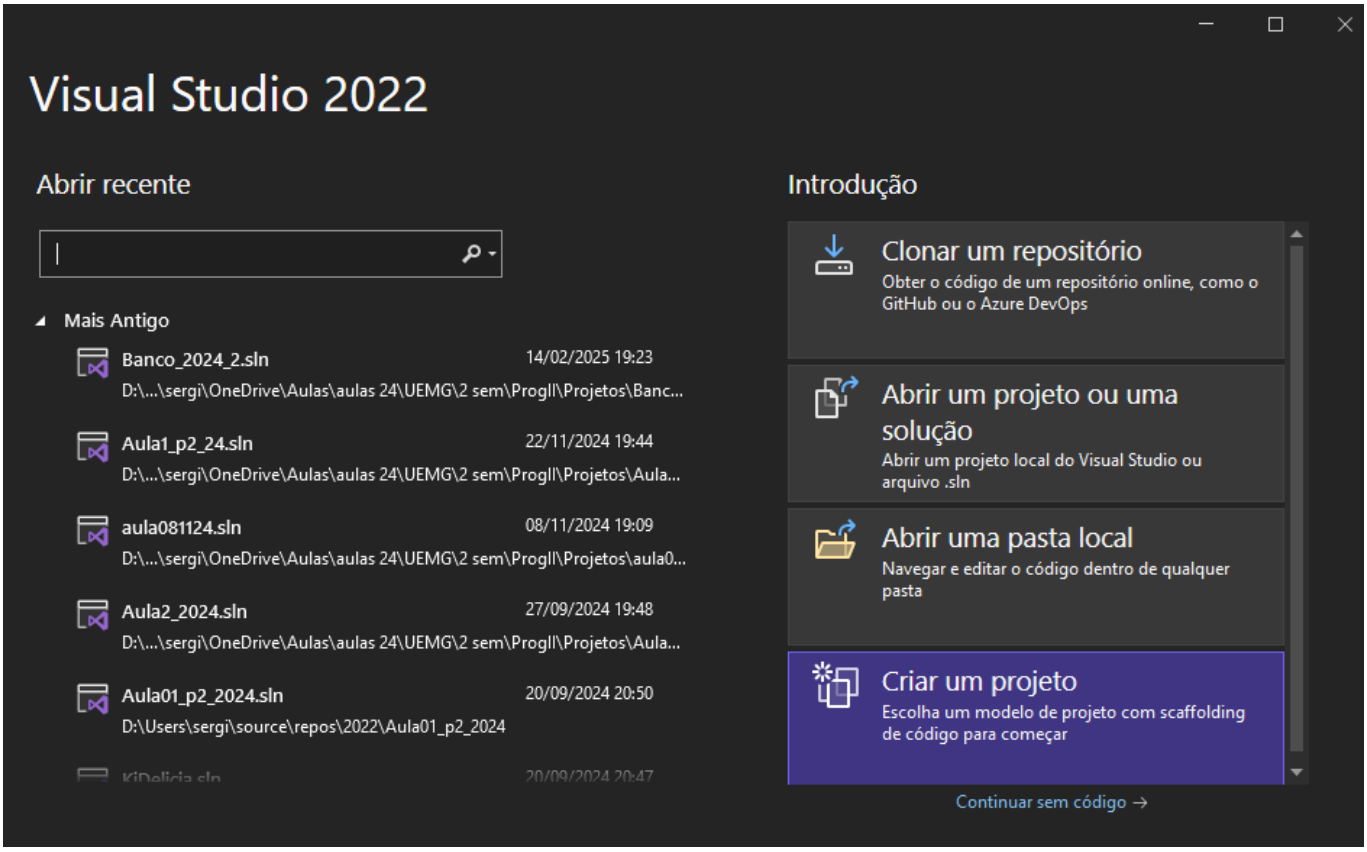
Introdução à Programação em C# - Console no Visual Studio

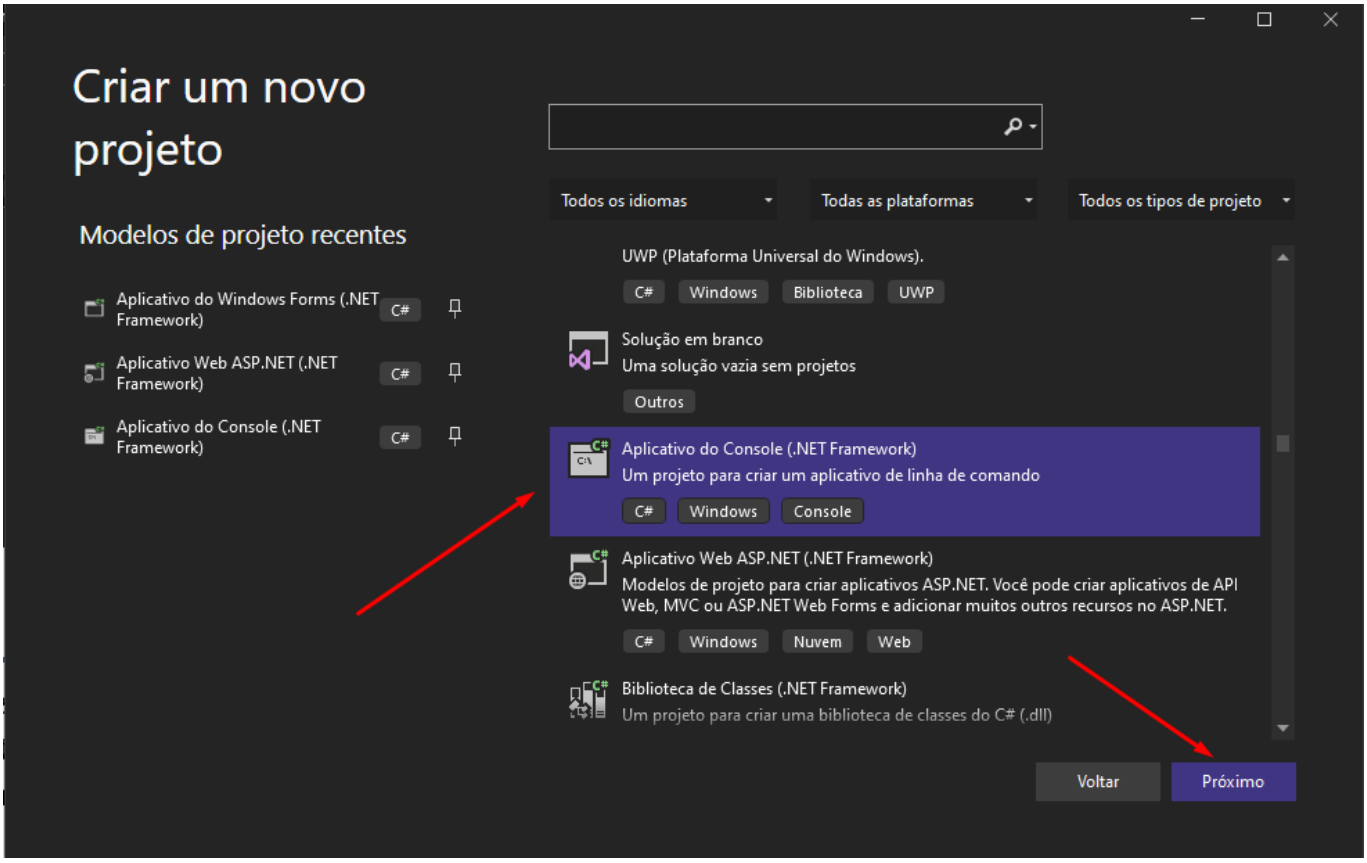
Introdução à Programação em C# - Console no Visual Studio

Abra o Visual Studio 2022

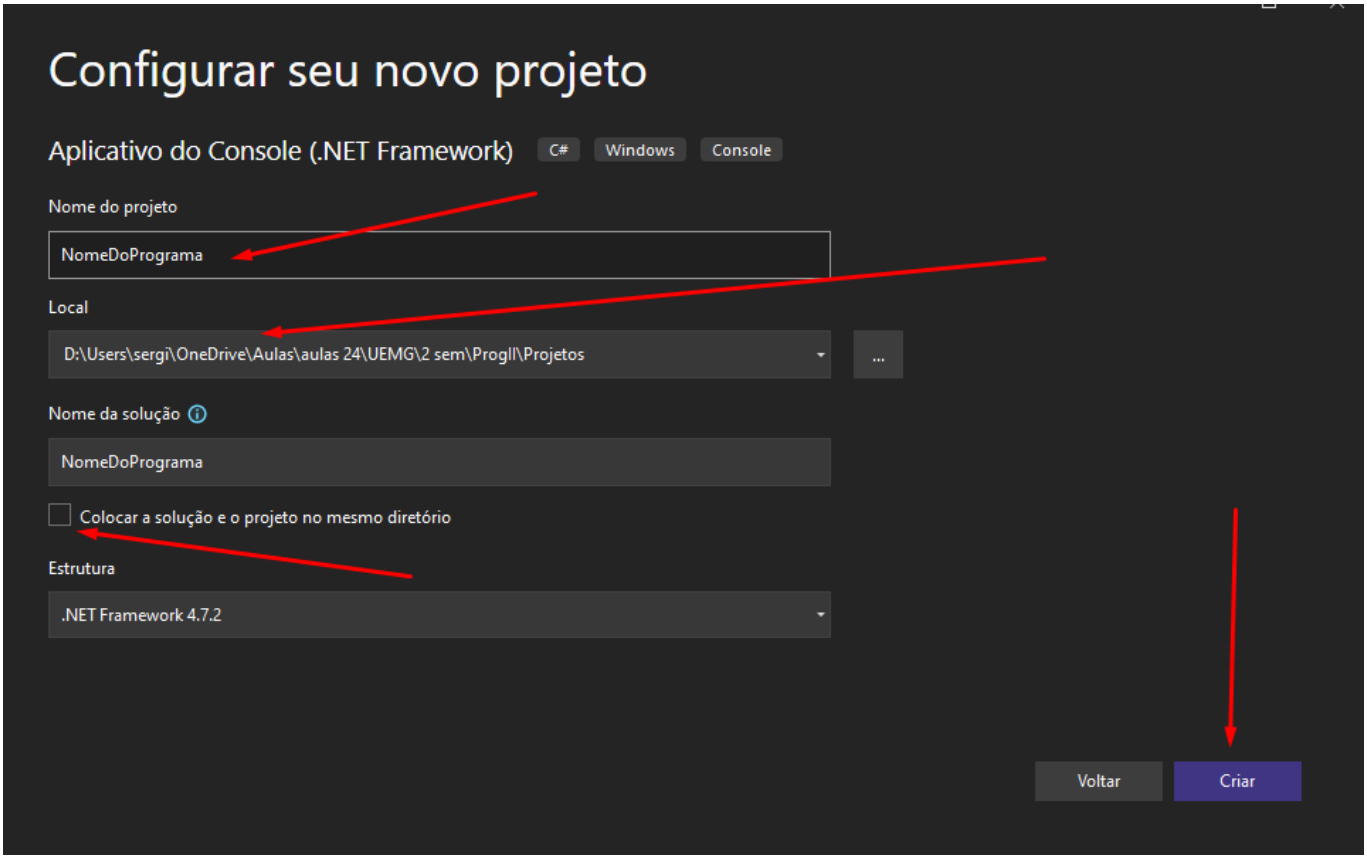
Selecione Criar Novo Projeto



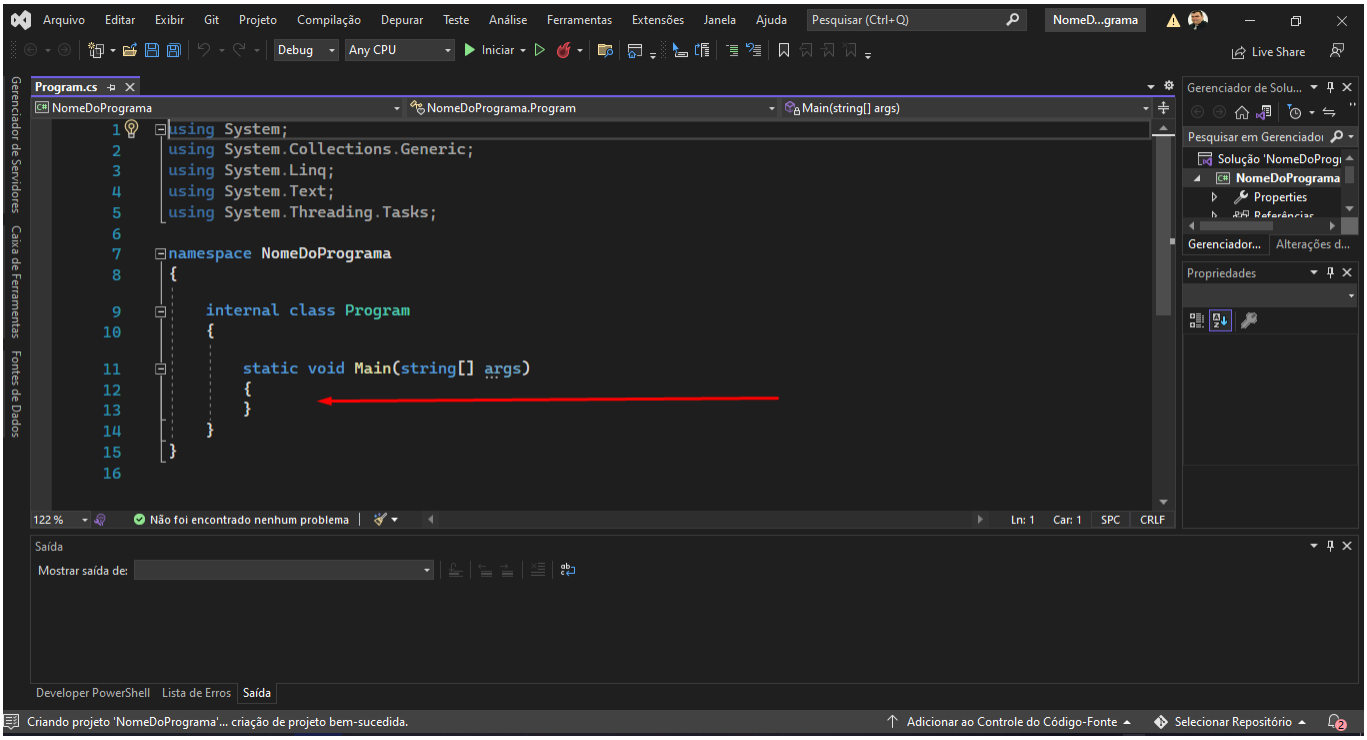
Escolha o tipo de projeto: Aplicativo do Console (.Net Framework), C#, Windows, Console



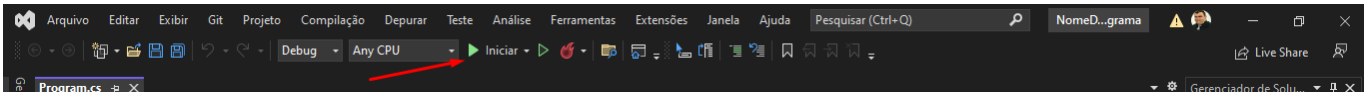
Digite o nome do seu projeto, escolha o local de salvamento e clique em Criar



Seu Código irá dentro de Main(string[] args), entre as chaves.



Para compilar e executar clique em Iniciar



1. Introdução

Esta apostila tem como objetivo introduzir a linguagem de programação C# para iniciantes, utilizando o modo console no Visual Studio. Serão abordados os conceitos básicos da linguagem, incluindo variáveis, entrada e saída de dados, estruturas de decisão e laços de repetição.

2. Tipos de Variáveis

Em C#, as variáveis são utilizadas para armazenar valores temporários na memória durante a execução de um programa.

Principais tipos de variáveis:
- int: números inteiros (ex: int idade = 25;)
- double: números decimais (ex: double peso = 70.5;)
- char: caracteres únicos (ex: char letra = 'A';)
- string: textos (ex: string nome = "Maria";)
- bool: valores booleanos (ex: bool ativo = true;)

3. Operadores Aritméticos e Lógicos

Operadores Aritméticos:
São usados para realizar operações matemáticas básicas.
- Soma: +
- Subtração: -
- Multiplicação: *
- Divisão: /
- Resto da divisão (módulo): %

Exemplo:

```
int a = 10;
int b = 3;
int soma = a + b;
int resto = a % b;
Console.WriteLine("Soma: " + soma);
Console.WriteLine("Resto: " + resto);
```

Operadores Relacionais:
Comparam valores e retornam verdadeiro ou falso.
- Igual: ==
- Diferente: !=
- Maior que: >
- Menor que: <
- Maior ou igual: >=
- Menor ou igual: <=

Operadores Lógicos:
Usados para combinar expressões booleanas.
- E lógico: &&
- Ou lógico: ||
- Negação: !

Exemplo:

```
bool resultado = (idade >= 18 && idade <= 60);
if (!resultado)
{
   Console.WriteLine("Fora da faixa etária permitida.");
}
```

3. Comandos de Entrada e Saída

Em aplicações de console em C#, os comandos de entrada e saída são fundamentais para interação com o usuário.

Saída de Dados:
- Console.Write(): escreve um texto no console sem quebrar linha.
- Console.WriteLine(): escreve um texto no console e adiciona uma quebra de linha.

Exemplo:

```
Console.Write("Digite seu nome: ");
string nome = Console.ReadLine();
Console.WriteLine("Olá, " + nome);
```

Outras funções úteis na saída:
- Console.Clear(): limpa todo o conteúdo exibido no console.
- Console.ForegroundColor: muda a cor do texto. Ex: Console.ForegroundColor = ConsoleColor.Red;
- Console.BackgroundColor: muda a cor de fundo do console. Ex: Console.BackgroundColor = ConsoleColor.Yellow;
- Console.ResetColor(): retorna as cores ao padrão original.

Exemplo com cores:

```
Console.ForegroundColor = ConsoleColor.Green;
Console.WriteLine("Texto em verde");
Console.ResetColor();
```

Entrada de Dados:
- Console.ReadLine(): lê uma linha do console e retorna uma string.
Para ler outros tipos de dados, convertemos a string com métodos como:
- int.Parse(), double.Parse(), bool.Parse()
- Convert.ToInt32(), Convert.ToDouble(), Convert.ToBoolean(), etc.

Exemplos:

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

Console.Write("Digite sua altura: ");
double altura = Convert.ToDouble(Console.ReadLine());
```

Tratamento de Erros:
É recomendável usar try/catch para evitar que o programa trave ao digitar um valor inválido.
Exemplo:

```
try
{
   Console.Write("Digite um número: ");
   int numero = int.Parse(Console.ReadLine());
}
catch (FormatException)
```

```
{
   Console.WriteLine("Entrada inválida. Digite um número inteiro.");
}
```

4. Estruturas de Decisão

As estruturas de decisão permitem que o programa tome caminhos diferentes com base em condições.

Exemplo com if/else:

```
int idade = 18;
if (idade >= 18)
{
   Console.WriteLine("Maior de idade.");
}
else
{
   Console.WriteLine("Menor de idade.");
}
```

Também existe o switch:

```
int opcao = 1;
switch(opcao)
{
   case 1:
      Console.WriteLine("Opção 1");
      break;
   case 2:
      Console.WriteLine("Opção 2");
      break;
   default:
      Console.WriteLine("Outra opção");
      break;
}
```

5. Laços de Repetição

Os laços de repetição permitem executar um bloco de código várias vezes.

while:

```
int i = 0;
while (i < 5)
{
   Console.WriteLine(i);
```

```
    i++;
}
```

for:

```
for (int i = 0; i < 5; i++)
{
    Console.WriteLine(i);
}
```

do-while:

```
int i = 0;
do
{
    Console.WriteLine(i);
    i++;
} while (i < 5);
```

6. Exercícios Práticos

1. Crie um programa que solicite ao usuário seu nome e idade, e exiba uma mensagem como: "Olá, Maria! Você tem 25 anos."

2. Escreva um programa que leia dois números inteiros, some-os e exiba o resultado.

3. Faça um programa que peça ao usuário um número e informe se ele é par ou ímpar.

4. Escreva um programa que simule um menu com três opções:
1 - Cadastrar usuário
2 - Exibir informações
3 - Sair
Use a estrutura switch para tratar a escolha do usuário.

5. Crie um programa que leia um número e imprima a tabuada desse número de 1 a 10.

6. Escreva um programa que conte de 1 até 10 utilizando o laço while.

7. Faça um programa que leia um número e informe se ele é positivo, negativo ou zero.

7. Desafio:

Crie um programa que simule uma senha de acesso. O programa deve pedir ao usuário para digitar uma senha. O usuário tem 3 tentativas para acertar a senha correta ("1234"). Caso acerte, exiba "Acesso permitido". Se errar 3 vezes, exiba "Acesso bloqueado".