



C#

👉 Bibliotecas

👉 .Net

👉 C#

👉 Entendendo estrutura de um projeto c#

👉 Módulos ou pacotes:

🔗 Apagar módulo

🔗 Importar módulo

👉 Onde o programa começa

🔗 Códigos

🔗 Variáveis

🔗 Tipos de Variáveis:

🔗 Alterar valor da Variáveis

🔗 Outras formas de declarar variáveis

🔗 Constantes

🔗 Capturando entrada do usuário

🔗 Operadores aritméticos

🔗 Condicionais - IF (se)

🔗 ELSE (Se não)

🔗 ELSE IF

🔗 Operadores lógicos

🔗 Operadores lógicos PRÁTICA (PARSE)

🔗 Funções

👉 Bibliotecas 🔗

São módulos/pacotes que estendem a linguagem de programação.

Por exemplo existem bibliotecas que permitem manipular placa de vídeo do computador, manipular arquivos do sistema operacional, criar aplicações que se comunicam com a internet, manipular imagem, áudio, vídeo e etc...

E cada linguagem tem uma biblioteca diferente.

👉 .Net 🔗

A Microsoft criou uma plataforma (conjunto de bibliotecas) o .Net

Ambiente de execução, permite que varias linguagens podem usar o .Net

C# roda em cima do .Net

Linguagens muito parecidas: F# e [VB.net](#) (VisualBasic) também rodam no mesmo ambiente e nas mesmas bibliotecas

Visual Studio - Ambiente de desenvolvimento integrado, ferramenta para criar as aplicações

Visual studio code - focado mais para desenvolvimento web

👉 C# 🔗

Também roda em outras plataformas (ambiente):

.Net Framework - roda apenas em Windows, usada para criar aplicação Web, desktop, jogos...

.Net Core - é multiplataforma (Windows, Linux, Mac), usada para desenvolver aplicações Web

Xamarin (zamarin) - roda no Iphone, Android e Ios, usada para desenvolver aplicativos

i O que é possível criar com C#?

Tudo!

👉 Entendendo estrutura de um projeto c# 🔗

Dentro do Visual Studio irá abrir uma estrutura que se chama **Program.cs**, os arquivos da linguagem C# tem o arquivo com extensão .cs

Program.cs é a estrutura principal

```
Program.cs
Projeto01
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
6
7 namespace Projeto01
8 {
9     0 referências
10    internal class Program
11    {
12        0 referências
13        static void Main(string[] args)
14        {
15        }
16    }
```

🔧 Módulos ou pacotes: 🔗

```
1 using System;
2 using System.Collections.Generic;
3 using System.Linq;
4 using System.Text;
5 using System.Threading.Tasks;
```

C# é dividido em vários módulos, cada conjunto de funcionalidade está dentro de um módulo, isso reduz o tamanho do programa deixando ele mais leve, inserindo somente o que for usar.

1. using System; = serve para que o C# consiga manipular coisas dentro do sistema operacional do usuário
2. using System.Collections.Generic;
3. using System.Linq; = para trabalhar com um conjunto grande de dados
4. using System.Text; = módulo para manipular texto
5. using System.Threading.Tasks; = serve basicamente para que o programa rode em vários núcleos do processador

🔧 Apagar módulo 🔗

Excluir as linhas que estão no módulo

🔧 Importar módulo 🔗

using nome_do_modulo;

🔧 Onde o programa começa 🔗

```
7 namespace Projeto01
8 {
9     0 referências
10    internal class Program
11    {
12        0 referências
13        static void Main(string[] args)
14        {
15        }
16    }
```

Essas linhas de código são mais focadas em modelagem a objetos (POO)

```
1 namespace Projeto01
2 {
3     internal class Program
```

Início e fim do programa, onde será criado o programa.

```
7 namespace Projeto01
8 {
9     0 referências
10    internal class Program → Início do programa
11    {
12        0 referências
13        static void Main(string[] args)
14        {
15        }
16    } → Fim do programa
```

Função principal Main será o primeiro código a ser executado no programa

Tudo que está dentro do Main é executado assim que o programa é aberto

```
1 static void Main(string[] args)
2 {
3 }
```

🔧 Códigos 🔗

Exibe uma mensagem de texto na tela

```

7  namespace Projeto01
8  {
9      0 referências
10     internal class Program
11     {
12         0 referências
13         static void Main(string[] args)
14         {
15             Console.WriteLine("Hello World!");
16             Console.ReadLine();
17         }
18     }
19 }

```

Readline = espera o usuário pressionar enter para fechar

\n = quebra uma linha

WhiteLine = comando que quebra linha sem precisar adicionar \n em toda final de frase

🔧 Variáveis

São locais na memória RAM do computador onde consigo guardar informações onde posso recuperar essas informações em qualquer momento.

var nome_da_variavel = seu_valor;

Declarando variáveis

🔧 Tipos de Variáveis:

Int = números inteiros (156 ou 4352462)

Float = decimais, entre vírgulas

Bool = true or false (só pode ter esses dois valores)

string = dados do tipo texto entre aspas duplas ""

char = caracteres entre aspas simples "

definir o tipo

nome = não pode começar com numero, de A-Z, numeros de 0 -9, não pode ter número no início, pode ter o caractere _ e não pode ter variável reservada do C#

valor

Não pode ter variáveis com o mesmo nome no mesmo lugar

Em int não utiliza aspas

Em string utiliza aspas duplas

Em float deve colocar f no final do valor adicionado

Em bool true or false não utiliza aspas

```
1 int Segunda_Guerra_Mundial = 1942;
```

```
1 string cor_favorita = "Vermelho";
```

```
1 float velocidade_f1 = 294.48f;
```

```
1 bool Segunda_Guerra_Mundial_Aconteceu = false;
```

Para exibir essas variáveis na tela usa o código:

```
1 Console.WriteLine (nome_da_variavel);
```

```

namespace Projeto01
{
    0 referências
    internal class Program
    {
        0 referências
        static void Main(string[] args)
        {
            int Segunda_Guerra_Mundial = 1942;
            string cor_favorita = "Vermelho";
            float velocidade_f1 = 294.48f;
            bool Segunda_Guerra_Mundial_Aconteceu = false;

            Console.WriteLine(Segunda_Guerra_Mundial);
            Console.WriteLine(cor_favorita);
            Console.WriteLine(velocidade_f1);
            Console.WriteLine(Segunda_Guerra_Mundial_Aconteceu);

            Console.ReadLine();
        }
    }
}

```

🔧 Alterar valor da Variáveis

= é atribuição é o recebe

nome_da_variavel = valor_nome;

não precisa colocar o tipo na frente, se não via entender que estou criando uma nova variável

```
1 velocidade_f1 = 300.21f;
```

```
1 cor_favorita = "Roxo";
```

```
namespace Projeto01
{
    namespace
    {
        internal class Program
        {
            [STAThread]
            static void Main(string[] args)
            {
                int Segunda_Guerra_Mundial = 1942;
                string cor_favorita = "Rosa";
                float velocidade_f1 = 294.48f;
                bool Segunda_Guerra_Mundial_Aconteceu = false;

                Console.WriteLine(Segunda_Guerra_Mundial);
                Console.WriteLine(cor_favorita);
                Console.WriteLine(velocidade_f1);
                Console.WriteLine(Segunda_Guerra_Mundial_Aconteceu);

                velocidade_f1 = 389.21f;
                cor_favorita = "Roxo";

                Console.WriteLine(velocidade_f1);
                Console.WriteLine(cor_favorita);

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

```
C:\Users\Alves\source\repos\Projeto01\Projeto01\bin\Debug\Projeto01.exe
1942
Rosa
294.48
False
389.21
Roxo
```

🔗 Outras formas de declarar variáveis

// = são comentários

/* */ = comentários em multiplas linhas

Tipo simples de declarar:

Omitindo o tipo de variavel colocar a palavra Var ao invés de colocar o tipo (int, float, string, bool, char)

O var já vai definir o tipo dela automaticamente mas ainda sim deve seguir as regras dos tipos

```
1 var idade = 25;
```

Tipo mais completo:

dynamic

Nele o c# já define o tipo dela sem precisar adicionar e também caso eu queira alterar o valor dela eu posso mudar para outro tipo sem precisar alterar o tipo dela. Mudando por exemplo de int para string.

Ela é dinamica, armazena variaveis de vários tipos, mas ele quebra muito o padrão do C#

```
dynamic idadeIsis = 25;
Console.WriteLine(idadeIsis);

idadeIsis = "Rosa";
Console.WriteLine(idadeIsis);
```

```
C:\Users\Alves\source\repos\Projeto01\Projeto01\bin\Debug\Projeto01.exe
25
Rosa
```

🔗 Constantes

São muito parecidas com as variáveis

Ao definir uma constante posso definir o valor dela e nunca mais posso mudar durante a execução do programa, essa é a diferença.

Questão de segurança evitando acidente como mudar o valor de uma variavel sem querer

Definir nome + tipo + nome_da_variável = valor

```
1 const float pi = 3.14f;
```

🔗 Capturando entrada do usuário

Console.WriteLine = exibe a mensagem na tela do usuario

Console.ReadLine = captura o valor e atribui a uma variável

```
Console.WriteLine("Escreva seu nome: ");
string nome = Console.ReadLine();
Console.ReadLine();
```

outro exemplo:

```
string nome = "";
Console.WriteLine("Escreva seu nome: ");
nome = Console.ReadLine();

Console.ReadLine();
```

Com mais informações do usuario

```

Console.WriteLine("Escreva seu nome: ");
string nome = Console.ReadLine();
Console.WriteLine("Sua idade é: ");
string idade = Console.ReadLine();

Console.ReadLine();

```

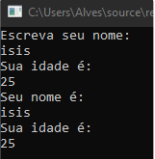
Exibindo a mensagem na tela do usuário

```

Console.WriteLine("Escreva seu nome: ");
string nome = Console.ReadLine();
Console.WriteLine("Sua idade é: ");
string idade = Console.ReadLine();
Console.WriteLine("Seu nome é: ");
Console.WriteLine(nome);
Console.WriteLine("Sua idade é: ");
Console.WriteLine(idade);

Console.ReadLine();

```



```

C:\Users\Alves\source\int...
Escreva seu nome:
isis
Sua idade é:
25
Seu nome é:
isis
Sua idade é:
25

```

🛠 Operadores aritméticos [↗](#)

Criar a variável tipo int ou float + nome_variavel = operação

Console.WriteLine(nome_davariavel); = atribui a variável que será exibida

Console.ReadLine (); = Para exibir o resultado na tela

C# sempre segue a regra da ordem dos operadores: multiplicação, divisão, sub, adi

Caso eu declarar a variável com int e o resultado for float o programa vai cortar o valor decimal

```

int numerosoma = 10 + 10 + 30;
int numeromult = 10 * 50;
int numerodiv = 6 / 3;

Console.WriteLine(numerosoma);
Console.WriteLine(numeromult);
Console.WriteLine(numerodiv);

Console.ReadLine();

```

🛠 Condicionais - IF (se) [↗](#)

Condições são expressões que podem ter valor de resultado verdadeiro ou falso

10 > 2 = True

2 >= 250 = False

var nome = "Lima"

var == "Victor" = False

Operadores padrões:

> maior que

< menor que

>=

<=

== igualdade, comparação

!= diferente

Captura informação de acordo com o resultado True or False terá uma ação

IF (colocar qqrlr um dos operadores padrão) {

aqui irá exibir o que for verdadeiro

}

```
if (10 > 2 ) {
    Console.WriteLine("É verdade!");
}

Console.ReadLine();
```

```
C:\Users\Alves\source
verdade!
```

```
if (10 > 200 ) {
    Console.WriteLine("É verdade!");
}

Console.ReadLine();
```

Essa condição é false então não foi exibido nada na tela do usuário

🐞 ELSE (Se não) [🔗](#)

Caso seja falso vai exibir tal ação

Sempre deverá ser usado com o If

```
if (10 > 200)
{
    Console.WriteLine("É verdade!");
}
else {
    Console.WriteLine("É mentira!");
}

Console.ReadLine();
```

```
C:\Users\Alves\source
É mentira!
```

🐞 ELSE IF [🔗](#)

Deverá sempre antes do Else

Pode colocar quantos desejar

Caso o primeiro seja falso vai cair no else if

else if sendo verdadeiro vai executar o comando do else if

caso fosse falso ia cair para o else

Caso o primeiro(if) seja verdadeiro ele ignora todo resto

O primeiro(if) sendo false e o segundo(else if) é true ele executa o segundo(else if)

O primeiro(if) e o segundo(else if) sendo false ele executa o ultimo (else)

```
if (10 > 200)
{
    Console.WriteLine("É verdade!");
}
else if (20 == 20) {
    Console.WriteLine("Executou Elseif!");
}
else {
    Console.WriteLine("É mentira!");
}

Console.ReadLine();
```

if false, elseif true

```
C:\Users\Alves\source
Executou Elseif!
```

```

if (10 > 200)
{
    Console.WriteLine("É verdade!");
}
else if (20 != 20) {
    Console.WriteLine("Executou Elseif!");
}
else {
    Console.WriteLine("É mentira!");
}

Console.ReadLine();

```

if false, else if false, executou else

```

C:\Users\Alves>
É mentira!

```

Dá para trabalhar com as variáveis ao invés diretamente com os números

```

int a = 10;
int b = 20;
int c = 2;

if (a < b) // SE
{
    Console.WriteLine("É verdade!");
}
else if (a > c)
{
    Console.WriteLine("Executou Elseif!");
}
else // SE NÃO
{
    Console.WriteLine("É mentira!");
}

Console.ReadLine();

```

Operadores lógicos

É uma forma de colocar duas ou mais condições dentro de um condicional

&& = e/and

|| = ou/or

```

// &&(E ou AND) e ||(OU/OR)
// PARA TER ENTRADA GRATUITA:
// SERMULHER && IDADE >= 25 -> Entra na festa de graça

```

Nesse caso trabalhou com o and &&, as duas condições precisam ser verdadeiras para que o resultado seja TRUE. Se caso uma seja false então o resultado é FALSE.

TRUE && TRUE = TRUE

TRUE && FALSE = FALSE

FALSE && TRUE = FALSE

FALSE && FALSE = FALSE

|| só precisa que uma das condições seja verdadeira para ser TRUE

```

// &&(E ou AND) e ||(OU/OR)
// Entrar no jogo gratuitamente(TRUE)
// LEVARALIMENTO || LEVARBRINQUEDO => Entrar no jogo

```

TRUE || TRUE = TRUE

TRUE || FALSE = TRUE

FALSE || TRUE = TRUE

FALSE || FALSE = FALSE

```

int a = 10;
int b = 20;
int c = 2;

if (a < b && a > c)
{
    Console.WriteLine("É verdade!");
}
else if (20 != 20) {
    Console.WriteLine("Executou Elseif!");
}
else {
    Console.WriteLine("É mentira!");
}

Console.ReadLine();

```

TRUE (executa o primeiro e ignora os demais)

```

int a = 10;
int b = 20;
int c = 200;

if (a < b && a > c)
{
    Console.WriteLine("É verdade!");
}
else if (20 != 20) {
    Console.WriteLine("Executou Elseif!");
}
else {
    Console.WriteLine("É mentira!");
}

Console.ReadLine();

```

FALSE (o primeiro e o segundo foram false, executou o terceiro)

```

int a = 10;
int b = 20;
int c = 200;

if (a < b || a > c)
{
    Console.WriteLine("É verdade!");
}
else if (20 != 20) {
    Console.WriteLine("Executou Elseif!");
}
else {
    Console.WriteLine("É mentira!");
}

Console.ReadLine();

```

TRUE (uma condição foi verdadeira então executou o primeiro e ignorou os demais)

🔗 Operadores lógicos PRÁTICA (PARSE) [🔗](#)

No C# para ele toda informação que o usuário digita no CONSOLE para ele é uma string, uma informação textual, mesmo ele digitando um número ele vai ser tratado como uma string.

Para converter isso:

converter um texto para número no CONSOLE usa-se o PARSE (converter)

```
1 int idade = Console.ReadLine(); // aqui tenho uma variável que vai receber do usuario a sua idade
```

```
1 int idade = int.Parse(Console.ReadLine()); // aqui com o Parse ele vai converter o valor q o usuario digitar de um valor textual para um valor numérico
```

```

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

if (idade >= 0 && idade <= 11)
{
    Console.WriteLine("Você é uma criança!");
}
else if (idade >= 12 && idade <= 18)
{
    Console.WriteLine("você é um adolescente!");
}
else if (idade >= 19 && idade <= 60)
{
    Console.WriteLine("Você é um adulto!");
}
else {
    Console.WriteLine("Você é um idoso");
}

Console.ReadLine();

```