

# Configuração do Ambiente

## Instalação do Visual Studio 2022

O **Visual Studio 2022** é um dos ambientes de desenvolvimento integrado (IDE) mais populares para programar em C#.

### Passo a Passo para Configurar o Ambiente:

#### Baixar e instalar o Visual Studio 2022

1. Acesse o site oficial: <https://visualstudio.microsoft.com/pt-br/>
2. Baixe a versão **Community** (gratuita).
3. Execute o instalador.

### Opções a Selecionar na Instalação do Visual Studio 2022

Durante a instalação, o Visual Studio permite escolher **cargas de trabalho (workloads)**, que são pacotes de ferramentas e bibliotecas necessárias para determinados tipos de projetos.

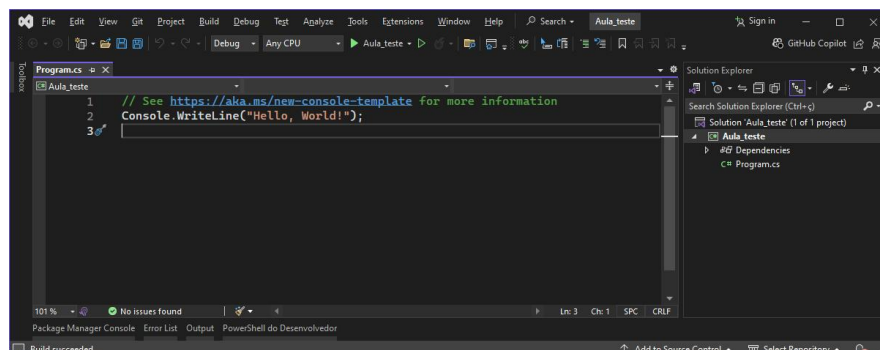
Para nossas aulas, selecione as seguintes cargas de trabalho:

1. **Desenvolvimento para Plataforma .NET**
  - a. Necessário para criar **aplicações de console** em C#.
  - b. Inclui o **.NET SDK** e o **.NET Runtime**.
2. **Desenvolvimento ASP.NET e Web**
  - a. Adiciona suporte para desenvolvimento web com **ASP.NET Core MVC**.
  - b. Inclui bibliotecas para desenvolvimento de **APIs REST**.
  - c. Permite trabalhar com **HTML, CSS e JavaScript**.
3. Se necessário **instale o SDK do .NET (se necessário)**.

## Criando o Primeiro Projeto

1. Abra o Visual Studio 2022
2. Clique em **Criar um novo projeto**
3. Escolha a opção **"Aplicação de Console (.NET Core)"**
4. Dê um nome ao projeto, escolha o local para salvar
5. Escolha o framework e clique em **Criar**

Agora estamos prontos para escrever nosso primeiro código em C#!



# Comandos de saída, concatenação e interpolação.

## Console.WriteLine()

Imprime uma mensagem no console e move o cursor para a próxima linha.

```
Console.WriteLine("Olá, mundo!");  
Console.WriteLine("Este é um exemplo de saída em C#.");
```

Saída no console:

```
Olá, mundo!  
Este é um exemplo de saída em C#.
```

## Console.Write()

Imprime uma mensagem no console sem mover o cursor para a próxima linha.

```
Console.Write("Primeira parte...");  
Console.Write(" Segunda parte na mesma linha.");
```

Saída no console:

```
Primeira parte... Segunda parte na mesma linha.
```

## Console.WriteLine() com Variáveis e concatenação

Você pode combinar texto e variáveis dentro do WriteLine().

```
string nome = "João";  
int idade = 25;  
Console.WriteLine("Nome: " + nome + ", Idade: " + idade);
```

Saída no console:

```
Nome: João, Idade: 25
```

## Console.WriteLine() com Interpolação de Strings

Uma forma mais moderna e legível de exibir variáveis.

```
string produto = "Notebook";  
double preco = 2999.99;  
Console.WriteLine($"Produto: { produto }, Preço: R$ {preco}");
```

Saída no console:

```
Produto: Notebook, Preço: R$ 2999,99
```

## Console.WriteLine() com Formatação

Você pode formatar números e textos com placeholders ({0}, {1}, ...).

```
Console.WriteLine("O número {0} multiplicado por {1} é igual a {2}", 5, 3, 5 * 3);
```

Saída no console:

```
O número 5 multiplicado por 3 é igual a 15
```

## Console.WriteLine() com Quebra de Linha e Tabulação

Você pode usar \n para quebra de linha e \t para tabulação.

```
Console.WriteLine("Linha 1\nLinha 2");  
Console.WriteLine("Coluna 1\tColuna 2\tColuna 3");
```

Saída no console:

```
Linha 1  
Linha 2  
Coluna 1   Coluna 2   Coluna 3
```

## Console.ReadKey();

- O programa pausa a execução até que o usuário pressione qualquer tecla.
- A tecla pressionada não precisa ser Enter; qualquer tecla funciona.
- Após pressionar, o programa continua a execução.

Se quiser que a tecla pressionada não apareça no console, use:

```
Console.ReadKey(true);
```

Isso evita que a tecla pressionada seja exibida na tela.

```
Console.WriteLine("Pressione qualquer tecla para continuar...");  
Console.ReadKey();
```

## Console.Clear()

Limpa a tela do console.

```
Console.Clear(); // Apaga tudo que foi exibido anteriormente.
```

# Entrada de dados e tipo de dados

Em C#, usamos `Console.ReadLine()` para capturar entradas do usuário. Aqui estão alguns exemplos de entrada de dados:

## Lendo um Texto (String)

O método `Console.ReadLine()` captura uma linha de texto digitada pelo usuário.

```
Console.Write("Digite seu nome: ");  
string nome = Console.ReadLine();  
Console.WriteLine($"Olá, {nome}!");
```

Saída no console:

```
Digite seu nome: João  
Olá, aluno!
```

## Lendo um Número Inteiro

Para capturar um número inteiro, usamos `int.Parse()` ou `Convert.ToInt32()`.

```
Console.Write("Digite sua idade: ");  
int idade = int.Parse(Console.ReadLine());  
Console.WriteLine($"Você tem {idade} anos.");
```

Saída no console:

```
Digite sua idade: 25  
Você tem 25 anos.
```

### Atenção!

Se o usuário digitar algo que não seja um número, o programa pode gerar erro.

## Lendo um Número Decimal (double)

Para capturar números decimais, usamos `double.Parse()` ou `Convert.ToDouble()`, levando em conta o separador decimal do sistema (. ou ,).

```
Console.Write("Digite um número decimal: ");  
double numero = double.Parse(Console.ReadLine());  
Console.WriteLine($"Você digitou: {numero}");
```

Saída no console:

```
Digite um número decimal: 3.14  
Você digitou: 3.14
```

## Lendo um Caractere Único

```
Console.Write("Digite uma tecla: ");  
char tecla = Convert.ToChar(Console.ReadLine());  
Console.WriteLine($"\\nVocê digitou: {tecla}");
```

Saída no console:

```
Digite uma tecla: A  
Você digitou: A
```

## Lendo um Valor Booleano

Podemos capturar true ou false como entrada do usuário.

```
Console.Write("Você gosta de programação? (true/false): ");  
bool gostaDeProgramar = bool.Parse(Console.ReadLine());  
Console.WriteLine($"Resposta: {gostaDeProgramar}");
```

Saída no console:

```
Você gosta de programação? (true/false): true  
Resposta: True
```

## Operadores aritméticos [+ , - , \* , / e %]

### Operador de Adição (+)

```
int a = 10;  
int b = 5;  
int soma = a + b;
```

```
Console.WriteLine($"Soma: {a} + {b} = {soma}"); // Saída: Soma: 10 + 5 = 15
```

### Operador de Subtração (-)

```
int x = 20;  
int y = 8;  
int subtracao = x - y;
```

```
Console.WriteLine($"Subtração: {x} - {y} = {subtracao}"); // Saída: Subtração: 20 - 8 = 12
```

### Operador de Multiplicação (\*)

```
int largura = 4;  
int altura = 3;
```

```
int area = largura * altura;

Console.WriteLine($"Área do retângulo: {largura} * {altura} = {area}");
// Saída: Área do retângulo: 4 * 3 = 12
```

## Operador de Divisão (/)

```
int total = 15;
int partes = 3;
int resultado = total / partes;

Console.WriteLine($"Divisão: {total} / {partes} = {resultado}"); // Saída: Divisão: 15 / 3 = 5
```

**Atenção:** Se ambos os operandos forem inteiros, a divisão retorna um número inteiro, descartando a parte decimal.

**Exemplo com double para divisão exata:**

```
double resultadoExato = 15.0 / 4;
Console.WriteLine($"Divisão exata: {resultadoExato}"); // Saída: 3.75
```

## Operador de Módulo (%)

O operador % retorna o **resto da divisão**.

```
int dividendo = 17;
int divisor = 4;
int resto = dividendo % divisor;

Console.WriteLine($"Resto da divisão: {dividendo} % {divisor} = {resto}");
// Saída: Resto da divisão: 17 % 4 = 1
```

## Exemplo Completo

```
using System;

class Program
{
    static void Main()
    {
        int a = 10, b = 3;

        Console.WriteLine($"Soma: {a} + {b} = {a + b}");
        Console.WriteLine($"Subtração: {a} - {b} = {a - b}");
        Console.WriteLine($"Multiplicação: {a} * {b} = {a * b}");
        Console.WriteLine($"Divisão: {a} / {b} = {a / b}"); // Retorna número inteiro
        Console.WriteLine($"Divisão exata: {(double)a / b}"); // Retorna número decimal
        Console.WriteLine($"Módulo: {a} % {b} = {a % b}");
    }
}
```

## Atribuição composta [ +=, -=, \*=, /=, %=]

Esses operadores realizam a operação e atribuem o valor ao próprio operando.

### Operador += (Adição e Atribuição)

```
int numero = 5;
numero += 3; // Equivalente a: numero = numero + 3;

Console.WriteLine($"Resultado de += : {numero}"); // Saída: 8
```

### Operador -= (Subtração e Atribuição)

```
int numero = 10;
numero -= 4; // Equivalente a: numero = numero - 4;

Console.WriteLine($"Resultado de -= : {numero}"); // Saída: 6
```

### Operador \*= (Multiplicação e Atribuição)

```
int numero = 7;
numero *= 2; // Equivalente a: numero = numero * 2;

Console.WriteLine($"Resultado de *= : {numero}"); // Saída: 14
```

### Operador /= (Divisão e Atribuição)

```
int numero = 20;
numero /= 4; // Equivalente a: numero = numero / 4;

Console.WriteLine($"Resultado de /= : {numero}"); // Saída: 5
```

**Atenção:** Se numero e o divisor forem inteiros, a divisão será inteira (descarta a parte decimal). Se precisar de um resultado com casas decimais, use double:

```
double valor = 20;
valor /= 3; // Equivalente a: valor = valor / 3;

Console.WriteLine($"Divisão exata: {valor}"); // Saída: 6.666666666666667
```

### Operador %= (Módulo e Atribuição)

```
int numero = 15;
numero %= 4; // Equivalente a: numero = numero % 4;
```

```
Console.WriteLine($"Resultado de %= : {numero}"); // Saída: 3
```

## Exemplo Completo

```
int num = 10;

Console.WriteLine($"Valor inicial: {num}");

num += 5; // Adição e atribuição
Console.WriteLine($"Após += 5: {num}");

num -= 3; // Subtração e atribuição
Console.WriteLine($"Após -= 3: {num}");

num *= 2; // Multiplicação e atribuição
Console.WriteLine($"Após *= 2: {num}");

num /= 4; // Divisão e atribuição
Console.WriteLine($"Após /= 4: {num}");

num %= 3; // Módulo e atribuição
Console.WriteLine($"Após %= 3: {num}");
```

### Resumo:

- += Adiciona e atribui.
- -= Subtrai e atribui.
- \*= Multiplica e atribui.
- /= Divide e atribui (atenção para divisão inteira!).
- %= Calcula o resto e atribui.

## Incremento e decremento [++, --]

Os operadores ++ (incremento) e -- (decremento) são usados para aumentar ou diminuir o valor de uma variável **em uma unidade**.

Eles podem ser utilizados em duas formas:

- **Pré-incremento** (++var) → Primeiro modifica o valor, depois usa a variável.
- **Pré-decremento** (--var) → Primeiro modifica o valor, depois usa a variável.
- **Pós-incremento** (var++) → Primeiro usa a variável, depois modifica o valor.
- **Pós-decremento** (var--) → Primeiro usa a variável, depois modifica o valor.



## Exemplo com ++ (Incremento)

```
int numero = 5;

Console.WriteLine($"Valor inicial: {numero}");

// Pós-incremento (usa o valor e depois soma 1)
Console.WriteLine($"Pós-incremento: {numero++}"); // Saída: 5
Console.WriteLine($"Depois do pós-incremento: {numero}"); // Saída: 6

// Pré-incremento (soma 1 antes de usar)
Console.WriteLine($"Pré-incremento: {++numero}"); // Saída: 7
```

## Exemplo com -- (Decremento)

```
int numero = 10;

Console.WriteLine($"Valor inicial: {numero}");

// Pós-decremento (usa o valor e depois subtrai 1)
Console.WriteLine($"Pós-decremento: {numero--}"); // Saída: 10
Console.WriteLine($"Depois do pós-decremento: {numero}"); // Saída: 9

// Pré-decremento (subtrai 1 antes de usar)
Console.WriteLine($"Pré-decremento: {--numero}"); // Saída: 8
```

## Exemplo Completo

```
int valor = 3;

Console.WriteLine($"Valor inicial: {valor}");

Console.WriteLine($"Pós-incremento: {valor++}"); // Usa 3, depois incrementa
Console.WriteLine($"Depois do pós-incremento: {valor}"); // Agora é 4

Console.WriteLine($"Pré-incremento: {++valor}"); // Incrementa antes de exibir (agora é 5)

Console.WriteLine($"Pós-decremento: {valor--}"); // Usa 5, depois decrementa
Console.WriteLine($"Depois do pós-decremento: {valor}"); // Agora é 4

Console.WriteLine($"Pré-decremento: {--valor}"); // Decrementa antes de exibir (agora é 3)
```

## Usando GetType()

No **C#**, a forma mais comum de obter o tipo de uma variável é utilizando o método `.GetType()`.

```
int numero = 10;
Console.WriteLine(numero.GetType()); // Saída: System.Int32
```

# Lista de exercícios para sua adaptação à linguagem

## 01 - Cálculo de Distância Percorrida

Crie um programa que leia o **tempo** (em horas) e a **velocidade média** (em km/h) e calcule a **distância percorrida** utilizando a fórmula:

$$\text{distância} = \text{Velocidade média} / \text{tempo}$$

*Exemplo:*

Digite o tempo (em horas): 3

Digite a velocidade média (km/h): 60

Distância percorrida: 180.00 km

## 02 - Cálculo da Área de uma Esfera

Crie um programa que leia o **raio** (r) de uma esfera e calcule a **área superficial** da esfera usando a fórmula:

$$A = 4 * \pi * r^2$$

Onde  $\pi$  é aproximadamente 3,14159.

*Exemplo:*

Digite o raio da esfera: 5

Área superficial: 314.16

## 03 - Cálculo do Volume de um Cone

Crie um programa que leia o **raio** (r) e a **altura** (h) de um cone e calcule o **volume** utilizando a fórmula:

$$V = 1/3 * \pi * r^2 * h$$

*Exemplo:*

Digite o raio do cone: 3

Digite a altura do cone: 5

Volume: 141.37

## 04 - Conversão de Temperatura (Escala Fahrenheit para Celsius)

Crie um programa que leia uma **temperatura** em **Fahrenheit** e converta para **Celsius** usando a fórmula:

$$C = 5 / 9 * (F - 32)$$

*Exemplo:*

Digite a temperatura em Fahrenheit: 100

Temperatura em Celsius: 37.8

## 05 - Cálculo do Salário com Desconto de Impostos

Crie um programa que leia o **salário bruto** de um trabalhador e aplique os seguintes descontos:

- Desconto de **INSS** de 8%
- Desconto de **Imposto de Renda** de 12%

Exiba o **salário líquido** após os descontos.

*Exemplo:*

Digite o salário bruto: 3000

Desconto de INSS: 240

Desconto de Imposto de Renda: 360

Salário líquido: 2400

## 06 - Cálculo da Média Ponderada

Crie um programa que leia **três notas** e seus respectivos **pesos**, e calcule a **média ponderada** utilizando a fórmula:

$$Mp = [ (N1 * P1) + (N2 * P2) + (N3 * P3) ] / (P1 + P2 + P3)$$

*Exemplo:*

Digite a primeira nota: 7

Digite o peso da primeira nota: 3

Digite a segunda nota: 8

Digite o peso da segunda nota: 2

Digite a terceira nota: 9

Digite o peso da terceira nota: 1

Média ponderada: 7.67

## 07 - Cálculo da Perda de Massa de um Corpo

Crie um programa que leia a **massa inicial** de um corpo (em kg) e calcule a **massa final** após perder 10% de sua massa inicial, 20% e 30%.

*Exemplo:*

Digite a massa inicial: 80

Massa final após perder 10%: 72.00

Massa final após perder 20%: 64.00

Massa final após perder 30%: 56.00

### Observação:

Trabalhe cada exercício de forma independente, realizando as **operações matemáticas** corretas e utilizando os operadores de forma precisa.

Ao final, **teste os cálculos** com diferentes valores para garantir que o programa está funcionando corretamente.

Faça **reflexões** sobre como os operadores podem ser usados de forma eficiente para resolver problemas matemáticos e lógicos.

# Estrutura de controle condicional

## IF

### Exemplo com if simples

Verifica se um número é positivo.

```
int numero = 10;

if (numero > 0)
{
    Console.WriteLine("O número é positivo.");
}
```

### Exemplo com if-else

Verifica se um número é par ou ímpar.

```
int numero = 7;

if (numero % 2 == 0)
{
    Console.WriteLine("O número é par.");
}
else
{
    Console.WriteLine("O número é ímpar.");
}
```

### Exemplo com if encadeado (else if)

Solicita um dia da semana ao usuário e retorna uma mensagem correspondente.

```
Console.WriteLine("Digite um número de 1 a 7 para o dia da semana: ");
int dia = Convert.ToInt32(Console.ReadLine());

if (dia == 1)
{
    Console.WriteLine("Domingo");
}
else if (dia == 2)
{
    Console.WriteLine("Segunda-feira");
}
else if (dia == 3)
{
    Console.WriteLine("Terça-feira");
}
else if (dia == 4)
{

```

```

        Console.WriteLine("Quarta-feira");
    }
    else if (dia == 5)
    {
        Console.WriteLine("Quinta-feira");
    }
    else if (dia == 6)
    {
        Console.WriteLine("Sexta-feira");
    }
    else if (dia == 7)
    {
        Console.WriteLine("Sábado");
    }
    else
    {
        Console.WriteLine("Número inválido! Digite um valor entre 1 e 7.");
    }
}

```

## Exemplo com if identado (aninhado)

Solicita uma letra ao usuário e verifica se é uma vogal ou consoante.

```

Console.Write("Digite uma letra: ");
string letra = Console.ReadLine().ToUpper();

if (letra.Length == 1) // Verifica se é uma única letra
{
    Console.WriteLine($"A letra digitada foi {letra} e é ");

    if (letra == "A")
    {
        Console.WriteLine("uma vogal.");
    }
    else
    {
        if (letra == "E")
        {
            Console.WriteLine("uma vogal.");
        }
        else
        {
            if (letra == "I")
            {
                Console.WriteLine("uma vogal.");
            }
            else
            {
                if (letra == "O")
                {
                    Console.WriteLine("uma vogal.");
                }
                else
                {
                    if (letra == "U")
                    {

```

```

        Console.WriteLine("uma vogal.");
    }
    else
    {
        Console.WriteLine("uma consoante.");
    }
}
}
}
}
else
{
    Console.WriteLine("Entrada inválida! Digite apenas uma única letra.");
}

```

## Switch

### Usando switch com números

Este código recebe um número de 1 a 7 e informa o dia correspondente.

```

Console.Write("Digite um número de 1 a 7: ");
int numero = Convert.ToInt32(Console.ReadLine());

switch (numero)
{
    case 1:
        Console.WriteLine("O dia correspondente é Domingo.");
        break;
    case 2:
        Console.WriteLine("O dia correspondente é Segunda-feira.");
        break;
    case 3:
        Console.WriteLine("O dia correspondente é Terça-feira.");
        break;
    case 4:
        Console.WriteLine("O dia correspondente é Quarta-feira.");
        break;
    case 5:
        Console.WriteLine("O dia correspondente é Quinta-feira.");
        break;
    case 6:
        Console.WriteLine("O dia correspondente é Sexta-feira.");
        break;
    case 7:
        Console.WriteLine("O dia correspondente é Sábado.");
        break;
    default:
        Console.WriteLine("Número inválido! Digite um número de 1 a 7.");
        break;
}

```

## Usando switch com letras

Este código recebe uma letra e informa se é uma vogal ou consoante.

```
Console.Write("Digite uma letra: ");
string letra = Console.ReadLine().ToUpper();

if (letra.Length == 1)
{
    switch (letra)
    {
        case "A":
        case "E":
        case "I":
        case "O":
        case "U":
            Console.WriteLine($"A letra digitada foi {letra} e é uma vogal.");
            break;
        default:
            Console.WriteLine($"A letra digitada foi {letra} e é uma consoante.");
            break;
    }
}
else
{
    Console.WriteLine("Entrada inválida! Digite apenas uma única letra.");
}
```

## Operador ternário [if ternário]

```
Console.Write("Digite um número: ");
int numero = Convert.ToInt32(Console.ReadLine());

string resultado = (numero % 2 == 0) ? "par" : "ímpar";

Console.WriteLine($"O número digitado foi {numero} e ele é {resultado}.");
```

## Lista de exercícios para sua adaptação à linguagem

### 01 - Identificação do Mês

Crie um programa que leia um número de 1 a 12 e informe o nome do mês correspondente. Caso o número esteja fora dessa faixa, exiba uma mensagem de erro.

### 02 - Calculadora de IMC

Crie um programa que leia o peso (em kg) e a altura (em metros) de uma pessoa e calcule o **Índice de Massa Corporal (IMC)**. O IMC deve ser calculado utilizando a fórmula:

$$\text{IMC} = \text{peso} / (\text{altura} * \text{altura})$$



Em seguida, classifique a pessoa de acordo com o IMC calculado:

- Abaixo de 18.5: **Abaixo do peso**
- De 18.5 a 24.9: **Peso normal**
- De 25 a 29.9: **Sobrepeso**
- De 30 ou mais: **Obesidade**

### 03 - Verificação de Ano Bissexto

Crie um programa que leia um ano e informe se ele é **bissexto** ou não. Um ano é bissexto se:

- É divisível por 4, mas não por 100.
- Ou é divisível por 400.

### 04 - Calculadora de Desconto

Crie um programa que leia o valor de uma compra e informe o preço final após aplicar um **desconto**. Se o valor da compra for superior a **R\$ 100**, aplique um **desconto de 10%**. Caso contrário, aplique um desconto de **5%**.

### 05 - Verificação de Triângulo

Crie um programa que leia os **três lados** de um triângulo (números positivos) e informe se o triângulo é:

- **Equilátero**: todos os lados são iguais.
- **Isósceles**: dois lados são iguais.
- **Escaleno**: todos os lados são diferentes.

Caso os lados informados não formem um triângulo válido (por exemplo, a soma de dois lados deve ser maior que o terceiro), exiba uma mensagem de erro.

### 06 - Ordem Crescente de Três Números

**Enunciado:**

Crie um programa que leia **três números** e exiba-os em ordem crescente. Se os números forem iguais, exiba uma mensagem dizendo que todos são iguais.

# Estrutura de repetição

## For

### Ordem Crescente (1 a 10)

```
// Contando de 1 a 10 em ordem crescente
for ( int i = 1; i <= 10; i++ )
{
    Console.WriteLine(i);
}
```

### Ordem Decrescente (10 a 1)

```
// Contando de 10 a 1 em ordem decrescente
for ( int i = 10; i >= 1; i-- )
{
    Console.WriteLine(i);
}
```

### Ordem Diferente (Pulando de 2 em 2)

```
// Contando de 2 em 2, começando de 1
for ( int i = 1; i <= 10; i += 2 )
{
    Console.WriteLine(i);
}
```

## While

### Exemplo: Contando de 1 a 10

```
int i = 1;

// Usando o laço while para contar de 1 a 10
while (i <= 10)
{
    Console.WriteLine(i);
    i++; // Incrementa 1 a cada iteração
}
```

## Do / While

```
int i = 1;

// Usando o laço do / while para contar de 1 a 10
do
{
    Console.WriteLine(i);
    i++; // Incrementa 1 a cada iteração
} while (i <= 10);
```

## Lista de exercícios

### 01 - Contagem de 1 a 100

Crie um programa para imprimir de **1 a 100** . Em seguida, faça o programa imprimir somente os números **ímpares** compreendidos entre 1 e 100 (inclusive).

### 02 - Tabuada de um Número

Crie um programa que leia um número e, usando o laço while, imprima a **tabuada** desse número de 1 a 10. Exemplo de saída para o número 5:

```
5 x 1 = 5
5 x 2 = 10
...
5 x 10 = 50
```

### 03. Soma de Números

Crie um programa que leia números positivos do usuário e calcule e imprima a soma desses números até que o usuário digite um número negativo (quando o programa deve parar de pedir entradas e mostrar a soma final).

### 04. Fatorial de um Número

Crie um programa que leia um número e calcule seu **fatorial**. O fatorial de um número N é dado pela multiplicação de todos os números inteiros de N até 1 (exemplo:  $5! = 5 \times 4 \times 3 \times 2 \times 1$ ). O programa deve exibir o valor do fatorial.

# Programação Orientada a Objetos

**Programação Orientada a Objetos (POO)** é uma forma de programar que organiza o código em **objetos**, que são representações de coisas do mundo real, como um carro, um aluno ou uma conta bancária. Cada objeto tem **características** (dados) e **ações** (funções).

Os principais conceitos da POO são:

- **Encapsulamento:** Protege os dados e só permite acesso controlado.
- **Herança:** Permite criar novos objetos a partir de outros já existentes.
- **Polimorfismo:** Permite que uma ação funcione de maneiras diferentes.
- **Abstração:** Esconde detalhes complexos e mostra apenas o necessário.

Isso torna os programas mais organizados, fáceis de entender e de reaproveitar.

## O que são Classes e Objetos?

Pense em uma **classe** como um molde ou uma receita. Ela define como algo deve ser, mas sozinha não faz nada. Já um **objeto** é algo real criado a partir dessa receita.

### Exemplo simples:

Imagine que uma **classe** seja o molde de um bolo. Ela define os ingredientes e o modo de preparo, mas não é um bolo de verdade. Quando seguimos a receita e assamos o bolo, criamos um **objeto** — um bolo real que podemos ver e comer!

## Diferença entre Classe e Objeto:

- **Classe** → É o modelo, a descrição ou a "receita" de um objeto.
- **Objeto** → É algo concreto, criado a partir da classe.

## Exemplo de uma classe (Molde)

```
class Carro
{
    public string cor;
    public string modelo;
}
```

## Exemplo de um objeto feito a partir da classe (molde) carro

```
Carro carro1 = new Carro();
carro1.cor = "Vermelho";
carro1.modelo = "Sedan";

Carro carro2 = new Carro();
carro2.cor = "Azul";
carro2.modelo = "SUV";
```

Aqui, **Carro** é a **classe** (o molde), e **carro1** e **carro2** são **objetos** (carros reais com cores e modelos diferentes).

## Exemplo - Criando uma classe endereço

```
class Endereco
{
    // Propriedades da classe Endereco
    public string Rua { get; set; }
    public string Bairro { get; set; }
    public string Cidade { get; set; }
    public string Estado { get; set; }
    public string CEP { get; set; }

    // Construtor opcional para inicializar os valores ao criar um objeto
    public Endereco(string rua, string bairro, string cidade, string estado, string cep)
    {
        Rua = rua;
        Bairro = bairro;
        Cidade = cidade;
        Estado = estado;
        CEP = cep;
    }

    // Método para exibir o endereço formatado
    public void ExibirEndereco()
    {
        Console.WriteLine($"Endereço: {Rua}, {Bairro}, {Cidade} - {Estado}, CEP: {CEP}");
    }
}
```

### Explicação:

- **public string Rua { get; set; }** → Propriedades que armazenam informações do endereço.
- **Construtor** (public Endereco(...)) → Facilita a criação do objeto já com valores.
- **Método** ExibirEndereco → Exibe o endereço formatado.

### Exemplo de uso:

```
class Program
{
    static void Main()
    {
        // Criando um objeto Endereco e atribuindo valores
        Endereco meuEndereco = new Endereco
            ("Rua das Flores", "Centro", "São Paulo", "SP", "01010-000");

        // Exibindo o endereço
        meuEndereco.ExibirEndereco();
    }
}
```

### Saída esperada no console:

Endereço: Rua das Flores, Centro, São Paulo - SP, CEP: 01010-000

## Exemplo - Criando uma classe Pessoa

```
class Pessoa
{
    private string nome;
    private int idade;
    private double peso;
    private double altura;

    // Propriedade Nome
    public string Nome {
        get { return nome; }
        set {
            if ( !string.IsNullOrEmpty(value) )
                nome = value;
            else
                throw new ArgumentException("O nome não pode ser vazio.");
        }
    }

    // Propriedade Idade com validação
    public int Idade {
        get { return idade; }
        set {
            if (value > 0)
                idade = value;
            else
                throw new ArgumentException("A idade deve ser maior que zero.");
        }
    }

    // Propriedade Peso com validação
    public double Peso {
        get { return peso; }
        set {
            if (value > 0)
                peso = value;
            else
                throw new ArgumentException("O peso deve ser maior que zero.");
        }
    }

    // Propriedade Altura com validação
    public double Altura {
        get { return altura; }
        set {
            if (value > 0)
                altura = value;
            else
                throw new ArgumentException("A altura deve ser maior que zero.");
        }
    }
}
```

```

    }

    // Construtor da classe
    public Pessoa(string nome, int idade, double peso, double altura) {
        Nome = nome;
        Idade = idade;
        Peso = peso;
        Altura = altura;
    }

    // Método para exibir informações da pessoa
    public void ExibirInformacoes() {
        Console.WriteLine
            ($"Nome: {Nome}, Idade: {Idade} anos, Peso: {Peso} kg, Altura: {Altura} m");
    }
}

```

### Explicação:

Nome → Não pode ser vazio.

Idade → Deve ser maior que zero.

Peso → Deve ser maior que zero.

Altura → Deve ser maior que zero.

**Uso de** throw new ArgumentException(...) → Lança um erro caso o valor não atenda às condições.

### Exemplo de uso:

```

class Program
{
    static void Main()
    {
        try
        {
            Console.Write("Digite o nome: ");
            string nome = Console.ReadLine();

            Console.Write("Digite a idade: ");
            int idade = int.Parse(Console.ReadLine());

            Console.Write("Digite o peso (kg): ");
            double peso = double.Parse(Console.ReadLine());

            Console.Write("Digite a altura (m): ");
            double altura = double.Parse(Console.ReadLine());

            Pessoa pessoa = new Pessoa(nome, idade, peso, altura);
            pessoa.ExibirInformacoes();
        }
        catch (FormatException)
    }
}

```

```

    {
        Console.WriteLine("Erro: Certifique-se de digitar valores numéricos corretamente.");
    }
    catch (ArgumentException ex)
    {
        Console.WriteLine($"Erro: {ex.Message}");
    }
}
}

```

### Saída esperada (caso o segundo objeto falhe):

Nome: Carlos, Idade: 30 anos, Peso: 75,5 kg, Altura: 1,75 m  
 Erro: A altura deve ser maior que zero.

## Lista de exercícios

### 01 - Classe "Produto" com Preço e Desconto

Crie uma classe Produto com as seguintes propriedades:

- Nome (string)
- Preço (double, deve ser maior que zero)
- Desconto (double, percentual entre 0 e 100)

Adicione um **método** chamado CalcularPrecoComDesconto() que retorna o preço final do produto após aplicar o desconto.

#### Exemplo de uso esperado:

```

Produto p = new Produto("Celular", 2000, 10);
Console.WriteLine($"Preço final: {p.CalcularPrecoComDesconto()}");
// Saída: Preço final: 1800

```

### 02 - Classe "Aluno" e Média das Notas

Crie uma classe Aluno com as propriedades:

- Nome (string)
- Nota1 (double, deve ser entre 0 e 10)
- Nota2 (double, deve ser entre 0 e 10)

Adicione um **método** chamado CalcularMedia() que retorna a média das duas notas.



### Exemplo de uso esperado:

```
Aluno aluno = new Aluno("Lucas", 8.5, 7.0);
Console.WriteLine($"Média do aluno: {aluno.CalcularMedia()}");
// Saída: Média do aluno: 7.75
```

## 03 - Classe "Carro" e Consumo de Combustível

Crie uma classe Carro com as propriedades:

- Modelo (string)
- ConsumoPorKm (double, quantos km por litro o carro faz, deve ser maior que zero)

Adicione um **método** chamado CalcularConsumo(double distancia) que recebe uma distância (em km) e retorna quantos litros de combustível serão necessários.

- $\text{Consumo} = \text{distância} / \text{consumoPorLitro}$

### Exemplo de uso esperado:

```
Carro carro = new Carro("Sedan", 12);
Console.WriteLine($"Combustível necessário: {carro.CalcularConsumo(240)} L");
// Saída: Combustível necessário: 20 L
```

## 04 - Classe "Funcionario" e Aumento de Salário

Crie uma classe Funcionario com as propriedades:

- Nome (string)
- Salario (double, deve ser maior que zero)

Adicione um **método** chamado AplicarAumento(double percentual) que aumenta o salário do funcionário pelo percentual informado.

### Exemplo de uso esperado:

```
Funcionario funcionario = new Funcionario("Mariana", 3000);
funcionario.AplicarAumento(10);
Console.WriteLine($"Novo salário: {funcionario.Salario}");
// Saída: Novo salário: 3300
```

## 05 - Classe "Retângulo" e Cálculo de Área e Perímetro

Crie uma classe Retangulo com as propriedades:

- Largura (double, deve ser maior que zero)
- Altura (double, deve ser maior que zero)

Adicione os métodos:

- `CalcularArea()` → Retorna a área do retângulo.
- `CalcularPerimetro()` → Retorna o perímetro do retângulo.

**Exemplo de uso esperado:**

```
Retangulo r = new Retangulo(5, 10);  
Console.WriteLine($"Área: {r.CalcularArea()}");    // Saída: Área: 50  
Console.WriteLine($"Perímetro: {r.CalcularPerimetro()}"); // Saída: Perímetro: 30
```

Esses exercícios ajudam a praticar **criação de classes, validação de dados, encapsulamento e métodos**.

