



البرمجة الإجرائية Procedural Programming

IPG202

الفصل السادس التعامل مع الملفات Files and Streams

الكلمات المفتاحية

ملف، مجرى، قراءة، كتابة، مجرى دخل، مجرى خرج، ملف وصول تسلسلي، ملف وصول عشوائي.

ملخص الفصل

يركز هذا الفصل على مفهوم الملفات، حيث يبين كيفية التعامل مع الملفات المخزنة على الأقراص وكيفية إنشائها وفتحها لتنفيذ عملية القراءة والكتابة مع التمييز بين الملفات التسلسلية وملفات الوصول العشوائي، ويختتم بتقديم بعض الأمثلة التي تساعد على اكتساب مهارات معالجة الملفات.

أهداف الفصل

بنهاية هذا الفصل سيكون الطالب قادراً على:

- إنشاء، القراءة من، الكتابة إلى، تعديل الملفات.
- استخدام الصنف File للحصول على معلومات عن الملفات على الحاسوب.
- معالجة الملفات ذات الوصول التسلسلي.
- معالجة الملفات ذات الوصول العشوائي.
- استخدام الأصناف FileStream، StreamReader، و StreamWriter للقراءة والكتابة في الملفات النصية.

محتويات الفصل

1. مقدمة
2. الملفات Files والمجاري Streams
3. القراءة والكتابة من وإلى الملفات في C#.
4. مثال تعليمي: القراءة والكتابة من وإلى ملف تسلسلي.
5. مثال تعليمي: القراءة والكتابة من وإلى ملف وصول عشوائي.
6. تمارين وأنشطة.

1- مقدمة.

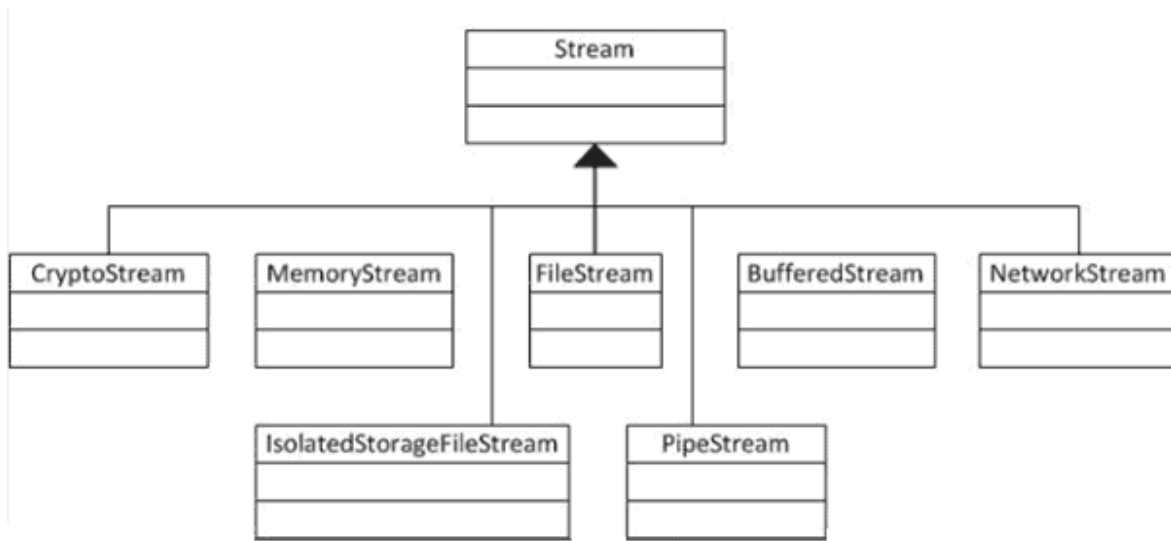
توفر المتغيرات والمصفوفات تخزيناً مؤقتاً للبيانات فقط إذ تفقد المتحولات المحلية بياناتها وقيمها عندما تصبح خارج مجال الرؤية أو التصريح الخاص بها أو عند انتهاء البرنامج. على النقيض من ذلك ، تستخدم الملفات للاحتفاظ بكميات كبيرة من البيانات على المدى الطويل، حتى بعد انتهاء البرنامج الذي أنشأ البيانات. تدعى البيانات المحفوظة في الملفات باسم البيانات الدائمة تقوم أجهزة الكمبيوتر بتخزين الملفات على أجهزة تخزين ثانوية ، مثل القرص الصلب وأقراص DVD والأشرطة. في هذا الفصل ، نشرح كيفية إنشاء وتحديث ومعالجة ملفات البيانات في برامج C# حيث نقدم لمحة على بعض أصناف معالجة الملفات في Framework Class Library. ثم نكتب بعض المقاطع البرمجية التي تظهر كيف يمكنك تحديد معلومات حول الملفات والمجلدات الموجودة على جهاز الكمبيوتر الخاص بك.

2- الملفات Files والمجاري Streams

ترى لغة C# إلى الملف على أنه تدفق stream من البايتات، حيث ينتهي كل ملف بعلامة تدل على نهاية الملف، حيث تحتوي لغة C# أصناف الدخل/ الخرج القياسية التالية الخاصة بالقراءة والكتابة من مصادر مختلفة كالملفات، الذاكرة، الشبكة، وسائط التخزين:

- **الصنف Stream:** يحتوي الصنف المجرد System.IO.Stream على طرائق قياسية لنقل البيانات (قراءة، كتابة، ... إلخ) إلى وسيط التخزين.
- **الصنف FileStream:** يقوم بقراءة وكتابة البيانات من/ إلى ملف فيزيائي سواء كان ملف نصي txt. أو تنفيذي exe. أو صورة أو أي ملف آخر (وهذا الصنف مشتق من الصنف Stream).
- **الصنف MemoryStream:** ويقوم بقراءة أو كتابة البيانات من وإلى الذاكرة.
- **الصنف BufferedStream:** يقوم بقراءة أو كتابة البيانات من مجاري أخرى لتحسين أداء بعض عمليات الدخل/الخرج.
- **الصنف NetworkStream:** يقوم بقراءة أو كتابة كتابة البيانات من الشبكة.
- **الصنف PipStream:** يقوم بقراءة أو كتابة البيانات من عمليات أخرى.
- **الصنف CryptoStream:** من أجل ربط مجاري البيانات إلى تحويلات التشفير.

يبين المخطط التالي هيكلية أصناف Stream:



الشكل 1- هيكلية أصناف Stream

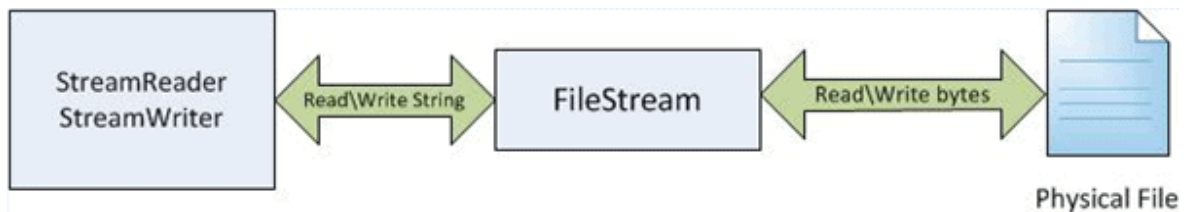
3- القراءة والكتابة من وإلى الملفات في C#

الصنف **StreamReader**: هو صنف مساعد من أجل قراءة المحارف من مجرى Stream ما من خلال تحويل البايتات إلى محارف باستخدام نظام الترميز، ويمكن أن يستخدم لقراءة المحارف والسلاسل المحرفية من مجاري أخرى مثل FileStream أو MemoryStream إلخ.

الصنف **StreamWriter** هو صنف مساعد لكتابة سلسلة محرفية إلى مجرى Stream من خلال تحويل المحارف إلى بايتات، ويمكن أن يستخدم لكتابة السلاسل المحرفية إلى مجاري أخرى مثل FileStream أو MemoryStream إلخ.

الصنف **BinaryReader**: هو صنف مساعد لقراءة أنماط البيانات الأصلية من البايتات.

الصنف **BinaryWriter**: يقوم بكتابة أنماط البيانات الأصلية بشكل ثنائي.



الشكل 2- مجاري الدخل والخرج.

يبين الشكل 2 أن الصنف FileStream يقوم بقراءة البايتات من ملف فيزيائي، ومن ثم يقوم الصنف StreamReader بقراءة السلاسل المحرفية من خلال تحويل هذه البايتات إلى سلاسل محرفية. وبنفس الطريقة، يأخذ الصنف StreamWriter السلاسل المحرفية ويقوم بتحويلها إلى بايتات وكتابتها الصنف FileStream الذي يقوم بدوره بكتابتها إلى الملف الفيزيائي، وبالتالي يتعامل الصنف FileStream مع البايتات، في حين أن الصنفين StreamReader و StreamWriter يتعاملان مع السلاسل المحرفية.

القراءة من ملف باستخدام StreamReader

يبين المثال التالي كيفية استخدام الصنف StreamReader لقراءة البيانات من ملف على القرص الصلب

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;           //Add this library to be able to use streams

namespace ReadingFiles
{
    class Program
    {
        static void Main(string[] args)
        {
            //Create an object of FileInfo for specified path
            FileInfo fi = new FileInfo("D:\\DummyFile.txt");

            //Open a file for Read\Write
            FileStream fs = fi.Open(FileMode.OpenOrCreate, FileAccess.Read, FileShare.Read);

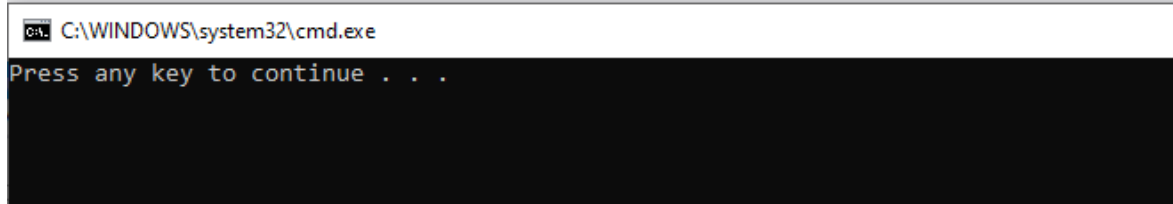
            //Create an object of StreamReader by passing FileStream object on which it needs to operates on
            StreamReader sr = new StreamReader(fs);

            //Use the ReadToEnd method to read all the content from file
            string fileContent = sr.ReadToEnd();
            Console.WriteLine(fileContent);
            //Close the StreamReader object after operation
            sr.Close();
            fs.Close();
        }
    }
}
```

لاحظ أن الطريقة `fi.open()` تملك ثلاثة بارامترات:

- البارامتر الأول **FileMode** وقد استخدم لإنشاء ملف وفتحه.
- البارامتر الثاني **FileAccess**: وقد استخدم للدلالة على أن العملية المطلوبة هي عملية قراءة من الملف.
- البارامتر الثالث **FileShare**: استخدم لمشاركة الملف مع المستخدمين الآخرين لأغراض القراءة عندما يكون الملف مفتوحاً.

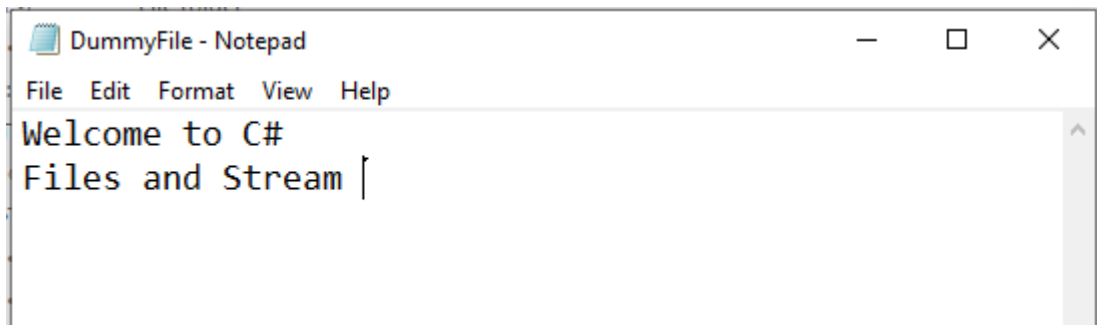
عند تنفيذ هذا البرنامج يعطي على خرجه:



```
C:\WINDOWS\system32\cmd.exe
Press any key to continue . . .
```

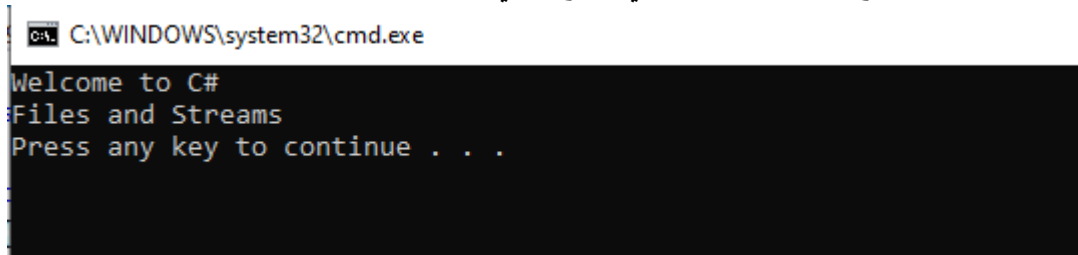
لقد قام هذا البرنامج بالبحث عن الملف `DummyFile.txt` في القرص D فلم يجده، لذلك قام بإنشاء ملف فارغ في المسار المحدد من ثم فتحه وقراءة محتواه (محتوى فارغ).

لنذهب إلى القرص D ونقوم بفتح الملف وكتابة السطرين التاليين يدوياً:



```
DummyFile - Notepad
File Edit Format View Help
Welcome to C#
Files and Stream |
```

الآن نعيد تنفيذ البرنامج السابق لنجد أنه يعطي الخرج التالي:



```
C:\WINDOWS\system32\cmd.exe
Welcome to C#
Files and Streams
Press any key to continue . . .
```

الكتابة إلى ملف باستخدام StreamWriter

يبين المثال التالي كيفية استخدام الصنف StreamWriter لكتابة البيانات إلى ملف على القرص الصلب

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO;           //Add this library to be able to use streams

namespace ReadingFiles
{
    class Program
    {
        static void Main(string[] args)
        {
            //Create an object of FileInfo for specified path
            FileInfo fi = new FileInfo("D:\\DummyFile.txt");

            //Open a file for Read\\Write
            FileStream fs = fi.Open(FileMode.OpenOrCreate, FileAccess.Write, FileShare.Read);

            //Create an object of StreamReader by passing FileStream object on which it needs to operates on
            StreamWriter sw = new StreamWriter(fs);

            //Use the WriteLine method to write to file
            sw.WriteLine("Writing to file");
            //Close the StreamReader object after operation
            sw.Close();
            fs.Close();
        }
    }
}
```

يقوم هذا البرنامج بالبحث عن الملف DummyFile.txt في القرص D فإن وجده يسمح محتواه ويكتب بدلاً عنها العبارة Writing to file وإن لم يجده يقوم بإنشائه وكتابة العبارة فيه.

استخدم الأمر Close لإغلاق المجاري والملفات التي تم إنشاؤها قبل الخروج من البرنامج.

ملاحظة: أعد تنفيذ البرنامج الخاص بالقراءة من الملفات وتحقق من محتوى الملف.

4- مثال تعليمي القراءة والكتابة من وإلى ملف تسلسلي

قمنا في المثال التالي باستخدام جميع التقنيات التي قمنا بتعملها في البرمجة الإجرائية (الطرائق ، معالجة الاستثناءات) للكتابة إلى ملف ومن قراءة محتواه وعرضه على الشاشة:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.IO; //Add this library to be able to use streams

namespace ReadingFiles
{
    class Program
    {
        static void writeObject(string path)
        {
            FileStream fs;
            StreamWriter fw;
            try
            {
                //create file stream object
                fs = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Write);
                //create writer object
                fw = new StreamWriter(fs);
                //write text to file
                fw.WriteLine("C# Programming");
                fw.WriteLine("C Programming");
                fw.WriteLine("C++ Programming");
                fw.Close();
                fs.Close();
            }
            catch (Exception e) { Console.WriteLine(e.Message); }
        }

        static void readObject(string path)
        {
            FileStream fs;
            StreamReader fr;
            try
            {
                //create file stream object
                fs = new FileStream(path, FileMode.Open, FileAccess.Read);
                //create reader objec
                fr = new StreamReader(fs);
            }
        }
    }
}
```



```

string content;
while (!fr.EndOfStream)
{
    content = fr.ReadLine();
    Console.WriteLine(content);
}
fr.Close();
fs.Close();
}
catch (Exception e) { Console.WriteLine(e.Message); }
}
static void Main(string[] args)
{
    writeObject("d:\\students.txt");
    readObject("d:\\students.txt");
    Console.Read();
}
}

```

يعطي هذا البرنامج على خرجه:

```

C:\WINDOWS\system32\cmd.exe
C# Programming
C Programming
C++ Programming

```

نشاط إضافي:

- عدل الطريقة WriteObject بحيث تقوم بإدخال أسماء 10 طلاب، ثم أعد تنفيذ البرنامج ولاحظ سلوكه.
- عدل الطريقة readObject بحيث تقوم بطباعة فقط أسماء الطلاب التي طولها أكثر من 5 أحرف، ثم أعد تنفيذ البرنامج ولاحظ سلوكه.

5- مثال تعليمي القراءة والكتابة من وإلى ملف وصول عشوائي

- الملفات ذات الوصول العشوائي هي ملفات يمكن الوصول فيها لأي تسجيلية مباشرة وبسرعة أي دون المرور بعدد كبير من التسجيلات.
- للكتابة في ملف ذو وصول عشوائي، نستخدم الصف `BinaryWriter`.
- للقراءة من ملف ذو وصل عشوائي، نستخدم الصف `BinaryReader`.
- يجب تحديد مكان الكتابة أو القراءة باستخدام الخاصية `Position`.
- نقوم في المثال التالي بتخزين بعض البيانات عن الطلاب ومن ثم قراءتها.
- يحوي صف الطالب `Student` ثلاثة خصائص: سلسلة نصية لرقم الطالب `stnumber` وسلسلة نصية لاسم الطالب `stname` والحجم الأعظمي لتسجيلية الطالب `recordsize`.
- تقوم الطريقة `addrecord` بتكرار الطلب من المستخدم إدخال رقم واسم الطالب ومن ثم إنشاء تسجيلية من النمط `Student` ومن ثم كتابتها في الملف.
- نستخدم المتغير الساكن `pos` للحفاظ على عدد التسجيلات.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.IO;
namespace RandomDemo
{
    class Program
    {
        public static int pos = 0;
        static void Main(string[] args)
        {
            addrecord("D:\\student.txt");
            readFromFile("D:\\student.txt");
        }
    }
    class Student
    {
        private string stnumber;
        private string stname;
        private int recordsize;
        public string stnumber
        { //stnumber property
            set { stnumber = value; }
            get { return stnumber; }
        }
        public string stname
        { //stname property
```

```

        set { stname = value; }
        get { return stname; }
    }
    public int size
    {
        get { return calsize(); }
    }
    private int calsize()
    {
        recordsize = 2 * 15 + 2 * 20; // max record size
        return recordsize;
    }
}
static void addrecord(string path)
{
    Student stu = new Student();
    String con = "y";
    while (con != "n")
    {
        Console.Write("Enter student number:");
        stu.stunumber = Console.ReadLine();
        Console.Write("Enter student name:");
        stu.stuname = Console.ReadLine();
        pos += 1; //update position
        writeToFile(path, stu, pos, stu.size);
        Console.WriteLine("Continue?y/n:");
        con = Console.ReadLine();
    }
}
static void writeToFile(string filename, Student obj, int pos, int size)
{
    FileStream fout;
    BinaryWriter bw;
    //create a file stream object
    fout = new FileStream(filename, FileMode.Append, FileAccess.Write);
    //create a binary writer object
    bw = new BinaryWriter(fout);
    //set file position where to write data
    fout.Position = pos * size;
    //write data
    bw.Write(obj.stunumber);
    bw.Write(obj.stuname);
    //close objects
    bw.Close();
    fout.Close();
}
static void readFromFile(string filename)

```

```

    {
        FileStream fn;
        BinaryReader br;
        Student stu = new Student();
        int currentrecord = 0;
        //open file to read data
        fn = new FileStream(filename, FileMode.Open, FileAccess.Read);
        br = new BinaryReader(fn);
        //read next record
        int i;
        for (i = 1; i <= (int)(fn.Length) / stu.size; i++)
        {
            currentrecord += 1; //update currentrecord position
            fn.Seek(currentrecord * stu.size, 0);
            stu.stunumber = br.ReadString().ToString();
            stu.stuname = br.ReadString().ToString();
            Console.WriteLine(stu.stunumber + "\t" + stu.stuname);
        }
        //update pos to the current position
        pos = currentrecord;
        //close objects
        br.Close();
        fn.Close();
    }
}

```

فيما يلي عينة عن التنفيذ:

```

Enter student number:1
Enter student name:lolo
Continue?y/n:
y
Enter student number:2
Enter student name:toto
Continue?y/n:
y
Enter student number:3
Enter student name:mimi
Continue?y/n:
n
1    lolo
2    toto
3    mimi
Press any key to continue . . .

```

6- تمارين وأنشطة

التمرين الأول:

- قم بكتابة برنامج لإدارة دليل هاتفي يسمح بالعمليات الأساسية: إضافة، حذف، تعديل، حذف. يتم تخزين الاسم ورقم الهاتف ورقم الجوال لكل شخص في الدليل.
- استخدم الملفات التسلسلية في نسخة أولى.
 - استخدم الملفات ذات الوصول العشوائي في نسخة ثانية.

التمرين الثاني:

يطلب استخدام الملفات في كتابة برنامج لإدخال أسماء الطلاب وعلاماتهم في 5 مقررات وتخزينها في ملف ومن ثم طباعة أسماء الطلاب ومعدلاتهم على الشاشة.

التمرين الثالث:

يطلب استخدام الملفات في تخزين قواسم عدد مدخل ومن ثم استخراج القواسم من الملف لاختبار فيما إذا كان العدد تاماً أم لا.

التمرين الرابع:

في هذا البرنامج لدينا ملف يدعى input.txt يحوي قائمة من القيم الصحيحة المخزنة كل قيمة على سطر منفصل. يطلب كتابة برنامج يقوم بإنشاء ملف يدعى output.txt بحيث يقوم البرنامج بقراءة الأرقام المخزنة في الملف input.txt رقماً رقماً ومن ثم اختبار كل رقم تمت قراءته فيما إذا كان أولياً أو لا، بحيث يتم تخزين الأعداد الأولية في الملف output.txt.