Curso: Programação Orientada a Objetos com C#
https://www.udemy.com/programacao-orientada-a-objetos-csharp
Prof. Nelio Alves

Notas de aula

Capítulo 1: Revisão de Lógica de Programação

Objetivos do capítulo:

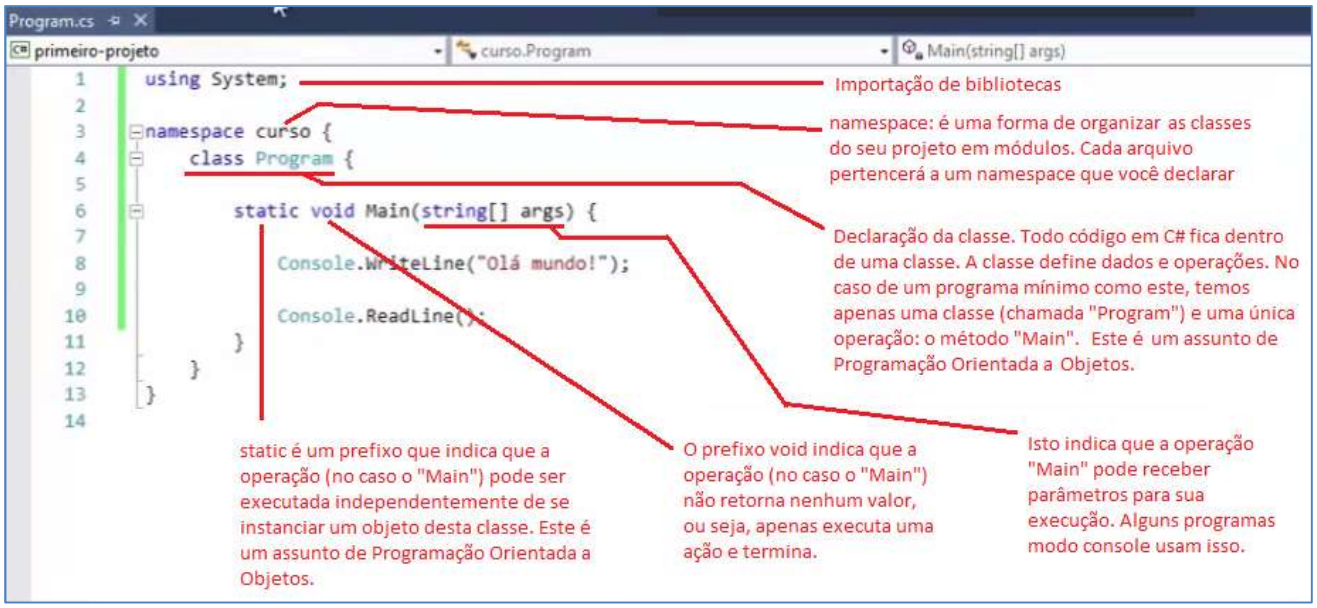
- Apresentar a linguagem C#
- Fazer uma breve revisão de Lógica de Programação usando C#

Video: Instalando o Microsoft Visual Studio

- **Versão instalada para o curso:** Visual Studio Community

- Na aba "Pacotes de idiomas", selecionamos:
  - Inglês
  - Português (Brasil)
- Na aba "Cargas de trabalho", selecionamos:
  - Desenvolvimento da plataforma do Windows Universal
  - Desenvolvimento da área de trabalho do .NET
  - Desenvolvimento para desktop com C++ (opcional)
- Na aba "Componentes individuais", selecionamos:
  - Ferramentas do Visual Studio para Unity (opcional)
  - SDK do Windows 8.1 (opcional - somente se seu Windows for 8.1)

Vídeo: Estrutura e cabeçalhos de um programa C#



Vídeo: Tipos básicos de dados em C#

Mais informações: https://msdn.microsoft.com/en-us/library/cs7y5x0x(v=vs.90).aspx

Tipos Inteiros:

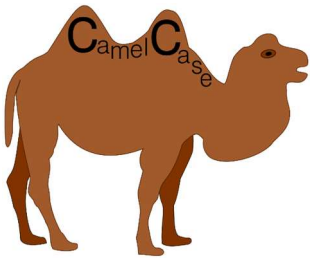
byte	0 .. 255
sbyte	-128 .. 127
short	-32,768 .. 32,767
ushort	0 .. 65,535
int	-2,147,483,648 .. 2,147,483,647
uint	0 .. 4,294,967,295
long	-9,223,372,036,854,775,808 .. 9,223,372,036,854,775,807
ulong	0 .. 18,446,744,073,709,551,615

Outros tipos:

float	-3.402823e38 .. 3.402823e38
double	-1.79769313486232e308 .. 1.79769313486232e308
decimal	-79228162514264337593543950335 .. 79228162514264337593543950335
char	A Unicode character.
string	A string of Unicode characters.
bool	True or False.
object	An object.

Nomes de variáveis

- Não pode começar com dígito: use uma letra ou _
- Não pode ter espaço em branco
- Não usar acentos ou til
- Sugestão: use o padrão "camel case"



Errado:
```
int 5minutos;
int salário;
int salário do funcionario;
```

Correto:
```
int _5minutos;
int salario;
int salarioDoFuncionario;
```

Vídeo: Como fazer saída de dados em C#

Para escrever na tela um texto qualquer

Sem quebra de linha ao final:

```
Console.Write("Olá mundo!");
```

Com quebra de linha ao final:

```
Console.WriteLine("Olá mundo!");
```

Para escrever o conteúdo de uma variável com ponto flutuante

Suponha uma variável tipo **double** declarada e iniciada:

```
double x = 10.35784;
```

```
Console.WriteLine(x);

Console.WriteLine(x.ToString("F2"));
Console.WriteLine(x.ToString("F4"));

Console.WriteLine(x.ToString("F2", CultureInfo.InvariantCulture));
```

Para concatenar vários elementos em um mesmo comando de escrita

Regra geral:

```
elemento1  +  elemento2  +  elemento3  +  ...  + elementoN
```

```
Console.WriteLine("RESULTADO = " + x);

Console.WriteLine("O valor do troco é " + x + " reais");

Console.WriteLine("O valor do troco é " + x.ToString("F2") + " reais");
```

Vídeo: Como fazer entrada de dados em C#

Para ler um texto (até a quebra de linha)

Suponha uma variável tipo **string** declarada:

```
string x;
```

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

Para ler um número inteiro (até a quebra de linha)

Suponha uma variável tipo **int** declarada:

```
int x;
```

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

Para ler um número com ponto flutuante (até a quebra de linha)

Suponha uma variável tipo **double** declarada:

```
double x;
```

```
x = double.Parse(Console.ReadLine());
x = double.Parse(Console.ReadLine(), CultureInfo.InvariantCulture);
```

using System.Globalization;

Para ler vários dados na mesma linha

```
using System;
using System.Globalization;

namespace curso {
    class Program {
        static void Main(string[] args) {

            string x;
            int y;
            double z;

            string[] vet = Console.ReadLine().Split(' ');

            x = vet[0];
            y = int.Parse(vet[1]);
            z = double.Parse(vet[2], CultureInfo.InvariantCulture);
            ...
```

Vídeo: Estrutura condicional (if) em C#

Sintaxe da estrutura condicional simples:

```
if ( condição ) {
    comando1
    comando2
}
```

Sintaxe da estrutura condicional composta:

```
if ( condição ) {
    comando1
    comando2
}
else {
    comando3
    comando4
}
```

Vamos listar a seguir os operadores lógicos, comparativos e aritméticos em C#:

Operadores lógicos:

Operador	Significado
&&	E
||	OU
!	NÃO

Operadores comparativos:

Operador	Significado
>	maior
<	menor
>=	maior ou igual
<=	menor ou igual
==	igual
!=	diferente

Operadores comparativos:

Operador	Significado
+	adição
-	subtração
*	multiplicação
/	divisão
%	resto da divisão

Vídeo: Estrutura repetitiva enquanto (while) em C#

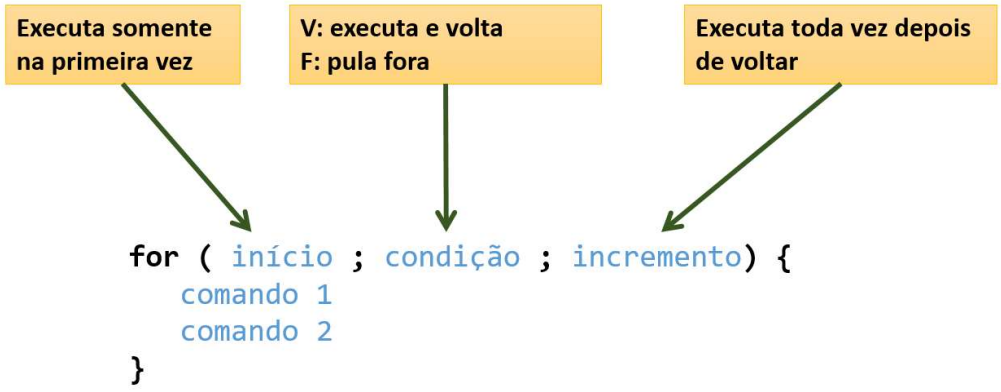
Sintaxe da estrutura repetitiva enquanto:

```
while ( condição ) {
    comando1
    comando2
}
```

Regra:
V: executa e volta
F: pula fora

Vídeo: Estrutura repetitiva para (for) em C#

Sintaxe e regra da estrutura repetitiva para:



Vídeo: Dicas de edição

Autoendentação: CTRL + K + D

Mudar o estilo das chaves:

- Ferramentas -> Opções -> Editor de Texto -> C# -> Estilo de Código -> Formatação -> Novas Linhas
- (desmarque tudo em "Novas opções de linha para chaves")