**17. Вход/Изход в .NET. Работа с файлове, директории, потоци, четци и писци.**

Потоците в обектно-ориентираното програмиране са една абстракция, с която се осъществява вход и изход от дадена програма. Потоците в C# като концепция са аналогични на потоците в други обектно-ориентирани езици, напр. Java, C++ и Delphi (Object Pascal).

**Потокът** е подредена серия от байтове, която служи като абстрактен канал за данни. Този виртуален канал свързва програмата с устройство за съхранение или пренос на данни (напр. файл върху хард диск), като достъпът до канала е последователен. Потоците предоставят средства за четене и запис на поредици от байтове от и към устройството. Това е стандартният механизъм за извършване на входно-изходни операции в .NET Framework.Потоците в .NET Framework се делят на две групи – **базови** и **преходни**. И едните, и другите, наследяват абстрактния клас **System.IO.Stream**, базов за всички потоци.Базовите потоци пишат и четат директно от някакъв външен механизъм за съхранение, като файловата система (например класът **FileStream**), паметта (**MemoryStream**) или данни, достъпни по мрежата (**NetworkStream**). По-нататък ще разгледаме класа **FileStream** в точката "Файлови потоци". Преходните потоци пишат и четат от други потоци (най-често в базови потоци), като при това посредничество добавят допълнителна функци-оналност, например буфериране (**BufferedStream**) или кодиране (**CryptoStream**). По-подробно ще разгледаме **BufferedStream** в точката "Буферирани потоци".За четене на данни от поток се използва методът **int Read(byte[] buffer, int offset, int count)**. Той чете най-много **count** на брой байта от текущата позиция на входния поток, увеличава позицията и връ-ща броя прочетени байтове или **0** при достигне края на потока. Четенето може да блокира за неопределено време. Например, ако при четене от мрежа извикаме метода **NetworkStream.Read(…)**, а не са налични данни за четене, операцията блокира до тяхното получаване. В такива случаи е уместно да се използва свойството **NetworkStream. DataAvailable**, което показва дали в потока има пристигнали данни, които още не са прочетени, т. е. дали последваща операция **Read()** ще блокира или ще върне резултат веднага.

**Писане в поток**

Методът **Write(byte[] buffer, int offset, int count)** записва в изход-ния поток **count** байта, като започва от зададеното отместване в байтовия масив. И тази операция е блокираща, т.е. може да предизвика забавяне за неопределено време. Не е гарантирано, че байтовете, записани в потока с **Write(…)**, са достигнали до местоназначението си след успеш-ното изпълнение на метода. Възможно е потокът да буферира данните и да не ги изпраща веднага.Файловите потоци в .NET Framework са реализирани в класа **FileStream**, който вече беше използван в примера за потоци. Като наследник на **Stream**, той поддържа всичките му методи и свойства (четене, писане, позициониране) и добавя някои допълнителни.Четенето и писането във файлови потоци, както и другите по-рядко използвани операции, се извършват както при всички наследници на класа **Stream** – с методите **Read()**, **Write()** и т. н.

Файловите потоци поддържат пряк достъп до определена позиция от файла чрез метода **Seek(…)**.

**Четците и писачите** (readers and writers) в .NET Framework са класове, които улесняват работата с потоците. При работа например само с файлов поток, програмистът може да чете и записва единствено байтове. Когато този поток се обвие в четец или писач, вече са позволени четенето и записа на различни структури от данни, например примитивни типове, текстова информация и други типове. Четците и писачите биват двоични и текстови. Двоичните четци и писачи осигуряват четене и запис на примитивни типове данни в двоичен вид – **ReadChar()**, **ReadChars()**, **ReadInt32()**, **ReadDouble()** и др. за четене и съответно **Write(char)**, **Write(char[])**, **Write(Int32)**, **Write(double)** – за запис. Може да се чете и записва и **string**, като той се представя във вид на масив от символи и префиксно се записва дължината му – **ReadString()**, респ. **Write(string)**.Текстовите четци и писачи осигуряват четене и запис на текстова инфор-мация, представена във вид на низове, разделени с нов ред. Базови текстови четци и писачи са абстрактните класове **TextReader** и **TextWriter**. Основните методи за четене и запис са следните: - **ReadLine()** – прочита един ред текст.

- **ReadToEnd()** – прочита всичко от текущата позиция до края на потока.

- **Write(…)** – вмъква данни в потока на текущата позиция.

**Работа с директории. Класове Directory и DirectoryInfo**

Класовете **Directory** и **DirectoryInfo** са помощни класове за работа с директории. Ще изброим основните им методи, като отбележим, че за **Directory** те са статични, а за **DirectoryInfo** – достъпни чрез инстанция.

- **Create()**, **CreateSubdirectory()** – създава директория или подди-ректория.

- **GetFiles(…)** – връща всички файлове в директорията.

- **GetDirectories(…)** – връща всички поддиректории на директорията.

- **MoveTo(…)** – премества (преименува) директория.

- **Delete()** – изтрива директория.

- **Exists()** – проверява директория дали съществува.

- **Parent** – връща горната директория.

- **FullName** – пълно име на директорията.