

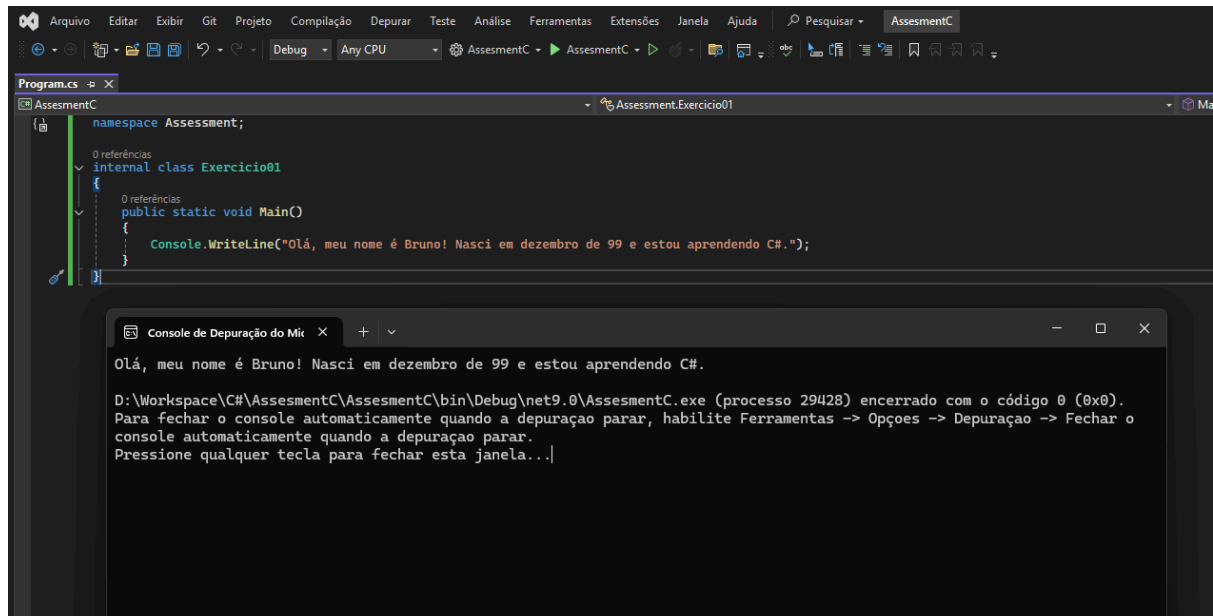
Instituto Infnet

Disciplina: Fundamentos de Desenvolvimento com C#

Discente: Bruno Martins

Assessment

Exercício 1: Criando e Executando seu Primeiro Programa



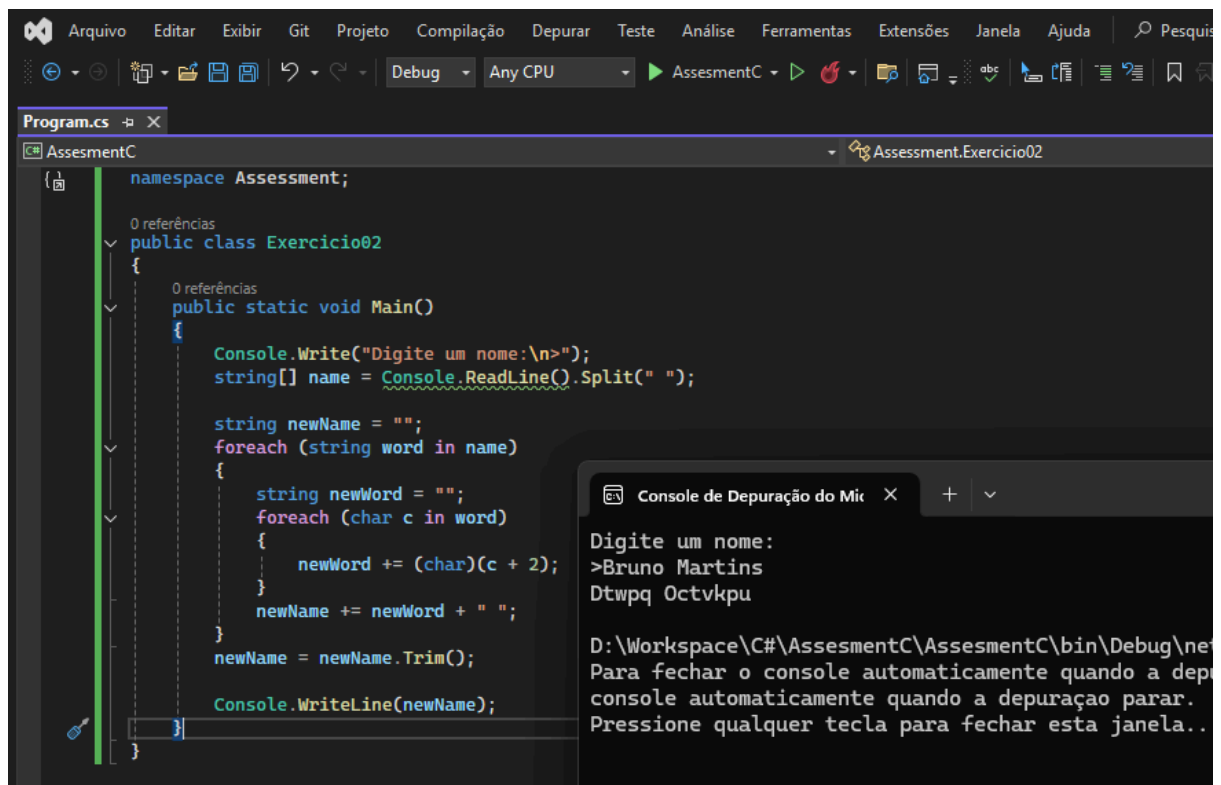
```
namespace Assessment;

internal class Exercicio01
{
    public static void Main()
    {
        Console.WriteLine("Olá, meu nome é Bruno! Nasci em dezembro de 99 e estou aprendendo C#.");
    }
}
```

Olá, meu nome é Bruno! Nasci em dezembro de 99 e estou aprendendo C#.

D:\Workspace\C#\AssesmentC\AssesmentC\bin\Debug\net9.0\AssesmentC.exe (processo 29428) encerrado com o código 0 (0x0). Para fechar o console automaticamente quando a depuração parar, habilite Ferramentas -> Opções -> Depuração -> Fechar o console automaticamente quando a depuração parar. Pressione qualquer tecla para fechar esta janela...

Exercício 2: Manipulação de Strings - Cifrador de Nome



```
namespace Assessment;

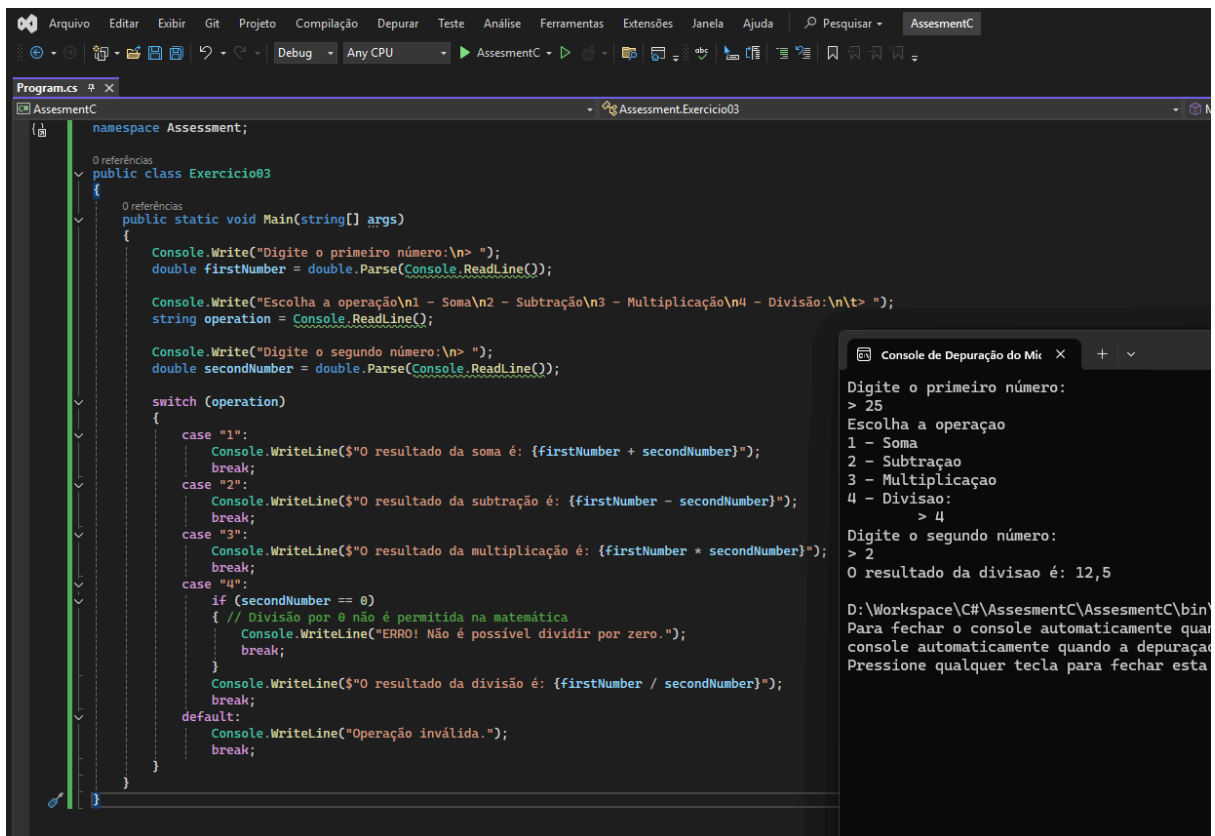
public class Exercicio02
{
    public static void Main()
    {
        Console.Write("Digite um nome:\n>");
        string[] name = Console.ReadLine().Split(" ");

        string newName = "";
        foreach (string word in name)
        {
            string newWord = "";
            foreach (char c in word)
            {
                newWord += (char)(c + 2);
            }
            newName += newWord + " ";
        }
        newName = newName.Trim();
        Console.WriteLine(newName);
    }
}
```

Digite um nome:
>Bruno Martins
Dtwpq Octvkpu

D:\Workspace\C#\AssesmentC\AssesmentC\bin\Debug\net9.0\AssesmentC.exe (processo 29428) encerrado com o código 0 (0x0). Para fechar o console automaticamente quando a depuração parar, habilite Ferramentas -> Opções -> Depuração -> Fechar o console automaticamente quando a depuração parar. Pressione qualquer tecla para fechar esta janela...

Exercício 3: Calculadora de Operações Matemáticas



The screenshot shows the Visual Studio IDE with a C# project named 'AssesmentC'. The file 'AssesmentC.Exercicio03.cs' is open, displaying the following code:

```
namespace Assessment;

public class Exercício03
{
    public static void Main(string[] args)
    {
        Console.WriteLine("Digite o primeiro número:\n> ");
        double firstNumber = double.Parse(Console.ReadLine());

        Console.WriteLine("Escolha a operação\n1 - Soma\n2 - Subtração\n3 - Multiplicação\n4 - Divisão:\n\t> ");
        string operation = Console.ReadLine();

        Console.WriteLine("Digite o segundo número:\n> ");
        double secondNumber = double.Parse(Console.ReadLine());

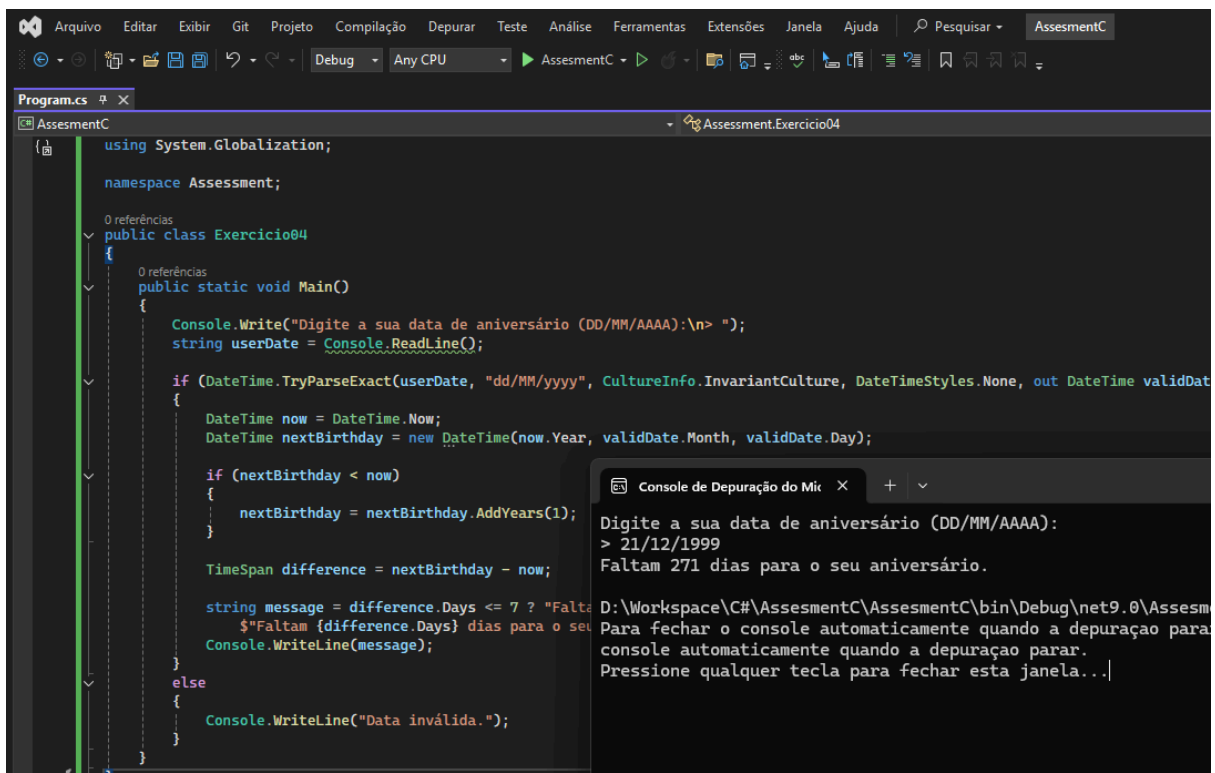
        switch (operation)
        {
            case "1":
                Console.WriteLine($"O resultado da soma é: {firstNumber + secondNumber}");
                break;
            case "2":
                Console.WriteLine($"O resultado da subtração é: {firstNumber - secondNumber}");
                break;
            case "3":
                Console.WriteLine($"O resultado da multiplicação é: {firstNumber * secondNumber}");
                break;
            case "4":
                if (secondNumber == 0)
                {
                    // Divisão por 0 não é permitida na matemática
                    Console.WriteLine("ERRO! Não é possível dividir por zero.");
                    break;
                }
                Console.WriteLine($"O resultado da divisão é: {firstNumber / secondNumber}");
                break;
            default:
                Console.WriteLine("Operação inválida.");
                break;
        }
    }
}
```

The 'Console de Depuração do Mix' window on the right shows the following output:

```
Digite o primeiro número:
> 25
Escolha a operação
1 - Soma
2 - Subtração
3 - Multiplicação
4 - Divisão:
> 4
Digite o segundo número:
> 2
O resultado da divisao é: 12,5

D:\Workspace\C#\AssesmentC\AssesmentC\bin\
Para fechar o console automaticamente quan
console automaticamente quando a depuração
Pressione qualquer tecla para fechar esta
```

Exercício 4: Manipulação de Datas - Dias até o Próximo Aniversário



The screenshot shows the Visual Studio IDE with a C# project named 'AssesmentC'. The file 'AssesmentC.Exercicio04.cs' is open, displaying the following code:

```
using System.Globalization;

namespace Assessment;

public class Exercício04
{
    public static void Main()
    {
        Console.WriteLine("Digite a sua data de aniversário (DD/MM/AAAA):\n> ");
        string userDate = Console.ReadLine();

        if (DateTime.TryParseExact(userDate, "dd/MM/yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime validDate))
        {
            DateTime now = DateTime.Now;
            DateTime nextBirthday = new DateTime(now.Year, validDate.Month, validDate.Day);

            if (nextBirthday < now)
            {
                nextBirthday = nextBirthday.AddYears(1);
            }

            TimeSpan difference = nextBirthday - now;

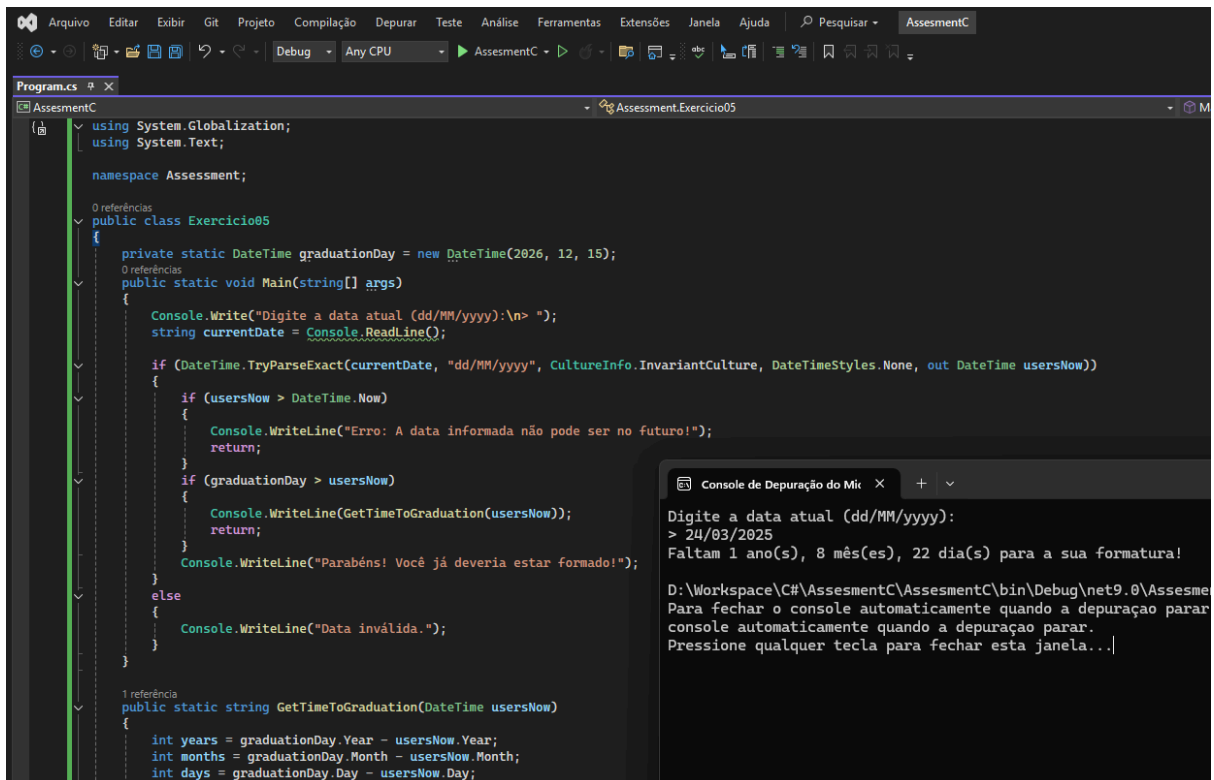
            string message = difference.Days <= 7 ? "Faltam " : "Faltam ";
            Console.WriteLine(message);
        }
        else
        {
            Console.WriteLine("Data inválida.");
        }
    }
}
```

The 'Console de Depuração do Mix' window on the right shows the following output:

```
Digite a sua data de aniversário (DD/MM/AAAA):
> 21/12/1999
Faltam 271 dias para o seu aniversário.

D:\Workspace\C#\AssesmentC\AssesmentC\bin\Debug\net9.0\AssesmentC.exe
Para fechar o console automaticamente quando a depuração para
console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela...
```

Exercício 5: Tempo Restante para Conclusão do Curso - Diferença Entre Datas



The screenshot shows the Visual Studio IDE with the file `AssesmentC\AssesmentC.Exercicio05` open. The code defines a class `Exercicio05` with a `Main` method that prompts the user for a date. It calculates the time remaining until a graduation date of December 15, 2026. The debug console shows the user inputting `24/03/2025`, and the program outputting the remaining time: 1 year, 8 months, and 22 days.

```
using System.Globalization;
using System.Text;

namespace Assessment;

public class Exercicio05
{
    private static DateTime graduationDay = new DateTime(2026, 12, 15);

    public static void Main(string[] args)
    {
        Console.WriteLine("Digite a data atual (dd/MM/yyyy):\n> ");
        string currentDate = Console.ReadLine();

        if (DateTime.TryParseExact(currentDate, "dd/MM/yyyy", CultureInfo.InvariantCulture, DateTimeStyles.None, out DateTime usersNow))
        {
            if (usersNow > DateTime.Now)
            {
                Console.WriteLine("Erro: A data informada não pode ser no futuro!");
                return;
            }
            if (graduationDay > usersNow)
            {
                Console.WriteLine(GetTimeToGraduation(usersNow));
                return;
            }
            Console.WriteLine("Parabéns! Você já deveria estar formado!");
        }
        else
        {
            Console.WriteLine("Data inválida.");
        }
    }

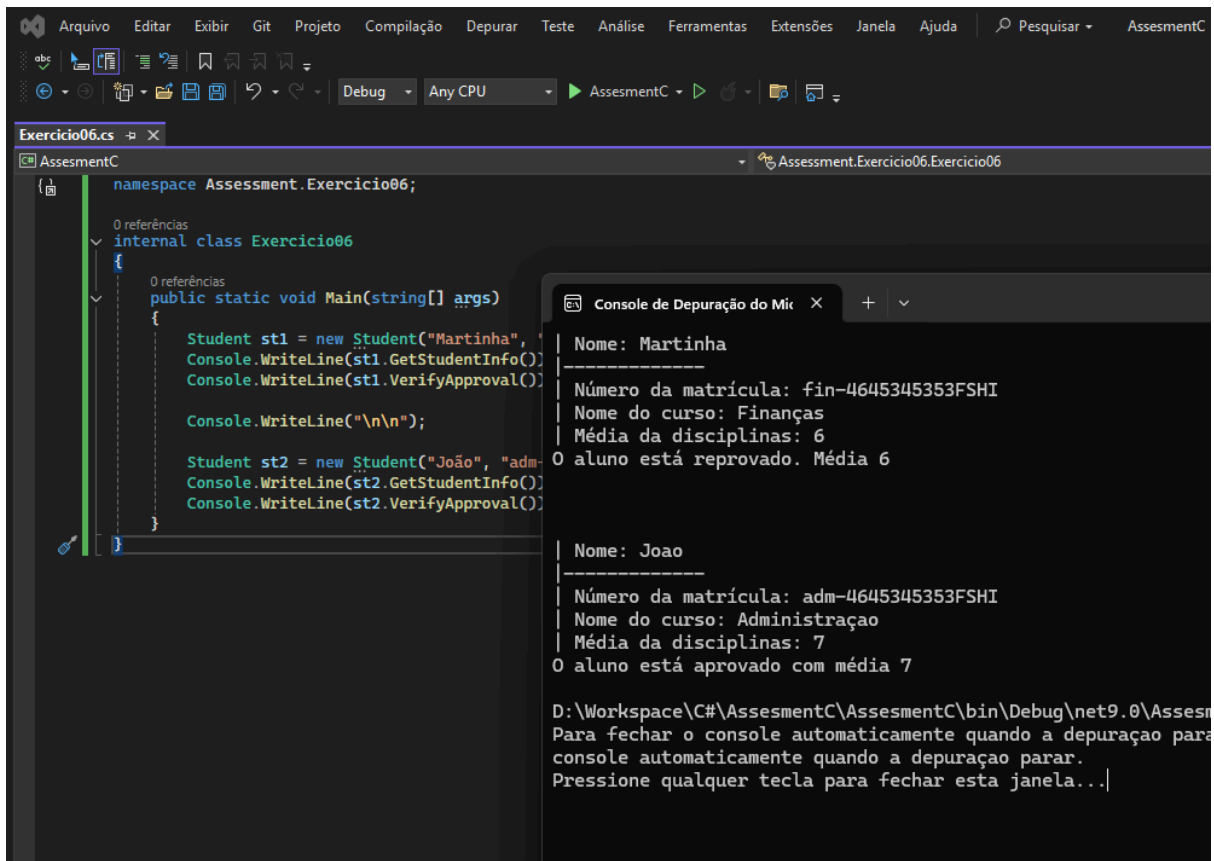
    public static string GetTimeToGraduation(DateTime usersNow)
    {
        int years = graduationDay.Year - usersNow.Year;
        int months = graduationDay.Month - usersNow.Month;
        int days = graduationDay.Day - usersNow.Day;
    }
}
```

Console de Depuração do Mik

```
Digite a data atual (dd/MM/yyyy):
> 24/03/2025
Faltam 1 ano(s), 8 mês(es), 22 dia(s) para a sua formatura!

D:\Workspace\C#\AssesmentC\AssesmentC\bin\Debug\net9.0\Assesme
Para fechar o console automaticamente quando a depuração parar
console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela...
```

Exercício 6: Cadastro de Alunos



The screenshot shows the Visual Studio IDE with the file `AssesmentC\AssesmentC.Exercicio06` open. The code defines a class `Exercicio06` with a `Main` method that creates two `Student` objects, `st1` and `st2`, and prints their information. The debug console shows the output for both students, including their names, IDs, courses, and average grades.

```
namespace Assessment.Exercicio06;

internal class Exercicio06
{
    public static void Main(string[] args)
    {
        Student st1 = new Student("Martinha", "fin-4645345353FSHI");
        Console.WriteLine(st1.GetStudentInfo());
        Console.WriteLine(st1.VerifyApproval());

        Console.WriteLine("\n\n");

        Student st2 = new Student("João", "adm-4645345353FSHI");
        Console.WriteLine(st2.GetStudentInfo());
        Console.WriteLine(st2.VerifyApproval());
    }
}
```

Console de Depuração do Mik

```
Nome: Martinha
-----
Número da matrícula: fin-4645345353FSHI
Nome do curso: Finanças
Média da disciplinas: 6
0 aluno está reprovado. Média 6

Nome: Joao
-----
Número da matrícula: adm-4645345353FSHI
Nome do curso: Administração
Média da disciplinas: 7
0 aluno está aprovado com média 7

D:\Workspace\C#\AssesmentC\AssesmentC\bin\Debug\net9.0\Assesme
Para fechar o console automaticamente quando a depuração parar
console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela...
```

Exercício 7: Banco Digital (Encapsulamento)

The screenshot shows the Visual Studio IDE with the file `Exercício07.cs` open. The code is in the `namespace Assessment.Exercício07` and contains an `internal class Exercício07` with a `Main` method. The `Main` method creates an `Account` object for João Silva with an initial balance of 1500.00. It then performs a deposit of 500.00, followed by two withdrawal attempts: 2500.00 (insufficient funds) and 200.00 (successful). The console output on the right shows the execution flow, including the initial balance, the deposit confirmation, and the withdrawal attempts and results.

```
namespace Assessment.Exercício07
{
    internal class Exercício07
    {
        public static void Main()
        {
            Account acc = new Account("João Silva", 1500.0);

            Console.WriteLine($"|Titular: {acc.ownerName}");
            Console.WriteLine($"|Saldo atual: R$ {acc.GetBalance()}\n");

            Console.WriteLine($"Depositando R$ 500...");
            Console.WriteLine(acc.Deposit(500.0));

            Console.WriteLine($"|Titular: {acc.ownerName}");
            Console.WriteLine($"|Saldo atual: R$ {acc.GetBalance()}\n");

            Console.WriteLine($"Tentativa de saque R$ 2500.00");
            Console.WriteLine(acc.Withdraw(2500.0));

            Console.WriteLine($"|Titular: {acc.ownerName}");
            Console.WriteLine($"|Saldo atual: R$ {acc.GetBalance()}\n");

            Console.WriteLine($"Tentativa de saque R$ 200.00");
            Console.WriteLine(acc.Withdraw(200.0));

            Console.WriteLine($"|Titular: {acc.ownerName}");
            Console.WriteLine($"|Saldo atual: R$ {acc.GetBalance()}\n");
        }
    }
}
```

Console de Depuração do Mic

```
|Titular: Joao Silva
|Saldo atual: R$ Saldo atual: R$ $1500

Depositando R$ 500...
Deposito de R$ 500 confirmado!
|Titular: Joao Silva
|Saldo atual: R$ Saldo atual: R$ $2000

Tentativa de saque R$ 2500.00
Saldo insuficiente para realizar o saque!
|Titular: Joao Silva
|Saldo atual: R$ Saldo atual: R$ $2000

Tentativa de saque R$ 200.00
Saque de R$ 200 confirmado!
|Titular: Joao Silva
|Saldo atual: R$ Saldo atual: R$ $1800

D:\Workspace\C#\Assessment\bin\Debug\net9.0\Assessment.exe
Para fechar o console automaticamente quando a depuração
console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela...
```

Exercício 8: Cadastro de Funcionários (Herança)

The screenshot shows the Visual Studio IDE with the file `Exercício08.cs` open. The code is in the `namespace Assessment.Exercício08` and contains an `internal class Exercício08` with a `Main` method. The `Main` method creates an `Employee` object for João (Developer, 5000.00) and a `Manager` object for Maria (Manager, 5000.00). The console output on the right shows the execution flow, including the creation of the objects and the output of the `GetEmpInfo` method for both.

```
namespace Assessment.Exercício08
{
    internal class Exercício08
    {
        public static void Main()
        {
            Employee emp = new Employee("João", "Developer", 5000.0);
            Console.WriteLine(emp.GetEmpInfo());

            Console.WriteLine();

            Manager manager = new Manager("Maria", "Manager", 5000.0);
            Console.WriteLine(manager.GetEmpInfo());
        }
    }
}
```

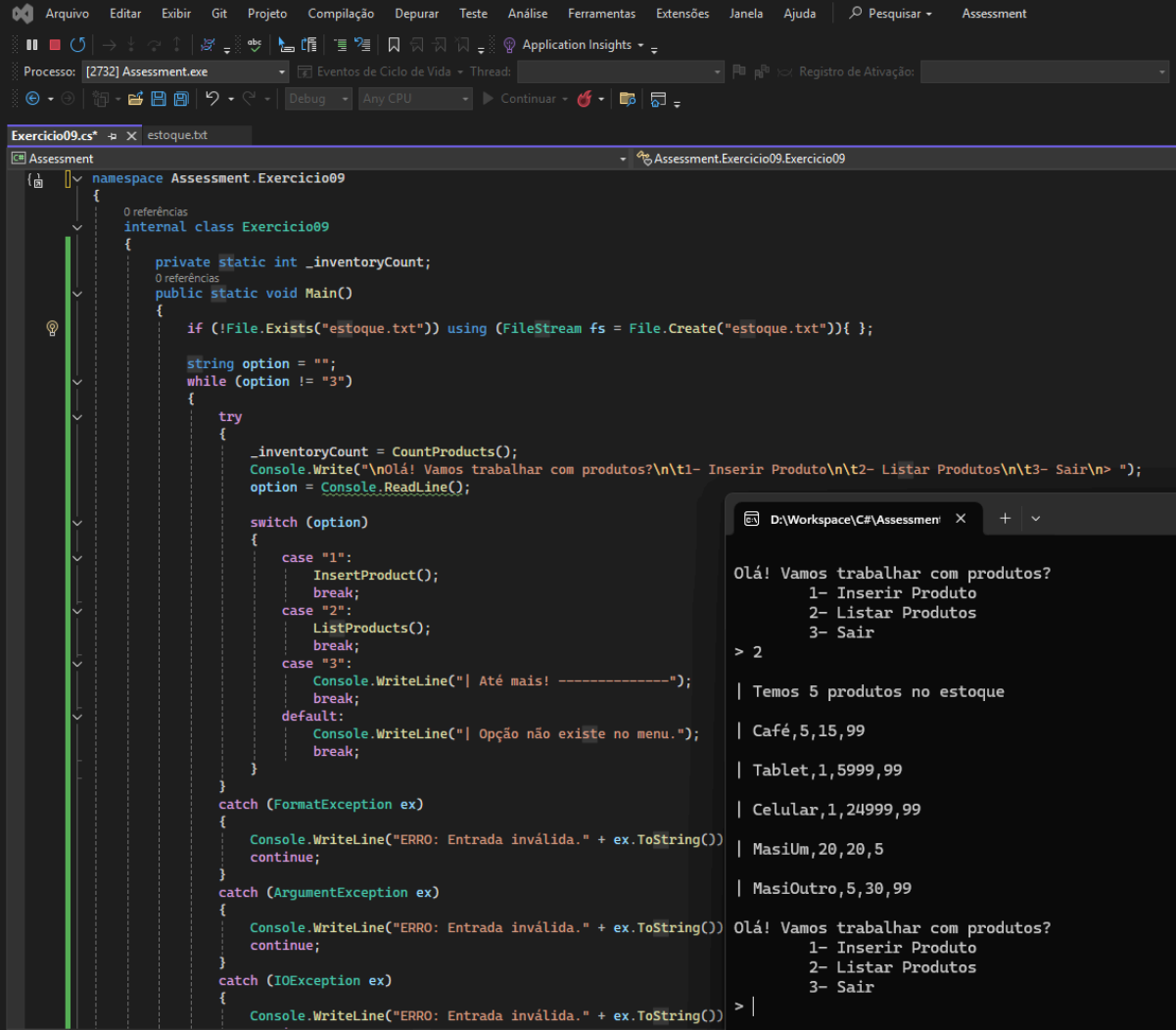
Console de Depuração do Mic

```
| Nome: Joao
| Cargo: Developer
| Salário Bruto: 5000

| Nome: Maria
| Cargo: Manager
| Salário Bruto: 5000

D:\Workspace\C#\Assessment\bin\Debug\net9.0\Assessment.exe
m o código 0 (0x0).
Para fechar o console automaticamente qua
Opções -> Depuração -> Fechar o console
Pressione qualquer tecla para fechar esta
```

Exercício 9: Controle de Estoque via Linha de Comando



The screenshot displays the Visual Studio IDE with the file `Exercício09.cs` open. The code defines an internal class `Exercicio09` with a static `_inventoryCount` and a `Main` method. The `Main` method prompts the user to work with products, offering options to insert, list, or exit. It uses a `switch` statement to handle these options, calling `InsertProduct()` or `ListProducts()` as appropriate. Error handling is implemented with `catch` blocks for `FormatException`, `ArgumentException`, and `IOException`.

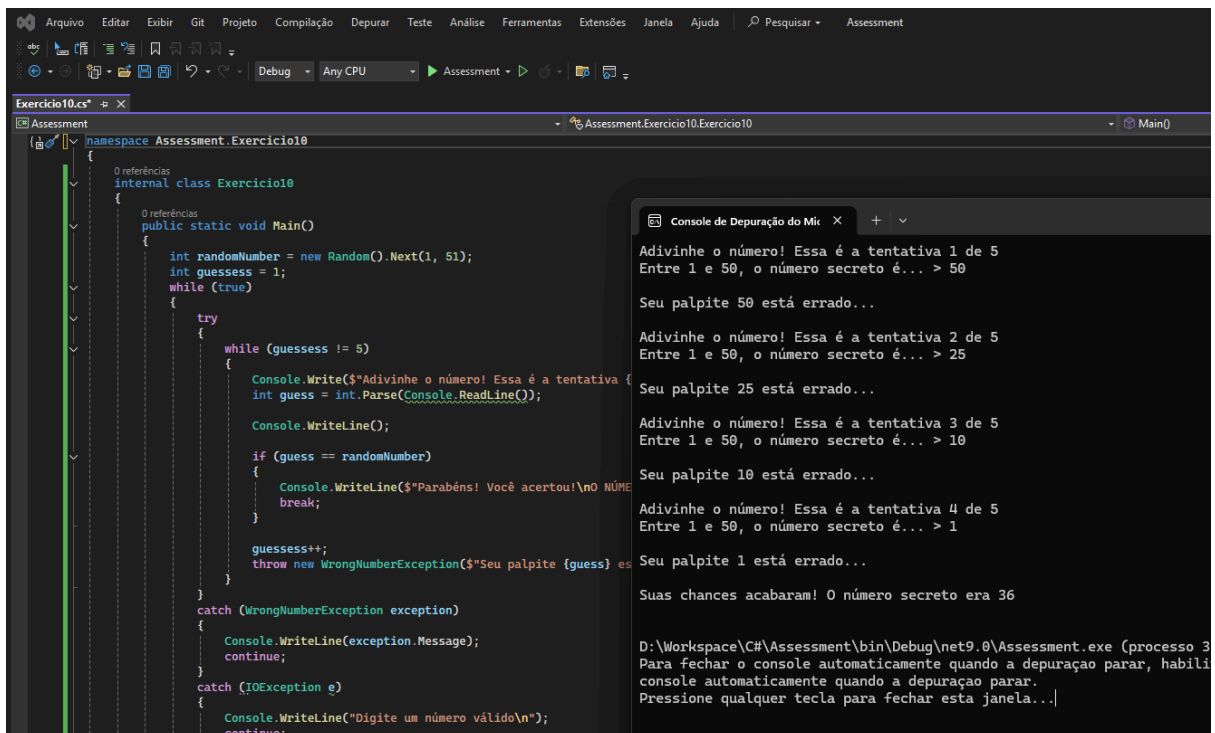
```
namespace Assessment.Exercicio09
{
    internal class Exercicio09
    {
        private static int _inventoryCount;
        public static void Main()
        {
            if (!File.Exists("estoque.txt")) using (FileStream fs = File.Create("estoque.txt")){ };

            string option = "";
            while (option != "3")
            {
                try
                {
                    _inventoryCount = CountProducts();
                    Console.WriteLine("\nOlá! Vamos trabalhar com produtos?\n\t1- Inserir Produto\n\t2- Listar Produtos\n\t3- Sair\n> ");
                    option = Console.ReadLine();

                    switch (option)
                    {
                        case "1":
                            InsertProduct();
                            break;
                        case "2":
                            ListProducts();
                            break;
                        case "3":
                            Console.WriteLine("\n| Até mais! -----");
                            break;
                        default:
                            Console.WriteLine("\n| Opção não existe no menu.");
                            break;
                    }
                }
                catch (FormatException ex)
                {
                    Console.WriteLine("ERRO: Entrada inválida." + ex.ToString());
                    continue;
                }
                catch (ArgumentException ex)
                {
                    Console.WriteLine("ERRO: Entrada inválida." + ex.ToString());
                    continue;
                }
                catch (IOException ex)
                {
                    Console.WriteLine("ERRO: Entrada inválida." + ex.ToString());
                }
            }
        }
    }
}
```

The console window shows the program's execution. It starts with the prompt "Olá! Vamos trabalhar com produtos?" and the menu options. The user enters "2" to list products, and the program displays the current inventory: 5 products. The list includes "Café,5,15,99", "Tablet,1,5999,99", "Celular,1,24999,99", "MasiUm,20,20,5", and "MasiOutro,5,30,99". The user then enters "3" to exit, and the program ends with the prompt "Olá! Vamos trabalhar com produtos?" and the menu options again.

Exercício 10: Jogo de Adivinhação



```
namespace Assessment.Exercício10
{
    internal class Exercício10
    {
        public static void Main()
        {
            int randomNumber = new Random().Next(1, 51);
            int guessess = 1;
            while (true)
            {
                try
                {
                    while (guessess != 5)
                    {
                        Console.WriteLine($"Adivinhe o número! Essa é a tentativa {guessess} de 5. Entre 1 e 50, o número secreto é... > ");
                        int guess = int.Parse(Console.ReadLine());

                        Console.WriteLine();

                        if (guess == randomNumber)
                        {
                            Console.WriteLine($"Parabéns! Você acertou! O número secreto era {randomNumber}");
                            break;
                        }

                        guessess++;
                        throw new WrongNumberException($"Seu palpite {guess} está errado...");
                    }
                }
                catch (WrongNumberException exception)
                {
                    Console.WriteLine(exception.Message);
                    continue;
                }
                catch (IOException ex)
                {
                    Console.WriteLine("Digite um número válido\n");
                    continue;
                }
            }
        }
    }
}
```

Console de Depuração do Mik

```
Adivinhe o número! Essa é a tentativa 1 de 5
Entre 1 e 50, o número secreto é... > 50

Seu palpite 50 está errado...

Adivinhe o número! Essa é a tentativa 2 de 5
Entre 1 e 50, o número secreto é... > 25

Seu palpite 25 está errado...

Adivinhe o número! Essa é a tentativa 3 de 5
Entre 1 e 50, o número secreto é... > 10

Seu palpite 10 está errado...

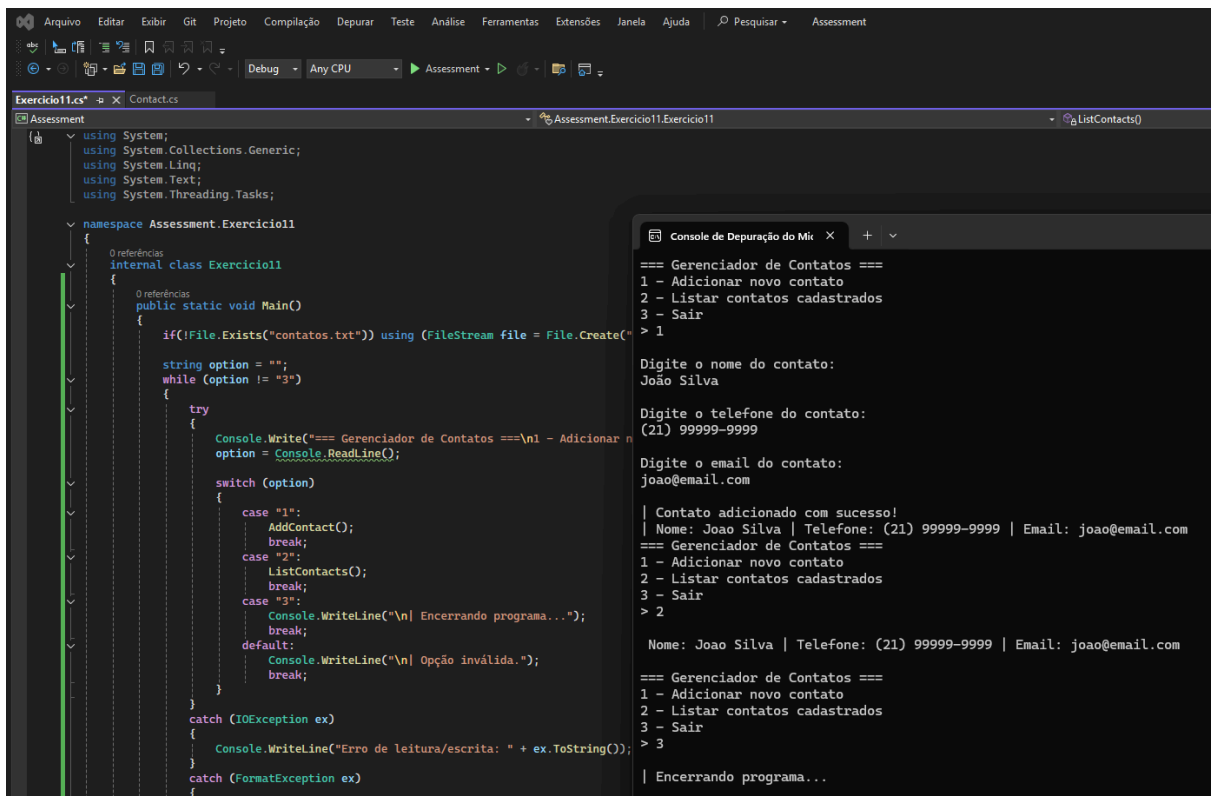
Adivinhe o número! Essa é a tentativa 4 de 5
Entre 1 e 50, o número secreto é... > 1

Seu palpite 1 está errado...

Suas chances acabaram! O número secreto era 36

D:\Workspace\C#\Assessment\bin\Debug\net9.0\Assessment.exe (processo 3)
Para fechar o console automaticamente quando a depuração parar, habilite
console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela...
```

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



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assessment.Exercício11
{
    internal class Exercício11
    {
        public static void Main()
        {
            if (!File.Exists("contatos.txt")) using (FileStream file = File.Create("contatos.txt"))
            {
            }

            string option = "";
            while (option != "3")
            {
                try
                {
                    Console.WriteLine("=== Gerenciador de Contatos ===\n1 - Adicionar novo contato\n2 - Listar contatos cadastrados\n3 - Sair\n> ");
                    option = Console.ReadLine();

                    switch (option)
                    {
                        case "1":
                            AddContact();
                            break;
                        case "2":
                            ListContacts();
                            break;
                        case "3":
                            Console.WriteLine("\n Encerrando programa...");
                            break;
                        default:
                            Console.WriteLine("\n Opção inválida.");
                            break;
                    }
                }
                catch (IOException ex)
                {
                    Console.WriteLine("Erro de leitura/escrita: " + ex.ToString());
                }
                catch (FormatException ex)
                {
                }
            }
        }

        static void AddContact()
        {
            Console.WriteLine("Digite o nome do contato:");
            string name = Console.ReadLine();

            Console.WriteLine("Digite o telefone do contato:");
            string phone = Console.ReadLine();

            Console.WriteLine("Digite o email do contato:");
            string email = Console.ReadLine();

            Console.WriteLine("Contato adicionado com sucesso!\nNome: {name} | Telefone: {phone} | Email: {email}");
        }

        static void ListContacts()
        {
            Console.WriteLine("=== Gerenciador de Contatos ===\n1 - Adicionar novo contato\n2 - Listar contatos cadastrados\n3 - Sair\n> ");
            string option = Console.ReadLine();

            while (option != "3")
            {
                try
                {
                    Console.WriteLine("Nome: Joao Silva | Telefone: (21) 99999-9999 | Email: joao@email.com");
                }
                catch (IOException ex)
                {
                    Console.WriteLine("Erro de leitura/escrita: " + ex.ToString());
                }
                catch (FormatException ex)
                {
                }
            }
        }
    }
}
```

Console de Depuração do Mik

```
=== Gerenciador de Contatos ===
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
> 1

Digite o nome do contato:
João Silva

Digite o telefone do contato:
(21) 99999-9999

Digite o email do contato:
joao@email.com

Contato adicionado com sucesso!
Nome: Joao Silva | Telefone: (21) 99999-9999 | Email: joao@email.com

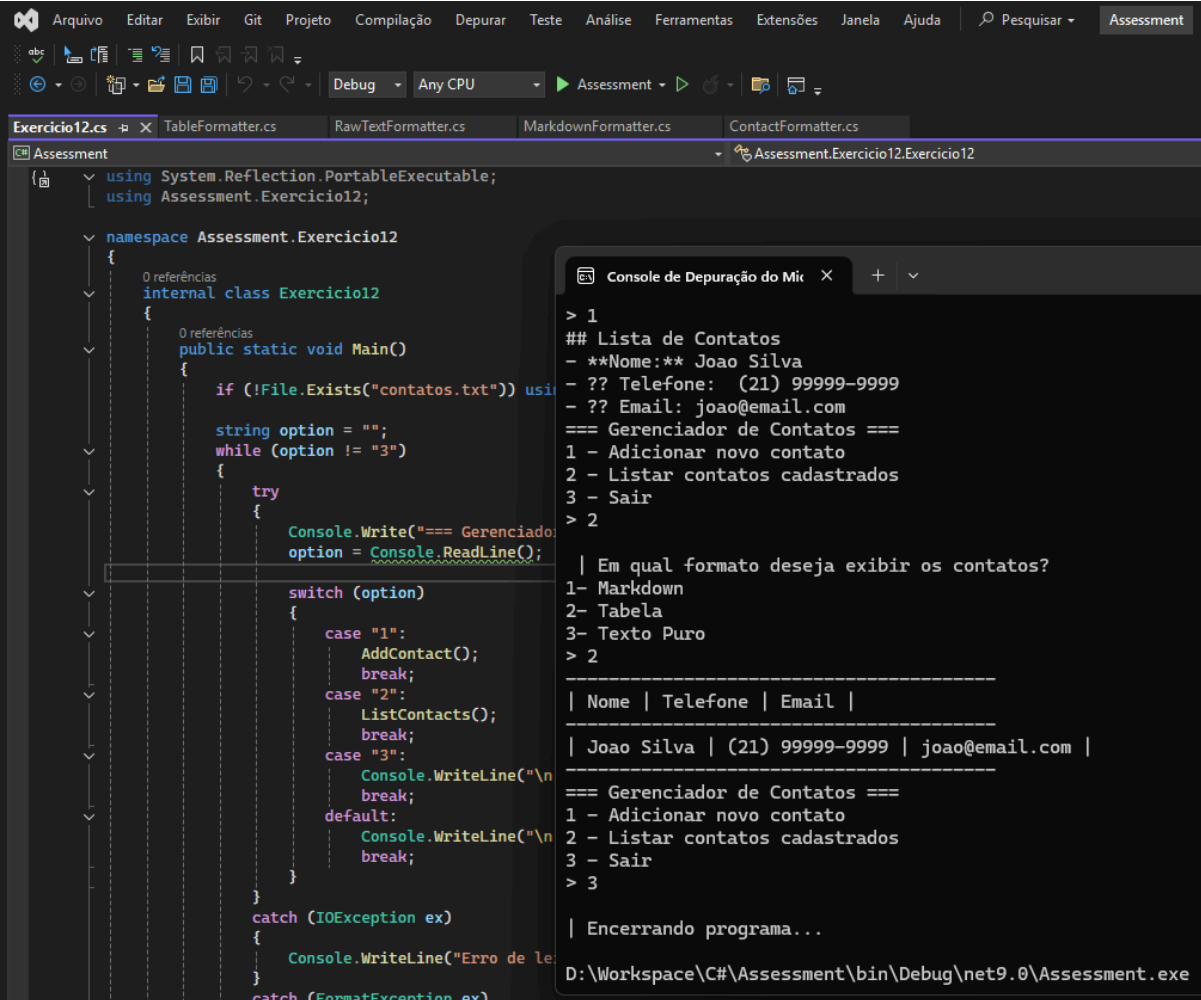
=== Gerenciador de Contatos ===
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
> 2

Nome: Joao Silva | Telefone: (21) 99999-9999 | Email: joao@email.com

=== Gerenciador de Contatos ===
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
> 3

Encerrando programa...
```

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



The screenshot shows the Visual Studio IDE with the file `Exercício12.cs` open. The code defines an internal class `Exercício12` with a `Main` method. The `Main` method checks if a file named `contatos.txt` exists. If it does, it reads the file and displays the contacts in a table format. If it doesn't, it prompts the user to add a new contact. The user enters the name `Joao Silva`, phone number `(21) 99999-9999`, and email `joao@email.com`. The program then displays the contacts in a table format.

```
using System.Reflection.PortableExecutable;
using Assessment.Exercício12;

namespace Assessment.Exercício12
{
    internal class Exercício12
    {
        public static void Main()
        {
            if (!File.Exists("contatos.txt"))
            {
                string option = "";
                while (option != "3")
                {
                    try
                    {
                        Console.WriteLine("=== Gerenciador de Contatos ===");
                        option = Console.ReadLine();

                        switch (option)
                        {
                            case "1":
                                AddContact();
                                break;
                            case "2":
                                ListContacts();
                                break;
                            case "3":
                                Console.WriteLine("\n");
                                break;
                            default:
                                Console.WriteLine("\n");
                                break;
                        }
                    }
                    catch (IOException ex)
                    {
                        Console.WriteLine("Erro de leitura");
                    }
                    catch (FormatException ex)
                    {
                        Console.WriteLine("Formato inválido");
                    }
                }
            }
            else
            {
                Console.WriteLine("Contatos encontrados no arquivo.");
                ListContacts();
            }
        }

        private static void AddContact()
        {
            Console.WriteLine("Adicionar novo contato");
            string nome = Console.ReadLine();
            string telefone = Console.ReadLine();
            string email = Console.ReadLine();
            AddContactToFile(nome, telefone, email);
        }

        private static void ListContacts()
        {
            Console.WriteLine("Listar contatos cadastrados");
            ListContactsFromFile();
        }

        private static void AddContactToFile(string nome, string telefone, string email)
        {
            using (StreamWriter writer = new StreamWriter("contatos.txt", true))
            {
                writer.WriteLine($"{nome};{telefone};{email}");
            }
        }

        private static void ListContactsFromFile()
        {
            using (StreamReader reader = new StreamReader("contatos.txt"))
            {
                while (!reader.EndOfStream)
                {
                    string line = reader.ReadLine();
                    string[] contact = line.Split(';');
                    Console.WriteLine($"{contact[0]} | {contact[1]} | {contact[2]}");
                }
            }
        }
    }
}
```

The console output shows the following sequence of events:

```
> 1
## Lista de Contatos
- **Nome:** Joao Silva
- ?? Telefone: (21) 99999-9999
- ?? Email: joao@email.com
=== Gerenciador de Contatos ===
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
> 2

| Em qual formato deseja exibir os contatos?
1- Markdown
2- Tabela
3- Texto Puro
> 2

-----
| Nome | Telefone | Email |
-----
| Joao Silva | (21) 99999-9999 | joao@email.com |
-----

=== Gerenciador de Contatos ===
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
> 3

| Encerrando programa...
D:\Workspace\C#\Assessment\bin\Debug\net9.0\Assessment.exe
```