



INSTITUTO INFNET

LÍVIA FARIA BRAZ

ASSESSMENT

Disciplina Regular 2: Fundamentos de
Desenvolvimento com C#

Professor: Luiz Paulo Maia

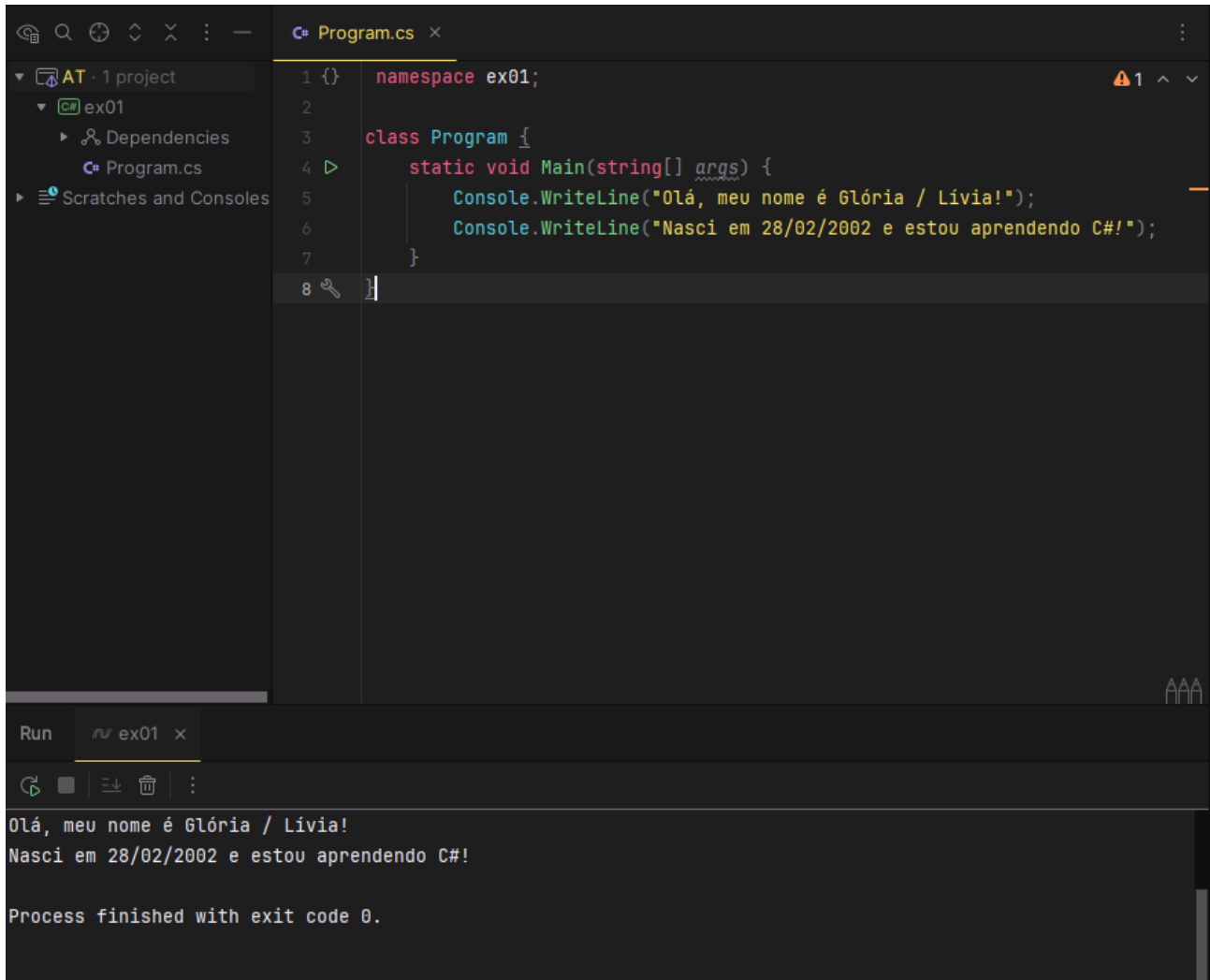
ITAÚNA - MG

08 de abril de 2025



Fundamentos de Desenvolvimento com C# - AT

➤ Exercício 01: Criando e Executando seu Primeiro Programa



```
1 {} namespace ex01;
2
3 class Program {
4     static void Main(string[] args) {
5         Console.WriteLine("Olá, meu nome é Glória / Livia!");
6         Console.WriteLine("Nasci em 28/02/2002 e estou aprendendo C#!");
7     }
8 }
```

Run ex01 x

Olá, meu nome é Glória / Livia!
Nasci em 28/02/2002 e estou aprendendo C#!

Process finished with exit code 0.

➤ Exercício 02: Manipulação de Strings - Cifrador de Nome

```
namespace ex02;

class Program {
    static void Main(string[] args) {

        Console.Write("Digite seu nome completo: ");
        string nome = Console.ReadLine();
        char[] nomeCodificado = new char[nome.Length];

        for (int i = 0; i < nome.Length; i++) {
```



Fundamentos de Desenvolvimento com C# - AT

```
char letra = nome[i];

if (char.IsLetter(letra)) {
    char letraBase = char.IsUpper(letra) ? 'A' : 'a';
    nomeCodificado[i] = (char)((letra - letraBase + 2) % 26) +
letraBase);
} else {
    nomeCodificado[i] = letra;
}
}

Console.WriteLine("Nome codificado: " + new string(nomeCodificado));
}
}
```

➤ Exercício 03: Calculadora de Operações Matemáticas

```
namespace ex03;

class Program {
    static void Main(string[] args) {

        Console.Write("Digite o primeiro número: ");
        if (!double.TryParse(Console.ReadLine(), out double num1))
        {
            Console.WriteLine("Número inválido.");
            return;
        }

        Console.Write("Digite o segundo número: ");
        if (!double.TryParse(Console.ReadLine(), out double num2))
        {
            Console.WriteLine("Número inválido.");
            return;
        }

        Console.WriteLine("\nEscolha a operação: \n1) Soma \n2) Subtração \n3)
Multiplicação \n4) Divisão");
        string opcao = Console.ReadLine();
        Console.WriteLine();

        switch (opcao) {
            case "1":
                Console.WriteLine($"Resultado: {num1 + num2}");
                break;
            case "2":
                Console.WriteLine($"Resultado: {num1 - num2}");
                break;
            case "3":
                Console.WriteLine($"Resultado: {num1 * num2}");
                break;
            case "4":
                if (num2 == 0)

```



Fundamentos de Desenvolvimento com C# - AT

```
        Console.WriteLine("Erro: Divisão por zero.");
    else
        Console.WriteLine($"Resultado: {num1 / num2}");
    break;
default:
    Console.WriteLine("Operação inválida.");
    break;
}
}
```

➤ Exercício 04: Manipulação de Datas - Dias até o Próximo Aniversário

```
namespace ex04;

class Program {
    static void Main(string[] args) {
        Console.Write("Digite sua data de nascimento (dd/MM/yyyy): ");
        if (!DateTime.TryParse(Console.ReadLine(), out DateTime nascimento))
        {
            Console.WriteLine("Data inválida.");
            return;
        }

        DateTime hoje = DateTime.Today;
        DateTime proximoAniversario = new DateTime(hoje.Year, nascimento.Month,
nascimento.Day);

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

        TimeSpan diferenca = proximoAniversario - hoje;

        Console.WriteLine($"Faltam {diferenca.Days} dias para seu próximo
aniversário!");

        if (diferenca.Days < 7)
        {
            Console.WriteLine("\nFalta pouco! Seu aniversário está chegando!");
        }
    }
}
```

➤ Exercício 05: Tempo Restante para Conclusão do Curso - Diferença Entre Datas

```
using System.Globalization;

namespace ex05;

class Program {
    static void Main(string[] args) {
```



Fundamentos de Desenvolvimento com C# - AT

```
DateTime dataFormatura = new DateTime(2028, 06, 30);

Console.WriteLine("Digite a data atual (dd/MM/yyyy): ");
string input = Console.ReadLine();

        if (DateTime.TryParseExact(input, "dd/MM/yyyy",
CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime dataAtual))
    {
        if (dataAtual > DateTime.Now)
        {
            Console.WriteLine("Erro: A data informada não pode ser no
futuro!");
            return;
        }

        if (dataAtual > dataFormatura)
        {
            Console.WriteLine("Parabéns! Você já deveria estar formada!");
            return;
        }

        TimeSpan diferenca = dataFormatura - dataAtual;
        DateTime dataTemp = new DateTime(1, 1, 1).Add(diferenca);

        int anos = dataFormatura.Year - dataAtual.Year;
        int meses = dataFormatura.Month - dataAtual.Month;
        int dias = dataFormatura.Day - dataAtual.Day;

        if (dias < 0)
        {
            meses--;
            dias += DateTime.DaysInMonth(dataAtual.Year, dataAtual.Month);
        }

        if (meses < 0)
        {
            anos--;
            meses += 12;
        }

        Console.WriteLine($"Faltam {anos} anos, {meses} meses e {dias} dias
para sua formatura!");

        if (anos == 0 && meses < 6)
        {
            Console.WriteLine("A reta final chegou! Prepare-se para a
formatura!");
        }
        else
        {
            Console.WriteLine("Data inválida! Use o formato dd/MM/yyyy.");
        }
    }
}
```



Fundamentos de Desenvolvimento com C# - AT

➤ Exercício 06: Cadastro de Alunos

```
namespace ex06.model;

public class Aluno {
    public string Nome { get; set; }
    public string Matricula { get; set; }
    public string Curso { get; set; }
    public double MediaNotas { get; set; }

    public void ExibirDados()
    {
        Console.WriteLine($"Nome: {Nome}\nMatrícula: {Matricula}\nCurso: {Curso}\nMédia: {MediaNotas}");
    }

    public string VerificarAprovacao()
    {
        return MediaNotas >= 7 ? "Aprovado" : "Reprovado";
    }
}
```

```
using ex06.model;

namespace ex06;

class Program {
    static void Main(string[] args) {

        Aluno aluna = new Aluno
        {
            Nome = "Glória Braz",
            Matricula = "20240428",
            Curso = "Engenharia de Software",
            MediaNotas = 8.2
        };

        aluna.ExibirDados();
        Console.WriteLine($"Status: {aluna.VerificarAprovacao()}");
    }
}
```

➤ Exercício 07: Banco Digital (Encapsulamento)

```
namespace ex07.models;

public class ContaBancaria {

    public string Titular { get; set; }
    private decimal Saldo;

    public void Depositar(decimal valor) {
```



Fundamentos de Desenvolvimento com C# - AT

```
        if (valor <= 0) {
            Console.WriteLine($"Tentativa de depósito: R$ {valor:F2}");
            Console.WriteLine("O valor do depósito deve ser positivo!\n");
            return;
        }

        Saldo += valor;
        Console.WriteLine($"Depósito de R$ {valor:F2} realizado com sucesso!");
    }

    public void Sacar(decimal valor) {
        if (valor > Saldo) {
            Console.WriteLine($"Tentativa de saque: R$ {valor:F2}");
            Console.WriteLine("Saldo insuficiente para realizar o saque!\n");
            return;
        }

        Saldo -= valor;
        Console.WriteLine($"Saque de R$ {valor:F2} realizado com sucesso!");
    }

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

```
using ex07.models;

namespace ex07;

class Program {
    static void Main(string[] args) {

        ContaBancaria conta = new ContaBancaria { Titular = "Luan Henrique" };

        Console.WriteLine($"Titular: {conta.Titular}");

        conta.Depositar(500);
        conta.ExibirSaldo();

        conta.Sacar(700); // Deve retornar o erro de saldo insuficiente
        conta.ExibirSaldo();

        conta.Depositar(-40); // Deve retornar o erro de valor negativo

        conta.Sacar(200);
        conta.ExibirSaldo();
    }
}
```



Fundamentos de Desenvolvimento com C# - AT

➤ Exercício 08: Cadastro de Funcionários (Herança)

```
namespace ex08.models;

public class Funcionario {
    public string Nome { get; set; }
    public string Cargo { get; set; }
    public double SalarioBase { get; set; }

    public virtual double CalcularSalario() {
        return SalarioBase;
    }
}
```

```
namespace ex08.models;

public class Gerente : Funcionario {
    public override double CalcularSalario() {
        return SalarioBase * 1.2;
    }
}
```

```
using ex08.models;

namespace ex08;

class Program {
    static void Main(string[] args) {
        Funcionario funcionario = new Funcionario {
            Nome = "Glória",
            Cargo = "Engenheira de Software",
            SalarioBase = 3200
        };

        Gerente gerente = new Gerente {
            Nome = "Luan",
            Cargo = "Gerente de Projetos",
            SalarioBase = 12000
        };

        Console.WriteLine($"{funcionario.Nome} - {funcionario.Cargo} - Salário: R$ {funcionario.CalcularSalario():F2}");
        Console.WriteLine($"{gerente.Nome} - {gerente.Cargo} - Salário: R$ {gerente.CalcularSalario():F2}");
    }
}
```

➤ Exercício 09: Controle de Estoque via Linha de Comando

```
namespace ex09.models;

public class Produto {
    public string Nome { get; set; }
    public int Quantidade { get; set; }
}
```




Fundamentos de Desenvolvimento com C# - AT

```
public double Preco { get; set; }  
}  
  
using System.ComponentModel.Design;  
using ex09.models;  
  
namespace ex09;  
  
class Parte01 {  
    static void Main(string[] args) {  
        Produto[] estoque = new Produto[5];  
        int count = 0;  
        int opcao;  
  
        do {  
            Menu();  
            opcao = int.Parse(Console.ReadLine());  
  
            if (opcao == 1)  
            {  
                if (count >= 5)  
                {  
                    Console.WriteLine("Limite de produtos atingido!");  
                }  
                else  
                {  
                    Produto p = new Produto();  
                    Console.Write("Nome: ");  
                    p.Nome = Console.ReadLine();  
                    Console.Write("Quantidade: ");  
                    p.Quantidade = int.Parse(Console.ReadLine());  
                    Console.Write("Preço: ");  
                    p.Preco = double.Parse(Console.ReadLine());  
                    estoque[count++] = p;  
                }  
            }  
            else if (opcao == 2)  
            {  
                for (int i = 0; i < count; i++)  
                {  
                    Console.WriteLine($"Produto: {estoque[i].Nome} | Quantidade:  
{estoque[i].Quantidade} | Preço: R$ {estoque[i].Preco:F2}");  
                }  
            }  
        } while (opcao != 3);  
    }  
  
    public static void Menu() {  
        Console.WriteLine();  
        Console.WriteLine("##### Menu #####");  
        Console.WriteLine("1) Inserir Produto");  
        Console.WriteLine("2) Listar Produtos");  
        Console.WriteLine("3) Sair");  
    }  
}
```



Fundamentos de Desenvolvimento com C# - AT

```
using ex09.models;

namespace ex09 {
    class Parte02 {
        static string caminhoArquivo =
@"C:\Users\Glória\Downloads\AT\ex09\estoque.txt";

        static void Main(string[] args) {
            int opcao;

            do {
                Menu();
                opcao = int.Parse(Console.ReadLine());

                if (opcao == 1) {
                    InserirProduto();
                }
                else if (opcao == 2) {
                    ListarProdutos();
                }
            } while (opcao != 3);
        }

        public static void Menu() {
            Console.WriteLine();
            Console.WriteLine("##### Menu #####");
            Console.WriteLine("1) Inserir Produto");
            Console.WriteLine("2) Listar Produtos");
            Console.WriteLine("3) Sair");
        }

        static void InserirProduto() {
            try {
                int quantidadeAtual = 0;
                if (File.Exists(caminhoArquivo)) {
                    quantidadeAtual = File.ReadAllLines(caminhoArquivo).Length;
                }

                if (quantidadeAtual >= 5) {
                    Console.WriteLine("Limite de produtos atingido!");
                    return;
                }

                Produto p = new Produto();
                Console.Write("Nome: ");
                p.Nome = Console.ReadLine();
                Console.Write("Quantidade: ");
                p.Quantidade = int.Parse(Console.ReadLine());
                Console.Write("Preço: ");
                p.Preco = double.Parse(Console.ReadLine());

                using (StreamWriter writer = File.AppendText(caminhoArquivo)) {
                    writer.WriteLine($"{p.Nome};{p.Quantidade};{p.Preco:F2}");
                }
            }
        }
    }
}
```



Fundamentos de Desenvolvimento com C# - AT

```
    }

    Console.WriteLine("Produto cadastrado com sucesso!");
}
catch (Exception ex) {
    Console.WriteLine("Erro ao inserir produto: " + ex.Message);
}
}

static void ListarProdutos() {
    try {
        if (!File.Exists(caminhoArquivo) || new
FileInfo(caminhoArquivo).Length == 0) {
            Console.WriteLine("Nenhum produto cadastrado.");
            return;
        }

        string[] linhas = File.ReadAllLines(caminhoArquivo);

        foreach (string linha in linhas) {
            string[] partes = linha.Split(';');

            if (partes.Length != 3) {
                Console.WriteLine("Linha com formato inválido: " + linha);
                continue;
            }

            string nome = partes[0];
            int quantidade = int.Parse(partes[1]);
            double preco = double.Parse(partes[2]);

            Console.WriteLine($"Produto: {nome} | Quantidade: {quantidade}
| Preço: R$ {preco:F2}");
        }
        catch (Exception ex) {
            Console.WriteLine("Erro ao listar produtos: " + ex.Message);
        }
    }
}
```

➤ Exercício 10: Jogo de Adivinhação

```
namespace ex10;

class Program {
    static void Main(string[] args) {
        Random rnd = new Random();
        int numeroSecreto = rnd.Next(1, 51);
        int tentativas = 5;

        while (tentativas > 0)
        {
```



Fundamentos de Desenvolvimento com C# - AT

```
        Console.WriteLine($"Tentativa {6 - tentativas}/5 - Adivinhe o número (1 a 50): ");
    try
    {
        int palpite = int.Parse(Console.ReadLine());

        if (palpite < 1 || palpite > 50)
        {
            Console.WriteLine("Número fora do intervalo!");
            continue;
        }

        if (palpite == numeroSecreto)
        {
            Console.WriteLine("Parabéns! Você acertou!");
            return;
        }
        else
        {
            Console.WriteLine("Errou! Tente novamente.");
        }

        tentativas--;
    }
    catch
    {
        Console.WriteLine("Entrada inválida!");
    }
}

Console.WriteLine($"Suas tentativas acabaram! O número era {numeroSecreto}.");
}
```

➤ Exercício 11: Manipulação de Arquivos - Cadastro e Listagem de Contatos

```
namespace ex11;

class Program {
    static void Main(string[] args) {
        string caminho = @"C:\Users\Glória\Downloads\AT\ex11\contatos.txt";
        int opcao;

        do
        {
            Console.WriteLine("\n=== Gerenciador de Contatos ===");
            Console.WriteLine("1 - Adicionar novo contato");
            Console.WriteLine("2 - Listar contatos cadastrados");
            Console.WriteLine("3 - Sair");
            Console.Write("Escolha uma opção: ");
            opcao = int.Parse(Console.ReadLine());

            if (opcao == 1)
```



Fundamentos de Desenvolvimento com C# - AT

```
{
    Console.WriteLine("Nome: ");
    string nome = Console.ReadLine();
    Console.WriteLine("Telefone: ");
    string telefone = Console.ReadLine();
    Console.WriteLine("Email: ");
    string email = Console.ReadLine();

    using (StreamWriter sw = File.AppendText(caminho))
    {
        sw.WriteLine($"{nome},{telefone},{email}");
    }

    Console.WriteLine("Contato cadastrado com sucesso!");
}
else if (opcao == 2)
{
    if (!File.Exists(caminho) || File.ReadAllLines(caminho).Length ==
0)
    {
        Console.WriteLine("Nenhum contato cadastrado.");
    }
    else
    {
        string[] contatos = File.ReadAllLines(caminho);
        Console.WriteLine("\nContatos cadastrados:\n");
        foreach (string contato in contatos)
        {
            string[] dados = contato.Split(',');
            Console.WriteLine($"Nome: {dados[0]} | Telefone: {dados[1]}
| Email: {dados[2]}");
        }
    }
} while (opcao != 3);

Console.WriteLine("Encerrando programa...");
}
```

➤ Exercício 12: Manipulação de Arquivos com Herança e Polimorfismo - Formatos de Exibição

```
namespace ex12.models;

public class Contato {
    public string Nome { get; set; }
    public string Telefone { get; set; }
    public string Email { get; set; }
}
```

```
namespace ex12.models;

public abstract class ContatoFormatter {
```



Fundamentos de Desenvolvimento com C# - AT

```
public abstract void ExibirContatos(List<Contato> contatos);  
}
```

```
namespace ex12.models;  
  
public class MarkdownFormatter : ContatoFormatter {  
    public override void ExibirContatos(List<Contato> contatos) {  
        Console.WriteLine("## Lista de Contatos\n");  
        foreach (var c in contatos) {  
            Console.WriteLine($"- **Nome:** {c.Nome}\n- 📞 Telefone:  
{c.Telefone}\n- ✉ Email: {c.Email}\n");  
        }  
    }  
}
```

```
namespace ex12.models;  
  
public class RawTextFormatter : ContatoFormatter {  
    public override void ExibirContatos(List<Contato> contatos) {  
        foreach (var c in contatos) {  
            Console.WriteLine($"Nome: {c.Nome} | Telefone: {c.Telefone} | Email:  
{c.Email}");  
        }  
    }  
}
```

```
namespace ex12.models;  
  
public class TabelaFormatter : ContatoFormatter {  
    public override void ExibirContatos(List<Contato> contatos)  
    {  
        Console.WriteLine("-----");  
        Console.WriteLine("| Nome          | Telefone          | Email    |");  
        Console.WriteLine("-----");  
        foreach (var c in contatos)  
        {  
            Console.WriteLine($"| {c.Nome,-14} | {c.Telefone,-14} | {c.Email,-20}  
|");  
        }  
        Console.WriteLine("-----");  
    }  
}
```

```
using ex12.models;  
  
namespace ex12;  
  
class Program {  
    static string caminho = @"C:\Users\Glória\Downloads\AT\ex12\contatos.txt";  
  
    static void Main()  
    {  
        int opcao;  
        do  
        {
```



Fundamentos de Desenvolvimento com C# - AT

```
Console.WriteLine("\n1 - Adicionar novo contato\n2 - Listar contatos\n3  
- Sair");  
opcao = int.Parse(Console.ReadLine());  
  
if (opcao == 1)  
{  
    Console.Write("Nome: ");  
    string nome = Console.ReadLine();  
    Console.Write("Telefone: ");  
    string telefone = Console.ReadLine();  
    Console.Write("Email: ");  
    string email = Console.ReadLine();  
  
    using (StreamWriter sw = File.AppendText(caminho))  
    {  
        sw.WriteLine($"{nome},{telefone},{email}");  
    }  
  
    Console.WriteLine("Contato cadastrado com sucesso!");  
}  
else if (opcao == 2)  
{  
    if (!File.Exists(caminho))  
    {  
        Console.WriteLine("Nenhum contato cadastrado.");  
        continue;  
    }  
  
    List<Contato> contatos = new List<Contato>();  
    foreach (var linha in File.ReadAllLines(caminho))  
    {  
        var dados = linha.Split(',');  
        contatos.Add(new Contato { Nome = dados[0], Telefone =  
dados[1], Email = dados[2] });  
    }  
  
    Console.WriteLine("Escolha o formato de exibição:");  
    Console.WriteLine("1 - Markdown\n2 - Tabela\n3 - Texto Puro");  
    int formato = int.Parse(Console.ReadLine());  
  
    ContatoFormatter formatter = formato switch  
    {  
        1 => new MarkdownFormatter(),  
        2 => new TabelaFormatter(),  
        _ => new RawTextFormatter()  
    };  
  
    formatter.ExibirContatos(contatos);  
}  
  
} while (opcao != 3);  
  
Console.WriteLine("Encerrando programa...");  
}
```