



Aula 3 – Coleções e lista de tipos

Uma lista é uma coleção de objetos que pode ser desde uma lista de números inteiros, uma lista de strings ou uma lista de objetos da classe *Personagem*, por exemplo. A sintaxe de criação de uma lista está exemplificada conforme o exemplo abaixo:

É possível realizar diversas operações com lista, como busca, soma, adição de itens, remoção. Aprenderemos a usar aos poucos estas funcionalidades.

1. Crie uma pasta chamada **Aula03Colecoes**, abra o VS Code nesta pasta, exiba o terminal e crie um projeto do tipo console. Utilize o *explorer* para copiar a pasta *Models* do projeto *Aula02RH* para dentro da pasta do projeto atual e confirme que você conseguirá visualizar ela através do VS Code. Altere o namespace para conter *Aula03Colecoes* ao invés de *Aula02RH*
2. Declare uma lista do tipo *Funcionario* de maneira global, ou seja, fora de qualquer método, mas dentro da classe. A palavra *List* necessitará do using de *System.Collections.Generic* e *Funcionario* do using do namespace onde a classe está.

```
using System;
using System.Collections.Generic;
using Aula03Colecoes.Models;
using Aula03Colecoes.Models.Enuns;

namespace Aula03Colecoes
{
    0 references
    class Program
    {
        0 references
        static List<Funcionario> lista = new List<Funcionario>();

        0 references
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World");
        }
    }
}
```



3. crie um método que vai inserir elementos na lista e faça a chamada dele no método principal.

```
static void Main(string[] args)
{
    CriarLista();
}
1 reference
public static void CriarLista()
{
    Funcionario f1 = new Funcionario();
    f1.Id = 1;
    f1.Nome = "Neymar";
    f1.Cpf = "12345678910";
    f1.DataAdmissao = DateTime.Parse("01/01/2000");
    f1.Salario = 100.000M;
    f1.TipoFuncionario = TipoFuncionarioEnum.CLT;
    lista.Add(f1);
}
```

- Adicione mais funcionários a esta lista conforme os objetos abaixo. Não esqueça que os objetos devem estar dentro do método

```
Funcionario f2 = new Funcionario();
f2.Id = 2;
f2.Nome = "Cristiano Ronaldo";
f2.Cpf = "01987654321";
f2.DataAdmissao = DateTime.Parse("30/06/2002");
f2.Salario = 150.000M;
f2.TipoFuncionario = TipoFuncionarioEnum.CLT;
lista.Add(f2);

Funcionario f3 = new Funcionario();
f3.Id = 3;
f3.Nome = "Messi";
f3.Cpf = "135792468";
f3.DataAdmissao = DateTime.Parse("01/11/2003");
f3.Salario = 70.000M;
f3.TipoFuncionario = TipoFuncionarioEnum.Aprendiz;
lista.Add(f3);
```



```
Funcionario f4 = new Funcionario();
f4.Id = 4;
f4.Nome = "Mbappe";
f4.Cpf = "246813579";
f4.DataAdmissao = DateTime.Parse("15/09/2005");
f4.Salario = 80.000M;
f4.TipoFuncionario = TipoFuncionarioEnum.Aprendiz;
lista.Add(f4);

Funcionario f5 = new Funcionario();
f5.Id = 5;
f5.Nome = "Lewa";
f5.Cpf = "246813579";
f5.DataAdmissao = DateTime.Parse("20/10/1998");
f5.Salario = 90.000M;
f5.TipoFuncionario = TipoFuncionarioEnum.Aprendiz;
lista.Add(f5);

Funcionario f6 = new Funcionario();
f6.Id = 6;
f6.Nome = "Roger Guedes";
f6.Cpf = "246813579";
f6.DataAdmissao = DateTime.Parse("13/12/1997");
f6.Salario = 300.000M;
f6.TipoFuncionario = TipoFuncionarioEnum.CLT;
lista.Add(f6);
```

4. Crie outro método que será responsável por exibir os dados da lista. Faça a chamada do método no método main e teste o código.

```
public static void ExibirLista()
{
    string dados = "";
    for (int i = 0; i < lista.Count; i++)
    {
        dados += "=====\n";
        dados += string.Format("Id: {0} \n", lista[i].Id);
        dados += string.Format("Nome: {0} \n", lista[i].Nome);
        dados += string.Format("CPF: {0} \n", lista[i].Cpf);
        dados += string.Format("Admissão: {0:dd/MM/yyyy} \n", lista[i].DataAdmissao);
        dados += string.Format("Salário: {0:c2} \n", lista[i].Salario);
        dados += string.Format("Tipo: {0} \n", lista[i].TipoFuncionario);
        dados += "=====\n";
    }
    Console.WriteLine(dados);
}
```



5. Programe o método para realizar uma busca na lista através de uma propriedade do objeto Funcionario

```
public static void ObterPorId()
{
    lista = lista.FindAll(x => x.Id == 1);
}
```

6. Crie um método para adicionar um elemento na lista

```
public static void AdicionarItem()
{
    Funcionario fNovo = new Funcionario();
    fNovo.Id = 9;
    fNovo.Nome = "Ronaldo";
    fNovo.Cpf = "1111111110";
    fNovo.DataAdmissao = DateTime.Parse("17/05/1997");
    fNovo.Salario = 300.000M;
    fNovo.TipoFuncionario = TipoFuncionarioEnum.CLT;

    lista.Add(fNovo);

    ExibirLista();
}
```

Outros exemplos de Métodos usando listas

- Ordenando uma lista por critério. Será necessário o using System.Linq na sintaxe *OrderBy*

```
public static void Ordenar()
{
    lista = lista.OrderBy(x => x.Nome).ToList();
    ExibirLista();
}
```

- Contar Itens de uma lista

```
public static void ContarFuncionarios()
{
    int qtd = lista.Count();
    Console.WriteLine($"Existem {qtd} funcionários.");
}
```



- Somando valores da propriedade comum entre objetos de uma lista

```
public static void SomarSalarios()
{
    decimal somatorio = lista.Sum(x => x.Salario);
    Console.WriteLine(string.Format("A soma dos salários é {0:c2}.", somatorio));
}
```

- Filtrando dados de uma lista de acordo com critérios

```
public static void ExibirAprendizes()
{
    lista = lista.FindAll( x => x.TipoFuncionario == TipoFuncionarioEnum.Aprendiz);
    ExibirLista();
}
```

- Busca por nome aproximado. *ToLower* transforma os caracteres em minúsculo para não termos problemas de distinção. O contrário disso seria o *ToUpper*

```
public static void BuscarPorNomeAproximado()
{
    AdicionarItem();

    lista = lista.FindAll( x => x.Nome.ToLower().Contains("ronaldo"));

    ExibirLista();
}
```

- Filtrando um personagem por algum critério e removendo o mesmo da lista

```
public static void BuscarPorCpfRemover()
{
    Funcionario fBusca = lista.Find( x => x.Cpf == "01987654321");
    lista.Remove(fBusca);
    Console.WriteLine($"Personagem removido: {fBusca.Nome} \nLista Atualizada: \n ");

    ExibirLista();
}
```




- Removendo da lista de acordo com filtragem de Ids

```
public static void RemoverIdMenor4()
{
    lista.RemoveAll( x => x.Id < 4);
    ExibirLista();
}
```

- Dada a lista inicial, será feita uma busca através de um Id digitado

```
static void Main(string[] args)
{
    CriarLista();

    Console.WriteLine("Digite o Id do funcionário que você quer buscar");
    int idDigitado = int.Parse(Console.ReadLine());

    ObterPorId(idDigitado);
}

1 reference
public static void ObterPorId(int id)
{
    Funcionario fBusca = lista.Find( x => x.Id == id);

    Console.WriteLine($"Personagem encontrado: {fBusca.Nome}");
}
```

- Buscando os funcionários com salário acima de um valor digitado.

```
static void Main(string[] args)
{
    CriarLista();

    Console.WriteLine("Digite o salário para buscar todos acima deste valor: ");
    decimal salarioDigitado = decimal.Parse(Console.ReadLine());

    ObterPorSalario(salarioDigitado);
}

1 reference
public static void ObterPorSalario(decimal valor)
{
    lista = lista.FindAll( x => x.Salario >= valor);
    ExibirLista();
}
```



Exemplo de método adição de item com validação das propriedades

- Crie o método abaixo. Perceba que o retorno dele é booleano

```
public static bool AdicionarFuncionario(Funcionario fNovo)
{
    if(fNovo.Salario == 0)
    {
        Console.WriteLine("Valor do salário não pode ser 0");
        return false;
    }
    else
    {
        lista.Add(fNovo);
        return true;
    }
}
```

- Faça a chamada para o método conforme abaixo

```
static void Main(string[] args)
{
    CriarLista();
    bool fAdicionado = false;

    while(fAdicionado == false)
    {
        Funcionario f = new Funcionario();

        Console.WriteLine("Digite o nome: ");
        f.Nome = Console.ReadLine();

        Console.WriteLine("Digite o salário: ");
        f.Salario = decimal.Parse(Console.ReadLine());

        Console.WriteLine("Digite a data de admissão: ");
        f.DataAdmissao = DateTime.Parse(Console.ReadLine());

        fAdicionado = AdicionarFuncionario(f);
    }
    ExibirLista();
}
```

Referências para o estudo de listas

<https://www.tutorialsteacher.com/csharp/csharp-list>

<https://www.dotnetperls.com/list>