

# Revisão

Neste módulo aprendemos

- Console, Variáveis, Constantes
- Tipos de dados
- Conversão de Dados
- Operadores (Aritméticos, Atribuição, Comparação e Lógicos)
- Estruturas Condicionais (If e Switch)
- Laços de Repetição (For, While e Do/While)
- Funções e Métodos
- Tipos de Referência e Tipo
- Estruturas
- Enumeradores

## Console

- dotnet new -O <name>
- dotnet build
- dotnet run
- dotnet clean

## Variáveis e constantes

Armazena um espaço na memória do computador p/ armazenar dados.

$\neq$  `int <name> = value;`

Constantes  $\rightarrow$  Armazenam do mesmo jeito que variáveis, mas esse valor não pode ser alterado: `const int <name> = value;`

## Tipos de Dados

Value types  $\rightarrow$  valores que ficam na stack da memória, onde repassa cópias, sendo independentes:

Built-in, structs e enums

$\rightarrow$  bytes, short, int, long, float, double, decimal, bool, string.

Reference types  $\rightarrow$  valores que ficam na parte heap da memória, onde armazena somente a referência a esses dados, garbage collector remove itens não usados. Sendo dependentes.  
Classes, objects e arrays

## Conversão de Dados

Conversão implícitas  $\rightarrow$  conversões onde o próprio c# consegue converter, precisa somente de tipos compatíveis  
`float  $\rightarrow$  int`  $\rightarrow$  é possível

Conversão explícitas  $\rightarrow$  conversões onde precisamos colocar o tipo a ser convertido antes do valor p/ convertê-lo  
ex: `int preco = 15;`  
`float rightPreco = (float) preco;`  $\rightarrow$  converte p/ float

Parse  $\rightarrow$  método que todos built-in types tem, onde convertem string em seu tipo que o chama.  
`int.Parse("100");`

Convert  $\rightarrow$  mas poderoso "podendo" converter p/ qualquer tipo: `Convert.ToInt32()`  
 $\rightarrow$  tipo p/ converter

## Operadores

- Aritméticos  $\rightarrow + - * /$
- Atribuição  $\rightarrow = += -= *= /=$
- Comparação  $\rightarrow > < >= <= !=$

Lógicos  $\rightarrow \& \&\& || !$

## Estrutura condicional

- `if (condicional) { }`
- `else if (condicional) { }`
- `else { }`

$\neq$  `switch (something) { }`  
`case "some": Console.WriteLine(); break;`  
`default: ...; break;`  
`}`

## Laços de Repetição

For

`for (int i = 0; i <= 10; i++)`  
`{`  
`// block`  
`}`

While

`while (condicional)`  
`{`  
`// block com algo p/ quebrar o loop`  
`}`

Do

`{`  
`// block`  
`} while (condicional);`