**INSTITUTO INFNET**

ESCOLA SUPERIOR DE TECNOLOGIA

GRADUAÇÃO EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS


# AT1

Aluno: Enzo Furtini

17 de mar. de 2025

# Exercícios de C# - Instituto Infnet

Este repositório contém uma série de exercícios em C# desenvolvidos para praticar conceitos fundamentais de programação.

## Índice



## Exercício 1 - Primeiro Programa

**Enunciado:** Crie um programa que imprima no terminal:

- Olá, meu nome é [Seu Nome]!
- Nasci em [sua data de nascimento] e estou aprendendo C#!

**Implementação:**

```
using System;

namespace AT_CSharp.Exercicios
{
    public static class Exercicio1
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 1 - Primeiro Programa ===");

            Console.Write("Digite seu nome: ");
            string nome = Console.ReadLine();

            Console.Write("Digite sua data de nascimento (dd/MM/yyyy): ");
            if (DateTime.TryParse(Console.ReadLine(), out DateTime
dataNascimento))
            {
                Console.WriteLine($"\nOlá, meu nome é {nome}!");
```

```
                Console.WriteLine($"Nasci em {dataNascimento:dd/MM/yyyy} e
estou aprendendo C#!");
            }

            Console.WriteLine("\nPressione qualquer tecla para
continuar...");
            Console.ReadKey();
        }
    }
}
```

## Exercício 2 - Cifrador de Nome

**Enunciado:** Crie um programa que receba seu nome completo e desloque cada letra duas posições para frente no alfabeto.

**Implementação:**

```
using System.Text;

namespace AT_CSharp.Exercicios
{
    public static class Exercicio2
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 2 - Cifrador de Nome ===");

            Console.Write("Digite seu nome completo: ");
            string nome = Console.ReadLine();

            string nomeCifrado = CifrarNome(nome);
            Console.WriteLine($"\nNome cifrado: {nomeCifrado}");

            Console.WriteLine("\nPressione qualquer tecla para
continuar...");
            Console.ReadKey();
        }

        private static string CifrarNome(string nome)
        {
            StringBuilder resultado = new StringBuilder();

            foreach (char c in nome)
            {
                if (char.IsLetter(c))
                {
                    char baseChar = char.IsUpper(c) ? 'A' : 'a';
                    char cifrado = (char)(((c - baseChar + 2) % 26) +
```

```
baseChar);
                    resultado.Append(cifrado);
                }
                else
                {
                    resultado.Append(c);
                }
            }

            return resultado.ToString();
        }
    }
}
```

## Exercício 3 - Calculadora

**Enunciado:** Crie um programa que solicite dois números e peça ao usuário para escolher uma operação matemática.

**Implementação:**

```
using System;

namespace AT_CSharp.Exercicios
{
    public static class Exercicio3
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 3 - Calculadora ===");

            double num1, num2;
            int operacao;

            do
            {
                Console.Write("Digite o primeiro número: ");
            } while (!double.TryParse(Console.ReadLine(), out num1));

            do
            {
                Console.Write("Digite o segundo número: ");
            } while (!double.TryParse(Console.ReadLine(), out num2));

            do
            {
                Console.WriteLine("\nEscolha a operação:");
                Console.WriteLine("1 - Soma");
                Console.WriteLine("2 - Subtração");
```

```
                Console.WriteLine("3 - Multiplicação");
                Console.WriteLine("4 - Divisão");
                Console.Write("Opção: ");
            } while (!int.TryParse(Console.ReadLine(), out operacao) ||
operacao < 1 || operacao > 4);

            double resultado = 0;
            string operacaoTexto = "";

            switch (operacao)
            {
                case 1:
                    resultado = num1 + num2;
                    operacaoTexto = "soma";
                    break;
                case 2:
                    resultado = num1 - num2;
                    operacaoTexto = "subtração";
                    break;
                case 3:
                    resultado = num1 * num2;
                    operacaoTexto = "multiplicação";
                    break;
                case 4:
                    if (num2 == 0)
                    {
                        Console.WriteLine("\nErro: Não é possível dividir por
zero!");
                        Console.WriteLine("\nPressione qualquer tecla para
continuar...");
                        Console.ReadKey();
                        return;
                    }
                    resultado = num1 / num2;
                    operacaoTexto = "divisão";
                    break;
            }

            Console.WriteLine($"\nResultado da {operacaoTexto}:
{resultado}");
            Console.WriteLine("\nPressione qualquer tecla para
continuar...");
            Console.ReadKey();
        }
    }
}
```

## Exercício 4 - Dias até o Próximo Aniversário

**Enunciado:** Crie um programa que peça sua data de nascimento e informe quantos dias faltam para seu próximo aniversário.

**Implementação:**

```
using System;

namespace AT_CSharp.Exercicios
{
    public static class Exercicio4
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 4 - Dias até o Próximo
Aniversário ===");

            DateTime dataNascimento;
            do
            {
                Console.Write("Digite sua data de nascimento (dd/MM/yyyy):
");
            } while (!DateTime.TryParse(Console.ReadLine(), out
dataNascimento));

            DateTime hoje = DateTime.Now;
            DateTime proximoAniversario = new DateTime(hoje.Year,
dataNascimento.Month, dataNascimento.Day);

            if (proximoAniversario < hoje)
            {
                proximoAniversario = proximoAniversario.AddYears(1);
            }

            TimeSpan diasRestantes = proximoAniversario - hoje;

            Console.WriteLine($"\nSeu próximo aniversário será em
{proximoAniversario:dd/MM/yyyy}");
            Console.WriteLine($"Faltam {diasRestantes.Days} dias para seu
aniversário!");

            if (diasRestantes.Days < 7)
            {
                Console.WriteLine("\nUau! Seu aniversário está chegando!
Prepare-se para comemorar! 🎉");
            }
```

```
            Console.WriteLine("\nPressione qualquer tecla para
continuar...");
            Console.ReadKey();
        }
    }
}
```

## Exercício 5 - Tempo até a Formatura

**Enunciado:** Implemente um programa que peça ao usuário a data atual e compare com a data prevista de sua formatura.

**Implementação:**

```
using System;

namespace AT_CSharp.Exercicios
{
    public static class Exercicio5
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 5 - Tempo até a Formatura ===");

            DateTime dataAtual = DateTime.Now;
            DateTime dataFormatura;

            do
            {
                Console.Write("Digite a data prevista de sua formatura
(dd/MM/yyyy): ");
            } while (!DateTime.TryParse(Console.ReadLine(), out
dataFormatura));

            if (dataFormatura < dataAtual)
            {
                Console.WriteLine("\nParabéns! Você já se formou! 🎓");
                Console.WriteLine("\nPressione qualquer tecla para
continuar...");
                Console.ReadKey();
                return;
            }

            TimeSpan tempoRestante = dataFormatura - dataAtual;

            Console.WriteLine($"\nTempo restante até a formatura:");
            Console.WriteLine($"Anos: {tempoRestante.Days / 365}");
            Console.WriteLine($"Meses: {(tempoRestante.Days % 365) / 30}");
            Console.WriteLine($"Dias: {tempoRestante.Days % 30}");
```

```
            if (tempoRestante.Days < 30)
            {
                Console.WriteLine("\nUau! Você está quase se formando!
Continue assim! 🎓");
            }

            Console.WriteLine("\nPressione qualquer tecla para
continuar...");
            Console.ReadKey();
        }
    }
}
```

## Exercício 6 - Cadastro de Alunos

**Enunciado:** Crie um programa que permita cadastrar alunos com nome, idade e nota.

**Implementação:**

```
using System;
using System.Collections.Generic;

namespace AT_CSharp.Exercicios
{
    public class Aluno
    {
        public string Nome { get; set; }
        public int Idade { get; set; }
        public double Nota { get; set; }

        public Aluno(string nome, int idade, double nota)
        {
            Nome = nome;
            Idade = idade;
            Nota = nota;
        }
    }

    public static class Exercicio6
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 6 - Cadastro de Alunos ===");

            List<Aluno> alunos = new List<Aluno>();
            bool continuar = true;

            while (continuar)
```

```
            {
                Console.WriteLine("\n1 - Cadastrar novo aluno");
                Console.WriteLine("2 - Listar alunos");
                Console.WriteLine("3 - Sair");
                Console.Write("Opção: ");

                if (int.TryParse(Console.ReadLine(), out int opcao))
                {
                    switch (opcao)
                    {
                        case 1:
                            CadastrarAluno(alunos);
                            break;
                        case 2:
                            ListarAlunos(alunos);
                            break;
                        case 3:
                            continuar = false;
                            break;
                        default:
                            Console.WriteLine("\nOpção inválida!");
                            break;
                    }
                }
            }
        }

        private static void CadastrarAluno(List<Aluno> alunos)
        {
            Console.Write("\nNome do aluno: ");
            string nome = Console.ReadLine();

            Console.Write("Idade: ");
            int idade;
            do
            {
                Console.Write("Idade: ");
            } while (!int.TryParse(Console.ReadLine(), out idade) || idade <
0);

            Console.Write("Nota (0-10): ");
            double nota;
            do
            {
                Console.Write("Nota (0-10): ");
            } while (!double.TryParse(Console.ReadLine(), out nota) || nota <
0 || nota > 10);

            alunos.Add(new Aluno(nome, idade, nota));
```

```
            Console.WriteLine("\nAluno cadastrado com sucesso!");
        }

        private static void ListarAlunos(List<Aluno> alunos)
        {
            if (alunos.Count == 0)
            {
                Console.WriteLine("\nNenhum aluno cadastrado!");
                return;
            }

            Console.WriteLine("\nLista de Alunos:");
            Console.WriteLine("----------------");
            foreach (var aluno in alunos)
            {
                Console.WriteLine($"Nome: {aluno.Nome}");
                Console.WriteLine($"Idade: {aluno.Idade}");
                Console.WriteLine($"Nota: {aluno.Nota:F2}");
                Console.WriteLine("----------------");
            }
        }
    }
}
```

## Exercício 7 - Banco Digital

**Enunciado:** Implemente um sistema simples de banco digital com operações de depósito e saque.

**Implementação:**

```
using System;

namespace AT_CSharp.Exercicios
{
    public class ContaBancaria
    {
        public string Titular { get; private set; }
        public double Saldo { get; private set; }
        public string NumeroConta { get; private set; }

        public ContaBancaria(string titular, string numeroConta)
        {
            Titular = titular;
            NumeroConta = numeroConta;
            Saldo = 0;
        }

        public void Depositar(double valor)
        {
```

```
            if (valor > 0)
            {
                Saldo += valor;
                Console.WriteLine($"\nDepósito de R$ {valor:F2} realizado com
sucesso!");
            }
            else
            {
                Console.WriteLine("\nValor de depósito inválido!");
            }
        }

        public void Sacar(double valor)
        {
            if (valor > 0 && valor <= Saldo)
            {
                Saldo -= valor;
                Console.WriteLine($"\nSaque de R$ {valor:F2} realizado com
sucesso!");
            }
            else
            {
                Console.WriteLine("\nValor de saque inválido ou saldo
insuficiente!");
            }
        }

        public void ExibirSaldo()
        {
            Console.WriteLine($"\nSaldo atual: R$ {Saldo:F2}");
        }
    }

    public static class Exercicio7
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 7 - Banco Digital ===");

            Console.Write("Digite o nome do titular: ");
            string titular = Console.ReadLine();

            Console.Write("Digite o número da conta: ");
            string numeroConta = Console.ReadLine();

            ContaBancaria conta = new ContaBancaria(titular, numeroConta);
            bool continuar = true;
```

```
            while (continuar)
            {
                Console.WriteLine("\n1 - Depositar");
                Console.WriteLine("2 - Sacar");
                Console.WriteLine("3 - Ver saldo");
                Console.WriteLine("4 - Sair");
                Console.Write("Opção: ");

                if (int.TryParse(Console.ReadLine(), out int opcao))
                {
                    switch (opcao)
                    {
                        case 1:
                            Console.Write("\nValor do depósito: R$ ");
                            if (double.TryParse(Console.ReadLine(), out
double valorDeposito))
                            {
                                conta.Depositar(valorDeposito);
                            }
                            break;
                        case 2:
                            Console.Write("\nValor do saque: R$ ");
                            if (double.TryParse(Console.ReadLine(), out
double valorSaque))
                            {
                                conta.Sacar(valorSaque);
                            }
                            break;
                        case 3:
                            conta.ExibirSaldo();
                            break;
                        case 4:
                            continuar = false;
                            break;
                        default:
                            Console.WriteLine("\nOpção inválida!");
                            break;
                    }
                }
            }
        }
    }
}
```

## Exercício 8 - Cadastro de Funcionários

**Enunciado:** Crie um sistema de cadastro de funcionários com nome, cargo e salário.

**Implementação:**

```
using System;
using System.Collections.Generic;

namespace AT_CSharp.Exercicios
{
    public class Funcionario
    {
        public string Nome { get; set; }
        public string Cargo { get; set; }
        public double Salario { get; set; }

        public Funcionario(string nome, string cargo, double salario)
        {
            Nome = nome;
            Cargo = cargo;
            Salario = salario;
        }
    }

    public static class Exercicio8
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 8 - Cadastro de Funcionários
===");

            List<Funcionario> funcionarios = new List<Funcionario>();
            bool continuar = true;

            while (continuar)
            {
                Console.WriteLine("\n1 - Cadastrar novo funcionário");
                Console.WriteLine("2 - Listar funcionários");
                Console.WriteLine("3 - Calcular folha de pagamento");
                Console.WriteLine("4 - Sair");
                Console.Write("Opção: ");

                if (int.TryParse(Console.ReadLine(), out int opcao))
                {
                    switch (opcao)
                    {
                        case 1:
                            CadastrarFuncionario(funcionarios);
                            break;
                        case 2:
                            ListarFuncionarios(funcionarios);
                            break;
                        case 3:
```

```
                        CalcularFolhaPagamento(funcionarios);
                        break;
                    case 4:
                        continuar = false;
                        break;
                    default:
                        Console.WriteLine("\nOpção inválida!");
                        break;
                }
            }
        }

        private static void CadastrarFuncionario(List<Funcionario>
funcionarios)
        {
            Console.Write("\nNome do funcionário: ");
            string nome = Console.ReadLine();

            Console.Write("Cargo: ");
            string cargo = Console.ReadLine();

            Console.Write("Salário: R$ ");
            double salario;
            do
            {
                Console.Write("Salário: R$ ");
            } while (!double.TryParse(Console.ReadLine(), out salario) ||
salario < 0);

            funcionarios.Add(new Funcionario(nome, cargo, salario));
            Console.WriteLine("\nFuncionário cadastrado com sucesso!");
        }

        private static void ListarFuncionarios(List<Funcionario>
funcionarios)
        {
            if (funcionarios.Count == 0)
            {
                Console.WriteLine("\nNenhum funcionário cadastrado!");
                return;
            }

            Console.WriteLine("\nLista de Funcionários:");
            Console.WriteLine("----------------------");
            foreach (var funcionario in funcionarios)
            {
                Console.WriteLine($"Nome: {funcionario.Nome}");
                Console.WriteLine($"Cargo: {funcionario.Cargo}");
```

```
                Console.WriteLine($"Salário: R$ {funcionario.Salario:F2}");
                Console.WriteLine("----------------------");
            }
        }

        private static void CalcularFolhaPagamento(List<Funcionario>
funcionarios)
        {
            if (funcionarios.Count == 0)
            {
                Console.WriteLine("\nNenhum funcionário cadastrado!");
                return;
            }

            double totalFolha = funcionarios.Sum(f => f.Salario);
            Console.WriteLine($"\nTotal da folha de pagamento: R$
{totalFolha:F2}");
        }
    }
}
```

## Exercício 9 - Controle de Estoque

**Enunciado:** Implemente um sistema de controle de estoque com produtos e quantidades.

**Implementação:**

```
using System;
using System.Collections.Generic;

namespace AT_CSharp.Exercicios
{
    public class Produto
    {
        public string Nome { get; set; }
        public int Quantidade { get; set; }
        public double Preco { get; set; }

        public Produto(string nome, int quantidade, double preco)
        {
            Nome = nome;
            Quantidade = quantidade;
            Preco = preco;
        }
    }

    public static class Exercicio9
    {
        public static void Executar()
        {
```

```
        Console.Clear();
        Console.WriteLine("=== Exercício 9 - Controle de Estoque ===");

        List<Produto> estoque = new List<Produto>();
        bool continuar = true;

        while (continuar)
        {
            Console.WriteLine("\n1 - Adicionar produto");
            Console.WriteLine("2 - Remover produto");
            Console.WriteLine("3 - Atualizar quantidade");
            Console.WriteLine("4 - Listar produtos");
            Console.WriteLine("5 - Calcular valor total do estoque");
            Console.WriteLine("6 - Sair");
            Console.Write("Opção: ");

            if (int.TryParse(Console.ReadLine(), out int opcao))
            {
                switch (opcao)
                {
                    case 1:
                        AdicionarProduto(estoque);
                        break;
                    case 2:
                        RemoverProduto(estoque);
                        break;
                    case 3:
                        AtualizarQuantidade(estoque);
                        break;
                    case 4:
                        ListarProdutos(estoque);
                        break;
                    case 5:
                        CalcularValorTotal(estoque);
                        break;
                    case 6:
                        continuar = false;
                        break;
                    default:
                        Console.WriteLine("\nOpção inválida!");
                        break;
                }
            }
        }
    }

    private static void AdicionarProduto(List<Produto> estoque)
    {
        Console.Write("\nNome do produto: ");
```

```
            string nome = Console.ReadLine();

            Console.Write("Quantidade: ");
            int quantidade;
            do
            {
                Console.Write("Quantidade: ");
            } while (!int.TryParse(Console.ReadLine(), out quantidade) ||
quantidade < 0);

            Console.Write("Preço unitário: R$ ");
            double preco;
            do
            {
                Console.Write("Preço unitário: R$ ");
            } while (!double.TryParse(Console.ReadLine(), out preco) || preco
< 0);

            estoque.Add(new Produto(nome, quantidade, preco));
            Console.WriteLine("\nProduto adicionado com sucesso!");
        }

        private static void RemoverProduto(List<Produto> estoque)
        {
            if (estoque.Count == 0)
            {
                Console.WriteLine("\nEstoque vazio!");
                return;
            }

            Console.Write("\nNome do produto a remover: ");
            string nome = Console.ReadLine();

            var produto = estoque.FirstOrDefault(p => p.Nome.Equals(nome,
StringComparison.OrdinalIgnoreCase));
            if (produto != null)
            {
                estoque.Remove(produto);
                Console.WriteLine("\nProduto removido com sucesso!");
            }
            else
            {
                Console.WriteLine("\nProduto não encontrado!");
            }
        }

        private static void AtualizarQuantidade(List<Produto> estoque)
        {
            if (estoque.Count == 0)
```

```
            {
                Console.WriteLine("\nEstoque vazio!");
                return;
            }

            Console.Write("\nNome do produto: ");
            string nome = Console.ReadLine();

            var produto = estoque.FirstOrDefault(p => p.Nome.Equals(nome,
StringComparison.OrdinalIgnoreCase));
            if (produto != null)
            {
                Console.Write("Nova quantidade: ");
                int novaQuantidade;
                do
                {
                    Console.Write("Nova quantidade: ");
                } while (!int.TryParse(Console.ReadLine(), out
novaQuantidade) || novaQuantidade < 0);

                produto.Quantidade = novaQuantidade;
                Console.WriteLine("\nQuantidade atualizada com sucesso!");
            }
            else
            {
                Console.WriteLine("\nProduto não encontrado!");
            }
        }

        private static void ListarProdutos(List<Produto> estoque)
        {
            if (estoque.Count == 0)
            {
                Console.WriteLine("\nEstoque vazio!");
                return;
            }

            Console.WriteLine("\nLista de Produtos:");
            Console.WriteLine("-----------------");
            foreach (var produto in estoque)
            {
                Console.WriteLine($"Nome: {produto.Nome}");
                Console.WriteLine($"Quantidade: {produto.Quantidade}");
                Console.WriteLine($"Preço: R$ {produto.Preco:F2}");
                Console.WriteLine("-----------------");
            }
        }

        private static void CalcularValorTotal(List<Produto> estoque)
```

```
        {
            if (estoque.Count == 0)
            {
                Console.WriteLine("\nEstoque vazio!");
                return;
            }

            double valorTotal = estoque.Sum(p => p.Quantidade * p.Preco);
            Console.WriteLine($"\nValor total do estoque: R$
{valorTotal:F2}");
        }
    }
}
```

## Exercício 10 - Jogo de Adivinhação

**Enunciado:** Crie um jogo onde o computador escolhe um número aleatório e o jogador tenta adivinhar.

**Implementação:**

```
using System;

namespace AT_CSharp.Exercicios
{
    public static class Exercicio10
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 10 - Jogo de Adivinhação ===");

            Random random = new Random();
            int numeroSecreto = random.Next(1, 101);
            int tentativas = 0;
            bool acertou = false;

            Console.WriteLine("Bem-vindo ao Jogo de Adivinhação!");
            Console.WriteLine("Tente adivinhar o número entre 1 e 100.");

            while (!acertou)
            {
                Console.Write("\nDigite sua tentativa: ");
                if (int.TryParse(Console.ReadLine(), out int tentativa))
                {
                    tentativas++;

                    if (tentativa == numeroSecreto)
                    {
                        acertou = true;
```

```
                    Console.WriteLine($"\nParabéns! Você acertou em
{tentativas} tentativas!");
                }
                else if (tentativa < numeroSecreto)
                {
                    Console.WriteLine("O número é maior!");
                }
                else
                {
                    Console.WriteLine("O número é menor!");
                }
            }
        }

        Console.WriteLine("\nPressione qualquer tecla para
continuar...");
        Console.ReadKey();
    }
}
```

## Exercício 11 - Gerenciador de Contatos

**Enunciado:** Implemente um sistema de gerenciamento de contatos com nome, telefone e email.

**Implementação:**

```
using System;
using System.Collections.Generic;

namespace AT_CSharp.Exercicios
{
    public class Contato
    {
        public string Nome { get; set; }
        public string Telefone { get; set; }
        public string Email { get; set; }

        public Contato(string nome, string telefone, string email)
        {
            Nome = nome;
            Telefone = telefone;
            Email = email;
        }
    }

    public static class Exercicio11
    {
        public static void Executar()
```

```
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 11 - Gerenciador de Contatos
===");

            List<Contato> contatos = new List<Contato>();
            bool continuar = true;

            while (continuar)
            {
                Console.WriteLine("\n1 - Adicionar contato");
                Console.WriteLine("2 - Remover contato");
                Console.WriteLine("3 - Buscar contato");
                Console.WriteLine("4 - Listar contatos");
                Console.WriteLine("5 - Sair");
                Console.Write("Opção: ");

                if (int.TryParse(Console.ReadLine(), out int opcao))
                {
                    switch (opcao)
                    {
                        case 1:
                            AdicionarContato(contatos);
                            break;
                        case 2:
                            RemoverContato(contatos);
                            break;
                        case 3:
                            BuscarContato(contatos);
                            break;
                        case 4:
                            ListarContatos(contatos);
                            break;
                        case 5:
                            continuar = false;
                            break;
                        default:
                            Console.WriteLine("\nOpção inválida!");
                            break;
                    }
                }
            }
        }

        private static void AdicionarContato(List<Contato> contatos)
        {
            Console.Write("\nNome: ");
            string nome = Console.ReadLine();
```

```
            Console.Write("Telefone: ");
            string telefone = Console.ReadLine();

            Console.Write("Email: ");
            string email = Console.ReadLine();

            contatos.Add(new Contato(nome, telefone, email));
            Console.WriteLine("\nContato adicionado com sucesso!");
        }

        private static void RemoverContato(List<Contato> contatos)
        {
            if (contatos.Count == 0)
            {
                Console.WriteLine("\nNenhum contato cadastrado!");
                return;
            }

            Console.Write("\nNome do contato a remover: ");
            string nome = Console.ReadLine();

            var contato = contatos.FirstOrDefault(c => c.Nome.Equals(nome,
StringComparison.OrdinalIgnoreCase));
            if (contato != null)
            {
                contatos.Remove(contato);
                Console.WriteLine("\nContato removido com sucesso!");
            }
            else
            {
                Console.WriteLine("\nContato não encontrado!");
            }
        }

        private static void BuscarContato(List<Contato> contatos)
        {
            if (contatos.Count == 0)
            {
                Console.WriteLine("\nNenhum contato cadastrado!");
                return;
            }

            Console.Write("\nNome do contato a buscar: ");
            string nome = Console.ReadLine();

            var contato = contatos.FirstOrDefault(c => c.Nome.Equals(nome,
StringComparison.OrdinalIgnoreCase));
            if (contato != null)
            {
```

```
                Console.WriteLine("\nContato encontrado:");
                Console.WriteLine($"Nome: {contato.Nome}");
                Console.WriteLine($"Telefone: {contato.Telefone}");
                Console.WriteLine($"Email: {contato.Email}");
            }
            else
            {
                Console.WriteLine("\nContato não encontrado!");
            }
        }

        private static void ListarContatos(List<Contato> contatos)
        {
            if (contatos.Count == 0)
            {
                Console.WriteLine("\nNenhum contato cadastrado!");
                return;
            }

            Console.WriteLine("\nLista de Contatos:");
            Console.WriteLine("------------------");
            foreach (var contato in contatos)
            {
                Console.WriteLine($"Nome: {contato.Nome}");
                Console.WriteLine($"Telefone: {contato.Telefone}");
                Console.WriteLine($"Email: {contato.Email}");
                Console.WriteLine("------------------");
            }
        }
    }
}
```

## Exercício 12 - Formatos de Exibição de Contatos

**Enunciado:** Crie um programa que exiba os contatos em diferentes formatos (texto, HTML, JSON).

**Implementação:**

```
using System;
using System.Collections.Generic;
using System.Text.Json;

namespace AT_CSharp.Exercicios
{
    public class Contato
    {
        public string Nome { get; set; }
        public string Telefone { get; set; }
        public string Email { get; set; }
```

```
        public Contato(string nome, string telefone, string email)
        {
            Nome = nome;
            Telefone = telefone;
            Email = email;
        }
    }

    public static class Exercicio12
    {
        public static void Executar()
        {
            Console.Clear();
            Console.WriteLine("=== Exercício 12 - Formatos de Exibição de
Contatos ===");

            List<Contato> contatos = new List<Contato>();
            bool continuar = true;

            while (continuar)
            {
                Console.WriteLine("\n1 - Adicionar contato");
                Console.WriteLine("2 - Exibir em formato texto");
                Console.WriteLine("3 - Exibir em formato HTML");
                Console.WriteLine("4 - Exibir em formato JSON");
                Console.WriteLine("5 - Sair");
                Console.Write("Opção: ");

                if (int.TryParse(Console.ReadLine(), out int opcao))
                {
                    switch (opcao)
                    {
                        case 1:
                            AdicionarContato(contatos);
                            break;
                        case 2:
                            ExibirFormatoTexto(contatos);
                            break;
                        case 3:
                            ExibirFormatoHtml(contatos);
                            break;
                        case 4:
                            ExibirFormatoJson(contatos);
                            break;
                        case 5:
                            continuar = false;
                            break;
                        default:
```

```
                    Console.WriteLine("\nOpção inválida!");
                    break;
                }
            }
        }
    }

    private static void AdicionarContato(List<Contato> contatos)
    {
        Console.Write("\nNome: ");
        string nome = Console.ReadLine();

        Console.Write("Telefone: ");
        string telefone = Console.ReadLine();

        Console.Write("Email: ");
        string email = Console.ReadLine();

        contatos.Add(new Contato(nome, telefone, email));
        Console.WriteLine("\nContato adicionado com sucesso!");
    }

    private static void ExibirFormatoTexto(List<Contato> contatos)
    {
        if (contatos.Count == 0)
        {
            Console.WriteLine("\nNenhum contato cadastrado!");
            return;
        }

        Console.WriteLine("\nLista de Contatos (Formato Texto):");
        Console.WriteLine("--------------------------------");
        foreach (var contato in contatos)
        {
            Console.WriteLine($"Nome: {contato.Nome}");
            Console.WriteLine($"Telefone: {contato.Telefone}");
            Console.WriteLine($"Email: {contato.Email}");
            Console.WriteLine("--------------------------------");
        }
    }

    private static void ExibirFormatoHtml(List<Contato> contatos)
    {
        if (contatos.Count == 0)
        {
            Console.WriteLine("\nNenhum contato cadastrado!");
            return;
        }
```

```
            Console.WriteLine("\nLista de Contatos (Formato HTML):");
            Console.WriteLine("<html>");
            Console.WriteLine("<head><title>Contatos</title></head>");
            Console.WriteLine("<body>");
            Console.WriteLine("<h1>Lista de Contatos</h1>");
            Console.WriteLine("<ul>");
            foreach (var contato in contatos)
            {
                Console.WriteLine($"<li>");
                Console.WriteLine($"<strong>Nome:</strong>
{contato.Nome}<br>");
                Console.WriteLine($"<strong>Telefone:</strong>
{contato.Telefone}<br>");
                Console.WriteLine($"<strong>Email:</strong>
{contato.Email}");
                Console.WriteLine("</li>");
            }
            Console.WriteLine("</ul>");
            Console.WriteLine("</body>");
            Console.WriteLine("</html>");
        }

        private static void ExibirFormatoJson(List<Contato> contatos)
        {
            if (contatos.Count == 0)
            {
                Console.WriteLine("\nNenhum contato cadastrado!");
                return;
            }

            Console.WriteLine("\nLista de Contatos (Formato JSON):");
            var options = new JsonSerializerOptions { WriteIndented = true };
            string json = JsonSerializer.Serialize(contatos, options);
            Console.WriteLine(json);
        }
    }
}
```