Работа с файлами и файловыми потоками

Получение информации о файле

Класс FileInfo содержит методы:

- Create(): создает файл
- Delete(): удаляет файл
- CopyTo(path): копирует файл по указанному пути
- MoveTo(destFileName): перемещает файл в новое место

Получение информации о файле

Класс FileInfo содержит свойтсва:

- Directory: получает родительский каталог
- DirectoryName: получает полный путь к родительскому каталогу
- Exists: указывает, существует ли файл
- Length: получает размер файла

Получение информации о файле

Класс FileInfo содержит свойтсва:

- Extension: получает расширение файла
- Name: получает имя файла
- FullName: получает полное имя файла

Пути к файлам

Абсолютный путь

```
string path1 = @"C:\Temp\ISPP\1.txt";  // для Windows
string path2 = "C:\\Temp\\ISPP\\1.txt";  // для Windows
string path3 = "/Users/student/Documents/1.txt";  // для MacOS или Linux
```

Пути к файлам

Относительный путь

```
string path4 = "MyDir\\content.txt";  // для Windows
string path5 = "MyDir/content.txt";  // для MacOS/Linux
```

Применение FileInfo

```
string path = @"C:\Temp\ISPP\1.txt";
FileInfo fileInfo = new FileInfo(path);
if (fileInfo.Exists)
{
    Console.WriteLine($"Имя файла: {fileInfo.Name}");
    Console.WriteLine($"Время создания: {fileInfo.CreationTime}");
    Console.WriteLine($"Размер: {fileInfo.Length}");
}
```

Удаление файла

```
string path = @"C:\app\content.txt";
FileInfo fileInf = new FileInfo(path);
if (fileInf.Exists)
{
  fileInf.Delete();
  // альтернатива с помощью класса File
  // File.Delete(path);
}
```

Запись в файл

Для записи в текстовый файл используется класс StreamWriter. Возможные конструкторы:

- StreamWriter(string path)
- StreamWriter(string path, bool append)
- StreamWriter(string path, bool append, System.Text.Encoding encoding)

Запись в файл

Методы класса StreamWriter:

- int Close(): закрывает записываемый файл
- void Write(string/int/char/double value): записывает в файл данные
- Task WriteAsync(string value): асинхронная версия метода Write.

Запись в файл

Методы класса StreamWriter:

- void WriteLine(string value): записывает в файл данные
- Task WriteLineAsync(string value): асинхронная версия метода WriteLine

Примеры

```
string path = "1.txt", text = "Hello World\nHello AKT";
using (StreamWriter writer = new StreamWriter(path, false){
        await writer.WriteLineAsync(text);// полная перезапись файла
}
 using (StreamWriter writer = new StreamWriter(path, true)) {
      await writer.WriteLineAsync("Addition");
       await writer.WriteAsync("4,5");// добавление в файл
   }
```

Чтение из файла

Класс StreamReader позволяет нам легко считывать весь текст или отдельные строки из текстового файла. Возможные конструкторы

- StreamReader(string path):
- StreamReader(string path, System.Text.Encoding encoding)

Чтение из файла

Методы StremReader:

- void Close(): закрывает считываемый файл
- int Peek(): возвращает следующий доступный символ, если символов больше нет, то возвращает -1
- int Read(): считывает и возвращает следующий символ в численном представлении.

Чтение из файла

Методы StremReader:

- Task<int> ReadAsync(): асинхронная версия метода Read
- string ReadLine(): считывает одну строку в файле
- string ReadLineAsync(): асинхронная версия метода ReadLine

Чтение из файла

Методы StremReader:

- string ReadToEnd(): считывает весь текст из файла
- string ReadToEndAsync(): асинхронная версия метода ReadToEnd

Примеры

Методы StremReader:

- string ReadToEnd(): считывает весь текст из файла
- string ReadToEndAsync(): асинхронная версия метода ReadToEnd

Примеры

```
string path = "1.txt";
// асинхронное чтение
using (StreamReader reader = new StreamReader(path)){
    string text = await reader.ReadToEndAsync();
    Console.WriteLine(text);
}
```

Примеры

```
string path = "1.txt";
// асинхронное чтение
using (StreamReader reader = new StreamReader(path)) {
    string line;
    while ((line = await reader.ReadLineAsync()) != null) {
        Console.WriteLine(line);
    }
}
```

Поиск данных в текстовых файлах

Поиск строки

```
// Чтение всего текста из файла
string text = File.ReadAllText(filePath);
if (text.Contains(searchString))
    Console.WriteLine("Строка найдена!");
```

Построчное чтение файла

```
string filePath = @"C:\path\to\your\textfile.txt";
string searchString = "Поисковая строка";
foreach (var line in File.ReadLines(filePath)){
   if (line.Contains(searchString))
      Console.WriteLine($"Найдено в строке: {line}");
}
```

Использование регулярных выражений

```
string filePath = @"C:\path\to\your\textfile.txt";
string pattern = @"\bПоисковая строка\b";
Regex regex = new Regex(pattern);
foreach (Match match in regex.Matches(File.ReadAllText(filePath)))
{
    Console.WriteLine($"Найдено: {match.Value}");
}
```

Запуск приложения с параметрами

Проверка указанных параметров

```
if (args.Length == 0){
   Console.WriteLine("Не переданы параметры. Используйте
формат:     MyApp.exe параметр1 параметр2 ...");
   return;
}
foreach (string arg in args)
   Console.WriteLine(arg);
```

Сборка проекта

Соберите проект, чтобы получить исполняемый файл .exe. Например, если ваш проект называется MyApp, вы получите файл MyApp.exe.

Запуск приложения с параметрами

Откройте командную строку (cmd) или PowerShell и перейдите в директорию, где находится собранный исполняемый файл. Затем выполните команду:

```
MyApp.exe параметр1 параметр2 …
```

Например:

```
MyApp.exe hello world
```