Importare dati da un file di testo in Excel con VBA

Essere in grado di manipolare i file di testo e/o CSV con VBA è una competenza che in programmazione risulta molto utile e in questa sezione verrà illustrato il modo per poterlo fare. Un file CSV è composto da dati separati tra di loro da virgole, infatti l'acronimo **CSV** indica Comma Separated Value, ovvero valori separati da virgole e i dati terminano con un ritorno a capo premendo il tasto Invio sulla tastiera (Fig. 1), mentre se un file ha ogni voce su una riga separata con il carattere di tabulazione, allora si dice che sia un file TXT (Fig. 2)

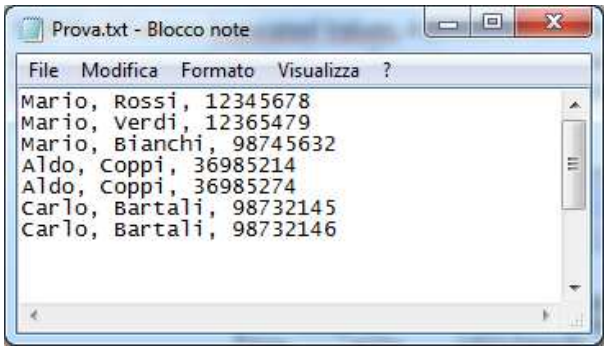

Fig. 1

Fig. 2

Il formato CSV generalmente viene utilizzato per importare e esportare i dati presenti in una tabella e ogni linea di testo del file rappresenta una riga della tabella, mentre le linee rappresentano i campi, divisi da un carattere separatore, che rappresentano i valori presenti nelle varie colonne. È possibile, naturalmente, aprire un file di testo direttamente da Excel, basta seguire il percorso **Dati - Carica dati esterni** dalla sezione Opzioni Testo della barra multifunzione di Excel e verrebbe in tal modo richiamata l'importazione guidata del testo. Tuttavia, è utile sapere che è possibile farlo a livello di programmazione usando VBA. Per meglio comprendere il procedimento in questa lezione andremo ad aprire il file CSV illustrato in *figura 1* spostando il codice numerico che si vede all'estrema destra in *figura 1* nella prima colonna.

Come prima operazione si apre Excel, si crea una nuova cartella di lavoro vuota, si clicca sulla colonna A e si formatta come testo, questo perché il codice nel file di testo è in formato numerico e se si lascia la colonna A formattata come generale (di default), si può ottenere un numero "strano". A questo punto si apre l'Editor di VBA per arrivare alla finestra del codice e si crea una nuova Sub chiamandola *ApriFile* e come prima riga di codice, si aggiunge la seguente:

```
Dim percorso As String
```

Questo codice imposta solo una variabile chiamata *percorso*, ora abbiamo bisogno di conoscere la posizione del file *Prova.csv*. Supponiamo di aver messo il file .CSV nella cartella Documenti, in questo caso è possibile utilizzare il comando **Application.DefaultFilePath** per conoscere il percorso predefinito dove vengono alloggiati i file dal sistema, a questo punto bisogna aggiungere solo il nome del file, preceduto da un backslash:

```
percorso = Application.DefaultFilePath & "\Prova.csv"
```

Se invece il file è stato inserito in qualche altra cartella allora si può fare qualcosa di simile a questo

```
percorso = "C:\Users\user\Documents\Test\Prova.csv"
```

Con questo listato il programma punta a una cartella denominata *Test* che si trova nella cartella *C: \ Users\user*, oppure si deve modificare il percorso per puntare alla cartella in cui è stato salvato il file Prova.csv.

Ora si deve aprire il file e si inizia con la parola chiave **Open**, quindi si specifica il nome del file, una modalità e un numero di file in questo modo

```
Open FileName For Mode As FileNumbe
```

La modalità di cui sopra deve essere sostituita da una delle seguenti operazioni:

- Append: viene utilizzato per aggiungere dati ad un file esistente
- Output: viene usato per scrivere in un file
- Input: viene utilizzato per leggere un file
- Binary: viene usato per leggere o scrivere dati in formato binario
- Random: viene utilizzato per inserire caratteri in un buffer di dimensioni set

Le modalità che ci interessano sono *Input* e *Output* e l'argomento *FileNumber* può essere qualsiasi numero compreso tra 1 e 511 e si precede il numero con il carattere **#**, quindi, se si sta aprendo un file a cui è stato assegnato un indice **#1**, se si apre un secondo file diverso il suo FileNumber sarebbe **# 2**, e così via, quindi il codice diventa:

```
Open percorso For Input As #1
```

Il file che vogliamo aprire è quello che abbiamo memorizzato nella variabile chiamata percorso, si tenga presente che è possibile digitare l'intero percorso del file in questo enunciato, racchiudendolo tra doppi apici:

```
Open "C:\Users\user\Documents\Test\Prova.csv" For Input As #1
```

Con questa istruzione abbiamo accesso al file in sola lettura, la riga successiva è quella di impostare una variabile per i numeri di riga, nel nostro caso possiamo usare una variabile come questa.

```
Nriga = 0
```

Al momento, abbiamo solo detto al VBA di aprire il file, non abbiamo eseguito nessuna azione sul file. Di solito come prima azione si dovrebbe eseguire la lettura dei dati nel file, useremo un ciclo *Do Until* per questo:

```
Do Until EOF (1)
.........
Loop
```

Si noti la condizione del Loop: **EOF** (1), che significa **End Of File**, mentre 1 tra parentesi tonde è il numero di file specificato in precedenza. All'interno del ciclo, inseriamo una linea di codice come la seguente:

```
Line Input # 1, LineaFile
```

Le prime tre voci prima della virgola si riferiscono alla lettura di una singola linea dal numero di file (# 1) e dopo la virgola, si indica a VBA dove si desidera posizionare questa linea, cosa che abbiamo deciso di fare tramite la variabile *LineaFile*. Ad ogni iterazione del Loop, verrà letta una nuova linea dal file di testo e posta nella variabile, tuttavia la linea avrà ancora tutte le virgole, quindi la prima linea sarà:

Mario, Rossi, 12345678

A questo punto è necessario analizzare le righe del file di testo in qualche modo, sia per togliere le virgole che per inserire i valori nelle righe del file Excel.

Un buon modo per analizzare una linea è usando la funzione **Split** tramite la quale è possibile inserire ogni elemento di una riga in un array, cosa che possiamo fare con un codice come il seguente:

```
RigaF = Split (LineaFile, ",")
```

Se osserviamo la riga di codice sopra esposto, notiamo che tra le parentesi tonde della funzione *Split*, abbiamo inserito la variabile *LineaFile* (che contiene la linea che vogliamo dividere), subito dopo è presente una virgola, e infine abbiamo il separatore che vogliamo cercare, nel nostro caso il separatore è la virgola. Quando è terminato il Loop, tramite la funzione Split otterremmo un array chiamato *RigaF* che conterrà tutte le linee del file pulite dal carattere separatore. Si deve tenere presente che il file di testo ha sempre tre elementi per riga (nome, cognome, codice), così sappiamo che l'array va da 0 a 2 posizioni. Ora possiamo andare avanti e mettere ogni elemento in una cella del foglio di calcolo in questo modo:

```
ActiveCell.Offset (Nriga, 0) .Value = RigaF (2)
ActiveCell.Offset (Nriga, 1) .Value = RigaF (1)
ActiveCell.Offset (Nriga, 2) .Value = RigaF (0)
```

Tra le parentesi tonde della funzione *Offset* abbiamo il numero di riga e il numero di colonna e stiamo usando la variabile chiamata Nriga per identificare le righe che devono ricevere i dati. Questa variabile è stata impostata a 0 in precedenza (che incrementeremo a breve), mentre invece le colonne sono sempre compensate a 0, 1 e 2, per cui un valore pari a 0, lo ricordiamo, si mantiene nella stessa colonna, un valore pari a 1 si sposta di 1 colonna, e un valore di 2 si sposta di 2 colonne dalla cella attiva.
A destra del segno di uguale, abbiamo il nostro array *RigaF*, poiché vogliamo che il codice numerico sia posto nella colonna A, abbiamo usato RigaF (2) che indica l'ultimo valore della riga letta nel file di testo e posto come primo valore nel foglio di lavoro dopo il segno di uguale. Abbiamo poi inserito RigaF (1) , che conterrà l'ultimo nome della riga del file di testo e infine, abbiamo RigaF (0) , che conterrà il primo nome della riga del file di testo. L'ultima cosa che rimane da fare all'interno del ciclo è quella di incrementare la variabile Nriga, altrimenti saremo bloccati sulla prima riga del foglio di calcolo, usando questo codice.

```
Nriga = Nriga + 1
```

Quando si apre un file, è necessario chiuderlo da qualche parte nel codice. Questo è abbastanza semplice, lo possiamo fare in questo modo:

```
Close # 1
```

Si digita la parola Close e poi, dopo uno spazio, il numero di file che si sta cercando di chiudere, in pratica tutto il codice dovrebbe essere simile a questo:

```
Sub ApriFile()
Dim percorso As String
    percorso = "C:\Users\user\Documents\Test\Prova.csv"
        Open percorso For Input As #1
Nriga = 0
Do Until EOF(1)
Line Input #1, LineaFile
RigaF = Split(LineaFile, ",")
ActiveCell.Offset(Nriga, 0).Value = RigaF(2)
ActiveCell.Offset(Nriga, 1).Value = RigaF(1)
ActiveCell.Offset(Nriga, 2).Value = RigaF(0)
Nriga = Nriga + 1
Loop
Close #1
End Sub
```

Provate il codice assicurandovi che la cella attiva del foglio di calcolo sia la cella A1 (la colonna A è quella che abbiamo formattato come testo), cliccate poi su un punto qualsiasi all'interno della Sub e premete F5 per eseguirla. Se poi tornate al foglio di lavoro si dovrebbero vedere i dati che sono stati importati dal file di testo

	A	B	C	D
1	12345678	Rossi	Mario	
2	12365479	Verdi	Mario	
3	98745632	Bianchi	Mario	
4	36985214	Coppi	Aldo	
5	36985274	Coppi	Aldo	
6	98732145	Bartali	Carlo	
7	98732146	Bartali	Carlo	
8				

Fig. 3

Scrittura di file di testo in Excel VBA

Abbiamo appena visto come importare i dati da un file di testo con VBA, ora vediamo come scrivere dei dati da un foglio di lavoro in un file di testo, prendendo come base dati quelli elencati in *Fig. 3* e scrivere di nuovo in un file Txt. Il primo compito è quello di trovare un modo per fare riferimento alle celle che ci interessano del foglio di lavoro che come vediamo è composto da 3 colonne e 7 righe, potremmo usare 2 cicli, 1 per scorrere le righe e un altro per scorrere le colonne.

```
For I = 1 To 7
For j = 1 To 3
Next j
Next i
```

Tuttavia, supponiamo che abbiamo aggiunto altre righe e colonne al foglio di calcolo e vorremmo un automatismo che calcoli quante righe e colonne contengano i nuovi dati. Il metodo migliore è quello di ottenere l'ultima riga con i dati in essa contenuti e l'ultima colonna, allora potremmo modificare i cicli in questo modo:

```
For I = 1 To UltimaR
For j = 1 To UltimaC
Next j
Next i
```

La domanda è: come possiamo ottenere i valori delle variabili *UltimaR* e *UltimaC*? Ci sono diverse metodi per trovare l'ultima riga e l'ultima colonna con i dati, un modo popolare per ottenere l'ultima riga con i dati, ad esempio, è questo:

```
UltimaR = Cells(1, "A").End(xlDown).Row
```

Questo codice va all'ultima riga del foglio e poi risale fino a trovare la prima cella nella colonna A che contiene qualcosa. Una tecnica simile è usata anche per trovare l'ultima colonna con i dati, basta sostituire *Row* con *Column*. A questo punto possiamo aprire il file di testo in scrittura, in questo modo:

```
Open percorso For Output As #1
```

Questo comando apre un file in modalità di scrittura e un file non esiste nel percorso indicato verrà creato e se il file esiste verrà sovrascritto. Attenzione, che se si desidera che i nuovi contenuti vengano aggiunti alla fine del file, allora si usa la parola chiave **Append** al posto di **Output**. Per scrivere nel file si deve inserire la seguente riga di codice:

```
Write #1, contenutoF
```

Si deve digitare la parola chiave **Write** seguita dal numero del file e subito dopo si inserisce una virgola seguita dall'array *contenutoF* che contiene i dati che si desiderano scrivere sul file. A questo punto possiamo vedere quale codice possiamo utilizzare, innanzi tutto si crea una nuova Sub e la chiamiamo *ScriviFile* e aggiungiamo 4 variabili ad inizio routine:

```
Dim percorso As String
Dim CellaD As String
Dim UltimaC As Long
Dim UltimaR As Long
```

Per ottenere l'ultima riga e colonna con i dati, aggiungiamo le seguenti due righe:

```
UltimaR = Cells(1, "A").End(xlDown).Row
UltimaC = Cells(1, "A").End(xlToRight).Column
```

È quindi necessario azzerare la variabile CellD e specificare il percorso del file

```
CellD = ""
percorso = Application.DefaultFilePath & "\Prova.csv"
```

Oppure se sappiamo dove si trova il file possiamo utilizzare il metodo già visto per la lettura del file

```
percorso = "C:\Users\user\Documents\Test\Prova.txt"
```

Come possiamo vedere il codice punta a un file chiamato Prova.txt nella cartella Documents, si tenga presente che se non c'è tale file, VBA allora lo creerà. La riga successiva da aggiungere è quella che apre il file in scrittura:

```
Open percorso For Output As #2
```

Si noti che il numero del file alla fine è il numero 2, se ricordate abbiamo già usato il numero 1 in precedenza, quindi ora cercheremo il numero 2 per evitare eventuali conflitti.

Su alcuni sistemi, si può ottenere un errore che indica che il file è già aperto quando si esegue il codice, se è così, chiudete Excel e riapritelo.

Il codice successivo da aggiungere è il doppio ciclo For per leggere i dati e prepararli per la scrittura nel file. Questo spezzone di codice è abbastanza complesso, quindi vediamo prima il listato e poi lo commenteremo.

```
For I = 1 To UltimaR
For j = 1 To UltimaC
If j = UltimaC Then
CellD = CellD + Trim(ActiveCell(I, j).Value)
Else
CellD = CellD + Trim(ActiveCell(I, j).Value) + ","
End If
Next j
Write #2, CellD
CellD = ""
Next i
```

Come abbiamo detto, stiamo eseguendo un ciclo sulle celle del foglio di calcolo, il ciclo esterno si occupa delle righe e il ciclo interno si occupa delle colonne, inoltre all'interno del ciclo interno, abbiamo utilizzato una condizione:

```
If j = UltimaC Then
CellD = CellD + Trim(ActiveCell(I, j).Value)
Else
CellD = CellD + Trim(ActiveCell(I, j).Value) + ","
End If
```

Il motivo per cui vogliamo sapere se **j** è uguale alla variabile *UltimaC* è a causa delle virgole, in quanto vogliamo che ogni riga nel nostro file di testo abbia i dati separati in questo modo:

Mario, Rossi, 12345678

In pratica stiamo scorrendo una cella alla volta del foglio di lavoro e ognuno degli elementi deve essere separato da una virgola, tuttavia, non c'è la virgola alla fine del terzo elemento, per cui se j è uguale a UltimaC non la aggiungiamo eseguendo questo codice

```
CellD = CellD + Trim(ActiveCell(I, j).Value)
```

In pratica qualunque sia la cella attiva (ActiveCell) avrà il suo valore inserito nella variabile CellD. Tuttavia, se j non è uguale a UltimaR allora viene eseguito, il codice

```
CellD = CellD + Trim(ActiveCell(I, j).Value) + ", "
```

L'unica differenza è la virgola alla fine, al primo ciclo interno, *CellD* conterrà il valore *12345678*, mentre al secondo ciclo vi aggiungerà *12345678, Mario,* e all'ultimo ciclo aggiungerà *12345678, Mario, Rossi*. Fuori dal ciclo interno, ma poco prima di Next, abbiamo queste due righe:

```
Write #2, CellD
CellD = “”
```

La prima riga è quella che scrive in realtà la nuova linea nel file di testo, mentre la seconda riporta la variabile CellD pari a una stringa vuota. Le ultime due righe di codice chiudono il file e stampano il messaggio Fatto

```
Close # 2
MsgBox (“Fatto”)
```

Tutto il codice è simile al seguente:

```
Sub ScriviFile()
Dim percorso As String, CellaD As String, UltimaC As Long, UltimaR As Long
UltimaR = Cells(1, “A”).End(xlDown).Row
UltimaC = Cells(1, “A”).End(xlToRight).Column
    percorso = “C:\Users\user\Documents\Test\Prova.txt”
    CellData = “”

Open percorso For Output As #2
For i = 1 To UltimaR
    For j = 1 To UltimaC
        If j = UltimaC Then
        CellD = CellD + Trim(ActiveCell(i, j).Value)
    Else
        CellD = CellD + Trim(ActiveCell(i, j).Value) + “,”
        End If
    Next j
    Write #2, CellD
    CellD = “”
Next i
Close #2
MsgBox (“Fatto”)
End Sub
```

Provate ad eseguire il codice e una volta individuato il nuovo file di testo, si dovrebbe trovare una cosa che assomiglia a questa:

Fig. 4

Non preoccupatevi per i doppi apici che VBA ha aggiunto all'inizio e alla fine di ogni riga