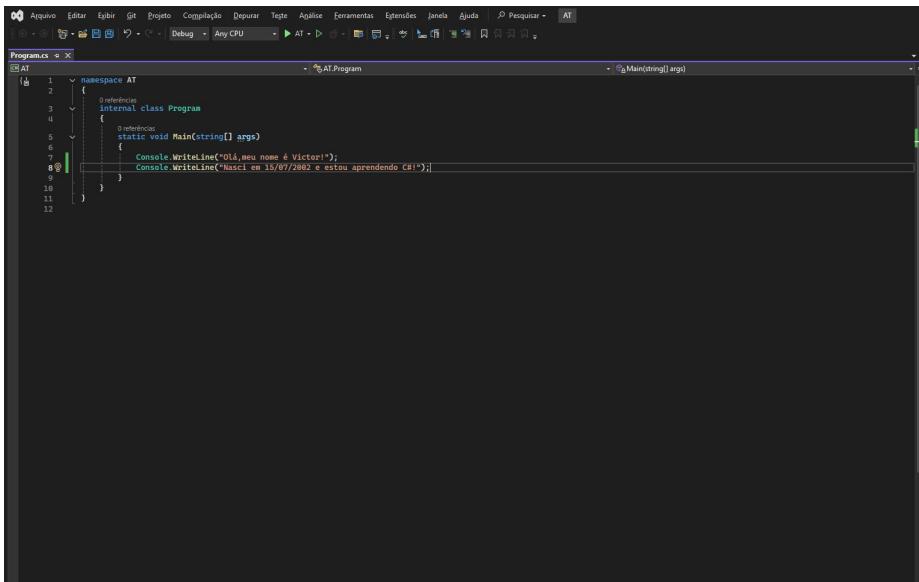


# Fundamentos de Desenvolvimento com C#

AT

Aluno: Victor Cezar

## Exercício 1: Criando e Executando seu Primeiro Programa



```
AT
1  namespace AT
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              Console.WriteLine("Olá, meu nome é Victor!");
8              Console.WriteLine("Nasci em 15/07/2002 e estou aprendendo C#!");
9          }
10     }
11 }
12 
```

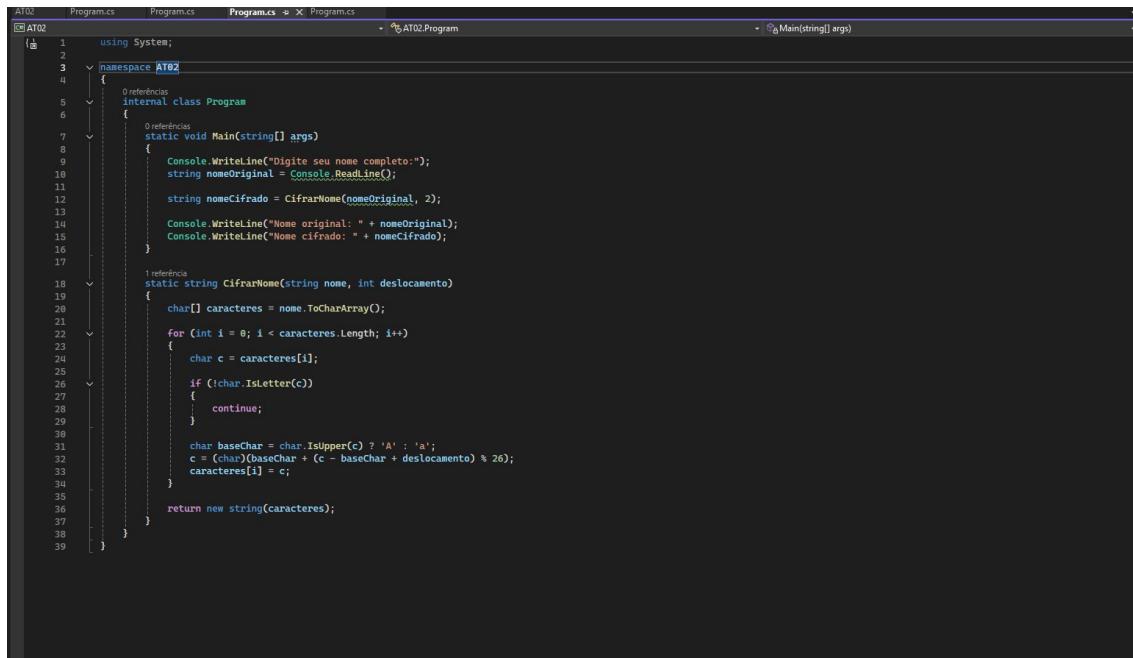
Resultado:



```
Olá, meu nome é Victor!
Nasci em 15/07/2002 e estou aprendendo C#!

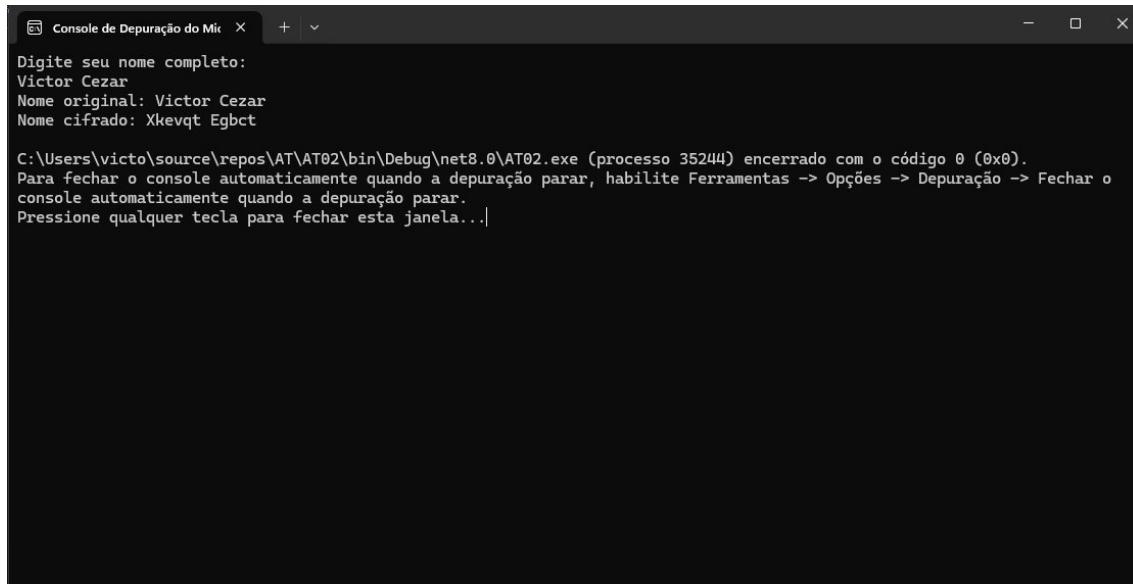
C:\Users\vitoco\source\repos\AT\AT\bin\Debug\net8.0\AT.exe (processo 32300) encerrado com o código 0 (0x0).
Para fechar o console automaticamente quando a depuração parar, habilite Ferramentas -> Opções -> Depuração -> Fechar o
console automaticamente quando a depuração parar.
Pressione qualquer tecla para fechar esta janela...]
```

## Exercício 2: Manipulação de Strings - Cifrador de Nome



```
AT02 Program.cs Program.cs Program.cs Program.cs
1  using System;
2
3  namespace AT02
4  {
5      internal class Program
6      {
7          static void Main(string[] args)
8          {
9              Console.WriteLine("Digite seu nome completo:");
10             string nomeOriginal = Console.ReadLine();
11
12             string nomeCifrado = CifrarNome(nomeOriginal, 2);
13
14             Console.WriteLine("Nome original: " + nomeOriginal);
15             Console.WriteLine("Nome cifrado: " + nomeCifrado);
16         }
17
18         static string CifrarNome(string nome, int deslocamento)
19         {
20             char[] caracteres = nome.ToCharArray();
21
22             for (int i = 0; i < caracteres.Length; i++)
23             {
24                 char c = caracteres[i];
25
26                 if (!char.IsLetter(c))
27                 {
28                     continue;
29                 }
30
31                 char baseChar = char.IsUpper(c) ? 'A' : 'a';
32                 c = (char)baseChar + (c - baseChar + deslocamento) % 26;
33                 caracteres[i] = c;
34             }
35
36         } return new string(caracteres);
37     }
38 }
```

Resultado:



```
Console de Depuração do Míx
Digite seu nome completo:
Victor Cezar
Nome original: Victor Cezar
Nome cifrado: Xkevqt Egbct

C:\Users\victo\source\repos\AT\AT02\bin\Debug\net8.0\AT02.exe (processo 35244) 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

```
AT03.cs
1  namespace AT03
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              double num1 = ObterNumeroValido("Digite o primeiro número:");
8              double num2 = ObterNumeroValido("Digite o segundo número:");
9
10             int opcao = ObterOperacaoValida();
11
12             double resultado = ExecutarOperacao(num1, num2, opcao);
13
14             ExibirResultado(resultado, opcao);
15         }
16     }
17
18     static double ObterNumeroValido(string mensagem)
19     {
20         double numero;
21         while(true)
22         {
23             Console.WriteLine(mensagem);
24             string entrada = Console.ReadLine();
25
26             if(double.TryParse(entrada, out numero))
27                 return numero;
28             else
29                 Console.WriteLine("Entrada inválida. Por favor, insira um número válido.");
30         }
31     }
32
33     static int ObterOperacaoValida()
34     {
35         int opcao;
36         while(true)
37         {
38             Console.WriteLine("Escolha a operação desejada:");
39             Console.WriteLine("1 - Soma");
40             Console.WriteLine("2 - Subtração");
41             Console.WriteLine("3 - Multiplicação");
42             Console.WriteLine("4 - Divisão");
43
44             string entrada = Console.ReadLine();
45
46             if(int.TryParse(entrada, out opcao) && opcao >= 1 && opcao <= 4)
47                 return opcao;
48             else
49                 Console.WriteLine("Opção inválida. Por favor, escolha uma operação entre 1 e 4");
50         }
51     }
52
53     static double ExecutarOperacao(double num1, double num2, int opcao)
54     {
55         switch (opcao)
56         {
57             case 1:
58                 return num1 + num2;
59             case 2:
60                 return num1 - num2;
61             case 3:
62                 return num1 * num2;
63             case 4:
64                 if (num2 == 0)
65                 {
66                     Console.WriteLine("\nErro: Não é possível dividir por zero.");
67                     return double.NaN;
68                 }
69                 return num1 / num2;
70             default:
71                 return double.NaN;
72         }
73     }
74
75     static void ExibirResultado(double resultado, int opcao)
76     {
77         if (resultado != double.NaN)
78         {
79             string operacao = opcao switch
80             {
81                 1 => "Soma";
82                 2 => "Subtração";
83                 3 => "Multiplicação";
84                 4 => "Divisão";
85                 _ => "Operação desconhecida";
86             };
87             Console.WriteLine($"O resultado da {operacao} é {resultado}");
88         }
89     }
90 }
91 
```

Resultado:

```
Console de Depuração do Microsoft Visual Studio
Digite o primeiro número:
5
Digite o segundo número:
4
Escolha a operação desejada:
1. Soma
2. Subtração
3. Multiplicação
4. Divisão
1

Resultado da Soma: 9
C:\Users\victo\source\repos\AT\AT03\bin\Debug\net8.0\AT03.exe (processo 31216) 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 4: Manipulação de Datas - Dias até o Próximo Aniversário

The screenshot shows the Microsoft Visual Studio IDE interface. The top menu bar has 'AT02' selected. There are three tabs open: 'Program.cs' (selected), 'Program.cs', and 'Program.cs'. The code editor displays C# code for a program named AT04. The code includes methods for obtaining birth date, calculating days until next birthday, and displaying the result.

```
AT02 Program.cs Program.cs Program.cs - ⑥ AT04.Program
1  namespace AT04
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              DateTime dataNascimento = ObterDataNascimento();
8
9              int diasFaltando = CalcularDiasParaProximoAniversario(dataNascimento);
10
11             ExibirResultado(diasFaltando);
12         }
13     }
14
15     static DateTime ObterDataNascimento()
16     {
17         DateTime dataNascimento;
18         while (true)
19         {
20             Console.WriteLine("Digite sua data de nascimento (formato: dd/MM/yyyy):");
21             string entrada = Console.ReadLine();
22
23             if (DateTime.TryParseExact(entrada, "dd/MM/yyyy", null, System.Globalization.DateTimeStyles.None, out dataNascimento))
24                 return dataNascimento;
25             else
26                 Console.WriteLine("Data inválida. Por favor, insira a data no formato correto.");
27         }
28     }
29
30     static int CalcularDiasParaProximoAniversario(DateTime dataNascimento)
31     {
32         DateTime dataAtual = DateTime.Today;
33
34         DateTime proximoAniversario = new DateTime(dataAtual.Year, dataNascimento.Month, dataNascimento.Day);
35
36         if (proximoAniversario < dataAtual)
37         {
38             proximoAniversario = proximoAniversario.AddYears(1);
39         }
40
41         return (proximoAniversario - dataAtual).Days;
42     }
43
44     static void ExibirResultado(int diasFaltando)
45     {
46         if (diasFaltando < 7)
47         {
48             Console.WriteLine($"\\nFaltam {diasFaltando} dias para o seu próximo aniversário.");
49         }
50         else
51         {
52             Console.WriteLine($"\\nFaltam {diasFaltando} dias para o seu próximo aniversário.");
53         }
54     }
55 }
```

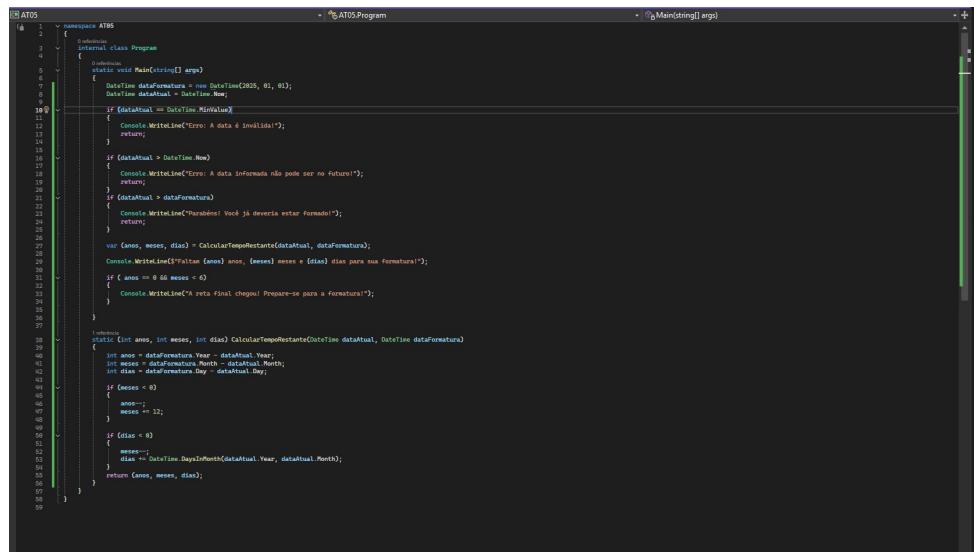
## Resultado

The screenshot shows the Windows Task Manager window titled 'Console de Depuração do Microsoft Visual Studio'. It displays the output of the application's execution. The user was prompted to enter their birth date, which they did as '15/07/2002'. The application then responded with 'Faltam 100 dias para o seu próximo aniversário.' (There are 100 days left until your next birthday).

```
Console de Depuração do Microsoft Visual Studio
Digite sua data de nascimento (formato: dd/MM/yyyy):
15/07/2002
Faltam 100 dias para o seu próximo aniversário.

C:\Users\victo\source\repos\AT\AT04\bin\Debug\net8.0\AT04.exe (processo 13912) 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 5: Tempo Restante para Conclusão do Curso - Diferença Entre Data



```
namespace AT05
{
    class Program
    {
        static void Main(string[] args)
        {
            DateTime dataFormatura = new DateTime(2025, 01, 01);
            DateTime dataAtual = DateTime.Now;

            if (dataAtual < DateTime.MinValue)
            {
                Console.WriteLine("Erro: A data é inválida!");
                return;
            }

            if (dataAtual > DateTime.MaxValue)
            {
                Console.WriteLine("Erro: A data informada não pode ser no futuro!");
                return;
            }

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

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

            if (anos == 0 & meses < 0)
            {
                Console.WriteLine("A reta final chegou! Prepare-se para a formatura!");
            }
        }

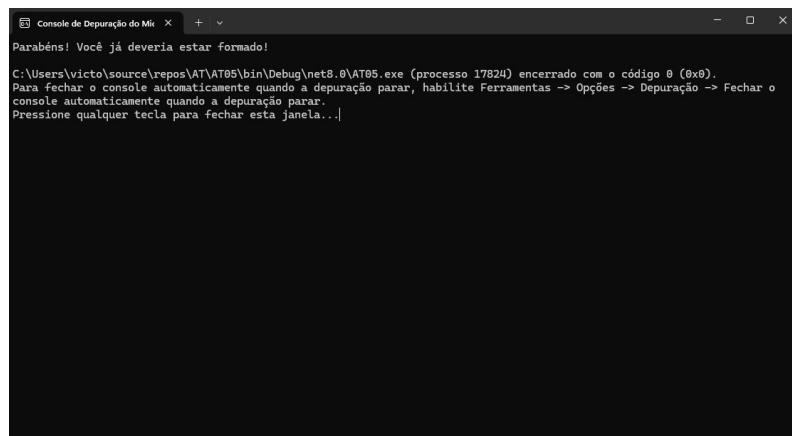
        static (int anos, int meses, int dias) CalcularTempoRestante(DateTime dataAtual, DateTime dataFormatura)
        {
            int anos = dataFormatura.Year - dataAtual.Year;
            int meses = dataFormatura.Month - dataAtual.Month;
            int dias = dataFormatura.Day - dataAtual.Day;

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

            if (dias < 0)
            {
                meses--;
                dias += DateTime.DaysInMonth(dataFormatura.Year, dataFormatura.Month);
            }
            return (anos, meses, dias);
        }
    }
}
```

### Resultados:

Data: 2025,01,01

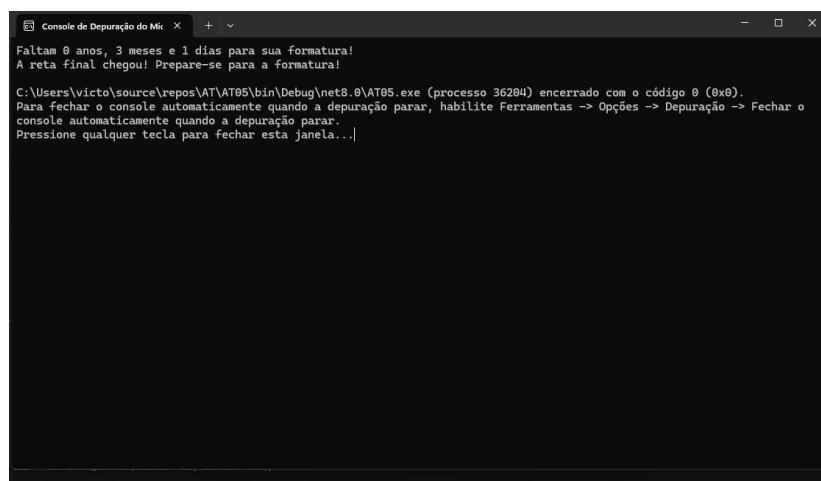


Console de Depuração do Míc

Parabéns! Você já deveria estar formado!

C:\Users\victo\source\repos\AT\AT05\bin\Debug\net8.0\AT05.exe (processo 17824) 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...

Data: 2025,07,07

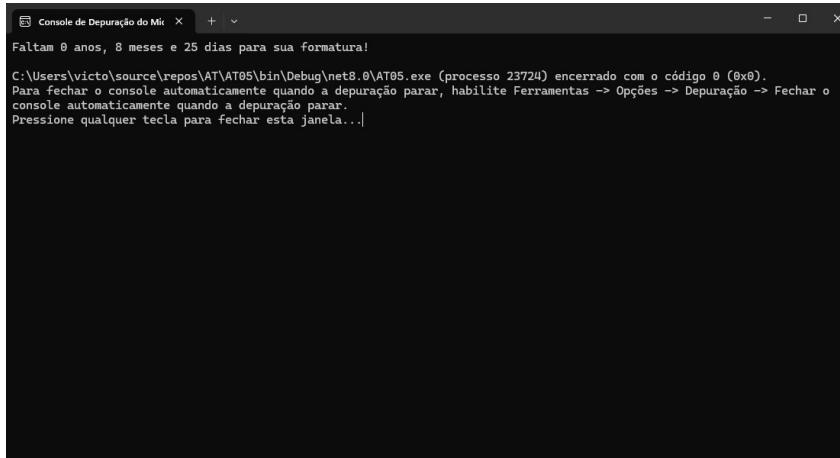


Console de Depuração do Míc

Faltam 0 anos, 3 meses e 1 dias para sua formatura!  
A reta final chegou! Prepare-se para a formatura!

C:\Users\victo\source\repos\AT\AT05\bin\Debug\net8.0\AT05.exe (processo 36204) 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...|

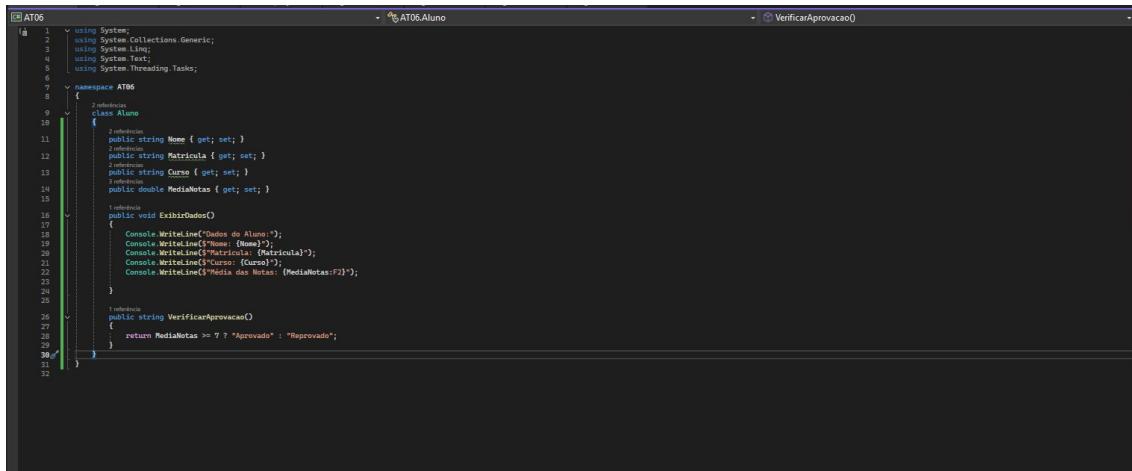
Data:2026,01,01



A screenshot of a Windows message box titled "Console de Depuração do Mi". The message reads: "Faltam 0 anos, 8 meses e 25 dias para sua formatura! C:\Users\vitco\source\repos\AT\AT05\bin\Debug\net8.0\AT05.exe (processo 23724) 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...".

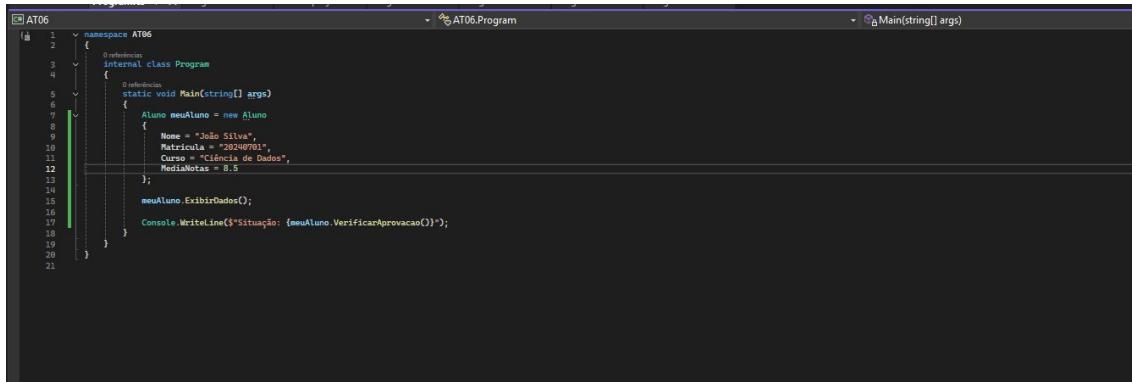
## Exercício 6: Cadastro de Alunos

### Classe Aluno:



```
AT06
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AT06
8  {
9      class Aluno
10     {
11         public string Nome { get; set; }
12         public string Matricula { get; set; }
13         public string Curso { get; set; }
14         public double MediaNotas { get; set; }
15
16         public void ExibirDados()
17         {
18             Console.WriteLine("Dados do Aluno:");
19             Console.WriteLine($"Nome: {Nome}");
20             Console.WriteLine($"Matrícula: {Matricula}");
21             Console.WriteLine($"Curso: {Curso}");
22             Console.WriteLine($"Média das Notas: {MediaNotas:F2}");
23         }
24
25         public string VerificarAprovacao()
26         {
27             if (MediaNotas >= 7)
28                 return "Aprovado";
29             else
30                 return "Reprovado";
31         }
32     }
}
```

### Classe Program:



```
AT06
1  namespace AT06
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              Aluno meuAluno = new Aluno
8              {
9                  Nome = "João Silva",
10                 Matricula = "38240701",
11                 Curso = "Ciência de Dados",
12                 MediaNotas = 8.5
13             };
14
15             meuAluno.ExibirDados();
16             Console.WriteLine($"Situação: {meuAluno.VerificarAprovacao()}");
17
18         }
19     }
20 }
```

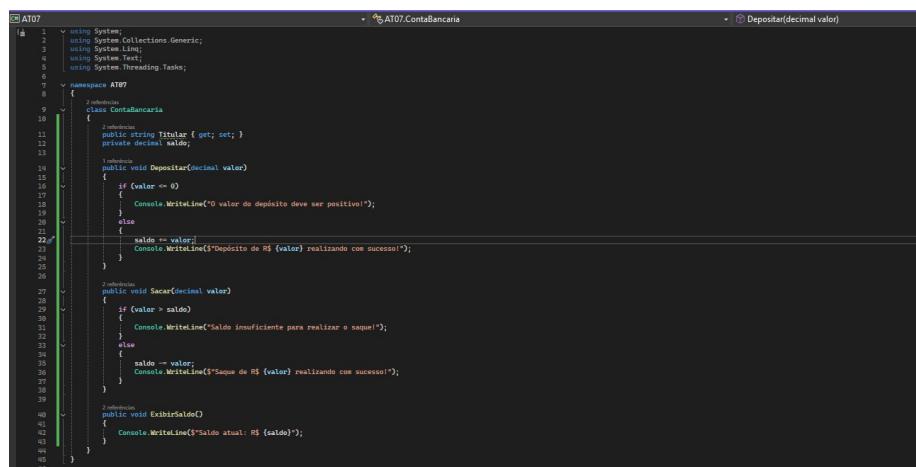
## Resultado:



```
Dados do Aluno:  
Nome: João Silva  
Matrícula: 20240701  
Curso: Ciência de Dados  
Média das Notas: 8.50  
Situação: Aprovado  
  
C:\Users\victo\source\repos\AT\AT06\bin\Debug\net8.0\AT06.exe (processo 11804) 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...|
```

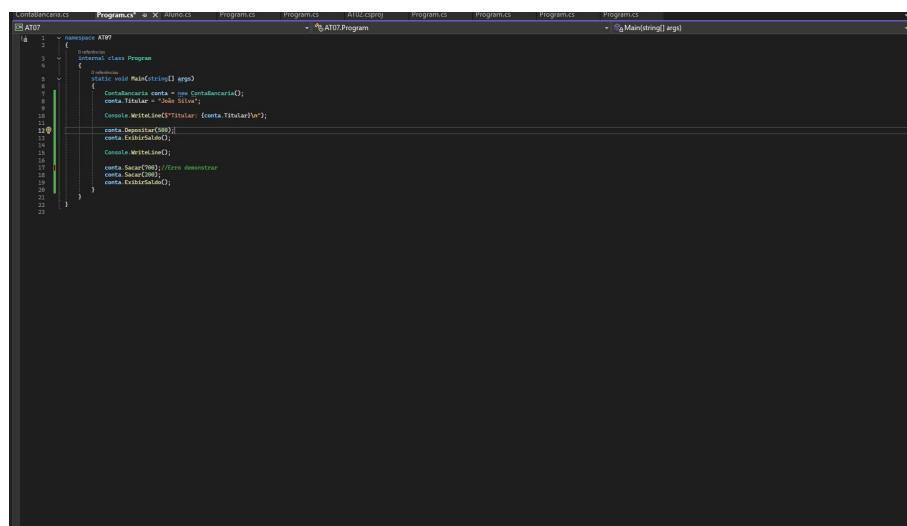
## Exercícios 7: Banco Digital (Encapsulamento)

### Classe ContaBancaria:



```
namespace AT07  
{  
    using System;  
    using System.Collections.Generic;  
    using System.Linq;  
    using System.Text;  
    using System.Threading.Tasks;  
  
    public class ContaBancaria  
    {  
        public string Titular { get; set; }  
        private decimal saldo;  
  
        public void Depositar(decimal valor)  
        {  
            if (valor <= 0)  
            {  
                Console.WriteLine("O valor do depósito deve ser positivo!");  
            }  
            else  
            {  
                saldo += valor;  
                Console.WriteLine($"Depósito de R$ {valor} realizado com sucesso!");  
            }  
        }  
  
        public void Sacar(decimal valor)  
        {  
            if (valor > saldo)  
            {  
                Console.WriteLine("Saldo insuficiente para realizar o saque!");  
            }  
            else  
            {  
                saldo -= valor;  
                Console.WriteLine($"Saque de R$ {valor} realizado com sucesso!");  
            }  
        }  
  
        public void ExibirSaldo()  
        {  
            Console.WriteLine($"Saldo atual: R$ {saldo}");  
        }  
    }  
}
```

### Classe Program:



```
internal class Program  
{  
    static void Main(string[] args)  
    {  
        ContaBancaria conta = new ContaBancaria();  
        conta.Titular = "João Silva";  
        Console.WriteLine($"Titular: {conta.Titular}");  
  
        conta.Depositar(1000);  
        conta.ExibirSaldo();  
        Console.WriteLine();  
        conta.Sacar(500);  
        conta.ExibirSaldo();  
    }  
}
```

## Resultado:

```
Console de Depuração do Mic X + - □ ×
Titular: João Silva
Depósito de R$ 500 realizando com sucesso!
Saldo atual: R$ 500
Saldo insuficiente para realizar o saque!
Saque de R$ 200 realizando com sucesso!
Saldo atual: R$ 300
C:\Users\victo\source\repos\AT\AT07\bin\Debug\net8.0\AT07.exe (processo 23172) 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...|
```

## Exercícios 8: Cadastro de Funcionários (Herança)

### Fucionário:

```
AT08
6  namespace AT08
7  {
8      public class Funcionario
9      {
10         public string Nome { get; set; }
11         public string Cargo { get; set; }
12         public decimal SalarioBase { get; set; }
13
14         public Funcionario(string nome, string cargo, decimal salarioBase)
15         {
16             Nome = nome;
17             Cargo = cargo;
18             SalarioBase = salarioBase;
19         }
20
21         public virtual decimal CalcularSalario()
22         {
23             return SalarioBase;
24         }
25     }
26 }
27
28
29
```

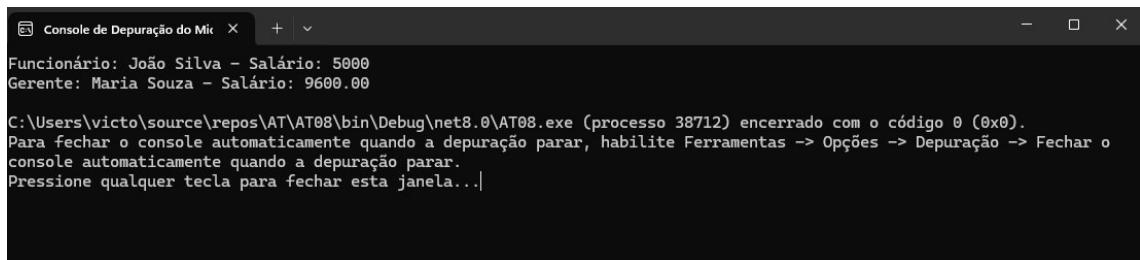
### Gerente:

```
AT08
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AT08
8  {
9      public class Gerente : Funcionario
10     {
11         private const decimal PercentualBonus = 0.28;
12
13         public Gerente(string nome, decimal salarioBase)
14         : base(nome, "Gerente", salarioBase)
15         {
16         }
17
18         public override decimal CalcularSalario()
19         {
20             return SalarioBase + (SalarioBase * PercentualBonus);
21         }
22     }
23 }
```

### Program:

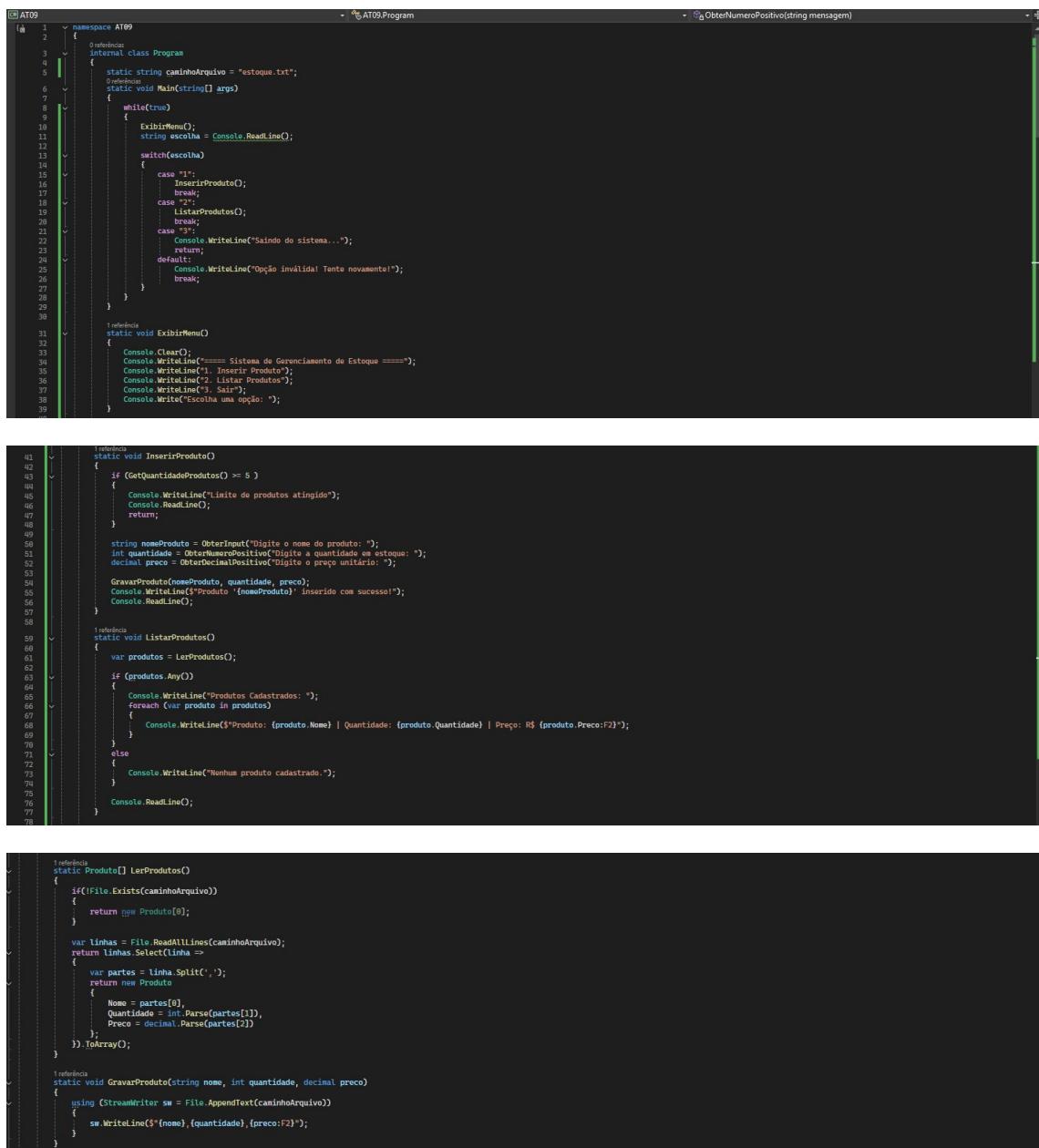
```
AT08
1  namespace AT08
2  {
3      internal class Program
4      {
5          static void Main(string[] args)
6          {
7              var funcionario = new Funcionario("João Silva", "Analista", 5000);
8              var gerente = new Gerente("Maria Souza", 6000);
9
10             Console.WriteLine($"Funcionario: {funcionario.Nome} - Salario: {funcionario.CalcularSalario()}");
11             Console.WriteLine($"Gerente: {gerente.Nome} - Salario: {gerente.CalcularSalario()}");
12         }
13     }
14 }
```

## Resultado:



```
Console de Depuração do Mi... + x
Funcionário: João Silva - Salário: 5000
Gerente: Maria Souza - Salário: 9600.00
C:\Users\victo\source\repos\AT\AT08\bin\Debug\net8.0\AT08.exe (processo 38712) 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...
```

## Exercícios 9 : Controle de Estoque via Linha de Comando



```
namespace AT09
{
    internal class Program
    {
        static string caminhoArquivo = "estoque.txt";
        static void Main(string[] args)
        {
            while(true)
            {
                ExibirMenu();
                string escolha = Console.ReadLine();
                switch(escolha)
                {
                    case "1":
                        InserirProduto();
                        break;
                    case "2":
                        ListarProdutos();
                        break;
                    case "3":
                        Console.WriteLine("Saindo do sistema...");
                        return;
                    default:
                        Console.WriteLine("Opção inválida! Tente novamente!");
                        break;
                }
            }
        }

        static void ExibirMenu()
        {
            Console.Clear();
            Console.WriteLine("===== Sistema de Gerenciamento de Estoque =====");
            Console.WriteLine("1. Inserir Produto");
            Console.WriteLine("2. Listar Produtos");
            Console.WriteLine("3. Sair");
            Console.WriteLine("Escolha uma opção: ");
        }

        static void InserirProduto()
        {
            if (GetQuantidadeProdutos() >= 5)
            {
                Console.WriteLine("Limite de produtos atingido");
                Console.ReadLine();
                return;
            }

            string nomeProduto = ObterInput("Digite o nome do produto: ");
            int quantidade = ObterNumeroPositivo("Digite a quantidade em estoque: ");
            decimal preco = ObterDecimalPositivo("Digite o preço unitário: ");

            GravarProduto(nomeProduto, quantidade, preco);
            Console.WriteLine($"Produto '{nomeProduto}' inserido com sucesso!");
            Console.ReadLine();
        }

        static void ListarProdutos()
        {
            var produtos = LerProdutos();
            if (produtos.Any())
            {
                Console.WriteLine("Produtos Cadastrados:");
                foreach (var produto in produtos)
                {
                    Console.WriteLine($"{produto.Nome} | Quantidade: {produto.Quantidade} | Preço: R$ {produto.Preco:F2}");
                }
            }
            else
            {
                Console.WriteLine("Nenhum produto cadastrado.");
            }
            Console.ReadLine();
        }

        static Produto[] LerProdutos()
        {
            if (!File.Exists(caminhoArquivo))
            {
                return new Produto[0];
            }

            var linhas = File.ReadAllLines(caminhoArquivo);
            return linhas.Select(LinhaToProduto);
        }

        static Produto LinhaToProduto(string linha)
        {
            var partes = linha.Split(',');
            return new Produto
            {
                Nome = partes[0],
                Quantidade = int.Parse(partes[1]),
                Preco = decimal.Parse(partes[2])
            };
        }

        static void GravarProduto(string nome, int quantidade, decimal preco)
        {
            using (StreamWriter sw = File.AppendText(caminhoArquivo))
            {
                sw.WriteLine($"{nome},{quantidade},{preco:F2}");
            }
        }
    }
}
```

```
1 referência
static int ObterNumeroPositivo(string mensagem)
{
    int numero;
    do
    {
        Console.WriteLine(mensagem);
        if (!int.TryParse(Console.ReadLine(), out numero) || numero <= 0);
    } while (!int.TryParse(Console.ReadLine(), out numero) || numero <= 0);
    return numero;
}

1 referência
static decimal ObterDecimalPositivo(string mensagem)
{
    decimal numero;
    do
    {
        Console.WriteLine(mensagem);
        if (!decimal.TryParse(Console.ReadLine(), out numero) || numero <= 0);
    } while (!decimal.TryParse(Console.ReadLine(), out numero) || numero <= 0);
    return numero;
}

1 referência
static string ObterInput(string mensagem)
{
    Console.WriteLine(mensagem);
    return Console.ReadLine();
}

1 referência
static int GetQuantidadeDeProdutos()
{
    return File.Exists(caminhoArquivo) ? File.ReadAllLines(caminhoArquivo).Length : 0;
}
```

```
102
103
104
105
106
107
108
109
110
111
```

Inserir:

```
== C:\Users\victo\source\repos\ ==
===== Sistema de Gerenciamento de Estoque =====
1. Inserir Produto
2. Listar Produtos
3. Sair
Escolha uma opção: 1
Digite o nome do produto: açaí
Digite a quantidade em estoque: 50
Digite o preço unitário: 10
Produto 'açaí' inserido com sucesso!
```

Listar:

```
== C:\Users\victo\source\repos\ ==
===== Sistema de Gerenciamento de Estoque =====
1. Inserir Produto
2. Listar Produtos
3. Sair
Escolha uma opção: 2
Produtos Cadastrados:
Produto: Cacau | Quantidade: 50 | Preço: R$ 2.00
Produto: Banana | Quantidade: 100 | Preço: R$ 5.00
Produto: açaí | Quantidade: 50 | Preço: R$ 10.00
```

## Exercícios 10: Jogo de Adivinhação

## JogoAdivinhacao:

```
AT10 1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AT10
8  {
9      class JogoAdivinhação
10     {
11         private const int MinNúmero = 1;
12         private const int MaxNúmero = 50;
13         private const int MaxTentativas = 5;
14
15         private readonly int númeroSecreto;
16         private int tentativasRestantes;
17
18         public JogoAdivinhação()
19         {
20             var random = new Random();
21             númeroSecreto = random.Next(MinNúmero, MaxNúmero + 1);
22             tentativasRestantes = MaxTentativas;
23         }
24
25         public void Iniciar()
26         {
27             Console.WriteLine($"Bem-vindo ao jogo de adivinhação! Tente acertar um número entre {MinNúmero} e {MaxNúmero}.");
28             Console.WriteLine($"Você tem {MaxTentativas} tentativas. Boa sorte!\n");
29
30             while (tentativasRestantes > 0)
31             {
32                 Console.WriteLine($"Tentativa {MaxTentativas - tentativasRestantes + 1} de {MaxTentativas}");
33                 Console.WriteLine("Digite seu palpite:");
34
35                 try
36                 {
37                     int palpite = ObterPalpiteValido();
38
39                     if(palpite == númeroSecreto)
40                     {
41                         Console.WriteLine($"Parabéns! Você acertou o número secreto {númeroSecreto}!");
42                         return;
43                     }
44
45                     tentativasRestantes--;
46
47                     if (tentativasRestantes > 0)
48                     {
49                         string dica = palpite < númeroSecreto ? "maior" : "menor";
50                         Console.WriteLine($"Errou! O número secreto é {dica} que {palpite}.");
51                         Console.WriteLine($"Você ainda tem {tentativasRestantes} tentativa(s)\n");
52                     }
53
54                 }
55
56                 catch (ArgumentException ex)
57                 {
58                     Console.WriteLine($"Erro: {ex.Message}\n");
59                 }
60             }
61             Console.WriteLine($"Game Over! Você esgotou todas as tentativas. O número secreto era {númeroSecreto}.");
62         }
63
64         private int ObterPalpiteValido()
65         {
66             if(!int.TryParse(Console.ReadLine(), out int palpite))
67             {
68                 throw new ArgumentException("Por favor, digite um número válido.");
69             }
70
71             if (palpite < MinNúmero || palpite > MaxNúmero)
72             {
73                 throw new ArgumentException($"O palpite deve estar entre {MinNúmero} e {MaxNúmero}.");
74             }
75
76             return palpite;
77         }
78     }
79 }
```

## Program:



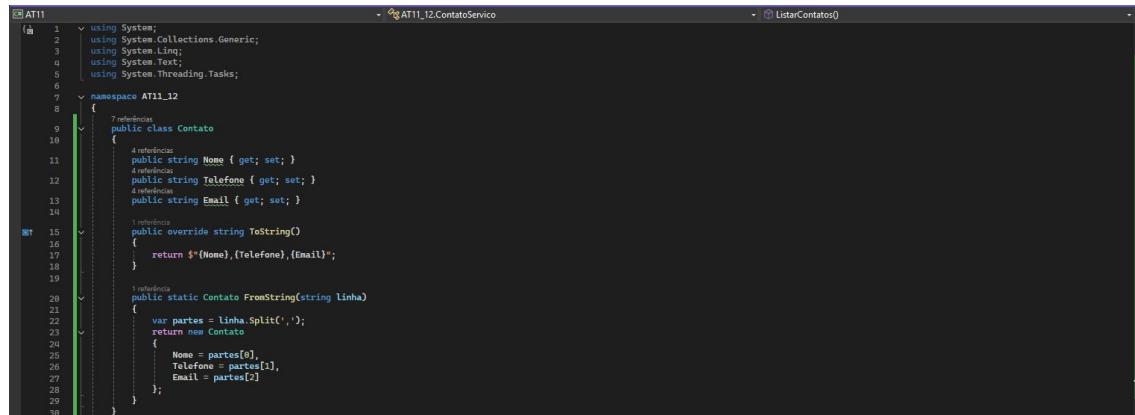
```
AT10
namespace AT10
{
    internal class Program
    {
        static void Main(string[] args)
        {
            var jogo = new Jogo();
            jogo.Iniciar();
            Console.WriteLine("\nPressione qualquer tecla para sair...");
            Console.ReadKey();
        }
    }
}
```

## Resultado:

```
C:\Users\Guilherme\PycharmProjects\ExerciciosPython> x + -  
|  
| Errou! O número secreto é menor que 48.  
| Você ainda tem 4 tentativa(s).  
| Tentativa 2 de 5  
| Dígite seu palpite:  
| 20  
| Errou! O número secreto é maior que 20.  
| Você ainda tem 3 tentativa(s).  
| Tentativa 3 de 5  
| Dígite seu palpite:  
| Errou! O número secreto é menor que 39.  
| Você ainda tem 2 tentativa(s).  
| Dígite seu palpite:  
| 30  
| Dígite seu palpite:  
| Errou! O número secreto é menor que 24.  
| Você ainda tem 1 tentativa(s).  
| Tentativa 4 de 5  
| Dígite seu palpite:  
| 22  
| Game Over! Você esgotou todas as tentativas. O número secreto era 21.  
| Pressione qualquer tecla para sair...  
|
```

## Exercícios 11: Manipulação de Arquivos - Cadastro e Listagem de Contatos

Contato:



```
AT11 1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.Threading.Tasks;
6
7  namespace AT11_12
8  {
9      public class Contato
10     {
11         public string Nome { get; set; }
12         public string Telefone { get; set; }
13         public string Email { get; set; }
14
15         public override string ToString()
16         {
17             return $"{Nome}, {Telefone}, {Email}";
18         }
19
20         public static Contato FromString(string linha)
21         {
22             var partes = linha.Split(',');
23             return new Contato
24             {
25                 Nome = partes[0],
26                 Telefone = partes[1],
27                 Email = partes[2]
28             };
29         }
30     }
31
32     public class ContatoRepository
33     {
34         private const string ArquivoContatos = "contatos.txt";
35
36         public void Adicionar(Contato contato)
37         {
38             try
39             {
40                 using (StreamWriter sw = new StreamWriter(ArquivoContatos, true))
41                 {
42                     sw.WriteLine(contato.ToString());
43                 }
44             }
45             catch (Exception ex)
46             {
47                 Console.WriteLine($"Erro ao salvar contato: {ex.Message}");
48             }
49         }
50
51         public List<Contato> ListarTodos()
52         {
53             var contatos = new List<Contato>();
54
55             if (!File.Exists(ArquivoContatos))
56             {
57                 return contatos;
58             }
59
60             try
61             {
62                 using (StreamReader sr = new StreamReader(ArquivoContatos))
63                 {
64                     string linha;
65                     while ((linha = sr.ReadLine()) != null)
66                     {
67                         if (!string.IsNullOrWhiteSpace(linha))
68                         {
69                             contatos.Add(Contato.FromString(linha));
70                         }
71                     }
72                 }
73             }
74             catch (Exception ex)
75             {
76                 Console.WriteLine($"Erro ao ler contatos: {ex.Message}");
77             }
78
79             return contatos;
80         }
81     }
82
83     public class ContatoService
84     {
85         private readonly ContatoRepository _repositorio;
86
87         public ContatoService()
88         {
89             _repositorio = new ContatoRepository();
90         }
91
92         public void AdicionarContato()
93         {
94             Console.Write("Nome: ");
95             string nome = Console.ReadLine();
96
97             Console.Write("Telefone: ");
98             string telefone = Console.ReadLine();
99
100            Console.Write("Email: ");
101            string email = Console.ReadLine();
102
103            var contato = new Contato
104            {
105                Nome = nome,
106                Telefone = telefone,
107                Email = email
108            };
109
110            _repositorio.Adicionar(contato);
111            Console.WriteLine("Contato cadastrado com sucesso!");
112        }
113
114         public void ListarContatos()
115         {
116             var contatos = _repositorio.ListarTodos();
117
118             if (contatos.Count == 0)
119             {
120                 Console.WriteLine("Nenhum contato cadastrado.\n");
121                 return;
122             }
123
124             Console.WriteLine("Contatos cadastrados:");
125             foreach (var contato in contatos)
126             {
127                 Console.WriteLine($"{Nome: contato.Nome} | Telefone: {contato.Telefone} | Email: {contato.Email}");
128             }
129             Console.WriteLine();
130         }
131
132     }
133
134 }
```

## Program:

```
1  using AT11_02;
2
3  namespace AT11
4  {
5      internal class Program
6      {
7          private static readonly ContatoService _contatoService = new ContatoService();
8
9          static void Main(string[] args)
10         {
11             bool sair = false;
12
13             while (!sair)
14             {
15                 ExibirMenu();
16                 string opcao = Console.ReadLine();
17
18                 switch (opcao)
19                 {
20                     case "1":
21                         _contatoService.AdicionarContato();
22                         break;
23                     case "2":
24                         _contatoService.ListarContatos();
25                         break;
26                     case "3":
27                         sair = true;
28                         Console.WriteLine("Encerrando programa...");
29                         break;
30                     default:
31                         Console.WriteLine("Opção inválida. Tente novamente.\n");
32                         break;
33                 }
34             }
35
36             ExibirMenu();
37         }
38     }
39 }
```

## Resultado:

### Adicionar:

```
== Gerenciador de Contatos ==
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
Escolha uma opção: 1
Nome: Victor
Telefone: 219999999999
Email: victorcezar@gmail.com
Contato cadastrado com sucesso!
```

### Listar:

```
== Gerenciador de Contatos ==
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
Escolha uma opção: 2
Contatos cadastrados:
Nome: Victor | Telefone: 2199038-8494 | Email: victorcesar212@gmail.com
Nome: Victor | Telefone: 219999999999 | Email: victorcezar@gmail.com

== Gerenciador de Contatos ==
1 - Adicionar novo contato
2 - Listar contatos cadastrados
3 - Sair
Escolha uma opção: |
```

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

Contato:

The screenshot shows a C# development environment with two open files:

- AT12**: This file contains the definition of the `Contato` class and its methods. It includes an overridden `ToString()` method that returns a string in the format `{Nome}, {Telefone}, {Email}`. It also contains a static method `FromString(string linha)` which splits the input string by comma and creates a new `Contato` object with the extracted values.
- AT12.ContatoServico**: This file defines an abstract class `ContatoFormatter` with an abstract method `ExibirContatos(List<Contato> contatos)`. It also contains two concrete implementations:
  - `MarkdownFormatter`: Implements `ContatoFormatter` and overrides `ExibirContatos` to output each contact on a new line in Markdown format: `## Lista de Contatos`, followed by the contact details separated by commas.
  - `TabelaFormatter`: Implements `ContatoFormatter` and overrides `ExibirContatos` to output a table where each row represents a contact. The columns are `Nome`, `Telefone`, and `Email`. The widths of the columns are calculated based on the maximum length of the corresponding field across all contacts. The table is separated by a separator line.

```

1  /*
2  * Author: Bruno Henrique
3  */
4
5  using System;
6  using System.Collections.Generic;
7
8  namespace AT12
9  {
10    public class Contato
11    {
12      public string Nome { get; set; }
13      public string Telefone { get; set; }
14      public string Email { get; set; }
15    }
16
17    public class ContatoRepository
18    {
19      private List<Contato> contatos = new List<Contato>();
20
21      public void AdicionarContato()
22      {
23        var contato = new Contato();
24
25        Console.WriteLine("Nome: ");
26        string nome = Console.ReadLine();
27
28        Console.WriteLine("Telefone: ");
29        string telefone = Console.ReadLine();
30
31        Console.WriteLine("Email: ");
32        string email = Console.ReadLine();
33
34        contato.Nome = nome;
35        contato.Telefone = telefone;
36        contato.Email = email;
37
38        _repository.Adicionar(contato);
39        Console.WriteLine("Contato cadastrado com sucesso!");
40      }
41
42      public void ListarContatos()
43      {
44        var contatos = _repository.ListarTodos();
45
46        if (contatos.Count == 0)
47        {
48          Console.WriteLine("Nenhum contato cadastrado.\n");
49          return;
50        }
51
52        ContatoFormatter formatter;
53        switch (formato)
54        {
55          case 1:
56            formatter = new MarkdownFormatter();
57            break;
58          case 2:
59            formatter = new TabelaFormatter();
60            break;
61          case 3:
62            formatter = new RawTextFormatter();
63            break;
64          default:
65            formatter = new RawTextFormatter();
66            break;
67        }
68
69        formatter.ExibirContatos(contatos);
70        Console.WriteLine();
71      }
72    }
73
74    public class ContatoService
75    {
76      private readonly ContatoRepository _repository;
77
78      public ContatoService()
79      {
80        _repository = new ContatoRepository();
81      }
82
83      public void AdicionarContato()
84      {
85        Console.WriteLine("Nome: ");
86        string nome = Console.ReadLine();
87
88        Console.WriteLine("Telefone: ");
89        string telefone = Console.ReadLine();
90
91        Console.WriteLine("Email: ");
92        string email = Console.ReadLine();
93
94        var contato = new Contato
95        {
96          Nome = nome,
97          Telefone = telefone,
98          Email = email
99        };
100
101        _repository.Adicionar(contato);
102        Console.WriteLine("Contato cadastrado com sucesso!");
103      }
104
105      public List<Contato> ListarTodos()
106      {
107        var contatos = new List<Contato>();
108
109        if (_repository.Exists("ArquivarContatos"))
110        {
111          return contatos;
112        }
113
114        try
115        {
116          using (StreamReader sr = new StreamReader("ArquivarContatos"))
117          {
118            string linha;
119            while ((linha = sr.ReadLine()) != null)
120            {
121              if (!string.IsNullOrWhiteSpace(linha))
122              {
123                contatos.Add(Contato.ParseString(linha));
124              }
125            }
126          }
127        }
128        catch (Exception ex)
129        {
130          Console.WriteLine($"Erro ao ler contatos: {ex.Message}");
131        }
132
133        return contatos;
134      }
135    }
136  }

```

## Program:

```

1  /*
2  * Author: Bruno Henrique
3  */
4
5  using System;
6  using System.Collections.Generic;
7
8  namespace AT12
9  {
10    internal class Program
11    {
12      private static readonly ContatoService _contatoService = new ContatoService();
13
14      static void Main(string[] args)
15      {
16        bool sair = false;
17        while (!sair)
18        {
19          ExibirMenu();
20          string opcao = Console.ReadLine();
21
22          switch (opcao)
23          {
24            case "1":
25              _contatoService.AdicionarContato();
26              break;
27            case "2":
28              ExibirTabelaFormatada();
29              break;
30            case "3":
31              formato = Console.ReadLine();
32              if (!int.TryParse(formato, out int formatoEscolhido) || formatoEscolhido <= 1 || formatoEscolhido >= 3)
33              {
34                _contatoService.ListarContatos(formatoEscolhido);
35              }
36              else
37              {
38                Console.WriteLine("Opção inválida. Escolha em formato padrão (Tudo Puro).");
39              }
40              break;
41            case "4":
42              sair = true;
43              Console.WriteLine("Encerrando programa...");
44            default:
45              Console.WriteLine("Opção inválida. Tente novamente.");
46              break;
47          }
48        }
49
50        privado static void ExibirMenu()
51        {
52          Console.WriteLine("1 - Gerenciar lista de contatos");
53          Console.WriteLine("2 - Adicionar novo contato");
54          Console.WriteLine("3 - Listar todos os contatos cadastrados");
55          Console.WriteLine("4 - Sair");
56        }
57
58        privado static void ExibirTabelaFormatada()
59        {
60          Console.WriteLine("Você escolheu a opção de exibição:");
61          Console.WriteLine("1 - Markdown");
62          Console.WriteLine("2 - Tabela");
63          Console.WriteLine("3 - Puro");
64          Console.Write("Digite sua opção: ");
65        }
66      }
67    }
68  }

```

Resultado:

Adicionar:

```
C:\Users\victo\source\repos\ X + ▾ - □ ×  
==== Gerenciador de Contatos ====  
1 - Adicionar novo contato  
2 - Listar contatos cadastrados  
3 - Sair  
Escolha uma opção: 1  
Nome: Nero  
Telefone: 219999999999  
Email: nero@gmail.com  
Contato cadastrado com sucesso!  
==== Gerenciador de Contatos ====
```

Listar:

Markdown:

```
Escolha o formato de exibição:  
1 - Markdown  
2 - Tabela  
3 - Texto Puro  
Digite sua opção: 1  
## Lista de Contatos  
  
- **Nome:** Victor  
- ?? Telefone: 21990388494  
- ?? Email: victorcesar212@gmail.com  
  
- **Nome:** Nero  
- ?? Telefone: 219999999999  
- ?? Email: nero@gmail.com
```

Tabela:

```
Escolha o formato de exibição:  
1 - Markdown  
2 - Tabela  
3 - Texto Puro  
Digite sua opção: 2  
  
| Nome | Telefone | Email |  
| --- | --- | --- |  
| Victor | 21990388494 | victorcesar212@gmail.com |  
| Nero | 219999999999 | nero@gmail.com |
```

Texto Puro:

```
Escolha o formato de exibição:  
1 - Markdown  
2 - Tabela  
3 - Texto Puro  
Digite sua opção: 3  
Nome: Victor | Telefone: 21990388494 | Email: victorcesar212@gmail.com  
Nome: Nero | Telefone: 219999999999 | Email: nero@gmail.com
```