

# Programação e Algoritmos



PROFESSOR EVERTON GONZALES SETTE

# O que é programação?

De forma geral, a programação é um processo de escrita, testes e manutenção de programas de computadores. Esses programas, por sua vez, são compostos por conjuntos de instruções determinados pelo programador que descrevem tarefas a serem realizadas pela máquina e atendem diversas finalidades.

# Hardwares, softwares e programação

Um computador é formado por componentes físicos, como a parte externa do seu celular, a CPU de um PC, a memória, o mouse, enfim... os meios pelos quais os sinais elétricos podem ser convertidos em dados, informações.

O software é o meio pelo qual a linguagem de máquina pode ser compilada ou interpretada, através de códigos criados em uma **linguagem intermediária**, para idiomas que conhecemos, como o português, o inglês e também para imagens, cores, números... enfim, uma série de dados que podem ser assimilados mais facilmente pelos seres humanos.

Mas, afinal, qual é a relação entre hardware, software e programação? A programação é exatamente quem possibilita a existência dos softwares e, por consequência, a utilização mais prática dos hardwares. Para poder dar origem aos softwares, a programação ganha uma linguagem própria que compõe códigos escritos por programadores.

# Linguagem de Programação

Um computador é formado por componentes físicos, como a parte externa do seu celular, a CPU de um PC, a memória, o mouse, ou seja, os meios pelos quais os sinais elétricos podem ser convertidos em dados, informações.

O software é o meio pelo qual a linguagem de máquina pode ser compilada ou interpretada, através de códigos criados em uma linguagem intermediária, para idiomas que conhecemos, como o português, o inglês e também para imagens, cores, números.

A programação é exatamente quem possibilita a existência dos softwares e, por consequência, a utilização mais prática dos hardwares. Para poder dar origem aos softwares, a programação ganha uma linguagem própria que compõe códigos escritos por programadores.

# Código Fonte

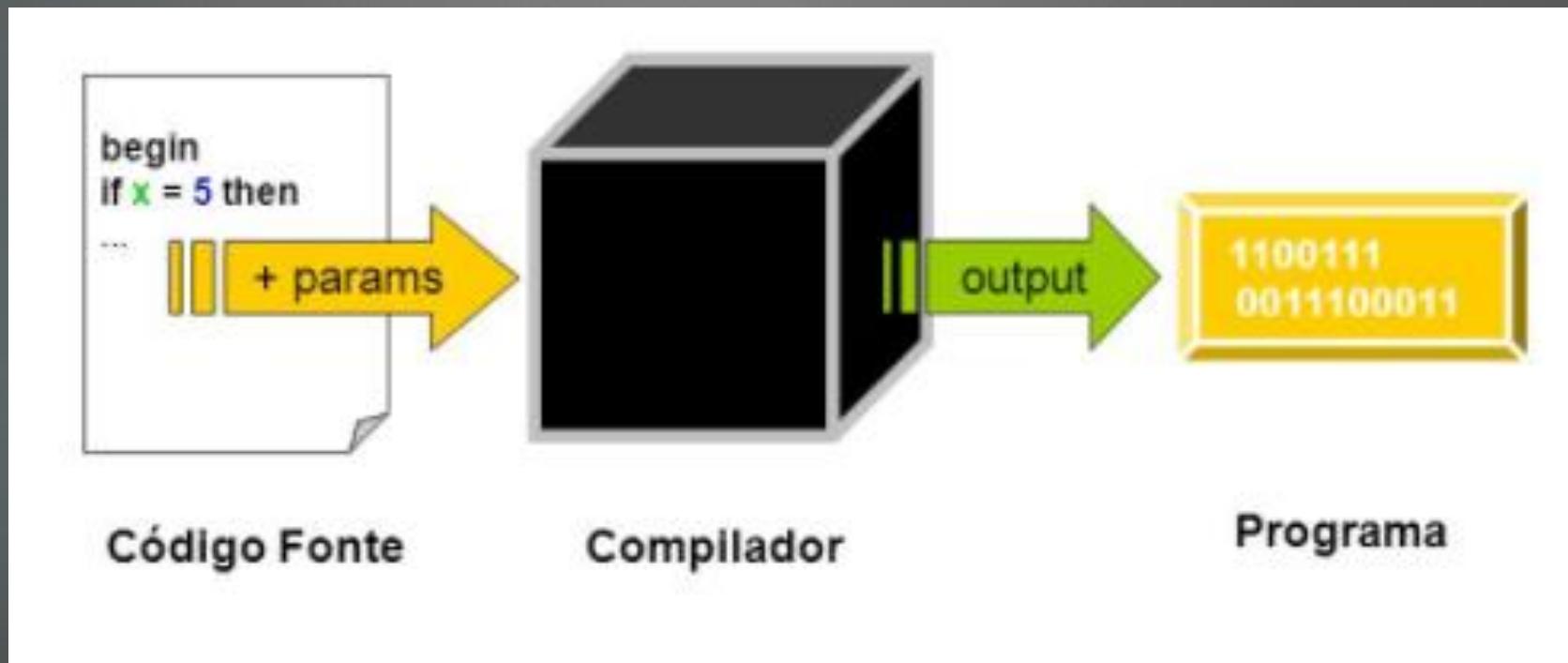
Código fonte é o conjunto de palavras e comandos escritos ordenadamente, de maneira lógica, que contém instruções em determinada linguagem de programação.

Há dois tipos de linguagem para escolher na hora de programar:

- As linguagens compiladas
- As linguagens interpretadas.

# Compilação

Compilar uma linguagem de programação nada mais é do que transformar o código fonte em algo que faça sentido para a máquina, ou seja, transformar o que o programador escreveu em linguagem de máquina.

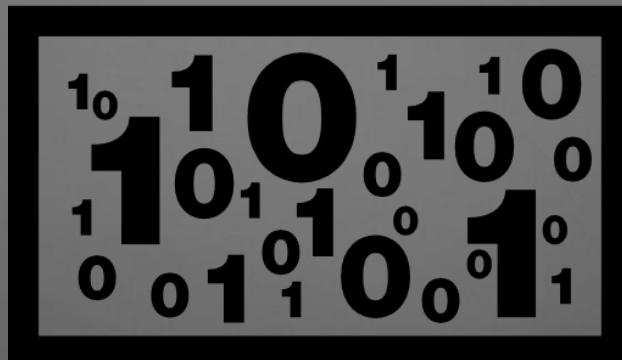


# Código binário

O sistema binário ou de base 2 é um sistema de numeração posicional em que todas as quantidades se representam com base em dois números, ou seja, zero e um (0 e 1). Os computadores digitais trabalham internamente com dois níveis de tensão, pelo que o seu sistema de numeração natural é o sistema binário.

0 – Não passa corrente elétrica.

1 – Passa corrente elétrica.



# Linguagens Baixo e Alto nível

Existem dois tipos de classificação de linguagens de programação, as de alto nível e baixo nível, a principal diferença entre elas é a seguinte:

- ▶ A linguagem de alto nível se aproxima mais com a linguagem humana.
- ▶ A linguagem de baixo nível está mais próxima da linguagem de máquina (Hardware).

# Evolução das linguagens

```
>Edit: A(dit C(pu D(lete F(ind I(nsrt J(mp R(place Q(uot X(chng Z(ap [1.1]
)
USES APPLESTUFF
PROGRAM SIREN
INTEGER CYCLES, PITCH

DO 30 CYCLES = 1, 15
    DO 10 PITCH = 30, 50
        CALL NOTE (PITCH, 1)
    CONTINUE
    DO 20 PITCH = 50, 30, -1
        CALL NOTE (PITCH, 1)
    CONTINUE
CONTINUE
* WRITE (*, 40)
*   CHAR(15), 'YOU'RE UNDER ARREST, JOE FORTRAN!', CHAR(14)
FORMAT (A, A, A)
END
```

```
namespace ConsoleApplication1.Model
{
    public class Usuario
    {
        public string login { get; set; }
        public string senha { get; set; }

        private Autenticacao autenticao;

        public Usuario(Autenticacao autenticao)
        {
            this.autenticao = autenticao;
        }

        public Boolean AutenticarUsuario(string login, string senha)
        {
            this.login = login;
            this.senha = senha;

            return autenticao.Autenticar(this.login, this.senha);
        }
    }
}
```

# Evolução das linguagens

10

Cadastro/Cientes :: Ordem de Serviço :: Relatórios :: Uilitários/sair

Ordem de serviço Nº 11/1 Data: 27/11/03

Cliente: WILSON ROBERTO DONATO Cód./Cliente: 001  
Endereço:  
Cidade: QWQ Bairro:  
Cep:  
Fone:

Marca Do Aparelho: Panasonic Série: C23E567-A00  
Modelo do Aparelho: P-300 Cor: Preto

Defeito: Nao tem alimentação (nao ascende o Led)

Solução: [redacted]

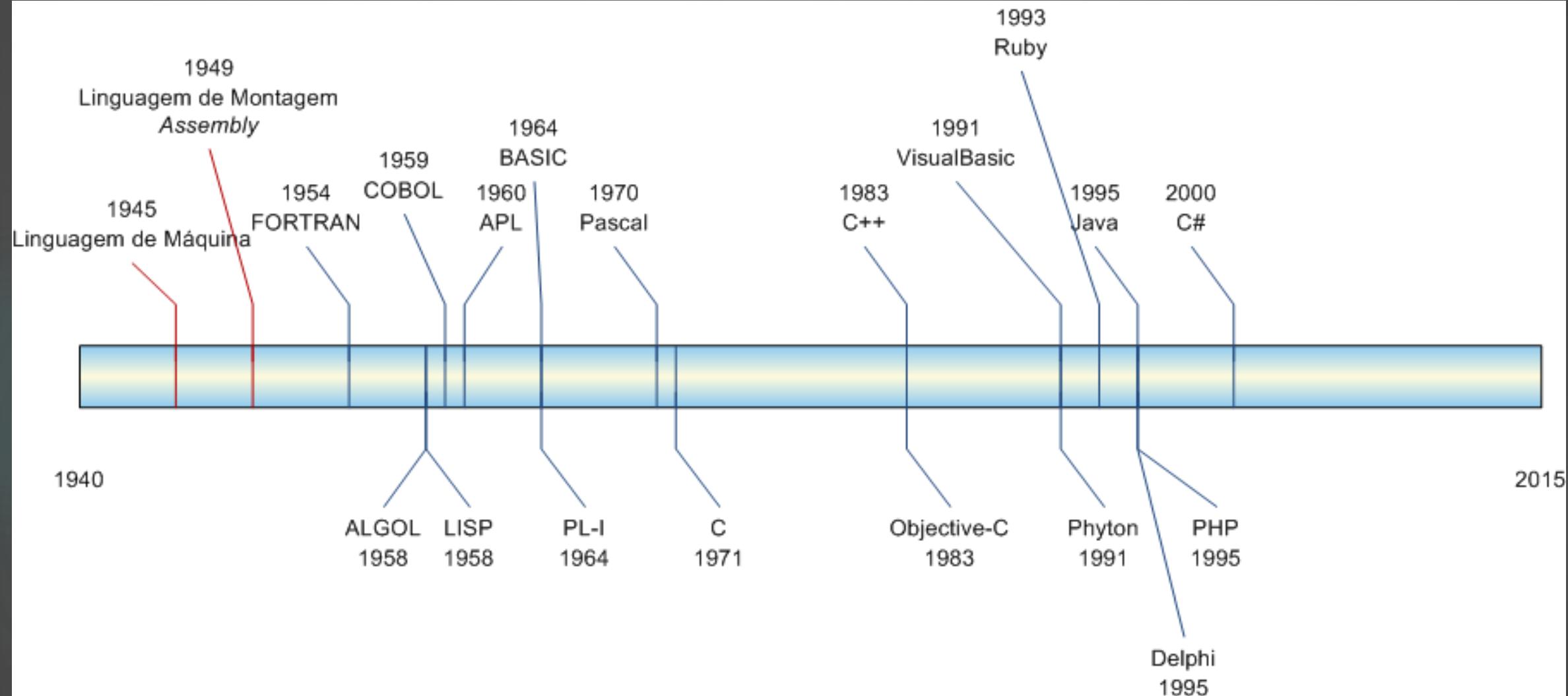
Componente subst: [redacted]

Valor R\$ 0.00 Técnico Responsável: [redacted] Data: / /

Quinta Cria uma ordem de Serviço 09:36



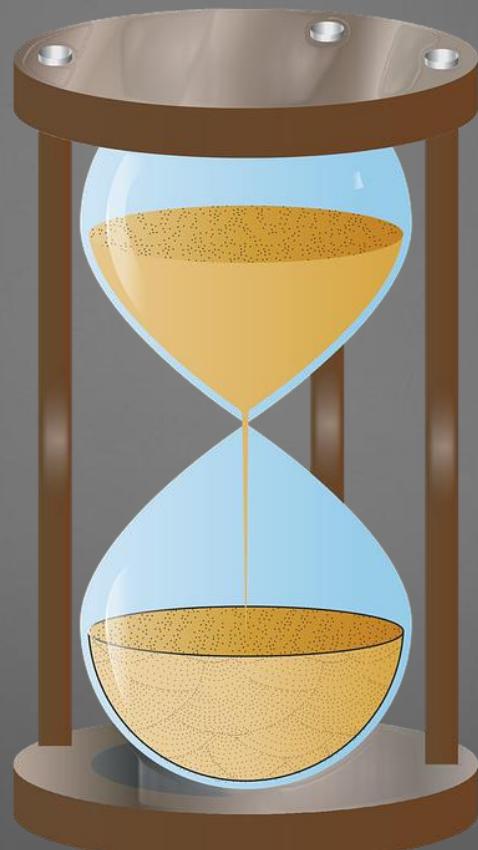
# Linha do tempo



# Linha do tempo

12

<https://www.youtube.com/watch?v=fcRR8eUZ8vo>



# Plataformas de desenvolvimento

13



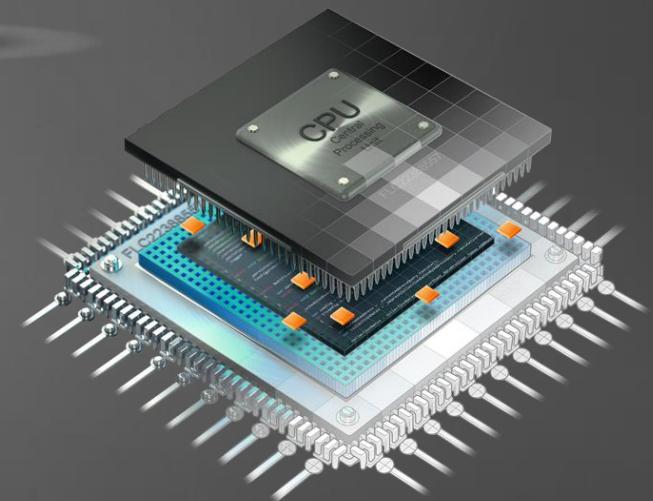
Desktop



Web



Mobile



Embedded

# Plataforma Desktop



- ▶ O que é?
- ▶ Aplicações
- ▶ Vantagens
- ▶ Desvantagens

# Principais linguagens

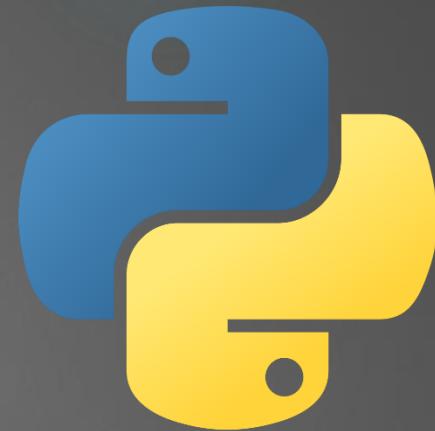
15



Ruby



python™



# Plataforma Web



- ▶ O que é?
- ▶ Aplicações
- ▶ Vantagens
- ▶ Desvantagens

# Principais linguagens

17



# Plataforma Mobile



- ▶ O que é?
- ▶ Aplicações
- ▶ Vantagens
- ▶ Desvantagens

# Principais linguagens

19



Java™

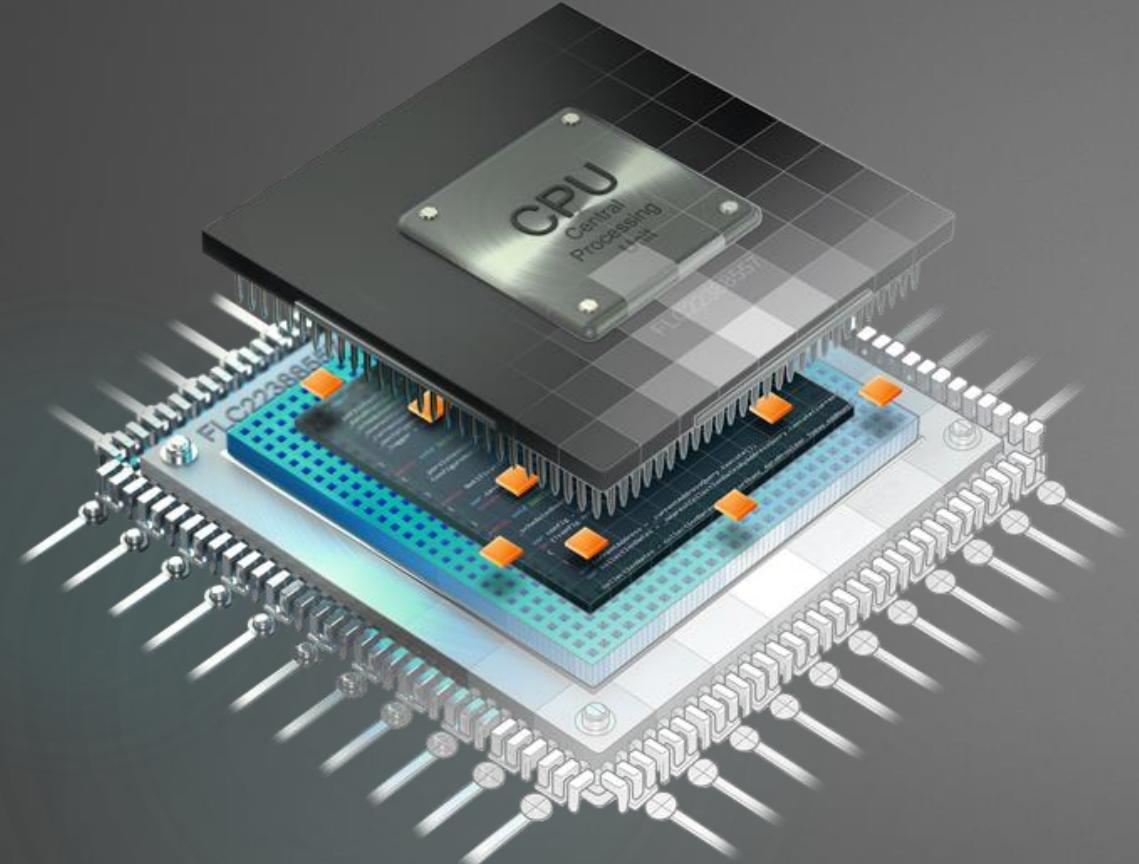


DART



Swift  
Kotlin

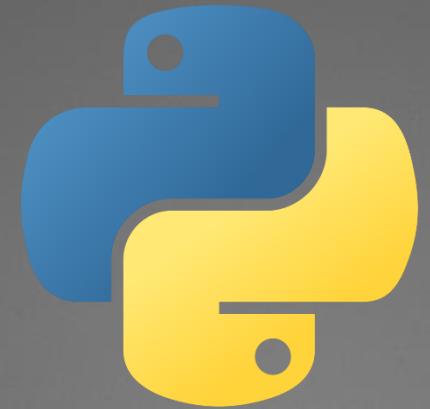
# Plataforma Embeded



- ▶ O que é?
- ▶ Aplicações
- ▶ Vantagens
- ▶ Desvantagens

# Principais linguagens

21

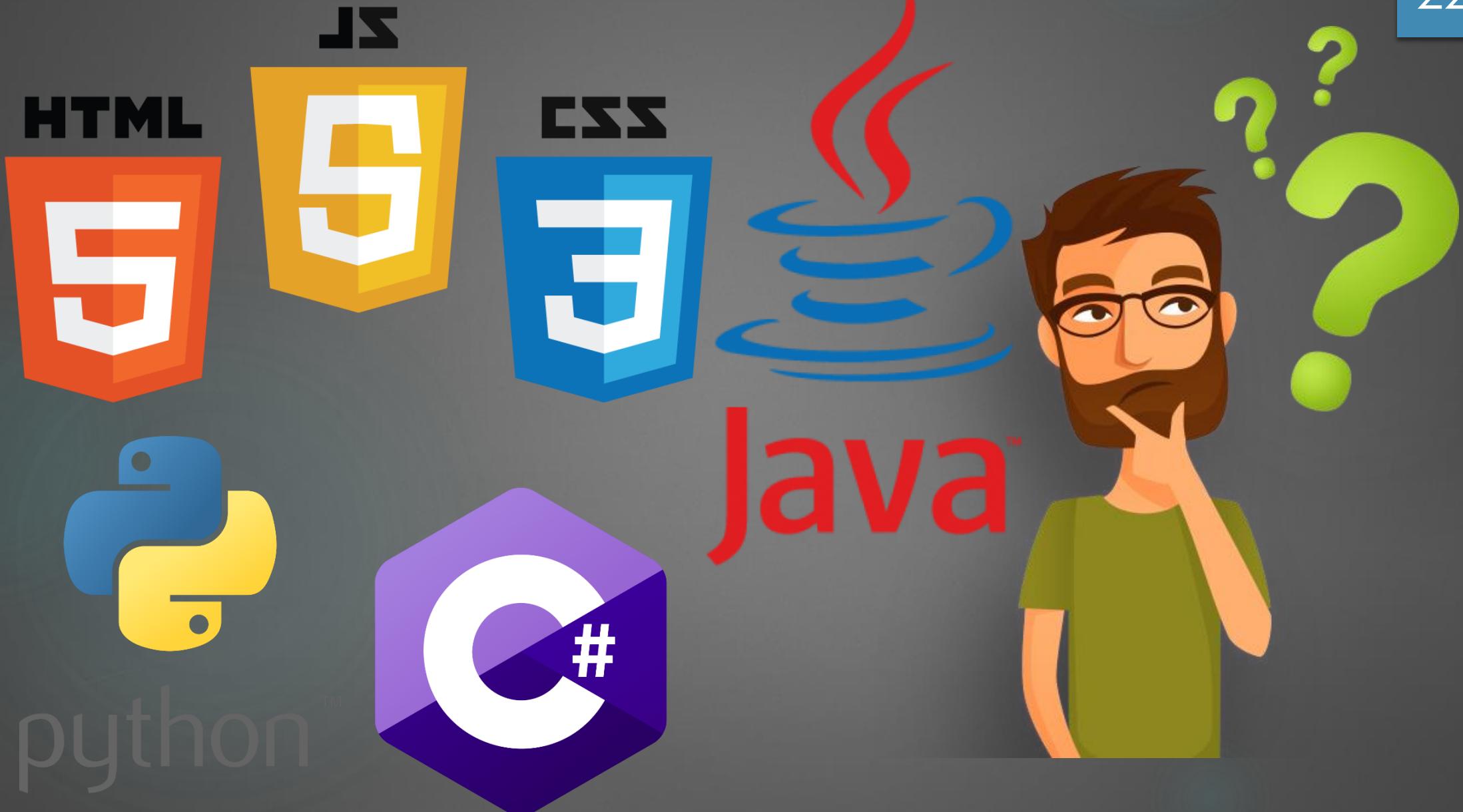


python™



# Qual linguagem aprender?

22



# Aprenda a programar!!



As linguagens mudam com o tempo,  
algumas surgem, outras desaparecem.



Saber desenvolver em apenas uma  
linguagem te limita profissionalmente.



Sua evolução profissional fica limitada a  
evolução da linguagem.



# Algoritmo

É formalmente uma sequência finita de passos que levam a execução de uma tarefa. Podemos pensar em algoritmo como uma receita, uma sequência de instruções para um fim específico. Para escrever um algoritmo precisamos descrever a sequência de instruções, de maneira simples e objetiva.

Todo algoritmo tem por padrão apresentar algumas características básicas:

- Partir de um ponto inicial e chegar a um ponto final;
- Não ser ambíguo (ter dupla interpretação);
- Poder receber dados externos e ser capaz de retornar resultados aos mesmos;
- Ter todas suas etapas alcançáveis em algum momento do programa.

**Seqüência Lógica** são passos executados até atingir um objetivo ou solução de um problema.

# Algoritmo

25

Ao montar um algoritmo, precisamos primeiro dividir o problema apresentado, em três fases fundamentais:

- **ENTRADA:** São os dados de entrada do algoritmo
- **PROCESSAMENTO:** São os procedimentos utilizados para chegar ao resultado final
- **SAÍDA:** São os dados já processados



Imaginamos agora como poderíamos descrever um algoritmo que fosse capaz de calcular a média dos alunos da turma no final do ano.

**Passo 1)** Quais são as entradas

As entradas neste caso consistirão de todas as notas finais dos alunos da turma, bem como do número de alunos da turma.

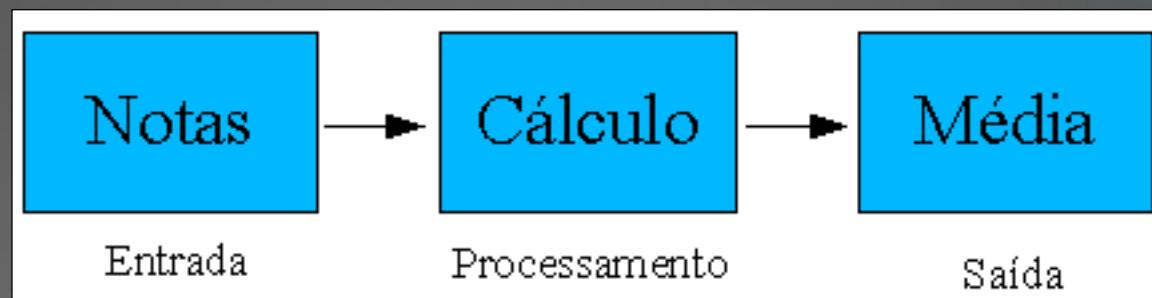
**Passo 2)** Como será calculada a média?

Neste caso iremos calcular a média dos alunos da turma da seguinte forma:

$$\text{Média} = \frac{n_1 + n_2 + \dots + n_n}{\text{número de notas dos alunos}}$$

**Passo 3)** Quais serão os dados de saída?

Neste caso o dado de saída será o valor da média. Dados de saída significam para nós que o algoritmo irá produzir durante o passo 2 um valor que depende da entrada fornecida no passo 1.



- ▶ Escreva então o algoritmo para calcular a média das notas de 4 alunos:
  - ▶ Receber a nota do primeiro aluno
  - ▶ Receber a nota do segundo aluno
  - ▶ Receber a nota do terceiro aluno
  - ▶ Receber a nota do quarto aluno
  - ▶ Somar todas as notas recebidas
  - ▶ Dividir o valor somado das notas por 4
  - ▶ Mostrar o resultado da divisão

# Algoritmo

1) Para "x" valendo 5

```
y ← x + 2  
z ← x + y  
x ← z / 2  
y ← x * 2  
z ← x * y
```

Quais os valores  
de x, y e z ?

Resposta = 6, 12 e 72

2) Sabendo que  
 $a = 1, b = 7, c = 2, d = 1,$

Qual o valor da expressão?

$a + b * c / d - a$

Resposta = 14

3) Para "x" valendo 10

```
y ← x + 1  
x ← y + 1  
y ← x + 1
```

se  $Y > X$   
 $x \leftarrow y * 2$   
 $y \leftarrow y - 5$   
senao  
 $x \leftarrow 0$   
 $y \leftarrow 1$

Quais os valores  
de x e y?

Resposta = 26 e 8



# Qual linguagem iremos aprender?

30





# Microsoft Visual Studio

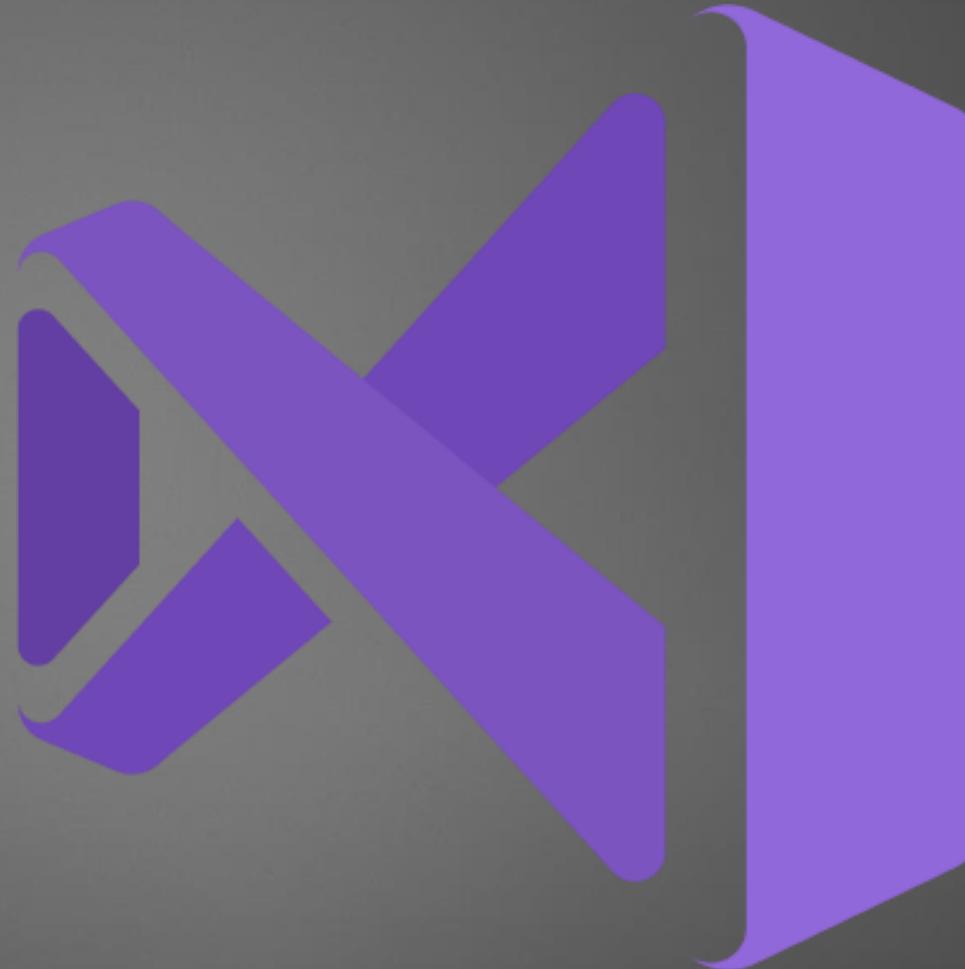
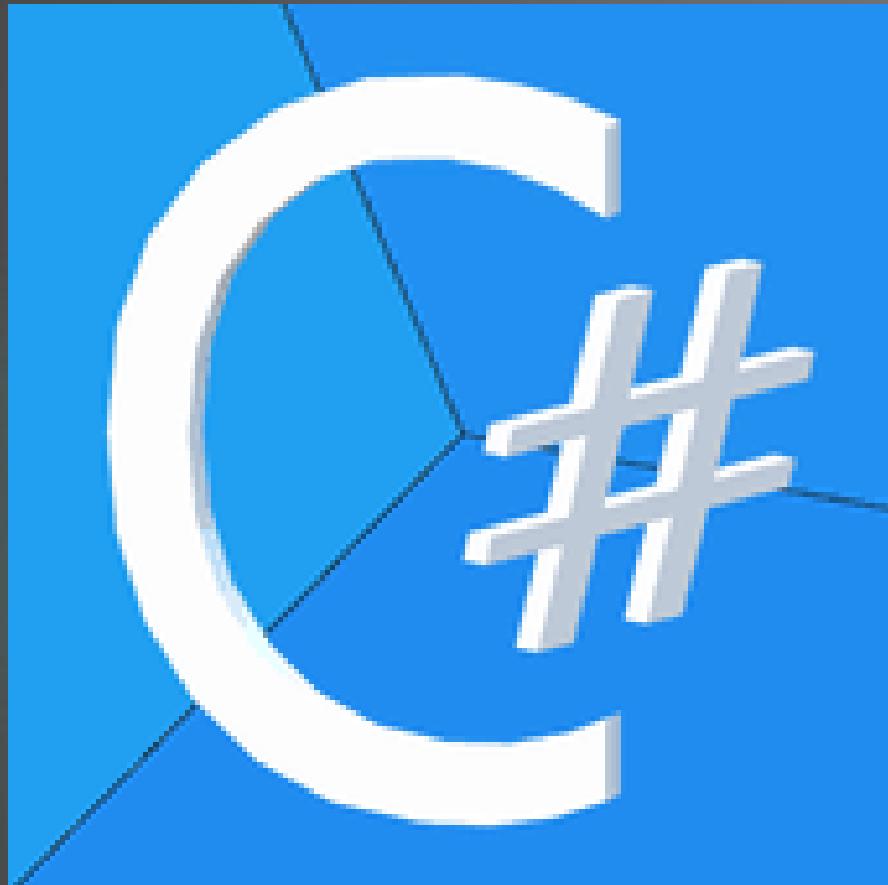
O Microsoft Visual Studio é um pacote de programas da Microsoft para desenvolvimento de software especialmente dedicado ao .NET Framework e às linguagens Visual Basic (VB), C, C++, C# (C Sharp) e J# (J Sharp). Também é um grande produto de desenvolvimento na área web, usando a plataforma do ASP.NET. As linguagens com maior frequência nessa plataforma são: VB.NET (Visual Basic.Net) e o C# (C Sharp).

# Linguagem C#

C# (C Sharp) é uma linguagem de programação interpretada multi-paradigma fortemente tipada, e, possuindo paradigmas de programação imperativa, funcional, declarativa, orientada a objetos e genérica C# foi desenvolvido(a) pela Microsoft como parte da plataforma .NET. Altamente escalável a fim de permitir que uma mesma aplicação possa ser executada em diversos dispositivos de *hardware*, independentemente destes serem PCs, *handhelds* ou qualquer outro dispositivo móvel.

A sua sintaxe orientada a objetos foi baseada no C++ mas inclui muitas influências de outras linguagens de programação, como Object Pascal e Java. A linguagem C# é compilada para bytecode e é interpretada pela máquina virtual chamada CLR (Common Language Runtime).

# Ferramentas utilizadas

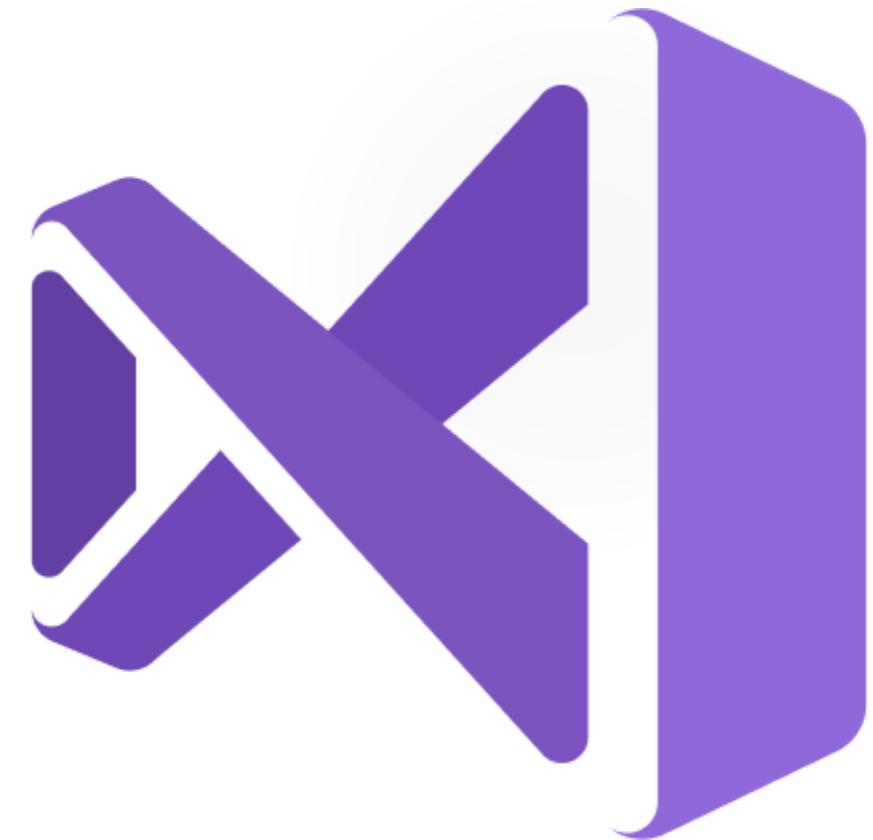


# Desenvolvimento Online

35

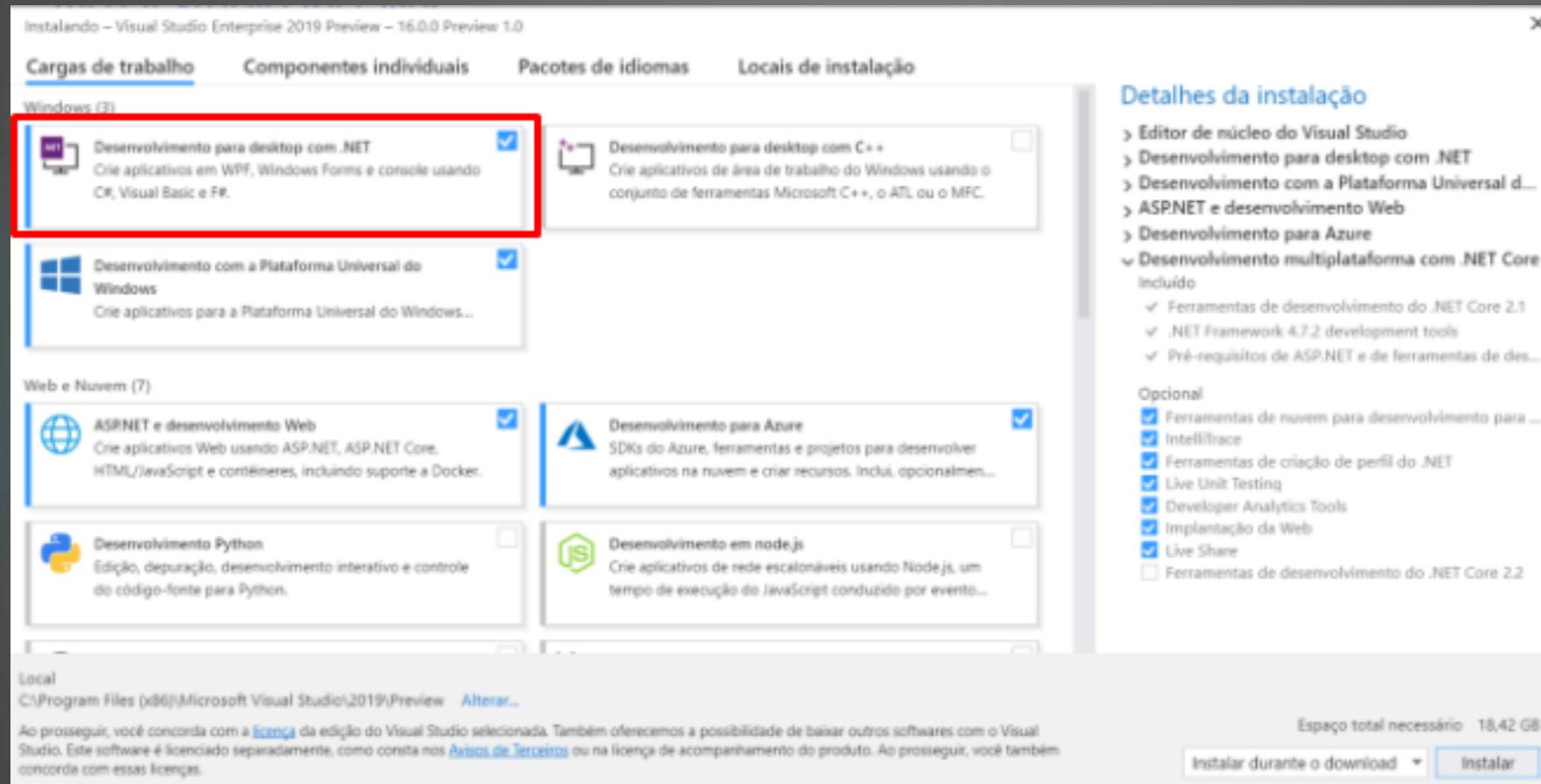
<https://repl.it/languages/csharp>

# Instalando e conhecendo o Visual Studio

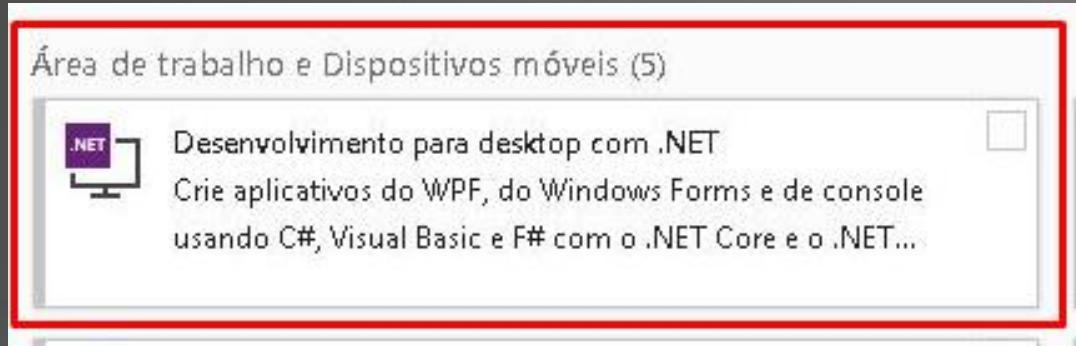


# Instalação do Visual Studio

<https://visualstudio.microsoft.com/pt-br/vs/community/>



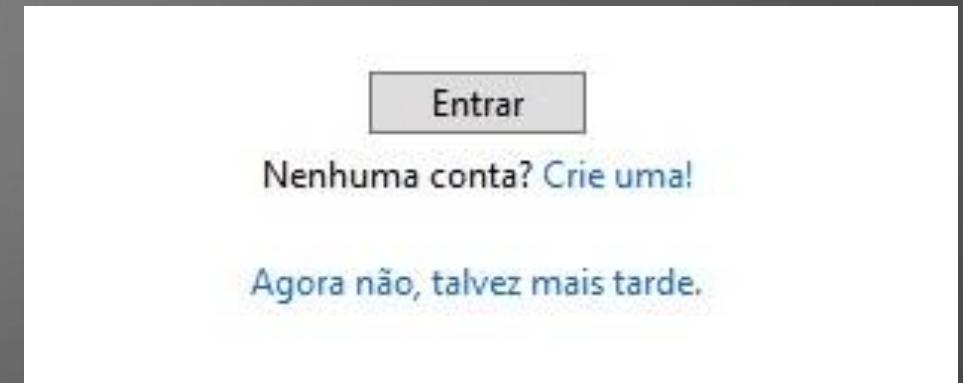
# Instalação do Visual Studio



Na instalação, selecione a opção para desenvolvimento Desktop

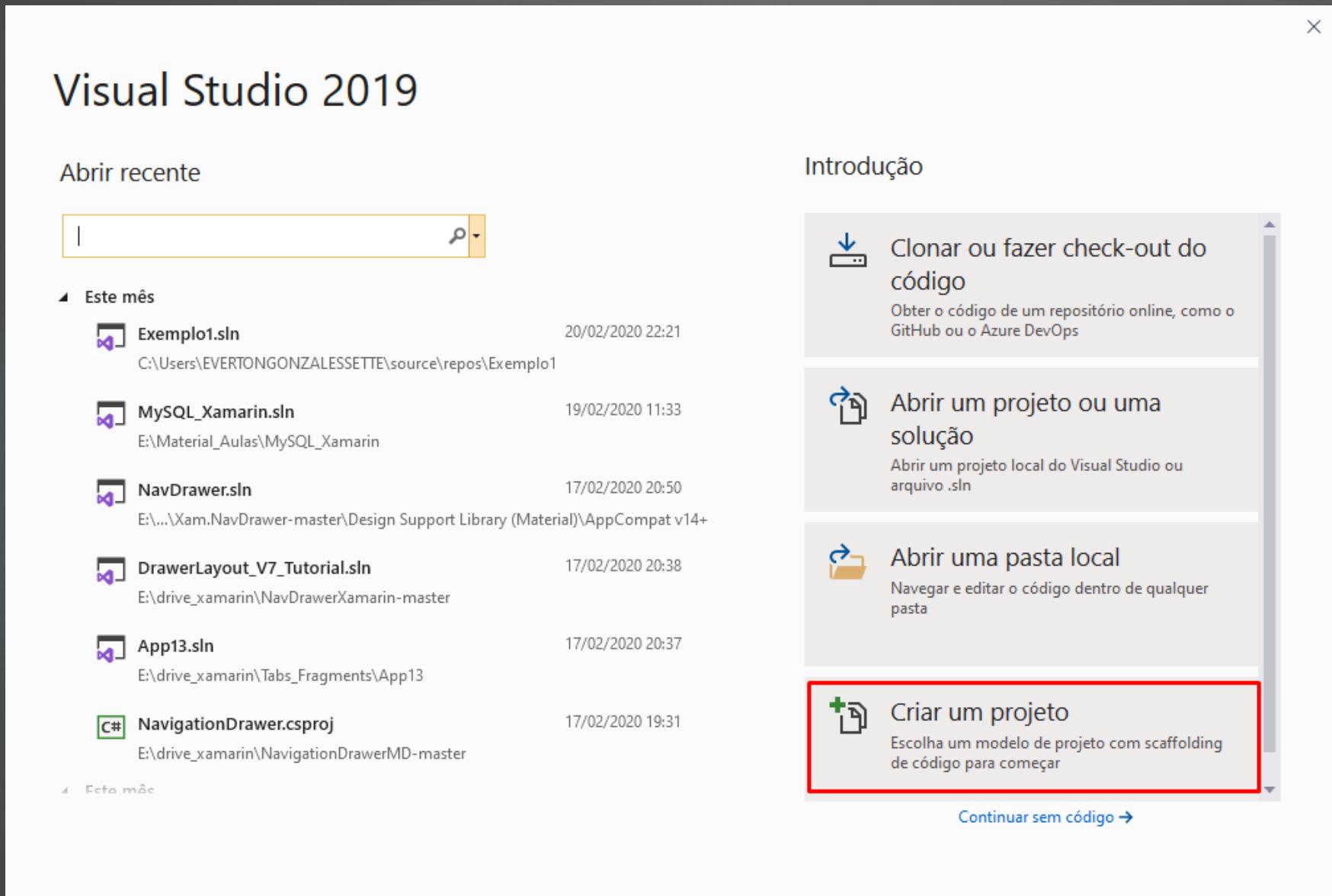


Quando finalizar a instalação logar com seu e-mail @etec e senha



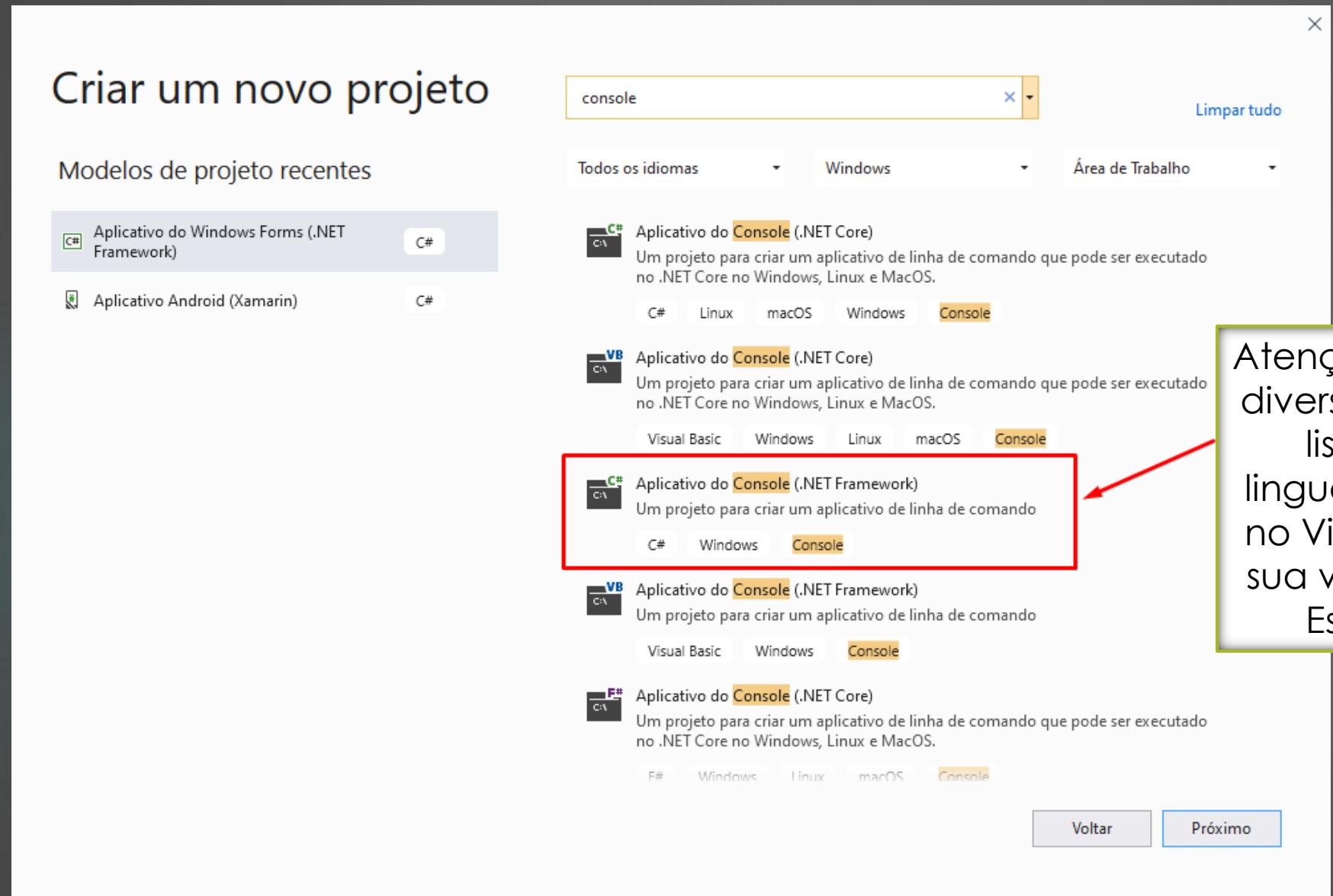
# Criando um novo projeto no VS

39



# Criando um novo projeto no VS

40



Atenção pois existem diversos Consoles na lista, todas as linguagens contidas no Visual Studio tem sua versão Console.  
Escolha o C#

# Criando um novo projeto no VS

41

## Configurar seu novo projeto

Aplicativo do Console (.NET Framework) C# Windows Console

Nome do projeto

ConsoleApp1

Nome do Projeto

Local

C:\Users\EVERTONGONZALESSETTE\source\repos

Seleciona o caminho  
do Projeto

Nome da solução

ConsoleApp1

Colocar a solução e o projeto no mesmo diretório

Cria automático  
um diretório para o  
projeto

Framework

.NET Framework 4.7.2

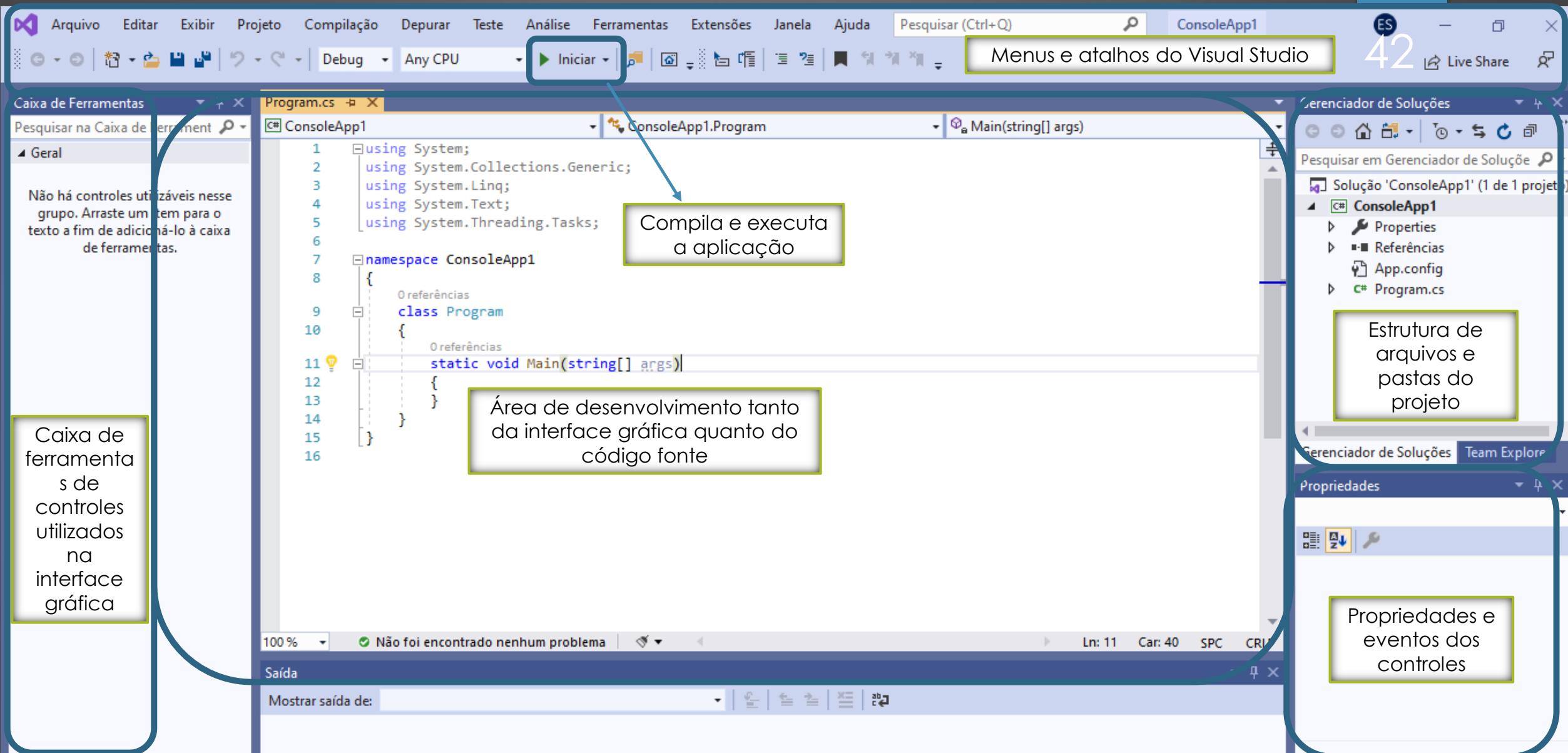
Permite a seleção da  
versão do .Net  
Framework

Clique em Criar para  
finalizar a criação

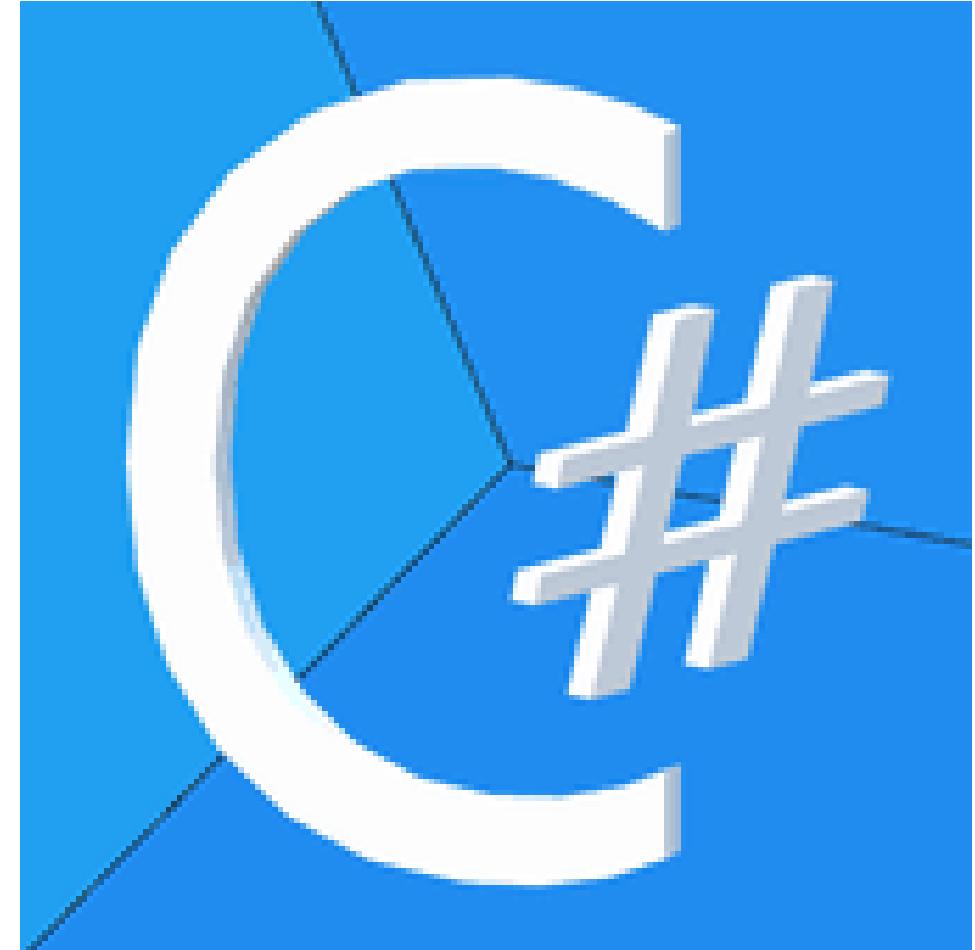
Voltar

Criar

# Interface da Ferramenta do VS



# Instalando e conhecendo o C# Shell



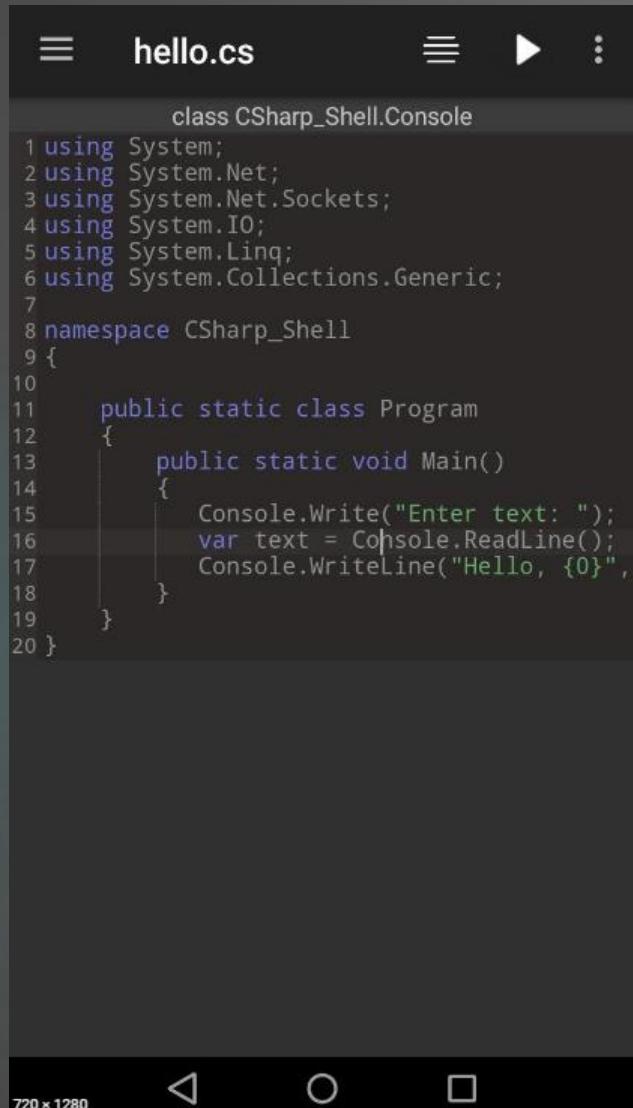
# Instalação do C# Shell

A screenshot of a mobile application store interface showing search results for "c# shell". The results are as follows:

- C# Shell (C# Offline Compiler)** by RsD • Ferramentas. Status: Instalado. This item is highlighted with a red border.
- C# Shell Packages Plugin** by NuGet. Rating: 4,6 ★. Size: 11 MB. Downloads: 1 mil+.
- C# Programming Compiler** by Kappsmart • Educação. Rating: 3,4 ★. Size: 6,7 MB. Downloads: 10 mil+.
- VB.NET Shell (Visual Basic Offline Co...)** by RsD • Ferramentas. Rating: 4,6 ★. Size: 73 MB. Downloads: 10 mil+.
- Learn C# Programming** by Techno Apps Development • Educação. Rating: 4,2 ★. Size: 3,6 MB. Downloads: 10 mil+.

# Ferramenta C# Shell

45



The screenshot shows the C# Shell application. At the top, there's a toolbar with icons for file operations. Below it is a code editor window titled "hello.cs" containing the following C# code:

```
class CSharp_Shell.Console
{
    using System;
    using System.Net;
    using System.Net.Sockets;
    using System.IO;
    using System.Linq;
    using System.Collections.Generic;
}

namespace CSharp_Shell
{
    public static class Program
    {
        public static void Main()
        {
            Console.WriteLine("Enter text: ");
            var text = Console.ReadLine();
            Console.WriteLine("Hello, {0}", text);
        }
    }
}
```

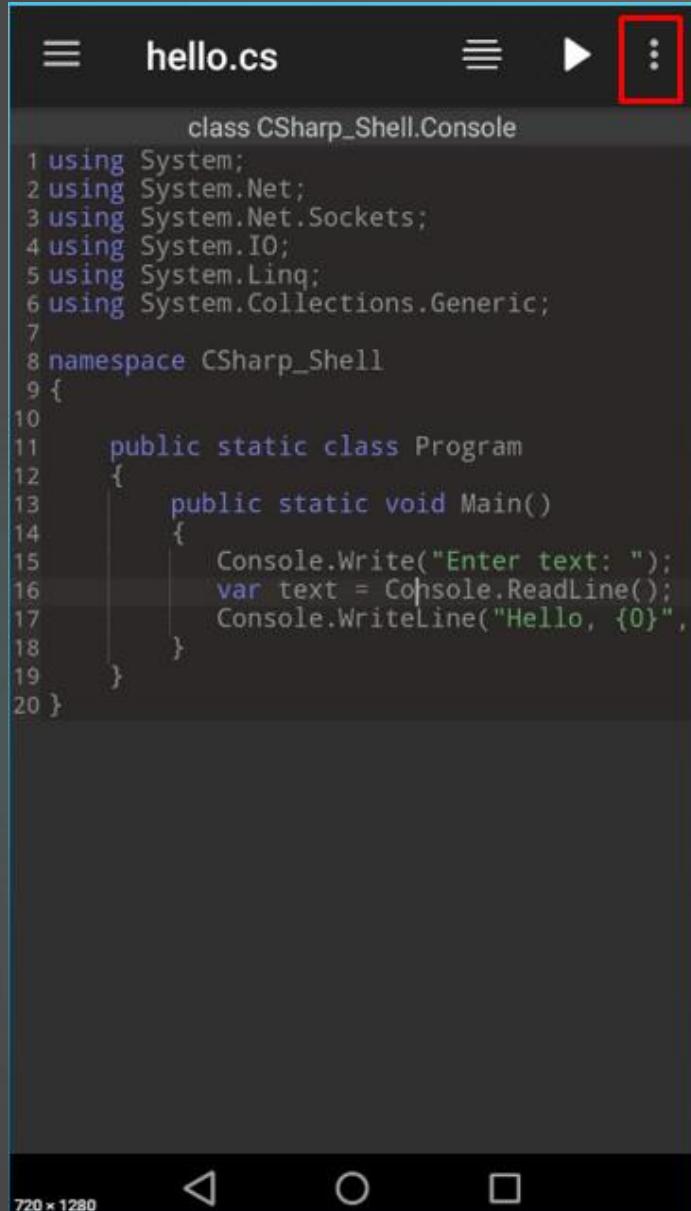
At the bottom of the application window is a terminal-like interface with three icons: a left arrow, a circle, and a square.



Essa é a tela inicial  
do C# Shell

# Criando um novo projeto

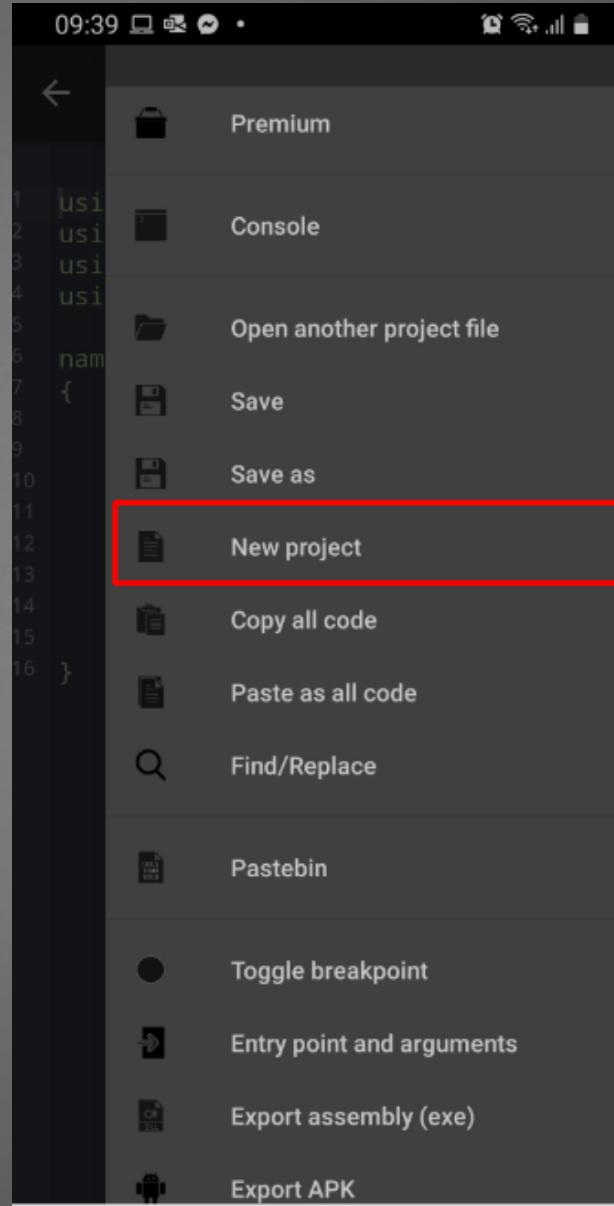
46



```
hello.cs
```

```
class CSharp_Shell.Console
1 using System;
2 using System.Net;
3 using System.Net.Sockets;
4 using System.IO;
5 using System.Linq;
6 using System.Collections.Generic;
7
8 namespace CSharp_Shell
9 {
10
11     public static class Program
12     {
13         public static void Main()
14         {
15             Console.WriteLine("Enter text: ");
16             var text = Console.ReadLine();
17             Console.WriteLine("Hello, {0}",
18         }
19     }
20 }
```

→  
Clique aqui  
para acessar  
as opções



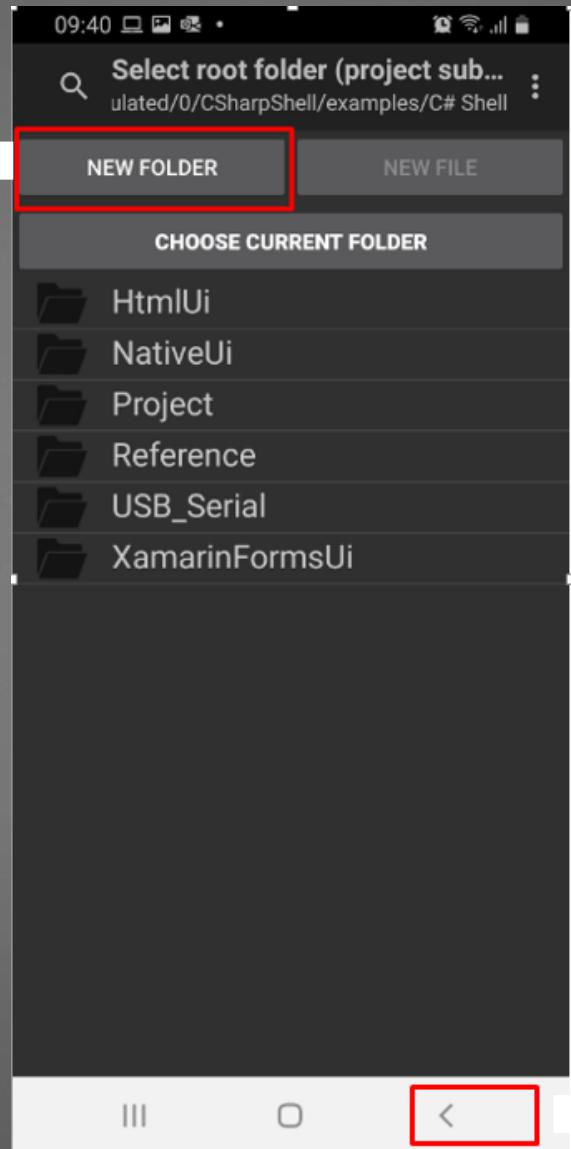
→  
Selecione a  
opção New  
project

# Criando um novo projeto

47

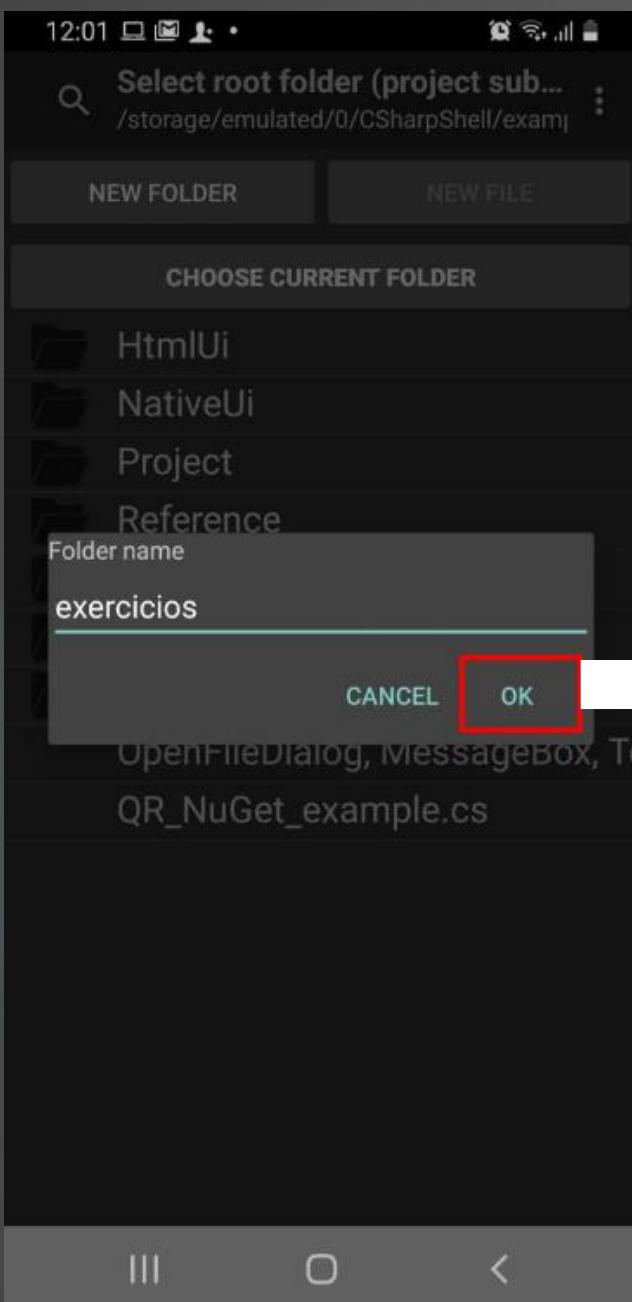
Crie uma pasta  
para seus projetos

2



Navegue na árvore  
de diretórios através  
desse botão

1



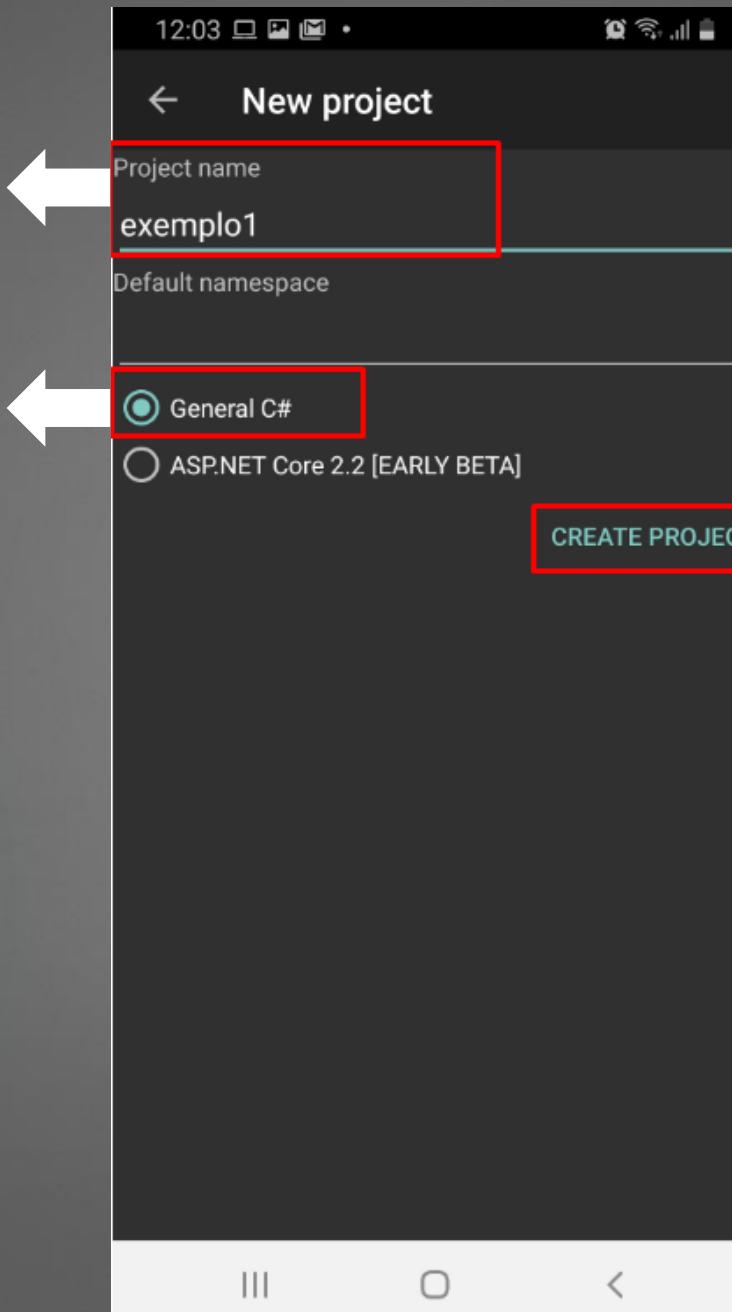
Digite o nome da  
pasta e clique em  
OK



Selecione a  
nova pasta  
para o projeto

1

Digite o nome do  
Projeto

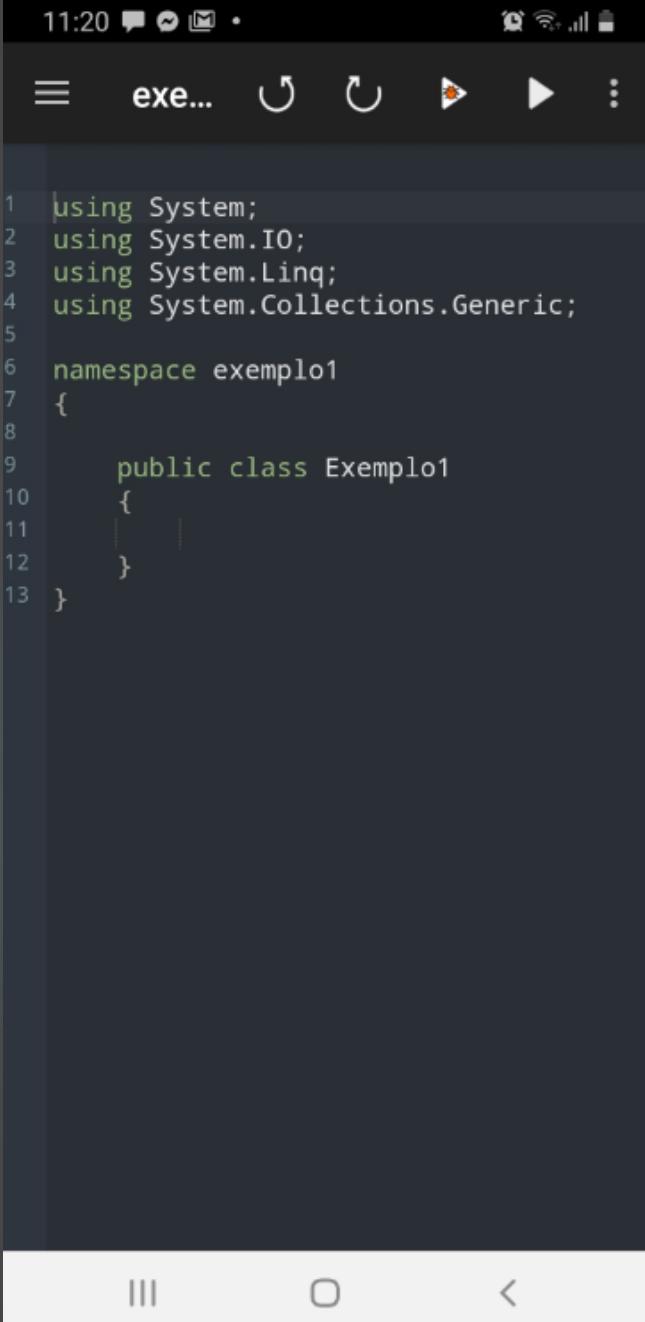


2

Selecionar a opção  
General C#

3

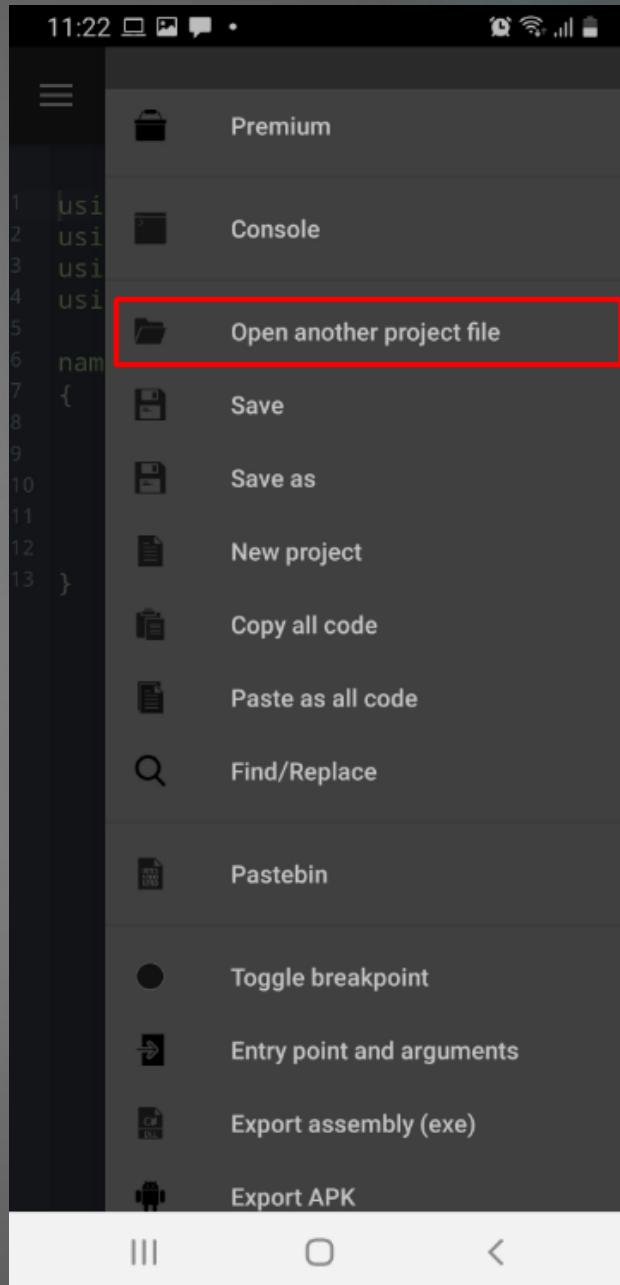
Clique em Create  
Project



```
1 using System;
2 using System.IO;
3 using System.Linq;
4 using System.Collections.Generic;
5
6 namespace exemplo1
7 {
8
9     public class Exemplo1
10    {
11    }
12 }
13 }
```



Tela do novo projeto. Podemos perceber que não foi criado o método Main, então vamos ter que fazer algumas alterações. Método Main é o que inicializa nossa aplicação.



Vamos abrir o projeto através dessa opção

Navegue até  
encontrar a pasta do Projeto, acesse a mesma



11:23



Open file

Shell/examples/C# Shell/teste/exemplo1

NEW FOLDER

NEW FILE

SELECT FILE

exemplo1.cs

exemplo1.cshpro

exemplo1.cs

DELETE

RENAME

COPY TO CLIPBOARD

1

Clique nos arquivos que aparecerão até aparecer essa mensagem. Clique em Delete

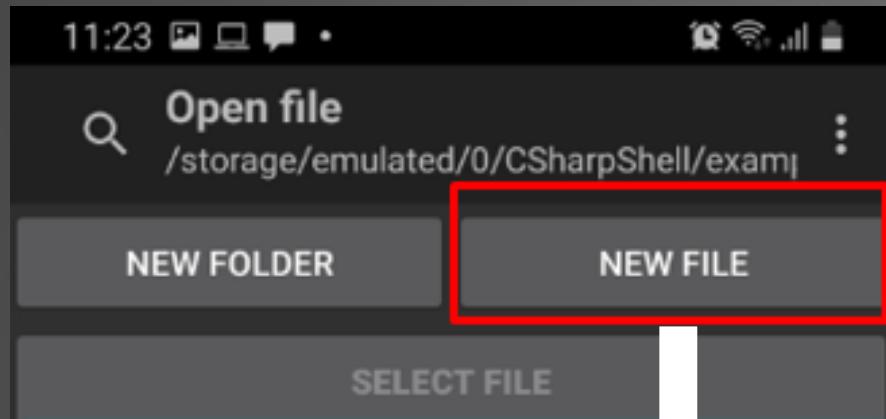
Delete 'exemplo1.cs'

NO

YES

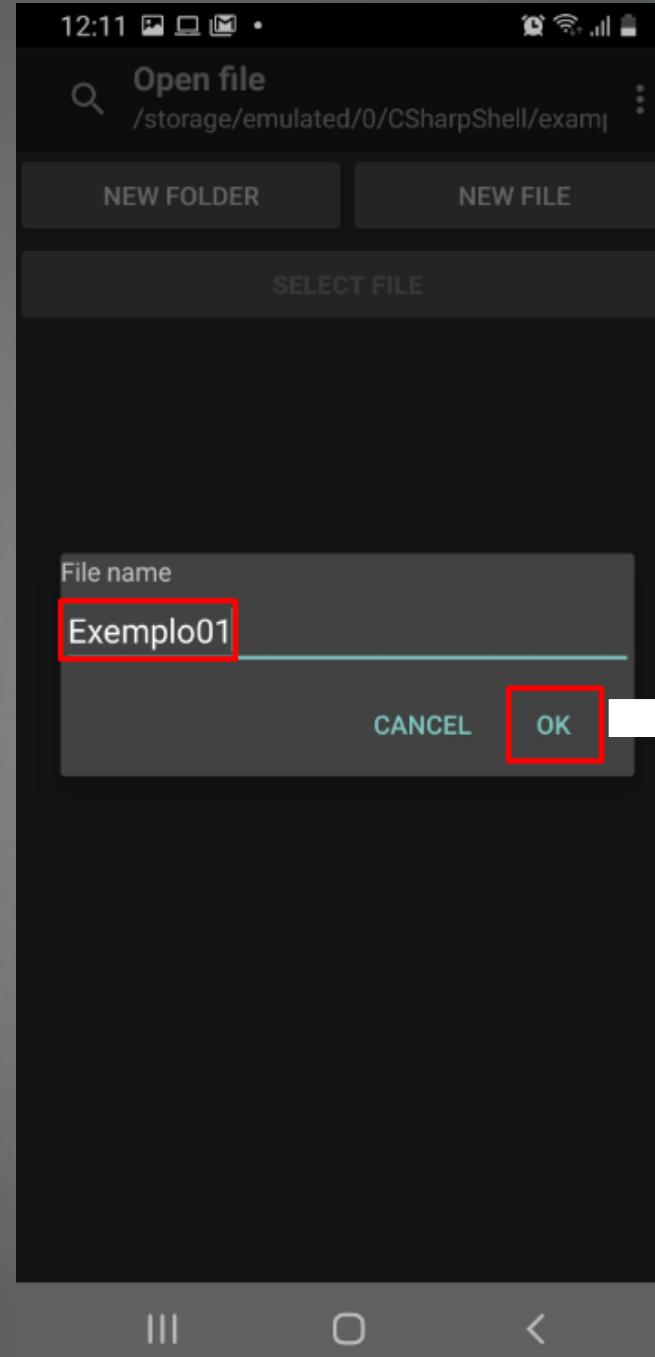
2

Clique em Yes para confirmar a exclusão



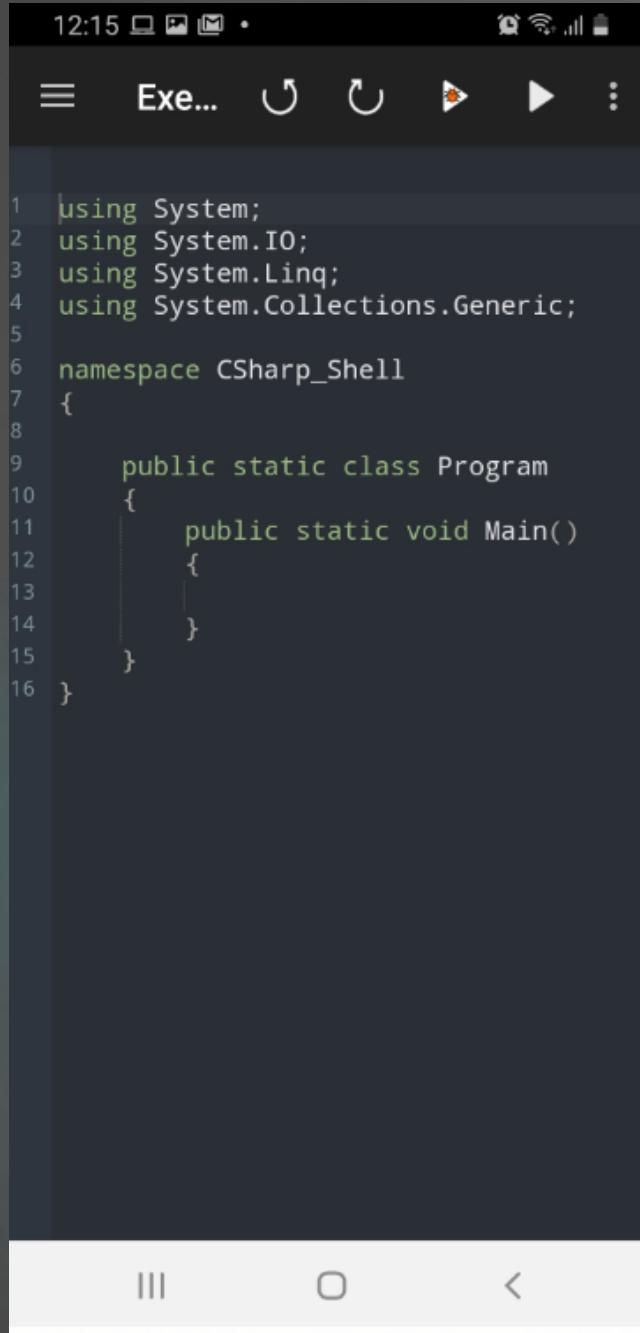
1

Após excluir os arquivos clique em New File



2

Digite o nome e clique em OK



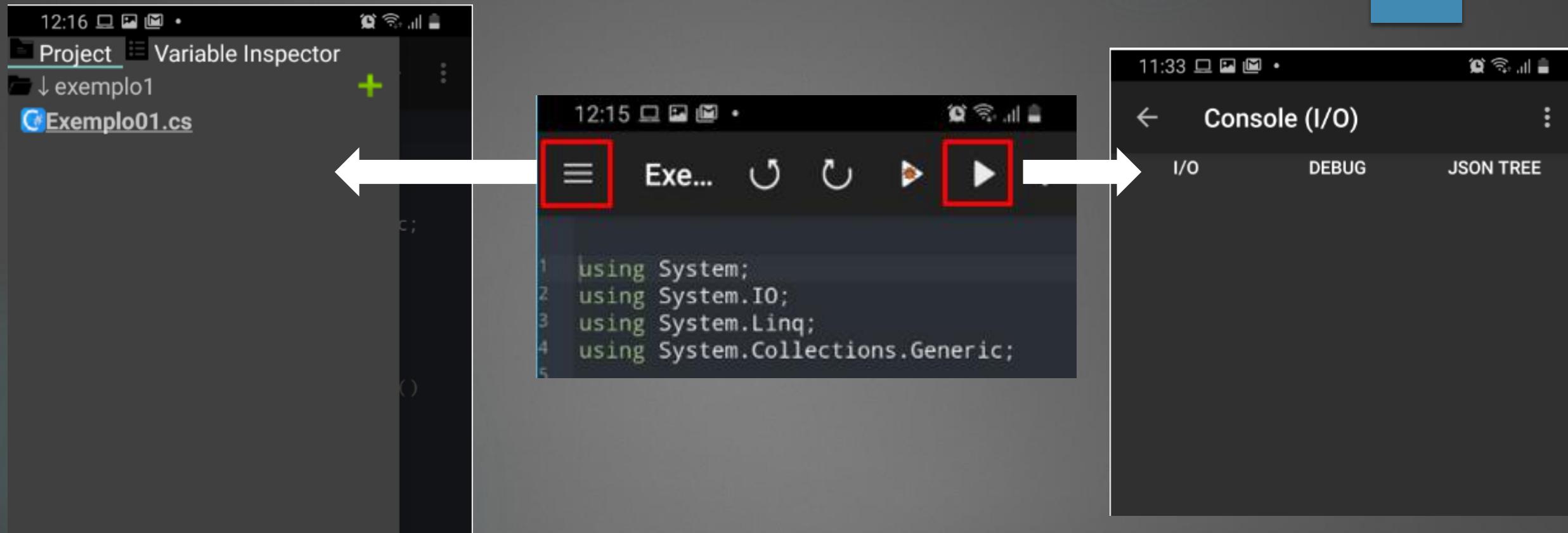
A screenshot of a mobile application interface, likely a code editor or terminal window. At the top, there is a dark header bar with icons for time (12:15), signal strength, and battery level. Below the header is a toolbar with several icons: a menu icon (three horizontal lines), an 'Exe...' button, a circular arrow, a play/pause button, a forward button, and a more options button (three dots). The main area is a code editor with a dark background and light-colored text. The code is as follows:

```
1 using System;
2 using System.IO;
3 using System.Linq;
4 using System.Collections.Generic;
5
6 namespace CSharp_Shell
7 {
8
9     public static class Program
10    {
11         public static void Main()
12        {
13        }
14    }
15 }
16 }
```

The bottom of the screen features a white navigation bar with three icons: three vertical lines, a square, and a left arrow.



Aparecerá a tela  
com o novo código e  
o método Main



Navegador dos  
arquivos do projeto

1

Compila e executa o  
código fonte

2

# Estrutura do código Console

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace ConsoleApp1
8  {
9      class Program
10     {
11         static void Main(string[] args)
12         {
13         }
14     }
15 }
```

Área de importações de bibliotecas

Nome do Projeto

O referências

Nome da Classe

O referências

static void Main(string[] args)

Método de inicialização do sistema

# Onde devo programar?

57

```
namespace ConsoleApp5
{
    O referências
    class Program
    {
        O referências
        static void Main(string[] args)
        {
            // Código a ser executado
        }
    }
}
```

Neste início toda nossa programação  
estará contida dentro do método Main, ou  
seja, devemos desenvolver nosso código  
dentro das {} desse método



# Métodos e Propriedades

58

 **Métodos ou funções** são ações que o nosso programa (no caso Console) irá executar a partir de um determinado comando. O comando sempre vem com abertura e fechamento de ( ) que dentro deles podem ou não conter informações. Que em métodos chamamos de parâmetros.

Exemplo: Exibir uma mensagem, limpar uma tela.

 **Propriedades** são comandos que alteram o estado ou características do nosso programa (no caso Console). Sempre quando queremos alterar alguma propriedade devemos colocar a propriedade o símbolo de atribuição (=) e o novo valor que essa propriedade irá receber.

Exemplo: Alterar a cor do Console, alterar a cor do texto, título do Console.

# Classe Console

Representa os fluxos de entrada, saída e erro padrão para aplicativos de console.

Principais métodos:

	<code>Write(String)</code>	Grava o valor de seqüência de caracteres especificada no fluxo de saída padrão.
	<code>WriteLine(String)</code>	Grava o valor de seqüência de caracteres especificada, seguido de terminador de linha atual, ao fluxo de saída padrão.
	<code>ReadLine</code>	Lê a próxima linha de caracteres do fluxo de entrada padrão.
	<code>.ReadKey()</code>	Obtém a próxima chave de caractere ou função pressionada pelo usuário. A tecla pressionada é exibida na janela do console.

Material de apoio: [https://msdn.microsoft.com/pt-br/library/system.console\(v=vs.100\).aspx](https://msdn.microsoft.com/pt-br/library/system.console(v=vs.100).aspx)

# Método Write

Existem basicamente dois métodos principais em Console que são: Escreva (Write) e Leia (Read).

O método Write é utilizado quando se deseja mostrar informações na tela do computador, ou seja, é um comando de saída de dados, ou quando queremos interagir com o usuário. Para simplificar, usa-se o método Write, quando se necessita mostrar algum dado para o usuário do algoritmo (e posteriormente do programa).

# Método ReadKey

Sua principal função é a captura de uma tecla aplicada pelo usuário e posteriormente verificar qual foi e aplicar algum tipo de teste.

Em Console utilizamos também no final do código para que possamos ver o resultado da nossa aplicação, já que aplicativos Console são utilizados profissionalmente para executar determinados processamentos e serem fechados automaticamente.

# Método Write - ReadKey

Veja o exemplo abaixo:

```
using System;

namespace Exemplo_01
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Olá Mundo");

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

Todas as linhas (com exceção de blocos de decisão e laço) terminam com ";"

Comando utilizado para que o usuário tecle "Enter" para encerrar a execução da aplicação. Caso contrário a aplicação finaliza sozinha.



Ao executar temos essa tela

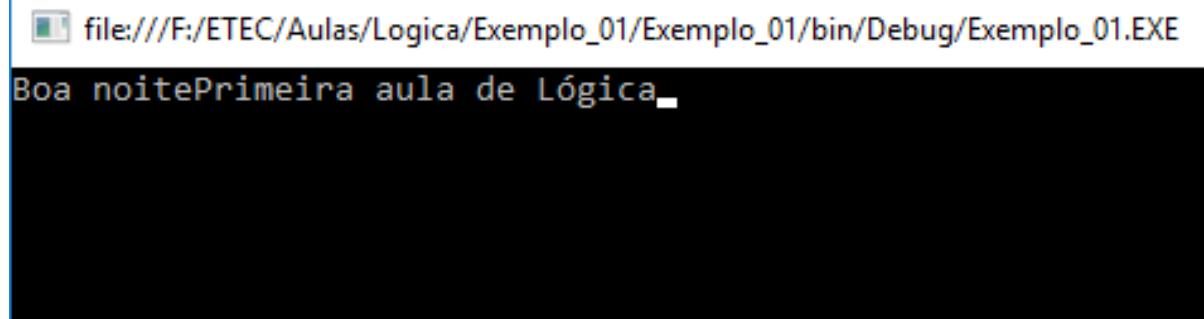
# Método Write X WriteLine

63

```
static void Main(string[] args)
{
    Console.Write("Boa noite");
    Console.Write("Primeira aula de Lógica");

    Console.ReadKey();
}
```

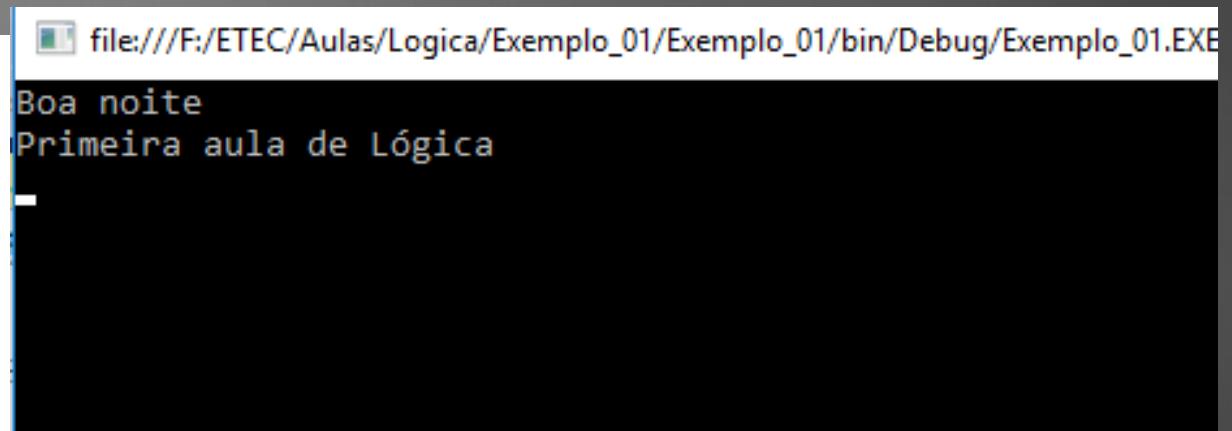
O método Write não pula linha, ou seja, é utilizado quando desejamos que a entrada do usuário fique na mesma linha da mensagem



```
static void Main(string[] args)
{
    Console.WriteLine("Boa noite");
    Console.WriteLine("Primeira aula de Lógica");

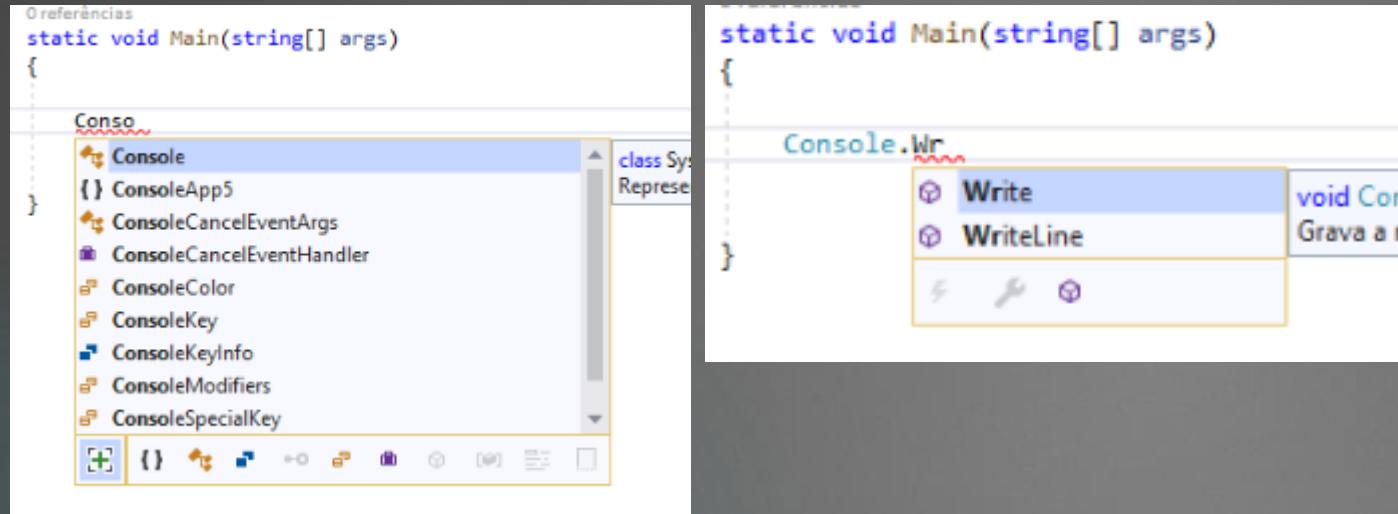
    Console.ReadKey();
}
```

O método WriteLine além de exibir a mensagem no Console também pula para a próxima linha.



# Dicas da ferramenta na programação

64



Sempre quando digitamos um comando aparecerá uma janela de opções. Através das setas do teclado podemos posicionar no comando que queremos e teclar Tab ou Enter que a ferramenta auto completará o código que selecionamos.

Ou simplesmente dar um duplo clique com o Mouse no comando que queremos a ferramenta completa.



# Tipos de dados em um WriteLine

65

## String

```
// Exibe uma String na tela  
Console.WriteLine("Primeira aula de Lógica");
```

Todo conteúdo String deverá estar entre “ ” (aspas duplas)

## Número Inteiro

```
// Exibe um número inteiro na tela  
Console.WriteLine(2);
```

Todo conteúdo numérico não deverá conter “ ” pois será considerado um String

## Número decimal

```
// Exibe um número decimal na tela  
Console.WriteLine(2.5);
```

Todo número decimal terá a parte inteira separada da decimal por ponto conforme formato americano

## Cálculo

```
Console.WriteLine("2 + 2");
```



```
Console.WriteLine(2 + 2);
```



Caso o cálculo esteja entre aspas também será considerada uma String

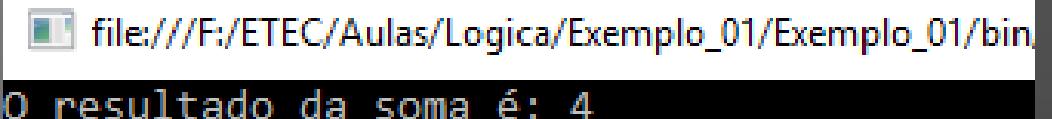
# Concatenação

**Concatenação** é um termo usado em computação para designar a operação de unir dois conteúdos que podem ser do tipo string, números, etc. Por exemplo, considerando as strings "casa" e "mento" a concatenação da primeira com a segunda gera a string "casamento".

Cada linguagem adota um símbolo para a concatenação, no C# utilizaremos o sinal de adição “+”.

Exemplo: Vamos concatenar o conteúdo de uma String com um Calculo (exibir o resultado de uma soma)

```
// Concatena uma string com um cálculo  
Console.WriteLine("O resultado da soma é: " + (2 + 2));
```



```
file:///F:/ETEC/Aulas/Logica/Exemplo_01/Exemplo_01/bin  
O resultado da soma é: 4
```

# Alguns outros métodos do Console

67



Beep – é um método que quando é executado emite um sinal sonoro em nosso programa.

```
static void Main(string[] args)
{
    Console.Beep();

    Console.ReadKey();
}
```



Clear – Limpa a tela do Console, apagando todos os caracteres existentes no mesmo.

```
static void Main(string[] args)
{
    Console.WriteLine("Olá mundo");

    Console.Clear();

    Console.ReadKey();
}
```

# Alterando propriedades Console

68



BackgroundColor – Altera a cor do fundo do Console

```
static void Main(string[] args)
{
    Console.BackgroundColor = ConsoleColor.Blue;

    Console.WriteLine("Olá mundo");

    Console.ReadKey();
}
```



Title – Altera o título da Janela Console

```
static void Main(string[] args)
{
    Console.Title = "Meu primeiro Programa";

    Console.ReadKey();
}
```



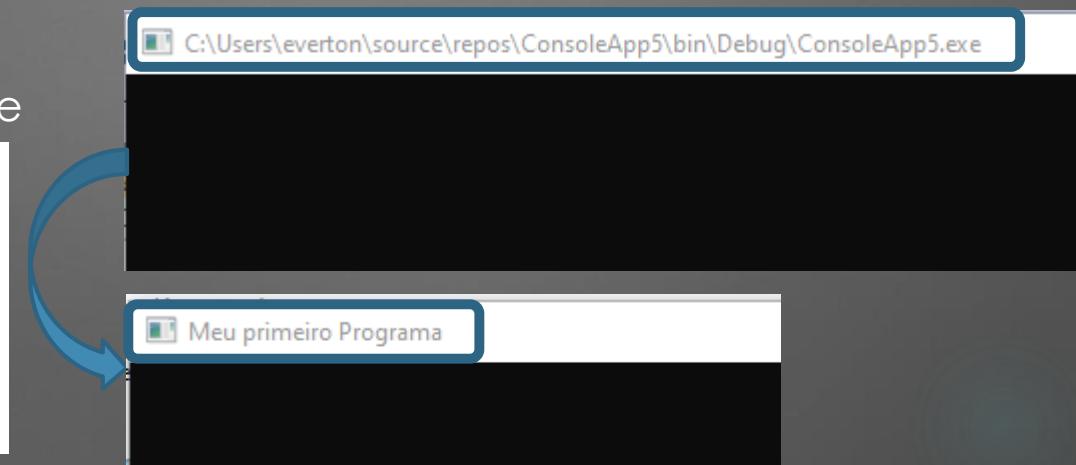
ForegroundColor – Altera a cor do texto do Console.

```
static void Main(string[] args)
{
    Console.ForegroundColor = ConsoleColor.DarkGreen;

    Console.WriteLine("Olá mundo");

    Console.ReadKey();
}
```

C:\Users\everton\source\repos\ConsoleApp5\b  
Olá mundo



# Comentários

Comentários é uma forma de documentação do código fonte onde permite o programador realizar anotações sobre determinadas rotinas.

Esses comentários são ignorados pelo compilador, ou seja, não serão executados



Comentário de apenas uma linha

```
static void Main(string[] args)
{
    // Esse comando exibe uma mensagem no Console
    Console.WriteLine("Olá mundo");

    Console.ReadKey();
}
```



Comentário de várias linhas

```
static void Main(string[] args)
{
    /* Esse tipo de comentário permite ao programador
     * inserir diversas linhas para um comentário
     * mais longo da rotina desenvolvida */

    Console.WriteLine("Olá mundo");

    Console.ReadKey();
}
```

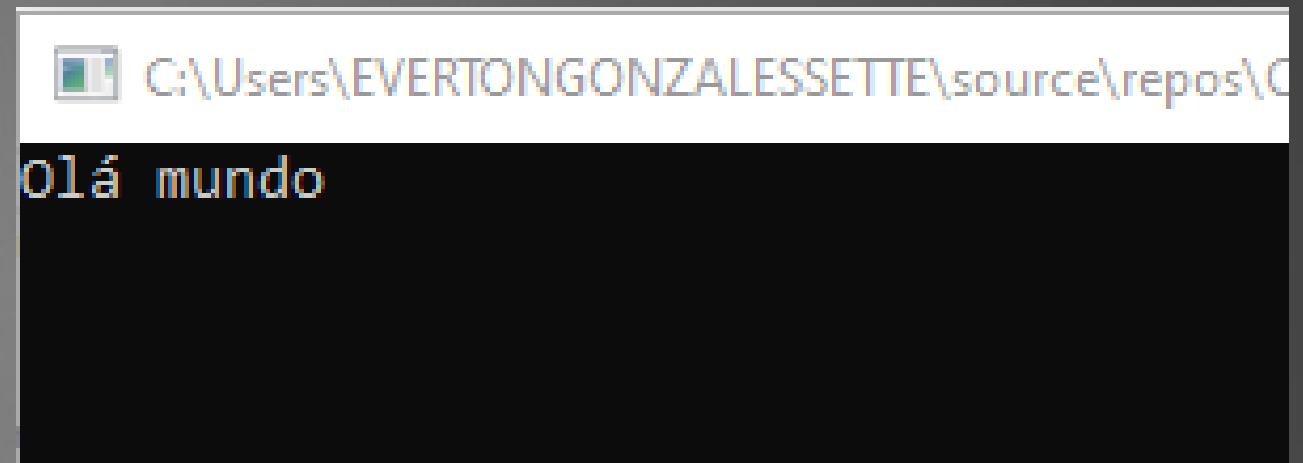


# Exemplo 01

70

```
namespace ConsoleApp3
{
    O referências
    class Program
    {
        O referências
        static void Main(string[] args)
        {
            Console.WriteLine("Olá mundo");

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



# Exemplo 02

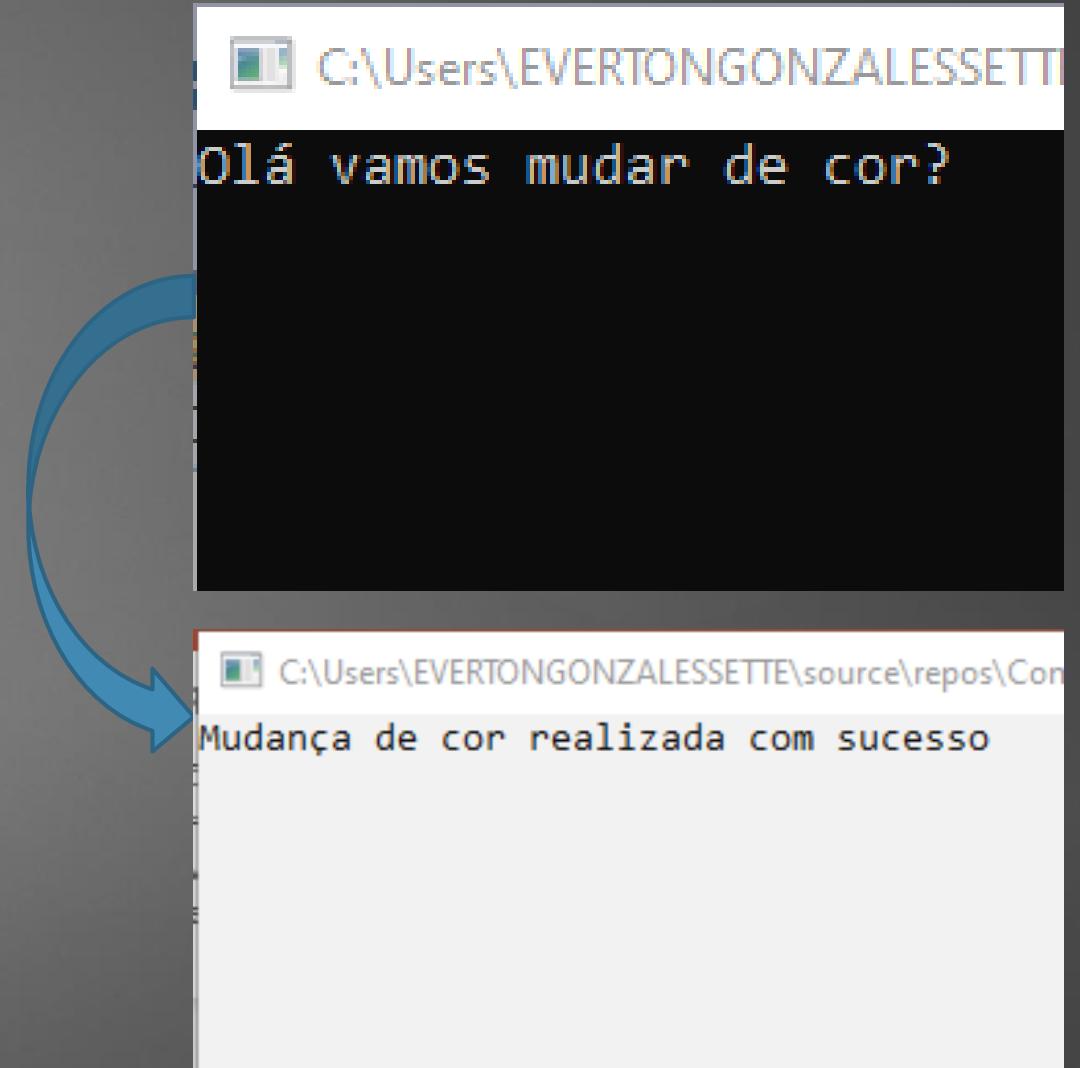
71

```
namespace ConsoleApp3
{
    O referências
    class Program
    {
        O referências
        static void Main(string[] args)
        {
            Console.WriteLine("Olá vamos mudar de cor?");
            Console.ReadKey();

            Console.BackgroundColor = ConsoleColor.White;
            Console.ForegroundColor = ConsoleColor.Black;

            Console.Clear();
            Console.WriteLine("Mudança de cor realizada com sucesso");

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



# Exemplo 03

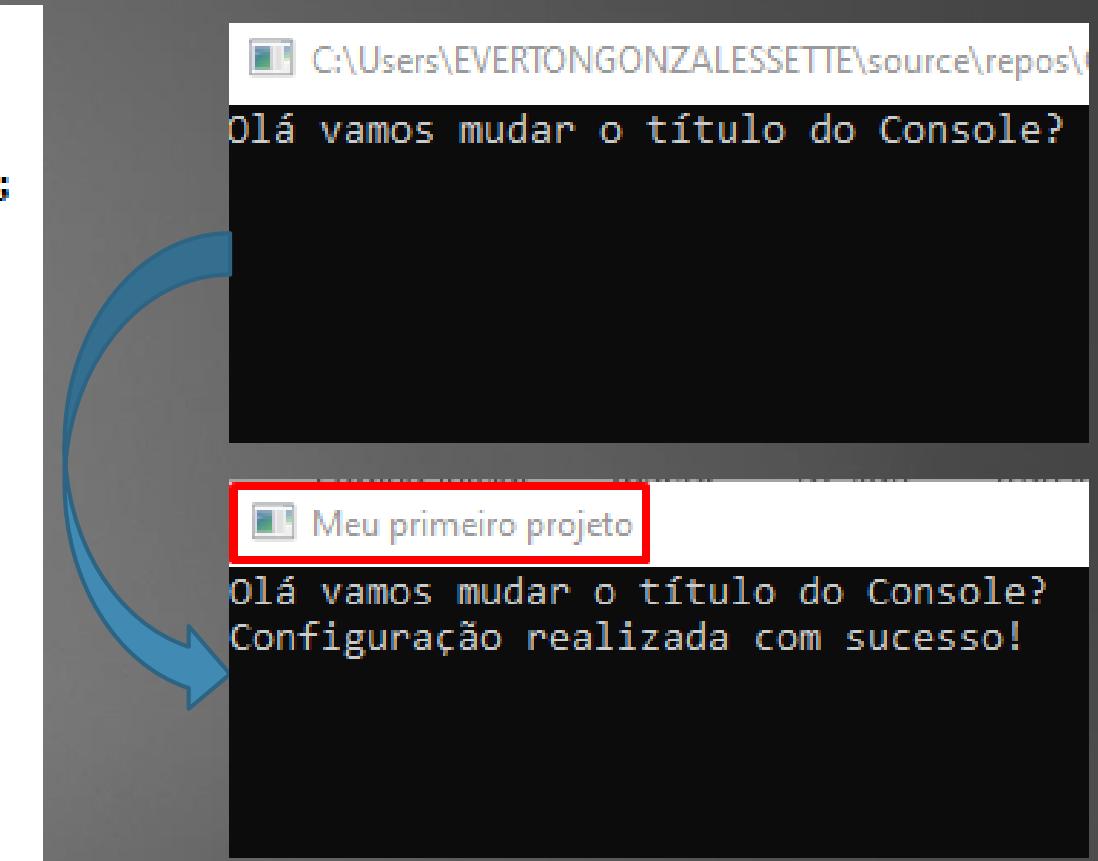
72

```
static void Main(string[] args)
{
    // Mensagem exibida na tela
    Console.WriteLine("Olá vamos mudar o título do Console?");
    // Aguarda uma tecla do usuário
    Console.ReadKey();

    // Altera o título do Console.
    Console.Title = "Meu primeiro projeto";

    // Exibe uma nova mensagem ao usuário
    Console.WriteLine("Configuração realizada com sucesso!");

    // Aguarda uma tecla do usuário
    Console.ReadKey();
}
```



# Dúvidas

73

