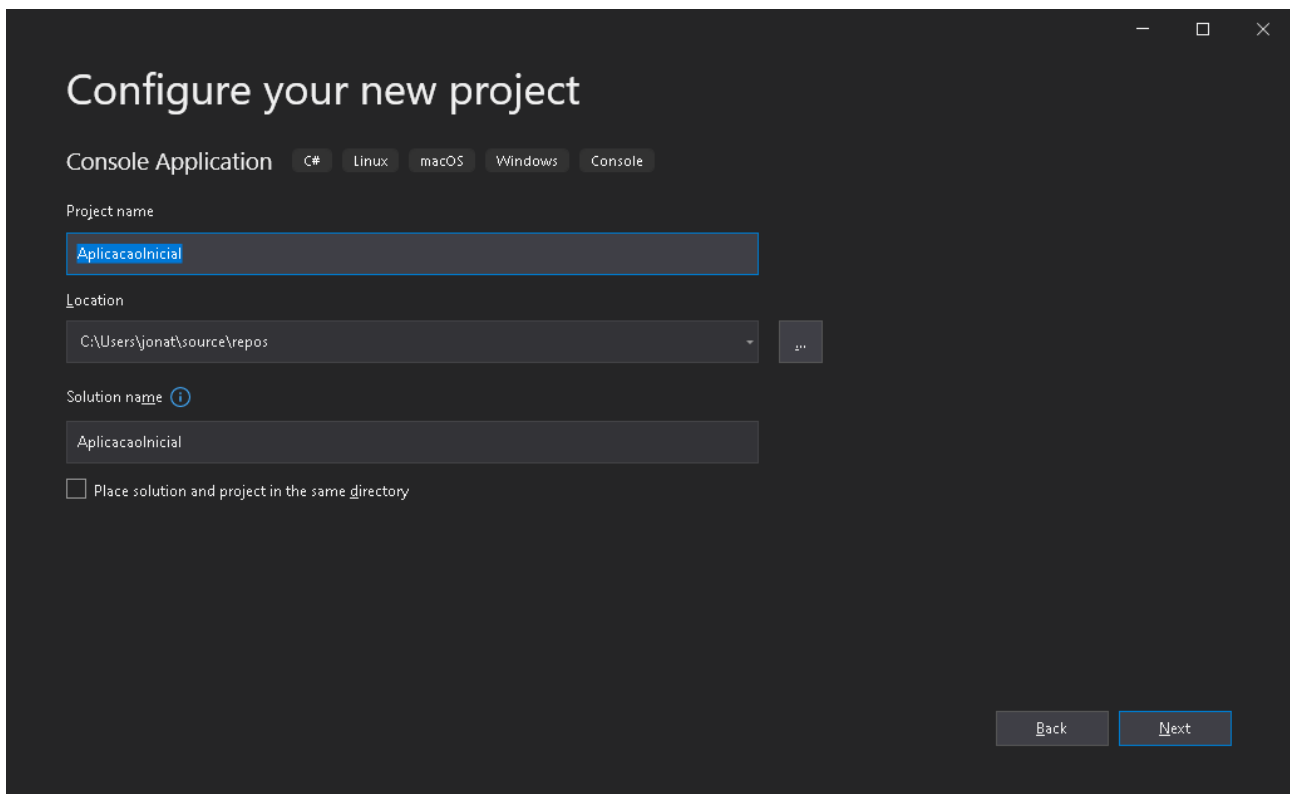
Ao abrir o Visual Studio, Deve-se selecionar a opção *Create New Project*:



Após isso, será necessário selecionar o *Template* do projeto. Neste caso, procurar por “console” no campo de buscas e selecione *Console Application* com a tag *C#*:

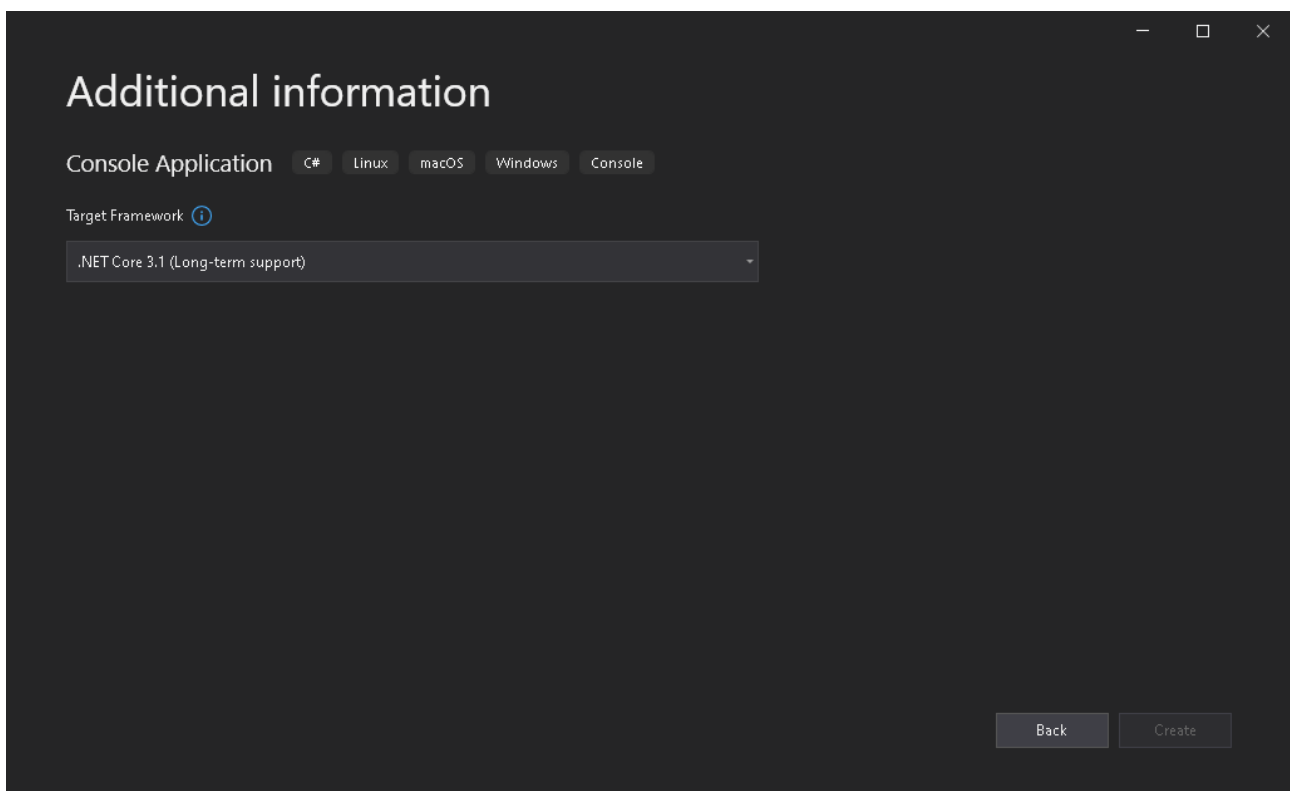


Na próxima tela, será solicitado nome da aplicação, a localização e o nome da solução.  
Caso seja necessário instalar em um outro diretório.



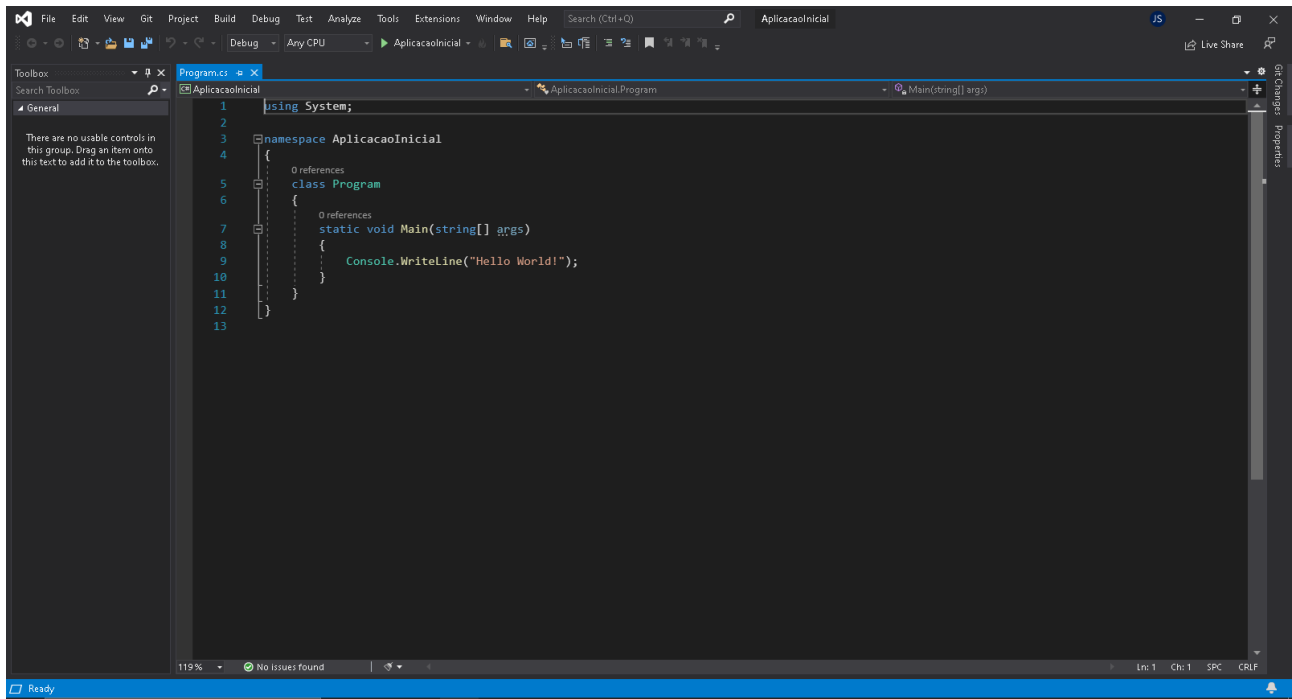
The screenshot shows the 'Configure your new project' window. At the top, it says 'Console Application' with tabs for 'C#', 'Linux', 'macOS', 'Windows', and 'Console'. Below this, there are three input fields: 'Project name' with the value 'AplicacaoInicial', 'Location' with the value 'C:\Users\jonat\source\repos', and 'Solution name' with the value 'AplicacaoInicial'. There is an information icon next to the 'Solution name' label. At the bottom left, there is a checkbox labeled 'Place solution and project in the same directory' which is currently unchecked. At the bottom right, there are two buttons: 'Back' and 'Next'.

Clicando em *Next*, será solicitada a versão do C# que o desenvolvedor pretende utilizar. Neste caso, utilize a versão *.NET Core 3.1 (Long-term support)*.

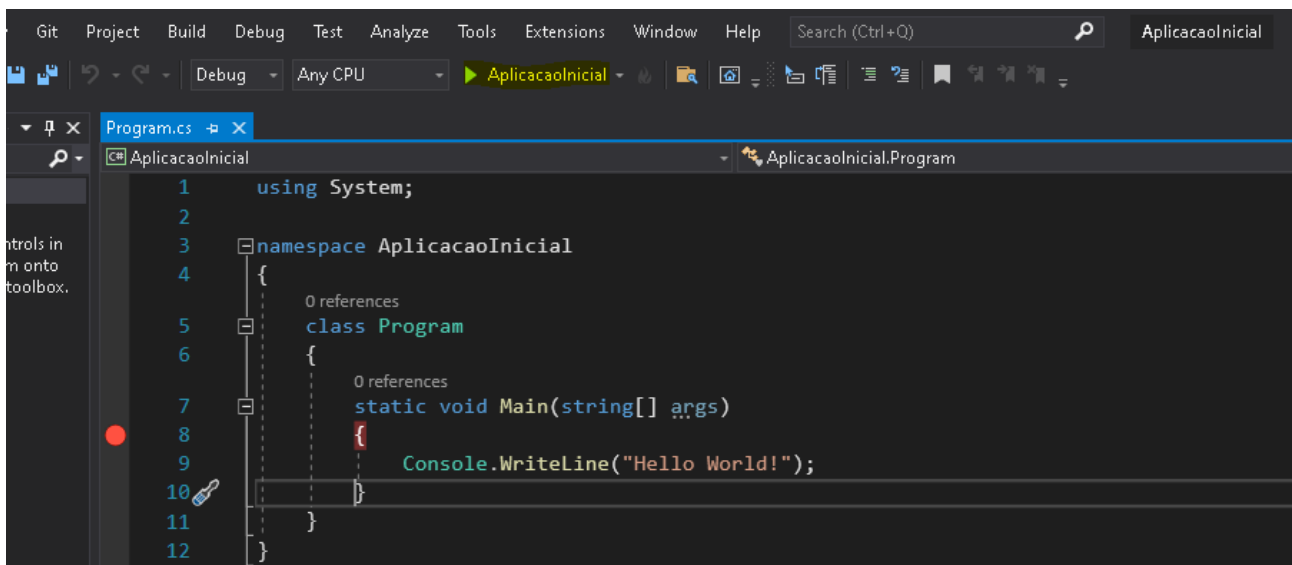


The screenshot shows the 'Additional information' window. It has the same header as the previous window: 'Console Application' with tabs for 'C#', 'Linux', 'macOS', 'Windows', and 'Console'. Below this, there is a 'Target Framework' label with an information icon, followed by a dropdown menu showing '.NET Core 3.1 (Long-term support)'. At the bottom right, there are two buttons: 'Back' and 'Create'.

Feito isso, a solução será aberta em sua classe principal.



Para executar o programa, pode-se pressionar a tecla F5 ou clicar no botão destacado, conforme abaixo:



## 4 Tipos de variáveis

O C#, como dito anteriormente, é uma linguagem de programação fortemente tipada. Dos tipos primitivos, há os seguintes\*:

```
class Program
{
    static void Main(string[] args)
    {
        //recebe um tipo verdadeiro ou falso (true ou false)
        bool v_bool = true;
        //recebe um valor inteiro entre 0 e 255
        byte v_bytes = 123;
        //recebe um valor inteiro entre -128 e 127
        sbyte v_sbytes = 127;
        //recebe um valor inteiro entre -32768 e 32767
        short v_short = 32767;
        //recebe um valor inteiro entre 0 e 65535
        ushort v_ushort = 62535;
        //recebe um valor inteiro entre -2147483648 e 2147483647
        int v_int = 2147483647;
        //recebe um valor inteiro entre 0 e 4294967295
        uint v_uint = 4294967295;
        //recebe um valor entre -9223372036854775808 e 9223372036854775807
        long v_long = 9223372036854775807;
        //recebe um valor inteiro entre 0 e 18446744073709551615
        ulong v_ulong = 18446744073709551615;
        //recebe um valor real com precisão de 7 dígitos
        float v_float = 3.1415926f;
        //recebe um valor real com precisão de até 16 dígitos.
        double v_double = 3.14159265359;
        //recebe um valor real com precisão de até 29 dígitos.
        decimal v_decimal = 4.000000000000000000000000000000M;
        //recebe um tipo de texto de apenas um caractere.
        char v_char = 'A';
        //recebe um tipo do enumerador
        EnumeradorTeste situacao = EnumeradorTeste.A;
        //é possível converter o enumerador para um número inteiro:
        int situacao_int = (int)EnumeradorTeste.A;
        //recebe uma cadeia de caracteres
        string v_string = "Jonathan Vinícius Suter";
    }
    enum EnumeradorTeste
    {
        A,
        B,
        C
    }
}
```

\*Embora o tipo *string* não seja um tipo primitivo, este será usado com frequência.

## 4.1 Regras para declaração de variáveis

Ao criar variáveis, não se deve utilizar palavras reservadas da linguagem, a lista das palavras estão [neste link](#);

Devem ser utilizados apenas textos, *underline* (   ) ou números, desde que estes últimos não sejam a primeira letra do nome da variável;

Não pode haver duas variáveis com o mesmo nome (salvo quando uma é variável é atributo de classe e outra de escopo local).

Outras regras mais voltadas para boas práticas (opcionais):

- Declarar as variáveis que utilizará mais ao início do programa, sempre que possível;
- Remover variáveis não utilizadas;
- Usar nomes significativos. Exemplo: resultadoDoTotalAposDesconto, notaMedia, etc..

OBS: o C# é uma linguagem *case sensitive*, ou seja, ela diferencia letras maiúsculas de minúsculas, sendo que uma variável com o nome “resultadoDoTotalAposDesconto” é diferente de “resultadodototalaposdesconto”.

## 5 Operadores aritméticos

Os operadores aritméticos são os operadores usados para operações matemáticas:

+	Soma ou adição;
-	Subtração;
*	Multiplicação;
/	Divisão
%	Módulo (Resto da divisão)

## Exemplos de uso:

```
class Program
{
    static void Main(string[] args)
    {
        int a1 = 100, b1 = 50, resultadoInt;
        double a2 = 50, b2 = 100, resultadoDouble;
        string texto1 = "Sou uma cadeia de texto", texto2 = " do C#", resultadoTextos;

        //Soma
        resultadoInt = a1 + b1;
        resultadoDouble = a2 + b2;
        //para cadeias de caracteres, a soma funciona como uma junção das cadeias de
        caracteres.
        resultadoTextos = texto1 + texto2;

        Console.WriteLine("Soma:");
        Console.WriteLine("resultadoInt = " + resultadoInt);
        Console.WriteLine("resultadoDouble = " + resultadoDouble);
        Console.WriteLine("resultadoTextos = " + resultadoTextos);

        //subtração
        resultadoInt = a1 - b1;
        resultadoDouble = a2 - b2;

        Console.WriteLine("Subtração:");
        Console.WriteLine("resultadoInt = " + resultadoInt);
        Console.WriteLine("resultadoDouble = " + resultadoDouble);

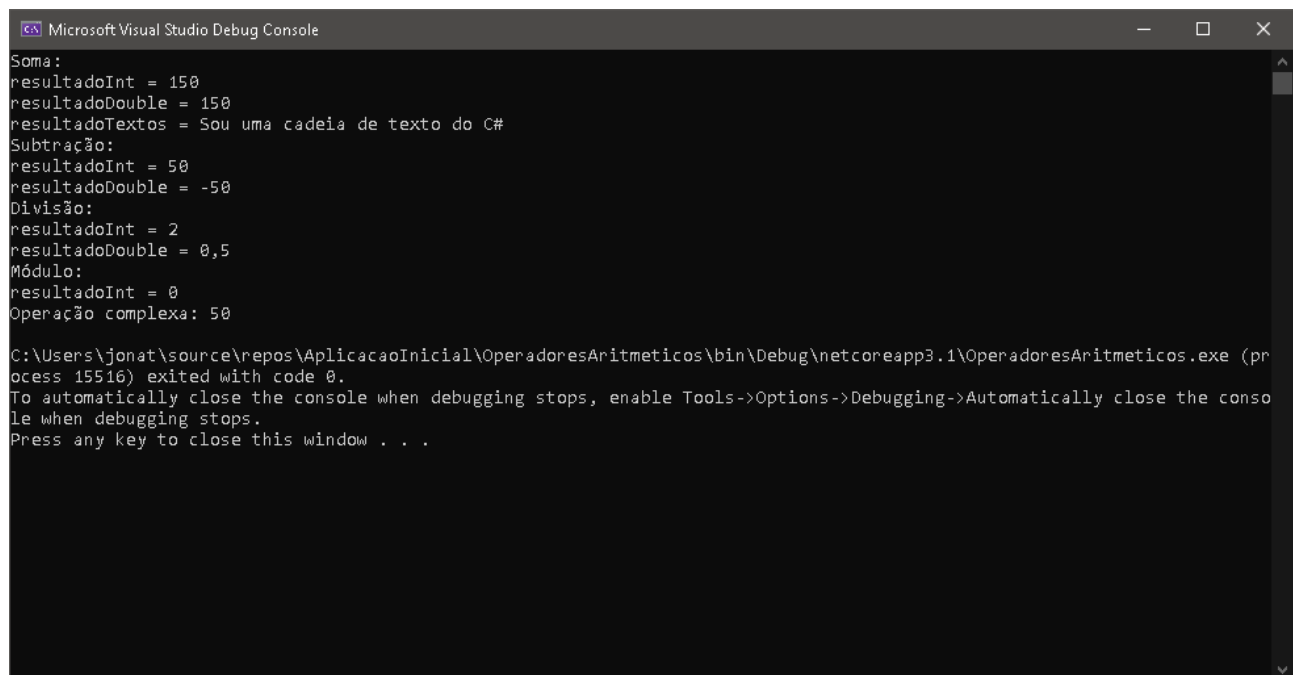
        //divisão
        resultadoInt = a1 / b1;
        resultadoDouble = a2 / b2;

        Console.WriteLine("Divisão: ");
        Console.WriteLine("resultadoInt = " + resultadoInt);
        Console.WriteLine("resultadoDouble = " + resultadoDouble);

        //módulo
        resultadoInt = a1 % b1;
        Console.WriteLine("Módulo: ");
        Console.WriteLine("resultadoInt = " + resultadoInt);

        //para fazer operações mais complexas, usa-se os parênteses para indicar a
        ordem das operações:
        resultadoDouble = ((a2 / b2) * 100);
        Console.WriteLine("Operação complexa: "+ resultadoDouble);
    }
}
```

Resultado da saída do console:



```
Microsoft Visual Studio Debug Console
Soma:
resultadoInt = 150
resultadoDouble = 150
resultadoTextos = Sou uma cadeia de texto do C#
Subtração:
resultadoInt = 50
resultadoDouble = -50
Divisão:
resultadoInt = 2
resultadoDouble = 0,5
Módulo:
resultadoInt = 0
Operação complexa: 50

C:\Users\jonat\source\repos\AplicacaoInicial\OperadoresAritmeticos\bin\Debug\netcoreapp3.1\OperadoresAritmeticos.exe (process 15516) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

## 6 Operadores relacionais

Operadores relacionais avaliam expressões com valores:

“==”	Igual
!=	Diferente
>	Maior que
<	Menor que
>=	Maior ou igual que
<=	Menor ou igual que



```

class Program
{
    static void Main(string[] args)
    {
        int a1 = 100, b1 = 50;
        string texto1 = "Jonathan", texto2 = "Jonathan";
        bool resultadoInt, resultadoString;

        //maior que
        resultadoInt = a1 > b1;
        Console.WriteLine("ResultadoInt: " + resultadoInt);

        //menor que
        resultadoInt = a1 >= b1;
        Console.WriteLine("ResultadoInt: " + resultadoInt);

        //menor ou igual que
        resultadoInt = a1 <= b1;
        Console.WriteLine("ResultadoInt: " + resultadoInt);

        //menor que
        resultadoInt = a1 < b1;
        Console.WriteLine("ResultadoInt: " + resultadoInt);

        //Igualdade
        resultadoInt = a1 == b1;
        Console.WriteLine("ResultadoInt: " + resultadoInt);

        //Diferença
        resultadoInt = a1 != b1;
        Console.WriteLine("ResultadoInt: " + resultadoInt);

        //Comparando strings
        resultadoString = texto1 == texto2;
        Console.WriteLine("ResultadoInt: " + resultadoString);
    }
}

```

Resultado da saída no console:

```

Microsoft Visual Studio Debug Console
ResultadoInt: True
ResultadoInt: True
ResultadoInt: False
ResultadoInt: False
ResultadoInt: False
ResultadoInt: True
ResultadoInt: True

C:\Users\jonat\source\repos\AplicacaoInicial\OperadoresRelacionais\bin\Debug\netcoreapp3.1\OperadoresRelacionais.exe (process 15636) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .

```

## 7 Operadores lógicos

Os operadores lógicos trabalham com operações de associação e retornam valores booleanos.

<b>&amp;&amp;</b>	AND (E)
<b>  </b>	OR (OU)
<b>!</b>	NOT (Negação)

```
class Program
{
    static void Main(string[] args)
    {
        int a1 = 100, b1 = 50;

        bool resultadoInt;

        resultadoInt = ((a1 > b1) && (a1 <= 200));
        Console.WriteLine("resultadoInt com o operador AND " + resultadoInt);

        resultadoInt = (a1 > b1) || (a1 > 100);
        Console.WriteLine("resultadoInt com o operador OU " + resultadoInt);

        resultadoInt = !resultadoInt;
        Console.WriteLine("resultadoInt com o operador de Negação " + resultadoInt);
    }
}
```

Os operadores lógicos sempre são usados para associar/avaliar duas expressões que retornam *true* ou *false*. E retornam *true* ou *false*.

O resultado da saída após a execução da aplicação.

```
Microsoft Visual Studio Debug Console

resultadoInt com o operador AND True
resultadoInt com o operador OU True
resultadoInt com o operador de Negação False

C:\Users\jonat\source\repos\AplicacaoInicial\OperadoresLogicos\bin\Debug\netcoreapp3.1\OperadoresLogicos.exe (process 14352) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the console when debugging stops.
Press any key to close this window . . .
```

## 8 Estrutura condicional (IF ELSE, IF ELSE IF)

São estruturas para tomada de decisão, que permitem executar um comportamento de acordo com o resultado de uma expressão.

```
if (expressao)
{
    //se o valor da expressao for verdadeira, executa esse bloco
}
else
{
    //se o valor da expressão for falso, executa esse bloco
}
```

Exemplo utilizando os blocos *if*:

```
class Program
{
    static void Main(string[] args)
    {
        int a1, b1;

        bool resultadoInt;

        //Console.ReadLine() serve para pegar a entrada de dados do usuário
        //Convert.ToInt32() converte o valor da entrada para um inteiro.
        Console.WriteLine("Insira um valor para a1:");
        a1 = Convert.ToInt32(Console.ReadLine());

        Console.WriteLine("Insira um valor para b1:");
        b1 = Convert.ToInt32(Console.ReadLine());

        resultadoInt = a1 > b1;

        //if else
        if (resultadoInt)
        {
            Console.WriteLine("a1 é maior que b1:");
        }
        else
        {
            Console.WriteLine("a1 é menor que b1:");
        }

        //if else if
        if (resultadoInt)
        {
            Console.WriteLine("a1 é maior que b1:");
        }
        else if(a1 > 100)
        {
            Console.WriteLine("a1 é maior que 100:");
        }
    }
}
```

## 9 Exercícios

- Faça um programa que receba o nome e a idade de uma pessoa e mostre uma mensagem se ela for maior de idade;
- Faça um programa em que a pessoa insira sua altura e peso e calcule o IMC;
- Faça um programa que receba dois números, um texto com a operação que ela deseja efetuar e efetue essa operação, mostrando em tela o resultado e a operação;

- Faça um programa que receba 3 notas de um aluno e identifique a maior nota, a menor nota e a média entre as notas;
- Faça um programa que receba o valor de um produto e o percentual de desconto e mostre o valor do produto com o desconto e calcule os impostos (com base no valor sem desconto) sobre e mostre também em tela:
  - Valor até 100: alíquota de 5%;
  - Valor entre 101 e 300: alíquota de 15%;
  - Valor entre 301 e 1000: alíquota de 25%;
- Escreva um programa que recebe o primeiro e o último nome de uma pessoa e imprima ambos em tela;
- Escreva um programa que receba dois pontos com coordenadas e imprima em tela a distância entre eles. Dados:  $d_{ab}^2 = (x_a - x_b)^2 + (y_a - y_b)^2$