Основы программирования

Средства ввода-вывода

Текстовые потоки данных

Текстовые потоки данных

Предназначены для работы с символьными данными в кодировке `Unicode`.

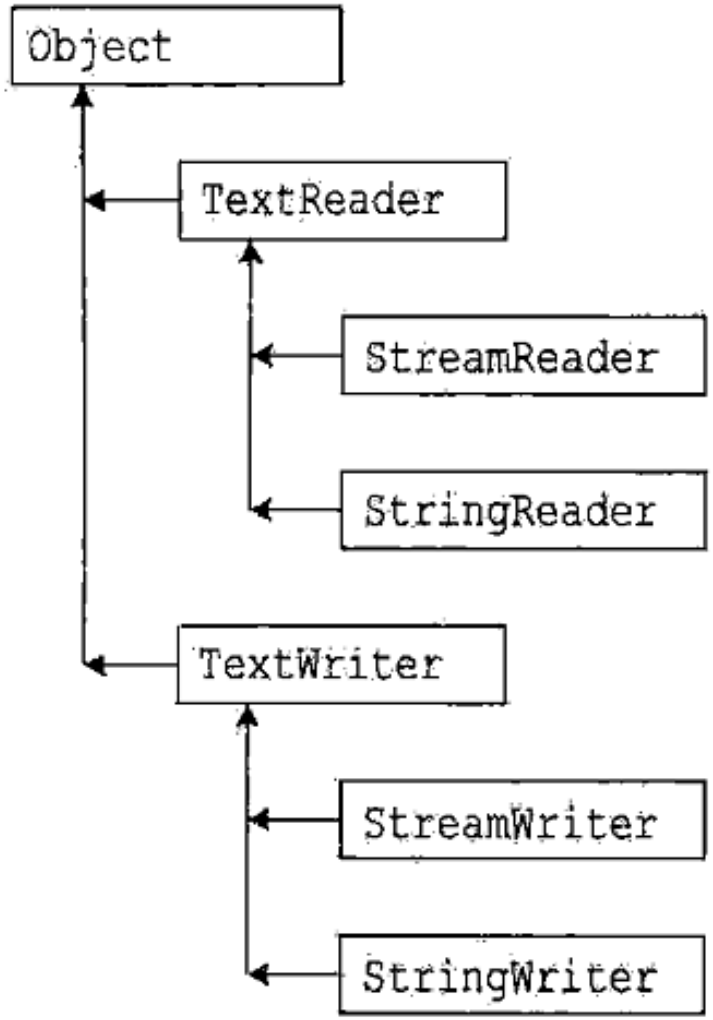
Кодировку потока можно настроить с помощью констант перечисления `Encoding` из пространства имен `System.Text`.

Текстовые потоки разделены на

- потоки чтения,
- потоки записи.

Каждая из групп имеет свой абстрактный базовый класс.

Схема наследования текстовых потоков



Класс TextReader

Является базовым абстрактным классом для символьных потоков чтения.

Класс TextReader. Некоторые методы

`Peek()` – возвращает следующий символ без перемещения курсора в потоке.

`Read()` – считывание текстовых данных из потока.

`ReadBlock()` – считывание из потока заданное количество символов в массив.

`ReadLine()` – считывание строки из потока. Пустая строка `(null)` означает конец потока.

`ReadToEnd()` – считывание всех символов до конца потока, начиная с текущей позиции курсора. Возвращается тип `string`.

`Close()` – закрытие потока.

Класс TextWriter

Является базовым абстрактным классом для текстовых потоков записи .

Класс TextWriter. Некоторые методы

`Close()` – закрытие потока.

`Flush()` – очистка буфера.

`NewLine` – свойство используется для хранения последовательности символов, означающих переход на новую строку (по умолчанию ‘\r\n’).

`Write()` – запись текста в поток.

`WriteLine()` – запись строки в поток, при этом курсор перемещается на новую строку.

Классы StreamReader и StreamWriter

Наследниками описанных абстрактных классов являются:

`StreamReader` – поток чтения из текстового файла,
`StreamWriter` – поток записи в текстовый файл.

Они реализуют функциональность своего предка для работы с файлом, как с текстовым потоком записи или чтения.

Имеют дополнительную функциональность, связанную с созданием и открытием файлов.

Чтение из файла и класс StreamReader

Позволяет считывать весь текст или отдельные строки из текстового файла.

Класс StreamReader. Некоторые конструкторы

`StreamReader(string path)` – через параметр `path` передается путь к считываемому файлу.

`StreamReader(string path, System.Text.Encoding encoding)` – параметр `encoding` задает кодировку для чтения файла.

Класс StreamReader. Некоторые методы

`Close()` – закрывает считываемый файл и освобождает все ресурсы.

`Peek()` – возвращает следующий доступный символ, если символов больше нет, то возвращает `-1`.

`Read()` – считывает и возвращает следующий символ в численном представлении.

Перегруженная версия:
`Read(char[] array, int index, int count)`, где
`array` - массив, куда считываются символы,
`index` - индекс в `array`, начиная с которого записываются считываемые символы и
`count` - максимальное количество считываемых символов.

`ReadLine()` – считывает одну строку в файле.

`ReadToEnd()` – считывает весь текст из файла.

Запись в файла и класс StreamWriter

Используется для записи в текстовый файл.

Класс StreamWriter. Некоторые конструкторы

`StreamWriter(string path)` – через параметр path передается путь к файлу, который будет связан с потоком

`StreamWriter(string path, bool append)` – параметр `append` указывает, надо ли добавлять в конец файла данные или же перезаписывать файл. Если равно `true`, то новые данные добавляются в конец файла. Если равно `false`, то файл перезаписываетсяя заново.

`StreamWriter(string path, bool append, System.Text.Encoding encoding)` – параметр `encoding` указывает на кодировку, которая будет применяться при записи.

Класс StreamWriter. Некоторые методы

`Close()` – закрывает записываемый файл и освобождает все ресурсы.

`Flush()` – записывает в файл оставшиеся в буфере данные и очищает буфер.

`Write()` – записывает в файл данные простейших типов, как `int`, `double`, `char`, `string` и т.д.

`WriteLine()` – также записывает данные, только после записи добавляет в файл символ окончания строки.

Пример

```
…
FileInfo f = new FileInfo(“my.txt“);

StreamWriter sw = f.CreateText();
sw.WriteLine(“Новый файл“);
for (int i = 0; i < 10; i++)      {
  sw.Write(i + “ “);                }
sw.Write(sw.NewLine);
sw.Close();

StreamReader sr = f.OpenText();
string s = null;
while ((s = sr.ReadLine()) != null)
Console.WriteLine(s);
sr.Close();
…
```

Пример. Комментарий

Методы `Write` и `WriteLine` класса `StreamWriter` имеют множество перегрузок для записи в файл значений разных типов (как и в классе `Console`).

Подробнее:
https://docs.microsoft.com/en-us/dotnet/api/system.io.streamwriter.write?view=net-6.0

В классе `StreamWriter` описано булевское свойство `AutoFlush`, если установить его в `true`, то очищение буфера потока будет производиться после каждой записи в поток. То есть каждый метод записи будет передавать данные через внутренний буфер потока прямо в файл.

Классы StringReader и StringWriter

Наследниками абстрактных классов `TextWriter` и `TextReader` являются также классы `StringWriter` и `StringReader` соответственно.

Эти классы позволяют работать с символьными потоками в динамической памяти аналогично потоку `MemoryStream`.

Пример

```
StringWriter stw = new StringWriter();
stw.WriteLine(”Поток в динамической памяти”);

for (int i = 0; i < 10; i++)      {
stw.Write(i + ” ”);                }
stw.Write(stw.NewLine);
stw.Close();

Console.WriteLine(stw.ToString());
StringReader str=new StringReader(”Поток в
динамической памяти”);
string s = null;
while ((s = str.ReadLine()) != null)
      Console.WriteLine(s);
str.Close();
…
```

Пример. Комментарии

Потоки в динамической памяти можно интерпретировать как временные хранилища данных, которые после формирования их содержимого нужно отправить в постоянное хранилище.

В классе `StringWriter` определен метод получения еще одного временного хранилища символьных данных – объекта класса `StringBuilder` – метод `GetStringBuilder`.

Отметим, что текстовые потоки

- не поддерживают произвольный доступ, и
- не имеют в своем составе методов управления курсором.

Ссылки, источники…

1. System.IO Namespace (Режим доступа https://docs.microsoft.com/ru-ru/dotnet/api/system.io?view=netcore-2.1).
2. C# и .NET | Работа с файлами. File и FileInfo (metanit.com) https://metanit.com/sharp/tutorial/5.3.php