Manipulação de arquivos texto em C# Console – *Utilizando classes abstratas do System.IO*

Você vai aprender como manipular arquivos texto em C# Console, usando a classe File e seus métodos. Arquivos texto são aqueles que armazenam dados em formato de texto, como por exemplo, um documento do Word ou um arquivo CSV. Manipular arquivos texto significa criar, escrever, ler, gravar, adicionar e excluir dados de um arquivo.

Para manipular arquivos texto em C#, você precisa usar a classe File, que faz parte do namespace System.IO. Essa classe fornece métodos estáticos para criar, copiar, mover, renomear e excluir arquivos. Além disso, ela também permite ler e escrever dados em um arquivo.

Para usar a classe File, você precisa importar o namespace System.IO no início do seu código:

```
using System.IO;
```

A seguir, vamos ver alguns dos principais métodos da classe File e como usá-los.

Criar um arquivo texto

Para criar um arquivo texto em C#, você pode usar o método File.Create(), que recebe como parâmetro o nome do arquivo (com o caminho completo) e retorna um objeto FileStream. Esse objeto representa um fluxo de dados que pode ser usado para ler ou escrever no arquivo.

Por exemplo:

```
FileStream fs = File.Create("C:\\meu_arquivo.txt");
```

Esse código cria um arquivo chamado meu_arquivo.txt na raiz do disco C. O objeto fs pode ser usado para escrever dados no arquivo.

Se você tentar criar um arquivo que já existe, o método File.Create() vai sobrescrever o arquivo existente com um novo arquivo vazio. Se você quiser evitar isso, você pode usar o método File.Exists() para verificar se o arquivo já existe ou não. Por exemplo:

```
if (File.Exists("meu_arquivo.txt"))
{
   Console.WriteLine("O arquivo já existe.");
}
else
{
   FileStream fs = File.Create("meu_arquivo.txt");
   fs.Close();
}
```

Escrever em um arquivo texto

Para escrever em um arquivo texto em C#, você pode usar o método File.WriteAllText(), que recebe como parâmetros o nome do arquivo (com o caminho completo) e a string que contém os dados a serem escritos. Esse método sobrescreve qualquer conteúdo existente no arquivo.

Por exemplo:

```
File.WriteAllText("C:\\meu_arquivo.txt", "Olá, mundo!");
```

Esse código escreve a string "Olá, mundo!" no arquivo meu_arquivo.txt na raiz do disco C.

Ler de um arquivo texto

Para ler de um arquivo texto em C#, você pode usar o método File.ReadAllText(), que recebe como parâmetro o nome do arquivo (com o caminho completo) e retorna uma string com todo o conteúdo do arquivo.

Por exemplo:

```
string conteudo = File.ReadAllText("C:\\meu_arquivo.txt");
```

Esse código lê todo o conteúdo do arquivo meu_arquivo.txt na raiz do disco C e armazena na variável conteudo.

Gravar em um arquivo texto

Para gravar em um arquivo texto em C#, você pode usar o método File.WriteAllBytes(), que recebe como parâmetros o nome do arquivo (com o caminho completo) e um array de bytes que contém os dados a serem gravados. Esse método sobrescreve qualquer conteúdo existente no arquivo.

Por exemplo:

```
byte[] dados = new byte[] { 65, 66, 67 }; // ABC em ASCII

File.WriteAllBytes("C:\\meu_arquivo.txt", dados);
```

Esse código grava os bytes 65, 66 e 67 (que correspondem aos caracteres A, B e C em ASCII) no arquivo meu_arquivo.txt na raiz do disco C.

Adicionar em um arquivo texto

Para adicionar em um arquivo texto em C#, você pode usar o método File.AppendAllText(), que recebe como parâmetros o nome do arquivo (com o caminho completo) e a string que contém

os dados a serem adicionados. Esse método não sobrescreve o conteúdo existente no arquivo, mas sim adiciona ao final dele.

Por exemplo:

```
File.AppendAllText("C:\\meu_arquivo.txt", "\nAté mais!");
```

Esse código adiciona a string "\nAté mais!" (quebra de linha seguida de "Até mais!") no final do arquivo meu_arquivo.txt na raiz do disco C.

Fechando o arquivo

Após toda sua manipulação no arquivo, você precisa fechá-lo usando o método fs.Close(). Por exemplo:

```
fs = File.Create("meu_arquivo.txt");

// escrever dados no arquivo, e ao final

fs.Close();
```

Excluir um arquivo texto

Para excluir um arquivo texto em C#, você pode usar o método File.Delete(), que recebe como parâmetro o nome do arquivo (com o caminho completo). Esse método remove o arquivo do sistema de arquivos.

Por exemplo:

```
File.Delete("C:\\meu_arquivo.txt");
```

Esse código exclui o arquivo meu_arquivo.txt na raiz do disco C.

Exercícios

Use os métodos da classe File para realizar as operações solicitadas. Você pode testar seu código usando o Visual Studio ou qualquer outro ambiente de desenvolvimento para C#.

1) Escreva um programa que crie um arquivo chamado numeros.txt na pasta Documentos do usuário logado (Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments); retorna o caminho de meus documentos do usuário logado) e escreva nele os números de 1 a 10, separados por vírgula.

2) Escreva um programa que leia o conteúdo do arquivo numeros.txt criado no exercício anterior e mostre na tela quantos números pares e ímpares existem nele.

3) Escreva um programa que grave no final do arquivo numeros.txt criado no exercício anterior a soma de todos os números nele contidos.

4) Escreva um programa que leia um nome de usuário e uma senha digitados pelo usuário e verifique se eles estão armazenados em um arquivo chamado usuarios.txt. Se estiverem, mostre na tela "Acesso permitido". Se não estiverem, mostre na tela "Acesso negado".

Apêndice:

A classe Stream

A classe Stream é uma classe abstrata que representa uma sequência de bytes que podem ser lidos ou escritos. Essa classe é a base para outras classes que permitem manipular arquivos texto de forma mais flexível e eficiente do que a classe File. Para usar essa classe e suas subclasses, você também precisa adicionar o namespace System.IO no início do seu código:

```
using System.IO;
```

Para abrir um arquivo texto para leitura ou escrita usando streams, você pode usar os métodos File.OpenRead() ou File.OpenWrite(), respectivamente. Esses métodos recebem como parâmetro o nome do arquivo e retornam um objeto FileStream. Por exemplo:

```
FileStream fs1 = File.OpenRead("meu_arquivo.txt");

FileStream fs2 = File.OpenWrite("meu_arquivo.txt");
```

Esses códigos abrem o arquivo meu_arquivo.txt para leitura ou escrita e atribuem os objetos FileStream correspondentes às variáveis fs1 e fs2. Você pode usar esses objetos para ler ou escrever dados no arquivo usando os métodos da classe Stream.

Para ler dados de um arquivo texto usando streams, você precisa de um objeto que converta os bytes em caracteres. Uma das classes que fazem isso é a classe StreamReader, que recebe como parâmetro um objeto FileStream e fornece métodos para ler strings de um stream. Por exemplo:

```
StreamReader sr = new StreamReader(fs1);

string linha = sr.ReadLine();
```

Esse código cria um objeto StreamReader a partir do objeto FileStream fs1 e usa o método sr.ReadLine() para ler uma linha do stream e atribuir à variável linha. Esse método retorna null quando não há mais linhas para ler.

Para escrever dados em um arquivo texto usando streams, você precisa de um objeto que converta os caracteres em bytes. Uma das classes que fazem isso é a classe StreamWriter, que recebe como parâmetro um objeto FileStream e fornece métodos para escrever strings em um stream. Por exemplo:

```
StreamWriter sw = new StreamWriter(fs2);
```

```
sw.WriteLine("Olá, mundo!");
```

Esse código cria um objeto StreamWriter a partir do objeto FileStream fs2 e usa o método sw.WriteLine() para escrever uma linha no stream com a string "Olá, mundo!". Esse método adiciona uma quebra de linha ao final da string.

Depois de ler ou escrever dados em um stream, você precisa fechar o stream usando o método Close(). Isso libera os recursos usados pelo stream e fecha o arquivo associado ao stream. Por exemplo:

```
sr.Close();

sw.Close();
```

Gabarito

1)

```
using System.IO;
namespace Exercicio1
{
   class Program
   {
      static void Main(string[] args)
      {
         // Obtém o caminho da pasta Documentos do usuário atual
         string pasta = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
         // Define o nome e o caminho completo do arquivo
         string nome = "numeros.txt";
         string caminho = Path.Combine(pasta, nome);
         // Define a string com os números de 1 a 10 separados por vírgula
         string numeros = "1,2,3,4,5,6,7,8,9,10";
         // Cria e escreve no arquivo
         File.WriteAllText(caminho, numeros);
      }
   }
}
```

2)

```
using System.IO;
namespace Exercicio2
{
   class Program
   {
      static void Main(string[] args)
      {
         // Obtém o caminho da pasta Documentos do usuário atual
```

```
        string pasta = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
        // Define o nome e o caminho completo do arquivo
        string nome = "numeros.txt";
        string caminho = Path.Combine(pasta, nome);
        // Lê o conteúdo do arquivo
        string numeros = File.ReadAllText(caminho);
        // Converte a string em um array de inteiros
        int[] array = Array.ConvertAll(numeros.Split(','), int.Parse);
        // Inicializa as variáveis para contar os números pares e ímpares
        int pares = 0;
        int impares = 0;
        // Percorre o array e verifica se cada número é par ou ímpar
        foreach (int n in array)
        {
           if (n % 2 == 0)
           {
              pares++;
           }
           else
           {
              impares++;
           }
        }
        // Mostra na tela os resultados
        Console.WriteLine($"O arquivo {nome} contém {pares} números pares e {impares}
números ímpares.");
      }
   }
}
```

3)

```
using System.IO;
namespace Exercicio3
{
   class Program
   {
      static void Main(string[] args)
      {
         // Obtém o caminho da pasta Documentos do usuário atual
         string pasta = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);
         // Define o nome e o caminho completo do arquivo
         string nome = "numeros.txt";
         string caminho = Path.Combine(pasta, nome);
         // Lê o conteúdo do arquivo
         string numeros = File.ReadAllText(caminho);
         // Converte a string em um array de inteiros
         int[] array = Array.ConvertAll(numeros.Split(','), int.Parse);
         // Inicializa a variável para somar os números
         int soma = 0;
         // Percorre o array e soma os números
         foreach (int n in array)
         {
            soma += n;
         }
         // Define a string com a soma dos números
         string resultado = $"\nA soma dos números é {soma}.";
         // Adiciona no final do arquivo
         File.AppendAllText(caminho, resultado);
      }
   }
}
```

4)

```
using System;
using System.IO;
namespace ConsoleApp
{
   class Program
   {
      static void Main(string[] args)
      {
         Console.Write("Digite o nome de usuário: ");
         string username = Console.ReadLine();
         Console.Write("Digite a senha: ");
         string password = Console.ReadLine();
         string[] lines = File.ReadAllLines("usuarios.txt");
         bool accessGranted = false;
         foreach (string line in lines)
         {
            string[] parts = line.Split(';');
            if (parts[0] == username && parts[1] == password)
            {
               accessGranted = true;
               break;
            }
         }
         if (accessGranted) //equivalente a if(accessGaranted == true)
         {
            Console.WriteLine("Acesso permitido");
         }
         else
         {
            Console.WriteLine("Acesso negado");
```

```
        }
      }
    }
}
```

Este código assume que o arquivo “usuarios.txt” contém uma lista de nomes de usuários e senhas separados por ponto e vírgula (;) em cada linha. Por exemplo:

joao;1234

maria;abcd

pedro;xyzw