

## Módulo 01 Variáveis, constantes e tipos de dados

### #01 Tipos Numéricos Inteiros

```
#region Inteiros
sbyte num1 = 10; // 8 bits, de -128 a 127
short num2 = 20; // 16 bits, -32.768 a 32.767
int num3 = 30; // 32 bits, -2.147.483.648 a 2.147.483.647
long num4 = 40L; // 64 bits, -9.223.372.036.854.775.808 a 9.223.372.036.854.775.807

// Integral sem sinal
byte num5 = 10; // 8 bits, intervalo de 0 a 255
ushort num6 = 20; // 16 bits, de 0 a 65.535
uint num7 = 30; // 32 bits, de 0 a 4.294.967.295
ulong num8 = 40L; // 64 bits, de 0 a 18.446.744.073.709.551.615

sbyte numero;
numero = 100;

numero = 120;
numero = num1;
#endregion

Console.WriteLine(numero); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

### #02 Tipos Numéricos Reais

```
#region Reais
float real1 = 100.75f; // 32 bits, de  $1,5 \times 10^{-45}$  a  $3,4 \times 10^{38}$ , precisão de 7 dígitos => Sufixo f, Ex.: 10.5f
double real2 = 12500.45; // 64 bits, de  $5,0 \times 10^{-324}$  a  $1,7 \times 10^{308}$ , precisão de 15 dígitos => Sufixo d (opcional), Ex.: 10.5d
decimal real3 = 752538.457m; // 128 bits, intervalo de pelo menos  $-7,9 \times 10^{-28}$  a  $7,9 \times 10^{28}$ , com precisão de pelo menos 28 dígitos

double valor;
valor = real1;
#endregion

Console.WriteLine(valor); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

### #04 Tipo Caractere

```
#region Caracteres
char letra = '\u0041'; //é o código do caracteres
char escape = '\n'; //aplica o escape é a \ assim dá a quebra de linha e para exibir o \ coloque o \\
char literal = 'C';
#endregion

Console.WriteLine(letra); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

### #05 Tipo Boolean

```
#region Boolean
bool verificar = false;
verificar = true;
#endregion

Console.WriteLine(verificar); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

#06 Tipo String, é uma cadeia de caracteres, no caso é textos ela pode ser inicializada com o valor null

```
#region String
string texto = @"Gabriel Artigas\n 1985 @@??$$"; //o @ permite o escape para usar o \n
string mensagem = null;
mensagem = texto;
#endregion

Console.WriteLine(mensagem); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

#07 Tipo Var, ela não tem seu tipo definido porém ela assume o tipo depois da definição e esse tipo fica contido até o final da sua execução

```
#region Var
var valor = 140; //aceita qualquer tipo, porém esse valor fica até o final
#endregion

Console.WriteLine(valor); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

#08 Tipo Objetos, o objetos é a base de todas as outras variáveis, no caso todas as variáveis herda da classe objetos, é a base de todos os tipos,

```
#region Objeto
object obj = false;
obj = 200;
obj = "Gabriel";
#endregion

Console.WriteLine(obj); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

#09 Constantes

```
#region Constantes
const double pi = 3.1415;
const string nome = "Gabriel";
#endregion

Console.WriteLine(nome); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

#10 Enum, é como um tipo de variável que você define os valores da enumeração

```
7 namespace _01VariaveisConstantesTipoDados
8 {
9     0 references
10     internal class Program
11     {
12         2 references
13         enum Notas
14         {
15             Minimo,
16             Media,
17             Maximo
18         }
19     }
20 }

#region Enumeração
Notas notasAlunos = Notas.Media;
#endregion

Console.WriteLine(notasAlunos); //exibe o valor
Console.ReadKey(); //para o console no valor de cima
```

#11 Struct, cria na raiz do projeto

```
4 references
struct Pessoa
{
    public string nome;
    public int idade;
    public double altura;
}

0 references
static void Main(string[] args)
{
    #region Estruturas
    Pessoa p1 = new Pessoa();
    p1.altura = 1.65;
    p1.idade = 35;
    p1.nome = "Gabriel";

    Pessoa p2 = new Pessoa()
    {
        nome = "Arthur",
        idade = 9,
        altura = 1.35
    };

    p1.nome = "Logan";
    #endregion

    Console.WriteLine(p1.nome + " tem a idade " + p1.idade + " e a altura " + p1.altura); //exibe o valor
    Console.ReadKey(); //para o console no valor de cima
}
```

Módulo 02 Entrada e saída de dados

## #01 Métodos de entrada e saída de dados no c#



## #02 Exercício inverte nomes

```
jos | %_02EntradaESaidaDeDados.Program | %Main(string[] args)

#region Exercício inverter nome
string nome1, nome2, nome3, nome4, auxiliar;

Console.WriteLine("Digite o nome #1: ");
nome1 = Console.ReadLine();

Console.WriteLine("Digite o nome #2: ");
nome2 = Console.ReadLine();

Console.WriteLine("Digite o nome #3: ");
nome3 = Console.ReadLine();

Console.WriteLine("Digite o nome #4: ");
nome4 = Console.ReadLine();

auxiliar = nome1;
nome1 = nome4;
nome4 = auxiliar;
auxiliar = nome2;
nome2 = nome3;
nome3 = auxiliar;

Console.WriteLine();
Console.WriteLine("Nomes inserido na sequência invertida");
Console.WriteLine(nome1);
Console.WriteLine(nome2);
Console.WriteLine(nome3);
Console.WriteLine(nome4);
#endregion
```

## Módulo 03 Conversão de tipos de dados

### #01 Conversão implícita de tipos numéricos

```
0 references
static void Main(string[] args)
{
    byte num1 = 100;
    ushort num2;
    num2 = num1; //conversão implícita, permite receber valores de outra variável

    float num3 = 1250.45f;
    double num5 = num3; //conversão implícita, permite receber valores de outra variável

    Console.WriteLine(num2);
    Console.ReadKey();
}
```

## #02 Conversão explícita de tipos numéricos

```
#region Conversão explícita
ushort num1 = 100;
byte num2 = (byte)num1; //força ele converter para o tipo byte

float num3 = 2500.32f;
int num4 = (int)num3;

char letra = (char)82;
#endregion
```

## #03 Método Parse

```
#region Método Parse
string txtNumero = "1985";
int numero = int.Parse(txtNumero); // converte para int o parse

byte num1 = byte.Parse("120");
double num2 = double.Parse("123.32");
#endregion
```

## #04 Classe Convert

```
#region Método Convert
string texto = Convert.ToString(29010290); // converte para string
double num1 = Convert.ToDouble(false); // retorna 0 por que 0 é falso
int nume = Convert.ToInt32('C');
#endregion
```

## Módulo 04 Operadores

#01 Operadores aritméticos

#02 Precedência de operadores aritméticos

#03 Operadores de incremento e decremento

#04 Operador de concatenação

#05 Operadores de atribuição

#06 Operadores de igualdade

#07 Operadores relacionais

#08 Operadores lógicos

#09

#10

## Módulo 19 Manipular arquivos e pastas

## #01 manipulação de arquivos e classe file

```
private void button1_Click(object sender, EventArgs e)
{
    string pasta = @"C:\curso\";
    string arquivo = "teste.txt";

    bool res = File.Exists(pasta + arquivo); // verifica se um arquivo existe
    File.Delete(pasta + arquivo); // apaga o arquivo

    if (!File.Exists(pasta + arquivo))
    {
        File.Create(pasta + arquivo).Close(); // cria um arquivo, e depois fecha
        label1.Text = "arquivo criado.";
    } else
    {
        label1.Text = "já existe.";
    }

    if (File.Exists(pasta + arquivo))
    {
        // copia um arquivo, 1 arquivo copia, 2 onde coloca e o nome, 3 permite sobrescrever o arquivo
        File.Copy(pasta + arquivo, pasta + "Copia.txt", true);
        label1.Text = "arquivo copiado.";
    }
    else
    {
        label1.Text = "não copiou.";
    }

    label1.Text = res.ToString();
}

// move o arquivo 1, e 2 onde coloca mais o nome
File.Move(pasta + arquivo, pasta + "Movido.txt");

// escrever no arquivo, 1 pasta e onde, 2 oque escrever, 3 o encoding se é utf8 etc
File.WriteAllText(pasta + arquivo, "Bom dia funcionou", Encoding.Default);

// para ler o arquivo de texto 1 origem, e o encoding
string textoDoArquivo = File.ReadAllText(pasta + arquivo, Encoding.Default);

File.GetCreationTime(pasta + arquivo); // pega o dia que o arquivo foi criado
```

## #02 A classe fileinfo

## #03 Manipulação de pastas a classe directory

## #04 A classe DirectoryInfo

## #05 Escrevendo arquivos de texto

## #06 Leitura de arquivos de texto

## #07 Leitura de arquivo binário

## #08 Escrevendo arquivos binário