como Func tem tipo de retorno

o lambda precisa ter o mesmo tipo do retorno da Func<T>

- `Func<int, int> cube = (int x) => x * x * x;`
  `Console.WriteLine(cube(3));`

- `(parameters) => expression-or-statement-block`

- `Func<int, int> cube = (int x) => { return x * x; };`

fim do bloco ↲ fim da linha

`Func<int, int> cube = (int x) => x * x;`

↳ melhor método, mais recente

- Alternativa:

↳ `Func<int, int> cube = (int x) => x * x;`

melhor alternativa

`int Cube (int x) => x * x;`

Extension Methods

- Acrescenta métodos a uma classe indiretamente
  - ↳ Não é preciso alterar classes, sobretudo classes base
  - ↳ **this**: representa o tipo de dados sobre o qual estamos a criar o método a criar o método de extensão
    - ↳ cria um método associado (extendendo) ao próprio tipo de dados
      - ↳ melhor que criar métodos utilitários

Nullable

- Tipos primitivos, por omissão, não podem ser null
  - ↳ Value types
    - ↳ `int i = null; // Erro compilador: tipo por valor não pode ser null`
- `Nullable<int> number = null;`
  `int? anotherNumber = null;`
  - ↳ ? "junto" ao tipo de dados (sintaxe)

útil para lidar com bases de dados
↳ pesquisar

- Linq começa sempre pela fonte de dados: equivalente ao FROM do TSQL