Arquivos , Streams e I/O

1- Verificando a existência do arquivo e exibindo a data de criação:

```
Console.Write("Digite o caminho do arquivo: ");
string caminho = Console.ReadLine();
if (File.Exists(caminho))
{
   FileInfo fileInfo = new FileInfo(caminho);
   Console.WriteLine($"O arquivo foi criado em {fileInfo.CreationTime}");
}
else
{
   Console.WriteLine("O arquivo não foi encontrado.");
}
```

2- Copiando um arquivo de um diretório para outro:

```
Console.Write("Digite o caminho do arquivo de origem: ");
string caminhoOrigem = Console.ReadLine();
Console.Write("Digite o caminho do diretório de destino: ");
string caminhoDestino = Console.ReadLine();
string nomeArquivo = Path.GetFileName(caminhoOrigem);
string caminhoDestinoCompleto = Path.Combine(caminhoDestino, nomeArquivo);
File.Copy(caminhoOrigem, caminhoDestinoCompleto, true);
Console.WriteLine("Arquivo copiado com sucesso.");
```

3- Escrevendo, adicionando e lendo informações em um arquivo:

```
string caminho = "arquivo.txt";
// Escreve informações no arquivo
using (StreamWriter streamWriter = File.CreateText(caminho))
{
   streamWriter.WriteLine("Primeira linha de texto");
}
// Adiciona mais informações ao arquivo
using (StreamWriter streamWriter = File.AppendText(caminho))
{
   streamWriter.WriteLine("Segunda linha de texto");
}
// Lê o conteúdo do arquivo e exibe na tela
using (StreamReader streamReader = File.OpenText(caminho))
{
   string conteudo = streamReader.ReadToEnd();
   Console.WriteLine(conteudo);
}
```

4- Criptografando um arquivo de texto com o algoritmo AES:

```
Console.Write("Digite o caminho do arquivo de origem: ");
string caminhoOrigem = Console.ReadLine();
Console.Write("Digite o caminho do arquivo de destino: ");
string caminhoDestino = Console.ReadLine();
```

```
string chave = "minhachave12345678";
// Lê o conteúdo do arquivo
string conteudo = File.ReadAllText(caminhoOrigem);
// Criptografa o conteúdo com o algoritmo AES
byte[] conteudoCriptografado;
using (Aes aes = Aes.Create())
{
   aes.Key = Encoding.UTF8.GetBytes(chave);
   aes.Mode = CipherMode.CBC;
   ICryptoTransform encryptor = aes.CreateEncryptor(aes.Key, aes.IV);
   using (MemoryStream memoryStream = new MemoryStream())
   {
      using (CryptoStream cryptoStream = new CryptoStream(memoryStream, encryptor,
CryptoStreamMode.Write))
      {
         using (StreamWriter streamWriter = new StreamWriter(cryptoStream))
         {
            streamWriter.Write(conteudo);
         }
         conteudoCriptografado = memoryStream.ToArray();
      }
   }
}
// Salva o conteúdo criptografado em um novo arquivo
File.WriteAllBytes(caminhoDestino, conteudoCriptografado);
Console.WriteLine("Arquivo criptografado com sucesso.");
```

5- Convertendo uma imagem para um arquivo de texto com codificação Base64:

Segue um exemplo de código C# que lê o conteúdo de um arquivo de imagem, converte-o para Base64 e salva o resultado em um novo arquivo de texto:

```
using System;
using System.IO;
class Program
{
   static void Main(string[] args)
   {
      Console.WriteLine("Digite o caminho do arquivo de imagem:");
      string imagePath = Console.ReadLine();
      Console.WriteLine("Digite o caminho do arquivo de destino:");
      string textFilePath = Console.ReadLine();
      // Lê o conteúdo do arquivo de imagem
      byte[] imageBytes = File.ReadAllBytes(imagePath);
      // Converte o conteúdo para Base64
      string base64String = Convert.ToBase64String(imageBytes);
      // Escreve o conteúdo em um novo arquivo de texto
```

```
        File.WriteAllText(textFilePath, base64String);
        Console.WriteLine("Arquivo de texto salvo com sucesso!");
    }
}
```

O programa solicita que o usuário digite o caminho do arquivo de imagem e o caminho do arquivo de destino. Em seguida, lê o conteúdo do arquivo de imagem, converte-o para Base64 e escreve o resultado em um novo arquivo de texto usando o método File.WriteAllText. Por fim, exibe uma mensagem informando que o arquivo foi salvo com sucesso.

6- Listar todos os arquivos em um diretório usando Directory e Path:

```
string caminho = @"C:\Users\Username\Documents";
string[] arquivos = Directory.GetFiles(caminho);
foreach (string arquivo in arquivos)
{
    Console.WriteLine(Path.GetFileName(arquivo));
}
```

7- Listar todos os subdiretórios em um diretório usando DirectoryInfo:

```
string caminho = @"C:\Users\Username\Documents";
DirectoryInfo diretorio = new DirectoryInfo(caminho);
foreach (var subdiretorio in diretorio.GetDirectories())
{
    Console.WriteLine(subdiretorio.Name);
}
```

8- Criar um novo diretório e criar um arquivo dentro desse diretório usando Directory e Path:

```
string caminhoDiretorio = @"C:\Users\Username\Documents\NovoDiretorio";
string nomeArquivo = "NovoArquivo.txt";
// Cria o diretório
Directory.CreateDirectory(caminhoDiretorio);
// Cria o arquivo dentro do diretório
string caminhoCompleto = Path.Combine(caminhoDiretorio, nomeArquivo);
File.Create(caminhoCompleto);
```

9- Copiar um arquivo de um diretório para outro usando File e Path:

```
string caminhoOrigem = @"C:\Users\Username\Documents\Origem\Arquivo.txt";
string caminhoDestino = @"C:\Users\Username\Documents\Destino";
string nomeArquivo = Path.GetFileName(caminhoOrigem);
string caminhoDestinoCompleto = Path.Combine(caminhoDestino, nomeArquivo);
File.Copy(caminhoOrigem, caminhoDestinoCompleto);
```

Arquivos , Streams e I/O

10- Mover um arquivo de um diretório para outro usando File e Path:

```
string caminhoOrigem = @"C:\Users\Username\Documents\Origem\Arquivo.txt";
string caminhoDestino = @"C:\Users\Username\Documents\Destino";
string nomeArquivo = Path.GetFileName(caminhoOrigem);string caminhoDestinoCompleto =
Path.Combine(caminhoDestino, nomeArquivo);
File.Move(caminhoOrigem, caminhoDestinoCompleto);
```

11- Listar todos os arquivos em um diretório e suas subpastas usando DirectoryInfo e Path:

```
string caminho = @"C:\Users\Username\Documents";
DirectoryInfo diretorio = new DirectoryInfo(caminho);
foreach (var arquivo in diretorio.GetFiles("*", SearchOption.AllDirectories))
{
   Console.WriteLine(arquivo.FullName);
}
```

12- Excluir um arquivo ou diretório usando Directory, DirectoryInfo e Path:

```
string caminhoDiretorio = @"C:\Users\Username\Documents\NovoDiretorio";
string caminhoArquivo = @"C:\Users\Username\Documents\NovoDiretorio\NovoArquivo.txt";
// Exclui o arquivo
File.Delete(caminhoArquivo);
// Exclui o diretório
Directory.Delete(caminhoDiretorio, true);
```

13-Renomear um arquivo usando File e Path:

```
string caminhoArquivo = @"C:\Users\Username\Documents\Arquivo.txt";
string novoNome = "NovoNome.txt";
string caminhoCompleto = Path.Combine(Path.GetDirectoryName(caminhoArquivo), novoNome);
File.Move(caminhoArquivo, caminhoCompleto);
```

14- Retornar o tamanho total de um diretório, incluindo todos os arquivos e subdiretórios, usando Directory

```
string diretorio = @"C:\Users\Usuario\Documents";

// Chama o método GetDirectorySize e imprime o resultado
long tamanhoTotal = GetDirectorySize(diretorio);
Console.WriteLine($"Tamanho total do diretório {diretorio}: {tamanhoTotal} bytes");

Console.ReadKey();

static long GetDirectorySize(string diretorio)
{
  // Verifica se o diretório existe
if (!Directory.Exists(diretorio))
{
   throw new DirectoryNotFoundException($"Diretório {diretorio} não encontrado.");
}

// Recupera o tamanho de todos os arquivos no diretório
long tamanhoTotal = 0;
foreach (string arquivo in Directory.GetFiles(diretorio, "*",
     SearchOption.AllDirectories))
{
            FileInfo info = new FileInfo(arquivo);
            tamanhoTotal += info.Length;
}
 return tamanhoTotal;
}
```

Nesse código, o método `GetDirectorySize` recebe uma string representando o caminho do diretório que você deseja calcular o tamanho total. Ele começa verificando se o diretório existe usando o método `Directory.Exists`. Se o diretório não for encontrado, o método lança uma exceção `DirectoryNotFoundException`.

O método então percorre todos os arquivos no diretório e em seus subdiretórios usando o método `Directory.GetFiles` com a opção `SearchOption.AllDirectories`. Para cada arquivo, ele recupera o tamanho em bytes usando a classe FileInfo e adiciona ao tamanho total.

Finalmente, o método retorna o tamanho total em bytes.

No método `Main`, basta chamar o método `GetDirectorySize` com o diretório desejado e imprimir o resultado. Nesse exemplo, o diretório "C:\Users\Usuario\Documents" é usado como exemplo, mas você pode substituí-lo pelo diretório desejado.

15- Criar um programa que retorne o nome do arquivo mais recente em um diretório usando a classe DirectoryInfo e a classe Path

```
// Diretório que você deseja pesquisar
string diretorio = @"C:\Users\Usuario\Downloads";

// Cria um objeto DirectoryInfo para o diretório
DirectoryInfo dirInfo = new DirectoryInfo(diretorio);

// Obtém todos os arquivos no diretório
FileInfo[] arquivos = dirInfo.GetFiles();

/ Inicializa a data de modificação mais recente e o nome do arquivo correspondente
DateTime ultimaModificacao = DateTime.MinValue;
string nomeArquivoMaisRecente = string.Empty;

// Percorre todos os arquivos para encontrar o mais recente
foreach (FileInfo arquivo in arquivos) {
  if (arquivo.LastWriteTime > ultimaModificacao) {
        ultimaModificacao = arquivo.LastWriteTime;
           nomeArquivoMaisRecente = arquivo.Name;
       }
 }
// Verifica se algum arquivo foi encontrado e imprime o nome do arquivo mais recente
if (!string.IsNullOrEmpty(nomeArquivoMaisRecente)) {
    Console.WriteLine($"O arquivo mais recente em {diretorio} é:
{Path.Combine(diretorio, nomeArquivoMaisRecente)}");
 }
 else {
   Console.WriteLine($"Não foi encontrado nenhum arquivo em {diretorio}");
}
```

Nesse código, o diretório que você deseja pesquisar é especificado em uma string `diretorio`. Em seguida, um objeto `DirectoryInfo` é criado para representar esse diretório.

O método `GetFiles` é então chamado em `dirInfo` para recuperar uma lista de todos os arquivos no diretório. O loop `foreach` percorre todos os arquivos, comparando a data de modificação de cada arquivo com uma variável `ultimaModificacao`, que é atualizada se uma data mais recente for encontrada.

Por fim, o nome do arquivo mais recente é armazenado na variável `nomeArquivoMaisRecente`. Se algum arquivo for encontrado, o caminho completo do arquivo é impresso usando a classe `Path` e o método `Combine`. Caso contrário, uma mensagem indicando que nenhum arquivo foi encontrado é exibida.

Note que o exemplo utiliza a data de modificação para determinar o arquivo mais recente. Você pode alterar a comparação para utilizar a data de criação ou outra propriedade do arquivo, caso deseje.