



CEUB

EDUCAÇÃO SUPERIOR

Programação Orientada a Objetos

ceub.br

Aula 02 –

Conceitos básicos de programação orientada a objetos com C#

Agenda

- Introdução a linguagem C#
- Arquitetura Plataforma .NET
- .NET Framework e .NET CORE
- Principais Vantagens da Linguagem C#
- Fundamentos de C#
- Estruturas de Decisão C#
- Estruturas de Repetição em C#
- Vetores
- Tratamento de Exceções
- Programação Orientada a Objetos com C#



Linguagem C# e o .Net

Introdução a linguagem C#

- **C# é uma linguagem elegante, orientada a objeto e fortemente tipada, que permite que os desenvolvedores criem uma variedade de aplicativos robustos e seguros executados no .NET Framework.**
- **C# (lê-se “c sharp”), trata-se de uma linguagem de programação desenvolvida pela Microsoft e lançada em julho de 2002.**
- **Construída do zero, sem se preocupar com compatibilidade de código legado, e a maioria das classes do framework .NET foram escritas com essa linguagem.**

Linguagem C#

- A linguagem é um dos recursos da plataforma .NET (pronuncia-se “dot net”);
- É uma linguagem orientada a objetos, cuja sintaxe foi baseada nas precursoras C++, Java e Object Pascal.
- Desse modo, programadores que conhecem pelo menos uma destas linguagens, podem facilmente aprender a programar em C#.

Linguagem C#

- **C# é uma linguagem moderna (diferente do Java que ficou obsoleto [edit: o Java 8 correu atrás do prejuízo e trouxe algumas “novidades” que as outras linguagens já tinham há anos]),**
- **C# é uma linguagem bem mais completa, recentemente passou a ter portabilidade com o Linux/Docker, graças ao projeto da Microsoft, o **.NET Core**.**
- **O .NET Core é uma plataforma de desenvolvimento de uso geral mantida pela Microsoft e pela comunidade .NET no GitHub. Ele é multiplataforma e dá suporte ao Windows, macOS e Linux, e pode ser usado em dispositivos, na nuvem e em cenários inseridos/IoT.**

Linguagem C#

- **O .NET Core é uma plataforma de desenvolvimento de uso geral mantida pela Microsoft e pela comunidade .NET no GitHub. Ele é multiplataforma e dá suporte ao Windows, macOS e Linux, e pode ser usado em dispositivos, na nuvem e em cenários inseridos/IoT.**
- **A versão mais recente é a Versão 8, que está muito próxima do padrão arquitetural da versão 6.**

Arquitetura Plataforma .NET

Com o **.NET 8/9/10** pode-se construir esse produto a partir de uma única base de código que os desenvolvedores (*Microsoft e a comunidade*) possam trabalhar e expandir juntos;

A partir da versão 5 haverá apenas um .NET, e você poderá usá-lo para segmentar Windows, Linux, macOS, iOS, Android, tvOS, watchOS e WebAssembly e outras tecnologias.

Serão introduzidas novas APIs .NET, recursos de runtime e recursos de linguagem como parte do .NET 5.

Arquitetura Plataforma .NET

Build anything with a unified platform

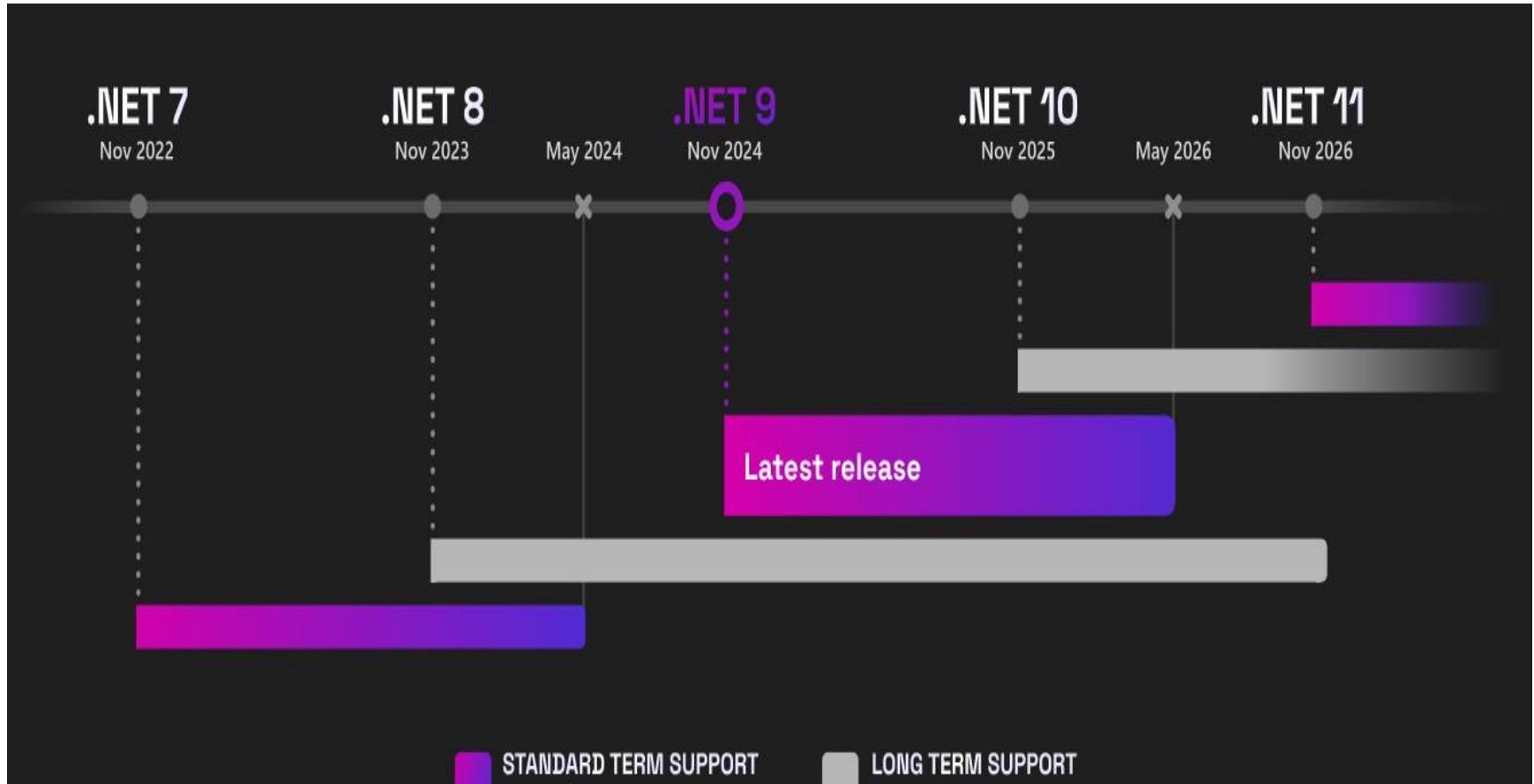


.NET Framework e .NET CORE

NET Framework é um ambiente de execução de tempo de execução que gerencia os aplicativos direcionados .NET Framework.

Ele consiste no **Common Language Runtime - CLR, que fornece gerenciamento de memória e outros serviços do sistema, além de em uma biblioteca de classes extensa, o que permite que programadores usem o código robusto e confiável para todas as áreas principais do desenvolvimento de aplicativos.**

.NET CORE



.NET Framework e .NET CORE

O **CLR (Common Language Runtime)** pode ser ainda o ambiente que **gerencia** a execução do código .

Ele fornece os serviços core como compilação de código, alocação de memória, gerenciamento de thread e coleta de lixo.

O CTS (Common Type System) força a tipagem de variáveis e garante que o código é executado em um ambiente seguro e também com segurança ao acesso ao código.

.NET Framework e .NET CORE

.NET Framework e .NET CORE são diferentes

O **.NET Framework** está na versão 4.8 e é usado para desenvolvimento de aplicações **Windows** usando Windows Forms , WPF e de aplicações Web usando ASP .NET MVC.

O **.NET Core, hoje na versão 8, é open-source e multiplataforma** e suporta UWP e as bibliotecas da ASP .NET Core.

A UWP é usada para criar aplicação Windows 10 ou 11 e a ASP .NET Core é usada para criar aplicações Web para Windows, Linux e Mac.

Principais Vantagens da Linguagem C#

- **Orientada a Objetos;**
- **Linguagem acessível para iniciantes;**
- **Linguagem multiplataforma;**
- **Uma das linguagens mais completas do mercado**
- **Mercado ativo para quem programa em C#**
- **Simplicidade: os projetistas de C# costumam dizer que essa linguagem é tão poderosa quanto o C++ e tão simples quanto o Visual Basic;**
- **Fortemente tipada:** isso ajudará a evitar erros por manipulação imprópria de tipos e atribuições incorretas;

IDE - Ambiente Integrado de Desenvolvimento

É um conjunto de softwares utilizado para a construção de programas.

Exemplos:

C# : Microsoft Visual Studio, Visual Studio Code

C++ : Code Blocks, Visual Studio Code

Java : Eclipse, NetBeans, Visual Studio Code

IDE - Ambiente Integrado de Desenvolvimento



Visual Studio 2022

Windows | Versão 17.1

O melhor IDE abrangente para desenvolvedores .NET e C++ no Windows. Totalmente empacotado com uma bela matriz de ferramentas e recursos para elevar e aprimorar cada estágio de desenvolvimento de software.

[Notas de versão](#) > [Comparar edições](#) > [Como fazer a instalação offline](#) >

Comunidade

IDE avançado, gratuito para estudantes, colaboradores de software livre e indivíduos

Download
gratuito

Professional

IDE profissional mais indicado para equipes pequenas

Avaliação
gratuita

Enterprise

Solução ponta a ponta escalonável para equipes de qualquer tamanho

Avaliação
gratuita

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

IDE - Ambiente Integrado de Desenvolvimento

Workloads

Individual components

Language packs

Windows (3)



Universal Windows Platform development

Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.



.NET desktop development

Build WPF, Windows Forms and console applications using the .NET Framework.



Desktop development with C++

Build classic Windows-based applications using the power of the Visual C++ toolset, ATL, and optional features like MFC and...



Web & Cloud (5)



ASP.NET and web development

Build web applications using ASP.NET, ASP.NET Core, HTML, JavaScript, and CSS.



Azure development

Azure SDK, tools, and projects for developing cloud apps and creating resources.



Node.js development

Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.



Data storage and processing

Connect, develop and test data solutions using SQL Server, Azure Data Lake, Hadoop or Azure ML.



Office/SharePoint development



Fundamentos de C#

Definições e estrutura de um programa em C#

Importações

- Usa a diretiva `using`
- Importações definem as bibliotecas que o programa deve utilizar;
- Sempre feito, quando necessário, no início do programa C#.

Namespace

- Utilizado para organização do código pastas
- São divisões (agrupamentos) lógicos das classes relacionadas
- A divisão física é por meio de pastas e não pelo namespace

Classe

- Elemento principal do programa
- Todo programa .NET em C# sempre terá pelo menos uma Classe
- Tudo que é executado estará aqui

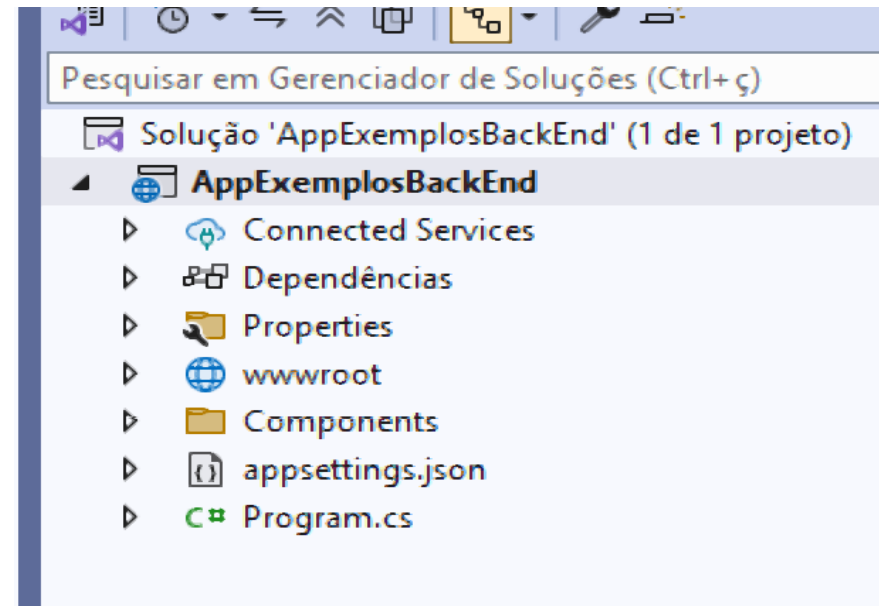
Métodos

- Principal, hoje não mais necessário, somente aparecerá se o usuário selecionar as funções de ordem superior

Definições e estrutura de um Projeto em C#

Estrutura de um projeto

- **Solução .sln**
- **Projeto .csproj**
- **Classe principal .cs**
- **Subpastas obj e bin**
- **Program.cs**



Definições e estrutura de um programa em C#

Exemplo

```
C#  
  
using System;  
  
namespace MeuPrimeiroPrograma  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            // Comentário: Exibe uma mensagem na tela  
            Console.WriteLine("Olá, mundo!");  
        }  
    }  
}
```



Conceitos Gerais sobre C# Tipo de Dados

Tipo de Dados e C#

Tipos de Dados em C#

Tipo	Tamanho	Valores Possíveis
bool	1 byte	true e false
byte	1 byte	0 a 255
sbyte	1 byte	-128 a 127
short	2 bytes	-32768 a 32767
ushort	2 bytes	0 a 65535
int	4 bytes	-2147483648 a 2147483647
uint	4 bytes	0 to 4294967295
long	8 bytes	-9223372036854775808L to 9223372036854775807L
ulong	8 bytes	0 a 18446744073709551615
float	4 bytes	Números até 10 elevado a 38. Exemplo: 10.0f, 12.5f
double	8 bytes	Números até 10 elevado a 308. Exemplo: 10.0, 12.33
decimal	16 bytes	números com até 28 casas decimais. Exemplo 10.991m, 33.333m
char	2 bytes	Caracteres delimitados por aspas simples. Exemplo: 'a', 'ç', 'o'

Tipos de Dados Primitivos

Tipos de dados primitivos são os tipos de dados básicos usados na construção de algoritmos. Alguns tipos de dados:

- **INTEIRO** - Int
- **REAL** - Double
- **CARACTER** - Char
- **LÓGICO** - Bool

Tipos de Dados em C#

Tipo	Tamanho	Valores Possíveis
bool	1 byte	true e false
byte	1 byte	0 a 255
sbyte	1 byte	-128 a 127
short	2 bytes	-32768 a 32767
ushort	2 bytes	0 a 65535
int	4 bytes	-2147483648 a 2147483647
uint	4 bytes	0 to 4294967295
long	8 bytes	-9223372036854775808L to 9223372036854775807L
ulong	8 bytes	0 a 18446744073709551615
float	4 bytes	Números até 10 elevado a 38. Exemplo: 10.0f, 12.5f
double	8 bytes	Números até 10 elevado a 308. Exemplo: 10.0, 12.33
decimal	16 bytes	números com até 28 casas decimais. Exemplo 10.991m, 33.333m
char	2 bytes	Caracteres delimitados por aspas simples. Exemplo: 'a', 'ç', 'o'

VARIÁVEL

A variável é um espaço na memória, ela também é identificada por um nome conhecido como identificador, que pode conter dados de algum determinado tipo primitivo.

Variável é tudo que está sujeito a variações, que é incerto, instável ou inconstante.

Os dados armazenados em uma variável são os seus conteúdos e estes podem variar durante a execução do programa à qual ele pertença.

Ex. $TOTAL = Produto * Quantidade$

Ex2. $MEDIA = (nota1+nota2+nota3)/4$

VARIÁVEL

Em C#, todas as variáveis utilizadas são declaradas seguindo a seguinte estrutura:

<tipo> <nome> = <valor inicial>;
(opcional)

EXEMPLO:

```
String nome, sobrenome;  
int idade;  
double salario;  
bool tem_filhos;
```

VARIÁVEL - REGRAS DE NOMECLATURA

A linguagem C# possui regras técnicas relacionadas à nomenclatura das variáveis. O nome (identificador) de uma variável é uma sequência limitada de caracteres que:

1. Deve começar com uma letra ou com o caractere `_`.
2. Ser for igual a uma palavra reservada, é necessário adicionar o prefixo `@`.
3. Pode conter letras e dígitos

VARIÁVEL - PALAVRAS RESERVADAS

Palavras reservadas em C#

abstract	as	base	bool	break
byte	case	catch	char	checked
class	const	continue	decimal	default
delegate	do	double	else	enum
event	explicit	extern	false	finally
fixed	float	for	foreach	goto
if	implicit	in	int	interface
internal	is	lock	long	namespace
new	null	object	operator	out
override	params	private	protected	public
readonly	ref	return	sbyte	sealed
short	sizeof	stackalloc	static	string
struct	switch	this	throw	true
try	typeof	uint	ulong	unchecked
unsafe	ushort	using	virtual	void
volatile	while			

Comandos de Entrada

COMANDOS DE ENTRADA

É utilizado para receber dados digitados pelo usuário, que serão armazenados em variáveis.

Em C# esse comando é:

```
(Console.ReadLine());
```

Ex: Suponha uma variável tipo String:

```
string x;
```

```
x = Console.ReadLine();
```

Essa ação corresponde a:

```
leia(X)
```

//O valor digitado pelo usuário será armazenado na variável X.

Comandos de Saída

COMANDOS DE SAÍDA

É utilizado para mostrar dados na tela ou na impressora.
Em C# esse comando é realizado da seguinte maneira:

`Console.Write();` **//sem quebra de linha ao final**

Ex:

```
Console.Write("Bom dia!");
```

ou

`Console.WriteLine();` **//com quebra de linha ao final**

Ex

```
Console.WriteLine("Bom dia!");
```

PRIMEIRO EXEMPLO – OLÁ MUNDO

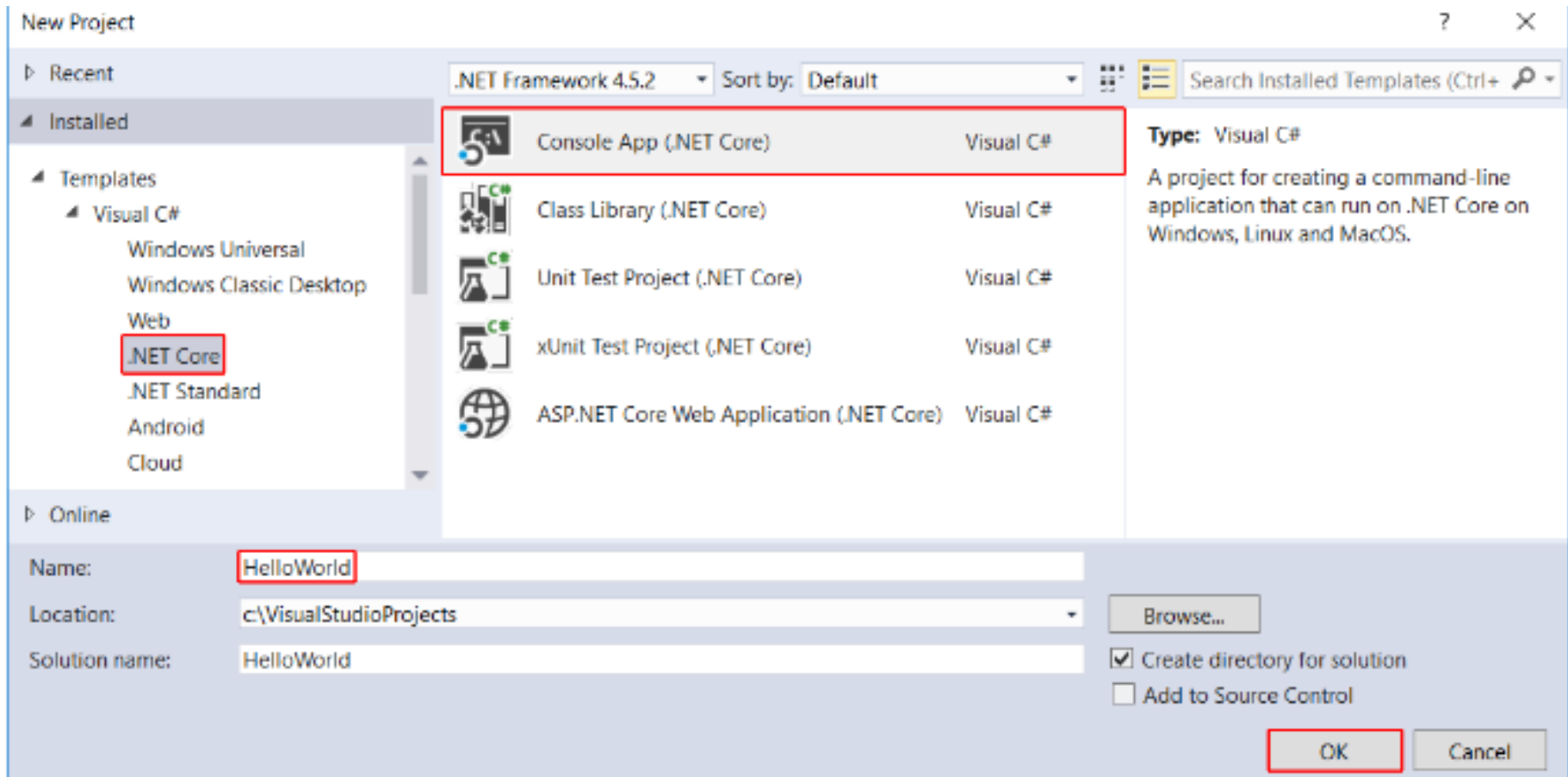
Um aplicativo simples Olá, Mundo

Comece criando um aplicativo de console simples "Olá, Mundo".

Siga estas etapas:

1. Inicie o Visual Studio 2017.
2. Selecione **Arquivo > Novo > Projeto** na barra de menus.
3. Na caixa de diálogo *Novo projeto**, selecione o nó **Visual C#** seguido pelo nó **.NET Core**.
4. Em seguida, selecione o modelo de projeto **Aplicativo de console (.NET Core)**.
5. Na caixa de texto **Name**, digite "HelloWorld". Selecione o botão **OK**.

PRIMEIRO EXEMPLO – OLÁ MUNDO



PRIMEIRO EXEMPLO – OLÁ MUNDO

Visual Studio 2019

Abrir recente

Pesquisar recentes (Alt+S)

Hoje



ExemploWinFormLP.sln

C:\Users\Acer\source\repos\ExemploWinFormLP

27/08/2020 19:23

Introdução



Clonar um repositório

Obter o código de um repositório online, como o GitHub ou o Azure DevOps



Abrir um projeto ou uma solução

Abrir um projeto local do Visual Studio ou arquivo .sln



Abrir uma pasta local

Navegar e editar o código dentro de qualquer pasta



Criar um projeto

Escolha um modelo de projeto com scaffolding de código para começar


[Continuar sem código →](#)



PRIMEIRO EXEMPLO – OLÁ MUNDO

Criar um novo projeto

Modelos de projeto recentes


 Aplicativo do Windows Forms (.NET Framework) C#


✕ Limpar tudo


Todos os idiomas


Todas as plataformas


Todos os tipos de projeto

 **Aplicativo do Windows Forms (.NET Framework)**
Um projeto para criar um aplicativo com uma interface do usuário do Windows Forms (WinForms)
C# Windows Área de Trabalho

 **Windows Forms App (.NET Core)**
Um projeto para criar um aplicativo com uma interface do usuário do Windows Forms (WinForms)
C# Windows Área de Trabalho

 **Biblioteca de Controles do Windows Forms (.NET Framework)**
Um projeto para criar controles a serem usados em aplicativos do Windows Forms (WinForms)
C# Windows Área de Trabalho Biblioteca

 **Aplicativo do Windows Forms (.NET Framework)**
Um projeto para criar um aplicativo com uma interface do usuário do Windows Forms (WinForms)
Visual Basic Windows Área de Trabalho

 **Biblioteca de Controles do Windows Forms (.NET Framework)**
Um projeto para criar controles a serem usados em aplicativos do Windows Forms (WinForms)

Voltar

Próximo



PRIMEIRO EXEMPLO – OLÁ MUNDO

Configurar seu novo projeto

Aplicativo do Windows Forms (.NET Framework) C# Windows Área de Trabalho

Nome do projeto

WinformsOLAMundo

Local

C:\Users\Acer\source\repos

Nome da solução ⓘ

WinformsOLAMundo

☐ Colocar a solução e o projeto no mesmo diretório

Framework

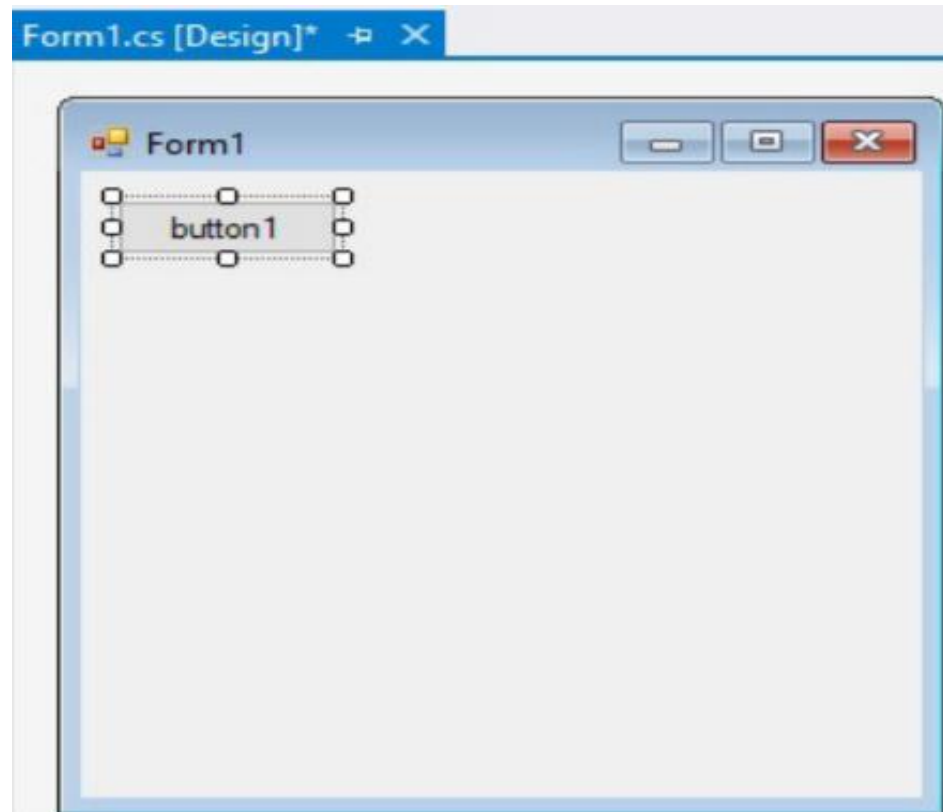
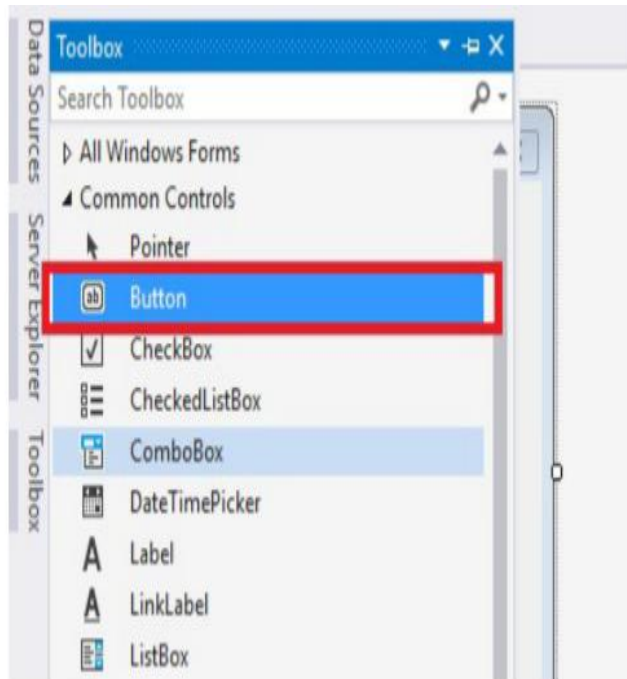
.NET Framework 4.7.2

Voltar

Criar

PRIMEIRO EXEMPLO – OLÁ MUNDO

Vamos adicionar um botão no formulário que, quando clicado, abrirá uma caixa de mensagem do Windows com a mensagem **OLÁ Mundo**.



PRIMEIRO EXEMPLO – OLÁ MUNDO

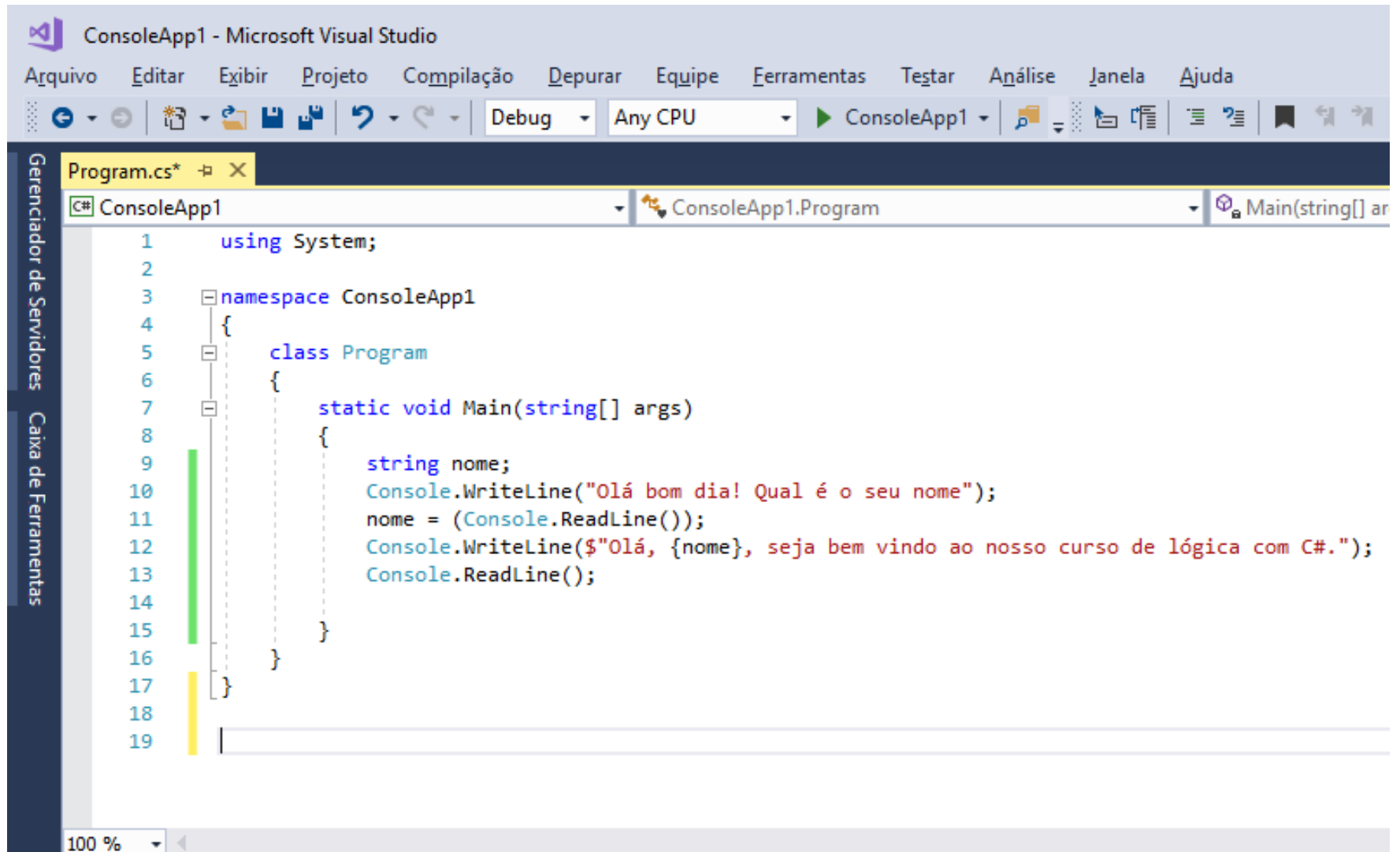
Após adicionar o botão, dê um duplo clique no botão que acabamos de adicionar para programarmos o que deve acontecer quando o botão for clicado.

O Visual Studio abrirá o código do formulário. Conforme Descrito a seguir

Exercício 2

Criar um programa usando o console do Windows em C# que pergunte seu nome e forneça como saída uma saudação. Olá seu nome... + , seja bem-vindo a nosso curso de lógica.

Exercício 2 - solução



The screenshot displays the Microsoft Visual Studio IDE with a C# console application named 'ConsoleApp1'. The interface includes a menu bar (Arquivo, Editar, Exibir, Projeto, Compilação, Depurar, Equipe, Ferramentas, Testar, Análise, Janela, Ajuda) and a toolbar with icons for navigation and execution. The 'Program.cs' file is open, showing the following code:

```
1 using System;
2
3 namespace ConsoleApp1
4 {
5     class Program
6     {
7         static void Main(string[] args)
8         {
9             string nome;
10            Console.WriteLine("Olá bom dia! Qual é o seu nome");
11            nome = (Console.ReadLine());
12            Console.WriteLine($"Olá, {nome}, seja bem vindo ao nosso curso de lógica com C#.");
13            Console.ReadLine();
14        }
15    }
16 }
17
18
19
```

The code is a simple console application that prompts the user for their name and displays a personalized greeting. The IDE interface also shows the 'Gerenciador de Servidores' and 'Caixa de Ferramentas' on the left side, and a status bar at the bottom indicating 100% zoom.

EXERCÍCIO 3

CRIAR UM PROGRAMA PARA CALCULAR O VALOR DA ÁREA DE UMA CIRCUNFERÊNCIA.

USAR A FORMULA $\text{PI} * \text{RAIO} * \text{RAIO}$ VALOR DE PI = 3,1416;

EXERCÍCIO 3 - PORTUGOL

CRIAR UMA ALGORITMO PARA CALCULAR O VALOR DA ÁREA DE UMA CIRCUNFERÊNCIA.

USAR A FORMULA $PI * RAIO * RAIO$ VALOR DE $PI = 3,1416$;

Algoritmo Area_circunferencia

VAR area , raio : Real

CONST $PI = 3.1416$

inicio

escreva('Informe o raio de uma Circunferência.');

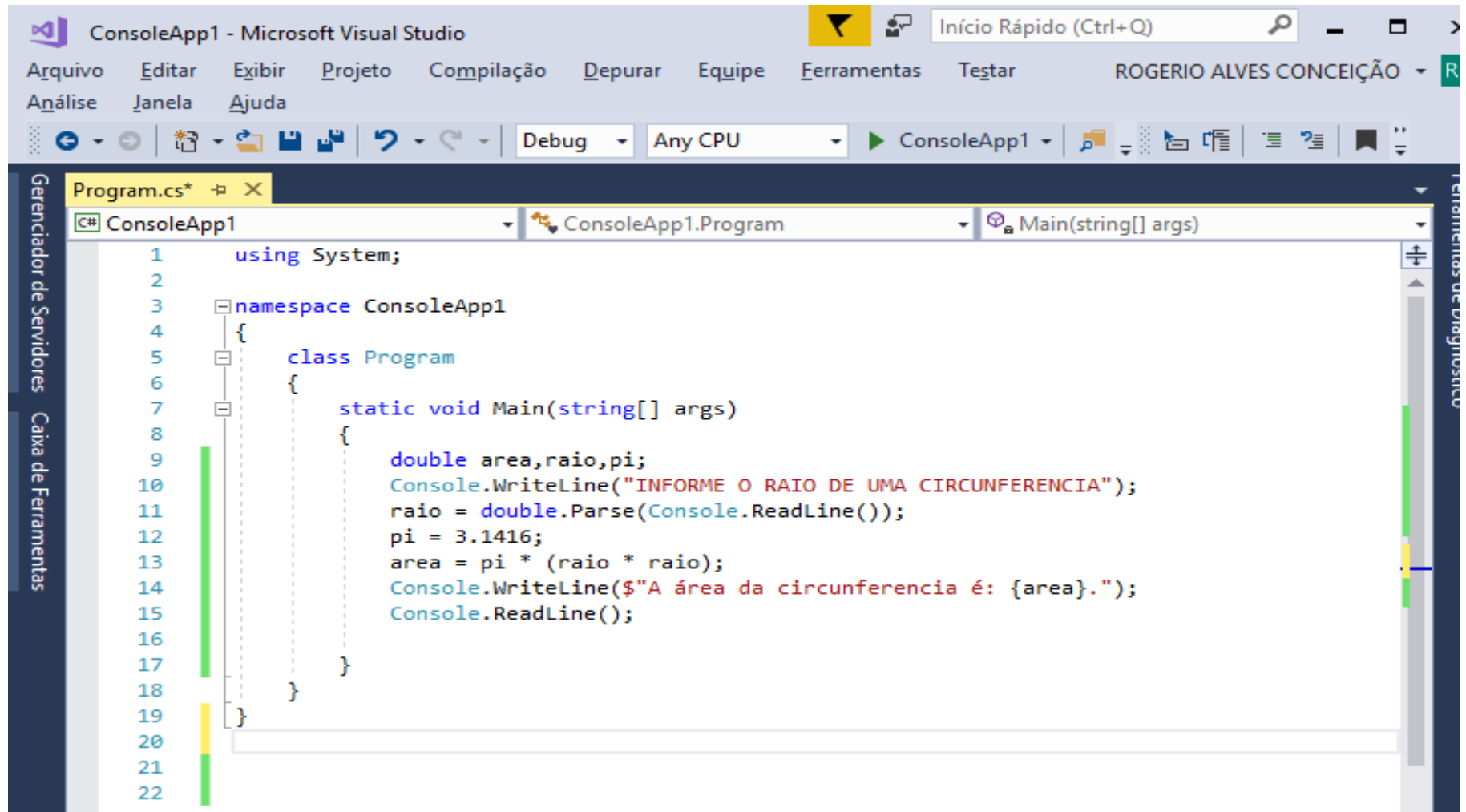
leia(raio);

area = $PI * (raio * raio)$;

escreva('A Área da circunferência é: ' , area);

fim

EXERCÍCIO 3 – C#



```
1  using System;
2
3  namespace ConsoleApp1
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              double area,raio,pi;
10             Console.WriteLine("INFORME O RAIO DE UMA CIRCUNFERENCIA");
11             raio = double.Parse(Console.ReadLine());
12             pi = 3.1416;
13             area = pi * (raio * raio);
14             Console.WriteLine($"A área da circunferencia é: {area}.");
15             Console.ReadLine();
16         }
17     }
18 }
19
20
21
22
```

Expressões Aritméticas

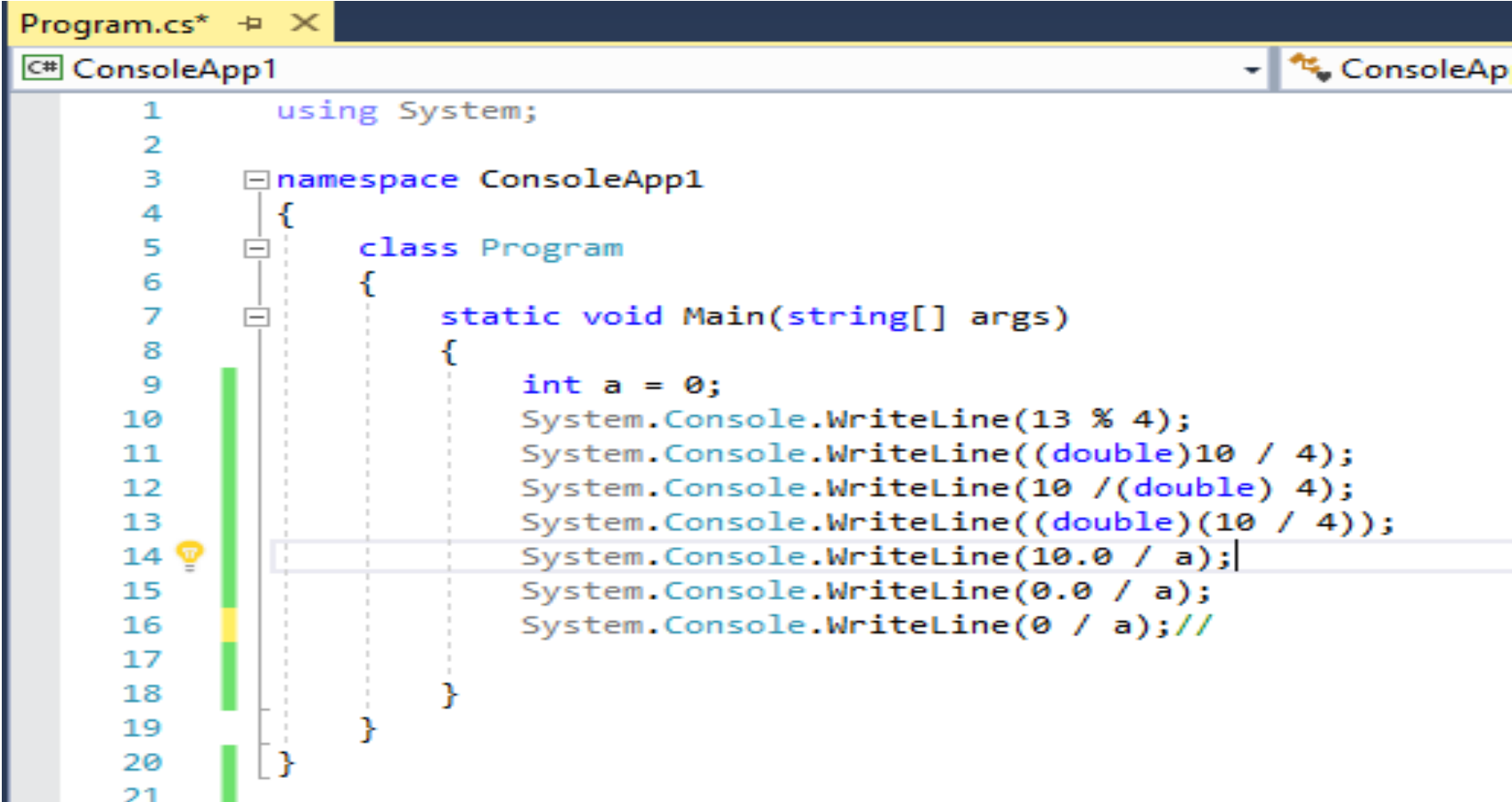
FUNÇÕES MATEMÁTICAS EM C#

Os operadores aritméticos funcionam de forma muito semelhante aos operadores da Matemática. Os operadores aritméticos são:

OPERADOR	SIMBOLO
ADIÇÃO	+
SUBTRAÇÃO	-
MULTIPLICAÇÃO	*
DIVISÃO	/
RESTO DA DIVISÃO	%

OPERADORES ARITMÉTICOS

Simulação – Criar um programa console em c# com a seguinte estrutura:



```
Program.cs* X
C# ConsoleApp1 ConsoleAp

1  using System;
2
3  namespace ConsoleApp1
4  {
5      class Program
6      {
7          static void Main(string[] args)
8          {
9              int a = 0;
10             System.Console.WriteLine(13 % 4);
11             System.Console.WriteLine((double)10 / 4);
12             System.Console.WriteLine(10 / (double) 4);
13             System.Console.WriteLine((double)(10 / 4));
14             System.Console.WriteLine(10.0 / a);
15             System.Console.WriteLine(0.0 / a);
16             System.Console.WriteLine(0 / a); //
17
18         }
19     }
20 }
21
```

FUNÇÕES MATEMÁTICAS EM C#

A classe **System.Math** oferece muitos campos constantes e métodos estáticos que você pode usar para fazer cálculos trigonométricos, logarítmicos, e outros cálculos matemáticos.

Por exemplo, o método **Pow** da classe **System.Math** pode ser usado para elevar um número a uma potência de x .

A seguir alguns exemplos de funções matemáticas e constantes existentes na classe **System.Math**:

Exemplos de algumas funções

Pow(x,y)	Obtém o valor de x elevado na y	Pow(2,4) = 16
Round(x,y)	Arredonda x para y casas decimais	Round(7.6758549,4) = 7.6759
Sqrt(x)	Obtém a raiz quadrada de x	Sqrt(169) = 13
Exp(x)	Obtém o exponencial (e elevado na x)	Exp(5.0) = 54.59...

OPERADORES ARITMÉTICOS

Exemplo – Criar um programa console em c# que calcule a área de um círculo. A Formula é $Area=PI \cdot R^2$. Use a classe Math.

OPERADORES ARITMÉTICOS

Exemplo – Criar um programa console em c# que calcule a área de um círculo. A Formula é $Area=PI \cdot R^2$. Use a classe Math.

```
using System;
namespace ConsoleApp2
{
    class Program
    {
        static void Main(string[] args)
        {
            double areaCirculo = 0;
            double RaioDoCirculo = 0;
            Console.WriteLine(" Informe o raio do Círculo : ");
            RaioDoCirculo = Convert.ToDouble(Console.ReadLine());
            areaCirculo = Math.PI * Math.Pow(RaioDoCirculo, 2);
            Console.WriteLine(" A área do círculo de raio " +
            RaioDoCirculo.ToString() + " é : " + areaCirculo.ToString());
            Console.ReadKey();
        }
    }
}
```

OPERADORES RELACIONAIS em C#

A Tabela abaixo apresenta os operadores relacionais que podem ser encontrados nas linguagens de programação C#, Java, C e C++.

OPERADOR	DESCRIÇÃO
==	IGUA A
!=	DIFERENTE
>	MAIOR QUE
<	MENOR QUE
>=	MAIOR OU IGUAL
<=	MENOR O U IGUAL

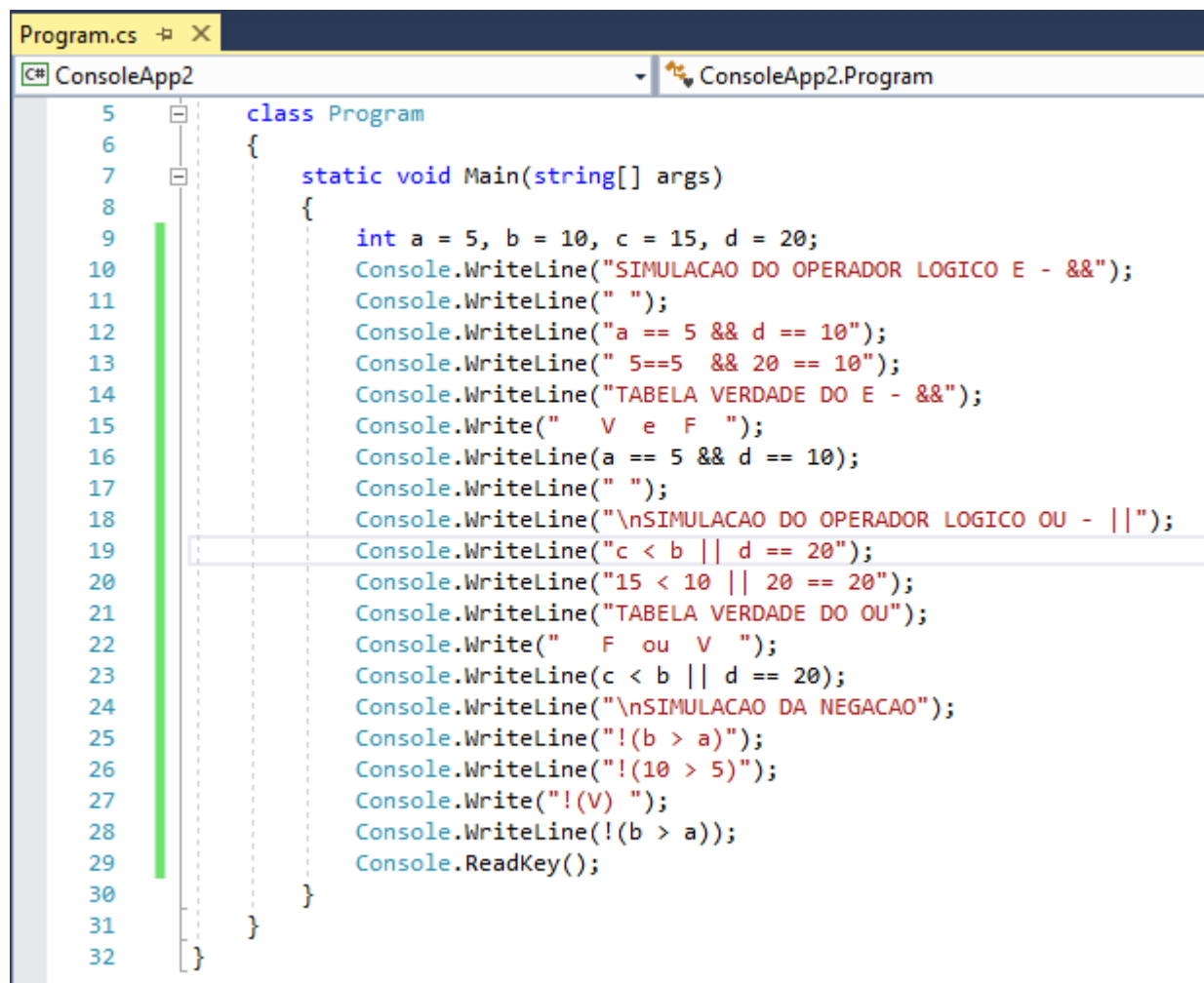
OPERADORES LÓGICOS em C#

Os operadores lógicos, junto com os operadores relacionais, são empregados na construção de expressões lógicas, que são aquelas expressões que sempre resultam num valor verdadeiro ou falso.

	Operador C#	DESCRIÇÃO
E	&&	Efetua a operação lógica E, ou também chamado de conjunção. Se ambas as expressões forem verdadeiras, o resultado será verdadeiro.
OU		Efetua a operação lógica Ou. Se uma das expressões (ou ambas) for verdadeira, então o resultado será verdadeiro.
XOR	^	XOR ou Disjunção exclusiva só será "V" se uma das partes for "F" e a outra "V" (independentemente da ordem) não podendo acontecer "V" ou "F" nos dois casos, caso aconteça a proposição resultante desta operação será falsa.
NÃO	(!)	Efetua a negação de uma expressão lógica. Se ela for verdadeira, a torna falsa. E vice-versa.

OPERADORES LÓGICOS

Simulação – Criar um programa console em c# com a seguinte estrutura:



```
Program.cs + X
C# ConsoleApp2 ConsoleApp2.Program

5 class Program
6 {
7     static void Main(string[] args)
8     {
9         int a = 5, b = 10, c = 15, d = 20;
10        Console.WriteLine("SIMULACAO DO OPERADOR LOGICO E - &&");
11        Console.WriteLine(" ");
12        Console.WriteLine("a == 5 && d == 10");
13        Console.WriteLine("5==5 && 20 == 10");
14        Console.WriteLine("TABELA VERDADE DO E - &&");
15        Console.Write(" V e F ");
16        Console.WriteLine(a == 5 && d == 10);
17        Console.WriteLine(" ");
18        Console.WriteLine("\nSIMULACAO DO OPERADOR LOGICO OU - ||");
19        Console.WriteLine("c < b || d == 20");
20        Console.WriteLine("15 < 10 || 20 == 20");
21        Console.WriteLine("TABELA VERDADE DO OU");
22        Console.Write(" F ou V ");
23        Console.WriteLine(c < b || d == 20);
24        Console.WriteLine("\nSIMULACAO DA NEGACAO");
25        Console.WriteLine("!(b > a)");
26        Console.WriteLine("!(10 > 5)");
27        Console.Write("!(V) ");
28        Console.WriteLine(!(b > a));
29        Console.ReadKey();
30    }
31 }
32 }
```


COMANDOS/OPERADORES DE ATRIBUIÇÃO

Um operador de atribuição altera o valor armazenado em uma variável.

= (atribuição simples)

Ex `int a=2;`

Operadores Aritméticos de Atribuição Reduzida – Esses operadores são usados para compor uma operação aritmética e uma atribuição, conforme é descrito na tabela a seguir:

Operador Aritmético	Descrição
<code>+</code> =	mais igual
<code>-</code> =	menos igual
<code>*</code> =	vezes igual
<code>/</code> =	dividido igual
<code>%</code> =	módulo igual

OBS: Os operadores de atribuição `+=`, `-=`, `*=`, `/=`, `%=`, `++` e `--` são chamados de operadores compostos, pois além de modificar o valor de uma variável, eles realizam a operação aritmética correspondente.

Por exemplo, a operação `X+=Y` o resultado é o mesmo que `X = X+Y`

COMANDOS/OPERADORES DE ATRIBUIÇÃO

Simulação – Criar um programa console em c# com a seguinte estrutura:

```
namespace ConsoleApp2
{
    class Program
    {
        static void Main(string[] args)
        {
            int X;
            X = 50;
            Console.WriteLine("Atribuição: {0}", X);
            X += 20;
            Console.WriteLine("Acumulando: {0}", X);
            X *= 2;
            Console.WriteLine("Multiplicando: {0}", X);
            X %= 6;
            Console.WriteLine("MÓDULO: {0}", X);
            X++;
            Console.WriteLine("INCREMENTO: {0}", X);
            X--;
            Console.WriteLine("DECREMENTO: {0}", X);
            Console.ReadKey();
        }
    }
}
```

Exercício de Fixação

Para cada exercícios, você deverá criar um novo projeto no Visual Studio do tipo Console Application.

- 1 - Crie um programa que solicite um nome, endereço e telefone e imprima estes dados na tela.
- 2 - Crie um programa que leia um número inteiro e imprima seu sucessor e se antecessor. Ex: Num1 = 3,
Saída: “O Número 3 possui o sucessor 4 e antecessor 2”;
- 3 - Crie um programa que leia três números e imprima sua média;
- 4 - Crie um programa que calcule a média final dos alunos do 1º semestre de lógica. Os alunos realizarão quatro provas: P1, P2, P3 e P4. Onde:
- 5 - Fazer um algoritmo que possa entrar com o saldo de uma aplicação e imprima o novo saldo, considerando o reajuste de 1%.

6 - Crie um programa que calcule a área de um trapézio. Sendo que a área do trapézio é dado pela seguinte fórmula:

$$A = \frac{(B + b) \cdot h}{2}$$

Estruturas de Decisão C#

Estruturas Condicionais/Decisão C#

Simples:

```
if ( <condição> ) {  
    <comando 1>  
    <comando 2>  
}
```

REGRA:

V: executa o bloco de comandos
F: pula o bloco de comandos

Composta:

```
if ( <condição> ) {  
    <comando 1>  
    <comando 2>  
}  
else {  
    <comando 3>  
    <comando 4>  
}
```

REGRA:

V: executa somente o bloco do **if**
F: executa somente o bloco do **else**

Estruturas Condicionais/Decisão C#

Estruturas de decisão encadeadas

```
if ( condição 1 ) {  
    comando 1  
    comando 2  
}  
else {  
    if ( condição 2 ) {  
        comando 3  
        comando 4  
    }  
    else {  
        comando 5  
        comando 6  
    }  
}
```

Estruturas Condicionais/Decisão C#

Estruturas de decisão Seleção Caso/

```
switch (variável ou valor)
{
    case valor1:
        // código 1
        break;
    case valor2:
        // código 2
        break;
    default:
        // código 3
        break;
}
```


Estruturas Condicionais/Decisão C#

Exemplo – Algoritmo para verificar se os números são iguais

```
static void Main(string[] args)
{
    int num1;
    int num2;
    Console.WriteLine("Digite um número:");
    num1 = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite outro número:");
    num2 = int.Parse(Console.ReadLine());
    if (num1 == num2)
    {
        Console.WriteLine("Os números são iguais");
    }
    Console.ReadKey();
}
```

Estruturas Condicionais/Decisão C#

Exemplo – Algoritmo para verificar se os números são iguais ou diferentes

```
static void Main(string[] args)
{
    int num1;
    int num2;
    Console.WriteLine("Digite um número:");
    num1 = int.Parse(Console.ReadLine());
    Console.WriteLine("Digite outro número:");
    num2 = int.Parse(Console.ReadLine());
    if (num1 == num2)
    {
        Console.WriteLine("Os números são iguais");
    }
    else
    {
        Console.WriteLine("Os números são diferentes");
    }
    Console.ReadKey();
}
```

Estruturas Condicionais/Decisão C#

Exemplo3

```
static void Main(string[] args)
{
    int num1;
    Console.WriteLine("Digite um número:");
    num1 = int.Parse(Console.ReadLine());
    switch (num1)
    {
        case 1:
            Console.WriteLine("O número que você selecionado é o 1");
            break;
        case 2:
            Console.WriteLine("O número que você selecionado é o 2");
            break;
        default:
            Console.WriteLine("Você digitou um número maior que 2");
            break;
    }
    Console.ReadKey();
}
```

Estruturas de Repetição em C#

Estruturas Repetição em C#

Estruturas Repetição Enquanto <condição> faça

```
while ( condição ) {  
    comando 1  
    comando 2  
}
```

Regra:

V: executa e volta
F: pula fora

Exemplo

```
static void Main(string[] args)  
{  
    int n = 1;  
    while (n < 6)  
    {  
        Console.WriteLine("O valor corrente é:{0}", n);  
        n++;  
    }  
    Console.ReadKey();  
}
```

Estruturas Repetição em C#

Estruturas Repetição Para <condição> faça

Executa somente
na primeira vez

V: executa e volta
F: pula fora

Executa toda vez depois
de voltar

```
for ( início ; condição ; incremento) {  
    comando 1  
    comando 2  
}
```

Exemplo

```
using System;  
namespace ConsoleApp1  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            for (int i = 1; i <= 5; i++)  
            {  
                Console.WriteLine(i);  
            }  
            Console.ReadKey();  
        }  
    }  
}
```

Estruturas Repetição em C#

Estruturas Repetição Faça Enquanto <condição>

```
do {  
    comando 1  
    comando 2  
} while ( condição );
```

Regra:

V: volta

F: pula fora

Exemplo

```
using System;  
namespace ConsoleApp1  
{  
    class Program  
    {  
        static void Main(string[] args)  
        {  
            int x = 0;  
            do  
            {  
                Console.WriteLine(x);  
                x++;  
            } while (x < 5);  
        }  
    }  
}
```

Exercícios C#

1 - Desenvolva um algoritmo que solicite a entrada de um número e calcule se o número é par ou impar.

```
static void Main(string[] args)
{
    int NumeroDigitado;
    Console.Write("Insira um número: "); //Exibe a mensagem
    NumeroDigitado = Convert.ToInt32(Console.ReadLine());
    //Lê e converte para int o número informado pelo usuário
    if (NumeroDigitado % 2 == 0)
    {
        //Número Par
        Console.WriteLine("Par");
    }
    else
    {
        //Número Impar
        Console.WriteLine("Impar");
    }
    Console.ReadKey();
}
```


Tratamento de Exceções

Sintaxe do tratamento de exceções

```
try {  
    // código factível de erro  
}  
catch (NullPointerException ex) {  
    // trata exceções de referência nula  
}  
catch {  
    // trata outras exceções  
}  
finally {  
    // executa sempre  
}
```

Arrays (vetores)

- **Estrutura de dados que contém um número certo de variáveis (elementos do array)**
- **Todos os elementos do array tem que ser do mesmo tipo**
- **Arrays são indexados a partir de zero (0)**
- **Arrays são objetos**
- **Arrays podem ser:**
 - **Unidimensionais: um array de ordem um**
 - **Multidimensionais: um array de ordem maior que um**
 - **Jagged: um array cujos elementos são arrays**
- **Arrays possuem métodos específicos para manipulação dos seus itens**

Arrays (vetores)

- **Para declarar um Array, basta adicionar um par de colchetes logo após a declaração do tipo dos elementos individuais**

```
private int[] meuVetorDeInteiros;  
public string[] meuVetorDeStrings;
```

- **Devem ser instanciados**

```
int[] codigos = new int[5];  
string[] nomes = new string[100];
```

- **Podemos criar um Array de Objetos**

```
object[] produtos = new object[50];
```

Arrays (vetores)

- Inicializando um array

```
int[] pedidos = {1, 4, 6, 8, 10, 68, 90, 98, 182, 500};
```

- Acessando membros do array

```
Cliente[] clientes = {  
    new Cliente("Rodrigo"),  
    new Cliente("Eduardo") };  
clientes[0].Idade = 20;  
clientes[1] = new Cliente("Marcelo");  
  
Console.WriteLine("{0} e {1}", clientes[0].Nome, clientes[1].Nome);
```

Arrays (vetores)

- Utilizando a instrução **foreach** para percorrer todos os itens de um Array

```
Cliente[] clientes = {  
    new Cliente("Rodrigo"),  
    new Cliente("Eduardo"),  
    new Cliente("Marcelo") };  
  
foreach (Cliente clienteDaVez in clientes)  
{  
    Console.WriteLine("Cliente {0}", clienteDaVez.Nome);  
}  
...  
Cliente Rodrigo  
Cliente Eduardo  
Cliente Marcelo
```

Vetor

Vetor é uma estrutura de dados linear unidimensional que armazena diversos valores de um mesmo tipo (estrutura homogênea).

Podem ser do tipo inteiro, real, caractere/ literal ou lógico.

Ex

VOGAIS :

0	1	2	3	4
A	E	I	O	U

NOTAS:

0	1	2	3	4	5	6	7	8	9
8.5	7.5	9.2	5.5	8.9	8.6	7.6	9.3	5.6	8.10

Vetor

O Vetor só pode ser acessado por sua posição(índice);

Sua alocação pode é estática e sequencial

OBS: uma vez alocado, o tamanho do vetor é fixado

Ex:

- vetor [5] NUMÉRICO ou;
- int vetor[10];

Atribuição

vetor[2] ← 10

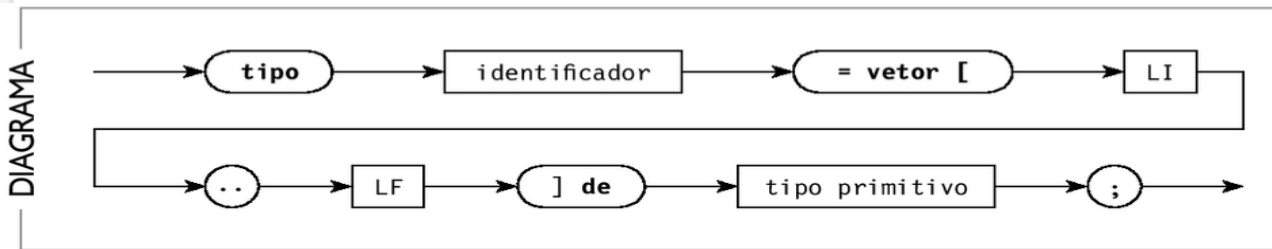
vetor[4] ← 5

0	1	2	3	4
		10		5

Vetor - Declaração

Existem várias formas de e declarar um Vetor, abaixo segue algumas delas:

Em portugol –Visual G



Declaração : <nome>:vetor[LI..LF] de <tipo_dado>

LI: representa o limite inicial do vetor

LF: representa o limite final do vetor

Tipo de dado: representa qualquer um dos tipos básicos de dados.

Vetor - Exemplo de declaração portugol

```
VAR IDADE    : vetor[1..20]  de integer  
      NOME     : vetor[1..30] de literal  
      NOTA     : vetor[1..4]  de real
```

Vetor - Exemplo de declaração portugol

Escreva um algoritmo que solicite ao usuário a entrada de 5 números, e que exiba o somatório desses números na tela.

Após exibir a soma, o programa deve mostrar também os números que o usuário digitou, um por linha.

Em portugol

```
ALGORITMO_SOMA_VETOR
Var
numeros : vetor [0..4] de inteiro
cont, soma : inteiro
INICIO
    PARA cont de 0 ate 4 faca
        | ESCREVA("Entre com um número:")
        | leia(numeros[cont])
        | soma <- soma + numeros[cont]
    FIM_PARA
    ESCREVA("A soma dos números é", soma)
    ESCREVA("Os números digitados foram:")
    PARA cont de 0 ate 4 faca
        | ESCREVA(numeros[cont])
    FIM_PARA
FIM_ALGORITMO
```

Em C#

```
static void Main(string[] args)
{
    int[] numero = new int[5];
    int soma = 0;
    for(int i = 0; i <= 4; i++)
    {
        Console.WriteLine("Entre com um número:");
        numero[i] =int.Parse(Console.ReadLine());
        soma = soma + numero[i];
    }
    Console.WriteLine("A soma dos números armazenados no vetor é {0}", soma);
    Console.WriteLine("");
    Console.WriteLine("Os número digitados foram:");

    for (int i = 0; i <= 4; i++)
    {
        Console.WriteLine(numero[i]);
    }
}
```

Vetor – Exemplo

Fazer um algoritmo que armazena 5 numeros. Estes números deverão ser multiplicados por 10. Ao final deverá ser mostrado os valores armazenados

Algoritmo vetor_exemplo

var

i: inteiro

MD: vetor[0..5] de real

Inicio

para i de 0 até 5 faça

MD[i] \leftarrow i * 10

fimpara

para i de 0 até 5 faça

Escreval ("O valor do armazenado no vetor MD é " + i + MD[i])

fimpara

Fimalgoritmo

A saída será 0, 10, 20, 30, 40, 50

Sendo armazenado no vetor MD os seguintes valores

0	1	2	3	4	5	Índice do vetor
0	10	20	30	40	50	Valor armazenado

Vetor - Exemplo

```
static void Main(string[] args)
{
    int[] numero = new int[6];
    for(int i = 0; i <= 5; i++)
    {
        numero[i] = i*10;
    }
    for (int i = 0; i <= 5; i++)
    {
        Console.WriteLine("Os valores armazenados no vetor  
número é: {0} ", numero[i]);
    }
}
```

0	1	2	3	4	5	Índice do vetor
0	10	20	30	40	50	Valor armazenado

Vetor - Exemplo de declaração portugol

EX: Dado o algoritmo abaixo qual será a saída?

Sabendo que o vetor MD é disposto conforme tabela a seguir:

MD	0	1	2	3	4	5	6	7	8	9
	8.5	1.5	2.5	3.0	5.0	1.0	7.6	9.3	5.6	8.10

algoritmo media_geral_vetor

var

i: inteiro

MD: vetor[1..8] de real

soma, media:real

inicio

soma \leftarrow 0

para i de 1 até 5 faça

leia MD[i]

soma \leftarrow soma + MD[i]

fimpara

media \leftarrow soma/5

escreva media

fimalgoritmo

Vetor

Vantagens:

- **Simplicidade;**
- **Acesso direto;**

Desvantagens

- **Tamanho fixo**

Vetor/Array em C#

- Vetores são também chamados de arranjos unidimensionais
- Em C# a primeira posição de um vetor é a posição 0.
- Um arranjo deve ser alocado previamente, antes de ser utilizado.
- Uma vez alocado, sua quantidade de elementos é fixa.

Vetor/Array em C#

Como criar um vetor?

declaração



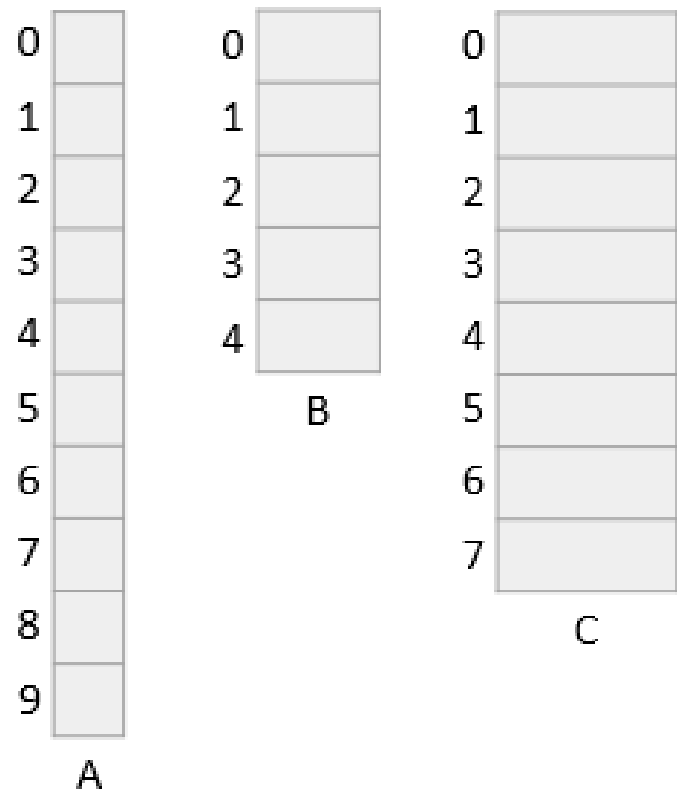
```
int[] A;  
double[] B;  
string[] C;
```

instanciação



```
A = new int[10];  
B = new double[5];  
C = new string[8];
```

Memória RAM



Vetor/Array em C#

Outra forma de criar um vetor (array) em C# é fazer a declaração e instaciação ao mesmo tempo

Ex

```
int[] numeros = new int[10];
```

Vetor números do tipo inteiro com 10 posições.

Vetor/Array em C#

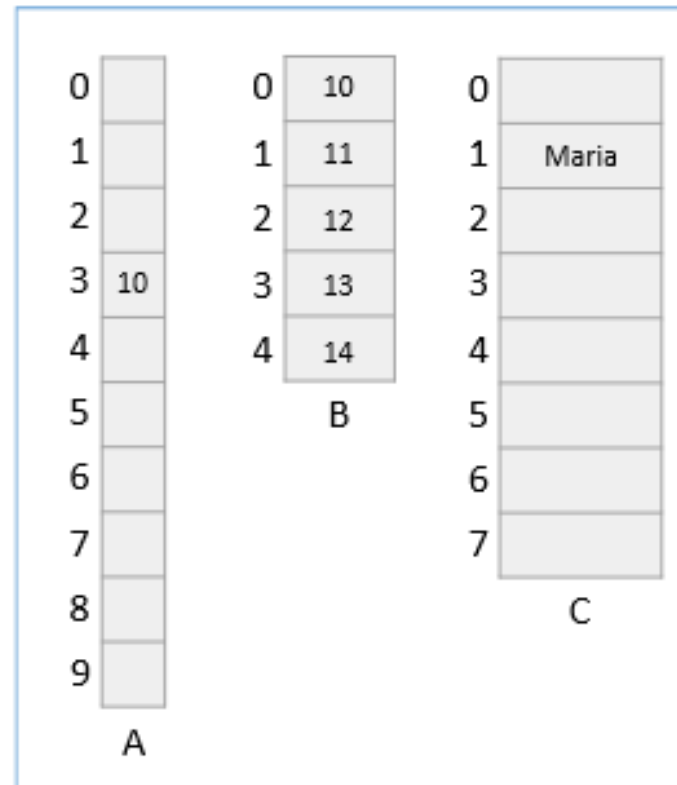
Como acessar os elementos de um vetor?

```
A[3] = 10;
```

```
for (int i=0; i<5; i++) {  
    B[i] = i + 10;  
}
```

```
C[1] = "Maria";
```

Memória RAM



Vetor/Array em C# Exemplos

```
int[] inteiros = new int[5];  
inteiros[0] = 1;  
inteiros[1] = 2;  
inteiros[2] = 3;  
inteiros[3] = 4;  
inteiros[4] = 5;
```

Ou de uma forma mais elegante

```
int[] umAoCinco = new int[] { 1, 2, 3, 4, 5 };
```

Vetor/Array em C#

Exemplos 1

```
using System;
namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] cores = new string[3];
            cores[0] = "Azul";
            cores[1] = "Vermelho";
            cores[2] = "Verde";
            Console.WriteLine(cores[0]);
            Console.WriteLine(cores[1]);
            Console.WriteLine(cores[2]);

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

Vetor/Array em C#

Exemplos 2

```
using System;
namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] cores = new string[3];
            cores[0] = "Azul";
            cores[1] = "Vermelho";
            cores[2] = "Verde";
            //Console.WriteLine(cores[0]);
            //Console.WriteLine(cores[1]);
            //Console.WriteLine(cores[2]);
            foreach (string cor in cores)
            {
                Console.WriteLine(cor);
            }
            Console.ReadKey();
        }
    }
}
```

Vetor/Array em C#

Exemplos

```
using System;
namespace ConsoleApp1
{
    class Program
    {
        static void Main(string[] args)
        {
            string[] nome = new string[3];
            int[] idade = new int[3];
            char[] sexo = new char[3];
            // ler o vetor de 150 aluno
            for (int i = 0; i < 3; i++)
            {
                Console.Write("Digite o seu nome: ");
                nome[i] = Console.ReadLine();
                Console.Write("Digite a sua idade: ");
                idade[i] = Int32.Parse(Console.ReadLine());
                Console.Write("Digite o seu sexo: ");
                sexo[i] = char.Parse(Console.ReadLine());
            }

            for (int i = 0; i < 3; i++)
            {
                Console.WriteLine("O seu nome e: {0} ", nome[i]);
                Console.WriteLine("O seu idade e: {0} ", idade[i]);
                Console.WriteLine("O seu sexo e: {0} ", sexo[i]);
            }

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

Vetor/Array em C#

Exercício

- 1 - Faça um algoritmo que leia 10 salários. Depois de lidos e armazenados, mostre o maior valor. Utilize vetores.**
- 2 - Armazene num vetor de 5 posições o salário de 5 pessoas. Se o salário for menor que 1000 reais, forneça um aumento de 10% e sobrescreva o valor antigo. Ao final, mostre a lista de salários atualizada.**
- 3 - Faça um programa q leia 5 valores reais. Armazene estes valores num vetor. Ao final, imprima a média aritmética destes valores.**

Vetor/Array em C#

Exercício

4 - Crie um programa que armazene 10 números digitados pelo usuário em dois vetores: um somente para números pares, e outro somente para números ímpares. Após, exiba os valores dos dois vetores na tela, em sequência.

Obs.: As posições que não receberem valores exibirão o número zero. Não se preocupe com isso por enquanto.

Correção dos Exercícios

Exercício 1

```
class Program
{
    static void Main(string[] args)
    {
        double[] salario = new double[10];
        double maior = 0.0;
        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine("Entre com o {0}º valor do salário",i+1);
            salario[i] = double.Parse(Console.ReadLine());

            if (salario[i] > maior)
            {
                maior = salario[i];
            }
        }
        Console.WriteLine("Os salários armazenados são :");
        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine(salario[i]);
        }
        Console.WriteLine("O maior salário armazenado é R$ {0} ", maior);
    }
}
```

Correção dos Exercícios

Exercício 2

```
static void Main(string[] args)
{
    double[] salario = new double[5];
    for (int i = 0; i < 5; i++)
    {
        Console.WriteLine("Entre com o valor do salário");
        salario[i] = double.Parse(Console.ReadLine());
        if (salario[i] < 1000.00)
        {
            Console.WriteLine("Os salários {0} será reajustado em 10%", salario[i]);
            salario[i] = salario[i] + (salario[i] * 10) / 100;
        }
    }
    Console.WriteLine("Os salários atualizados armazenados são :");
    for (int i = 0; i < 5; i++)
    {
        Console.WriteLine(salario[i]);
    }
    Console.ReadKey();
}
```

Correção dos Exercícios

Exercício 3

```
class Program
{
    static void Main(string[] args)
    {
        double[] salario = new double[5];
        double soma = 0.0;
        for (int i = 0; i < 5; i++)
        {
            Console.WriteLine("Entre com o valor {0}º do salário", i+1);
            salario[i] = double.Parse(Console.ReadLine());
            soma = soma + salario[i];
        }
        Console.WriteLine("Os salários armazenados são :");
        for (int i = 0; i < 5; i++)
        {
            Console.WriteLine(salario[i]);
        }
        Console.WriteLine("A média dos salários armazenado é R$ {0} ", soma/5);
    }
}
```

Correção dos Exercícios

Exercício 4

```
class Program
{
    static void Main(string[] args)
    {
        int[] numeroPar = new int[10];
        int[] numeroImpar = new int[10];
        int[] numero = new int[10];
        for (int i = 0; i < 10; i++)
        {
            Console.WriteLine("Entre com o {0}º número", i + 1);
            numero[i] = int.Parse(Console.ReadLine());
        }
        for (int i = 0; i < 10; i++)
        {
            if (numero[i] % 2 == 0)
            {
                numeroPar[i] = numero[i];
            }
            else
            {
                numeroImpar[i] = numero[i];
            }
        }
        Console.WriteLine("Os números pares digitados são:");
        for (int i = 0; i < 10; i++)
        {
            if (numeroPar[i] != 0)
            {
                Console.WriteLine(numeroPar[i]);
            }
        }
        Console.WriteLine("Os números ímpares digitados são:");
        for (int i = 0; i < 10; i++)
        {
            if (numeroImpar[i] != 0)
            {
                Console.WriteLine(numeroImpar[i]);
            }
        }
    }
}
```

ArrayLists

- **ArrayLists não tem tamanho definido**
- **Use o método Add(object) para adicionar um elemento ao fim do ArrayList**
- **Use os colchetes para acessar os elementos do ArrayList**
- **Está localizado no Namespace System.Collections**
- **Use o método Clear() para remover todos os elementos do array**
- **Uso recomendado para manipulação de objetos em Programação Orientada a Objetos**

ArrayLists

- Um ArrayList herda classe ArrayList que implementa a interface IList usando um array cujo tamanho é aumentando dinamicamente quando requerido.
- A propriedade **Count** fornece o número total de elementos atualmente armazenados no ArrayList.
- A propriedade **Capacity** obtém ou define o número de elementos que um ArrayList que pode conter.
- O método **Contains** determina se um elemento existe em um ArrayList;
- O método **Clear** remove todos os elementos de um ArrayList;
- O método **Insert** insere um elemento em um índice especificado em um ArrayList;
- Os objetos são adicionados ao final de um ArrayList usando o método **Add()** e removidos o método **Remove()**.
- O método **RemoveAt()** remove um elemento do ArrayList em um índice especificado.

Exemplo ArrayLists

```
using System;
using System.Collections;

namespace exemploArrayList
{
    class Program
    {
        static void Main(string[] args)
        {
            System.Collections.ArrayList teste = new ArrayList();
            teste.Add("Tópicos Avançados em Programação!");
            teste.Add("14/09/2020");
            teste.Add(2020);           //int
            teste.Add("xxxxx");       //string
            teste.Add(null);          //null
            teste.Add(true);          //boolean
            teste.Add("#####");      //string
            teste.Add(12.25);          //double
            teste.Add(18.45f);         //float

            Console.WriteLine("ArrayList teste");
            Console.WriteLine("capacidade = " + teste.Capacity);
            Console.WriteLine("tamanho   = " + teste.Count);
            Console.WriteLine("removendo o elemento xxxxx");
            teste.Remove("xxxxx");
            Console.WriteLine("inserindo um novo elemento : o fim");
            teste.Add("the end");
```

```
            Console.WriteLine("Percorrendo o ArrayList ");
            for (int indice = 0; indice < teste.Count; indice++)
            {
                Console.WriteLine("indice {0} - {1} ", indice, teste[indice]);
            }
            Console.WriteLine("Limpendo o teste ");

            teste.Clear();
            if (teste.Count == 0)
            {
                Console.WriteLine(" O teste esta vazio ....");
            }
            Console.ReadKey();
        }
    }
}
```

ArrayLists

- Transformando um Array em um ArrayList
- Transformando um ArrayList em um Array

```
Cliente[] clientes = empresa.RecuperaClientes();
```

```
//Convertendo o Array em ArrayList
```

```
ArrayList clientesAux = new ArrayList(clientes);  
clientesAux.Add(new Cliente("Rodrigo", 11));
```

```
//Convertendo o ArrayList em Array
```

```
clientes = new Clientes[clientesAux.Count];  
clientesAux.CopyTo(clientes);
```

```
...
```

```
Console.WriteLine(clientes[3].Nome);
```

```
...
```

```
Rodrigo
```


Indexadores

- Para criar indexadores utilize a palavra-chave **this** e as instruções **get** e **set**

```
public class CarroCompras
{
    private Produto[] produtosSelecionados;
    public CarroCompras()
    {
        produtosSelecionados = new Produto[10];
    }
    public Produto this[int i]
    {
        get { return produtosSelecionados[i]; }
        set { produtosSelecionados[i] = value; }
    }
}
```

Indexadores

- Consumindo indexadores

```
CarroCompras carro = new CarroCompras();
```

```
carro[0] = new Produto("Televisor");
```

```
carro[1] = new Produto("Geladeira");
```

```
carro[2] = new Produto("Computador");
```

```
carro[3] = new Produto("Microondas");
```

Eventos e Delegates

- **Conceitos:**
 - **Evento:** ação que pode ser gerenciada/manipulada através de código
 - **Delegate:** membro da classe responsável por “delegar” as ações correspondentes a ocorrência de um evento ao(s) manipulador(es) de eventos correspondentes
 - **Manipulador de Evento:** método responsável pela execução de ações em reação a ocorrência de um evento

Enumeradores

- Definindo Tipos Enumeradores

```
enum TipoDiretor  
{  
    Marketing,  
    RH,  
    Comercial,  
    Financeiro  
}
```

- Usando Tipos Enumeradores

```
TipoDiretor tpDiretor = TipoDiretor.Comercial;
```

- Mostrando Variáveis

```
Console.WriteLine("{0}", tpDiretor); //Mostra Comercial
```

Conversão de operadores

- **Conversão implícita**
 - Sempre possível, sem perda de precisão
 - Ex.: `long = int`
- **Conversão explícita**
 - É necessária uma verificação em tempo de execução (cast)
 - Ex.: `int = (int) long;`

Conversão de operadores

```
public static implicit operator Conta (int x) { return new Conta(x); }
```

```
public static explicit operator int (Conta c) { return c.saldo; }
```

```
Conta c = 3; //conversão implícita
```

```
int x = (int) c; //conversão explícita
```

Referências:

<https://owasp.org/www-community/attacks/>

<https://www.dio.me/articles/a-evolucao-da-arquitetura-de-software-backend>

<https://4future.com.br/index.php/2023/12/04/compreendendo-os-principios-solid/>

<https://medium.com/beelabacademy/princ%C3%ADpios-de-s-o-l-i-d-em-c-guia-pr%C3%A1tico-cbb1e6584284>

[PIRES, E. SOLID: Teoria e Prática.](#)

ANICHE, M. Orientação a Objetos e SOLID para ninjas. São Paulo – Casa do Código