Revisão

Neste módulo aprendemos
- Console, Variáveis, Constantes
- Tipos de dados
- Conversão de Dados
- Operadores (Aritméticos, Atribuição, Comparação e Lógicos)
- Estruturas Condicionais (If e Sqitch)
- Laços de Repetição (For, While e Do/While)
- Funções e Métodos
- Tipos de Referência e Tipo[redacted]
- 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.
  int <name> = value;
- Constantes → armazenam do mesmo jeito que variáveis, mas esse valor não pode ser alterado: const int <name> = value;

- Tipos de Dados

- Value Types → valores que ficam na stack da memória, onde repassa cópias, sendo independentes:
  Built in, structs e enums
  ↳ Bytes, short, int, long, float, double, decimal, bool, string.

- Reference Types → valores que ficam na parte Heap da memória, onde armazena somente a referência a esses dados, Garbage collector remove items não usados. Sendo dependentes.
  classes, objects e arrays

Conversão de Dados

- Conversão implicitas → Conversões onde o próprio C# consegue converter, precisa somente de tipos compatíveis
  float → int é possível
- Conversão explicitas → Conversões onde precisamos colocar o tipo a ser convertido antes do valor p/ convertermos ele: int preco = 15;
  float rightPreco = (float) preco; converte p/ float
- Parse → método que todos Built in Types tem, onde convertem string em seu tipo que o chama.
  int.Parse("100");
- Convert → mas podemos "rodeando" converter p/ qualquer tipo: Convert.ToInt32();
  ↳ Tipo p/ converter

- Operadores
  - Aritméticos → + - * /
  - Atribuição → = += -= *= /=
  - Comparadores → > < >= <= !=
  - Lógicos → && || !

Estruturas condicionais

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

```
switch(something)
{
  case "some": Console.Write(); break;
  default: ...; break;
}
```

- Laços de Repetição
  - For

```
for (int i=0; idade > i; i++)
{
  // Block
}
```

  - While

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

```
do
{
//Block
} while(condicional);
```

Funções e métodos

- body → public <return type> <Name> (<type Parameters> nameplaceholder)

```
{
    return ....
}
```

↳ Parâmetros opcionais: no final

```
int idade = 30;
```

Structs

- Definição → feita fora da classe:

/Product/

```
struct <name>
{
    // propriedades
    public <type> Name;

    // metodos
    public int Idade();
    {
    }

    // metodo construtor
    public Product(int Idade)
    {
        Idade = idade;
    }
}
```

executado assim que a struct é inicializado
→ mesmo nome do struct
→ não retorna nada

Dentro da classe main

- var product = new Product( )

metodo construtor

Enums

- Fora da classe
- Ajuda na formatação

```
Enum EExemplo
{
    Sorteino = 1,
    SLA = 2,
    oq = 3
}
```