GLI ARCHIVI

in

VISUAL BASIC

A cura del
Prof. Enea Ferri

Introduzione

Ricordiamo che un record:
- E' un insieme di informazioni, non necessariamente dello stesso tipo, che si riferiscono ad un elemento di un certo insieme;
- ogni singola informazione costituisce un CAMPO del record;
- ad ogni campo viene assegnato un NOME, che attribuisce un significato all'informazione (es. COGNOME, NOME…);
- ogni campo ha un tipo e una dimensione, esempio COGNOME AS string*10 è un campo di tipo stringa ed ha una dimensione di 10 byte
- la successione dei nomi dei campi costituisce il FORMATO del record, detto anche TRACCIATO Record
- la lunghezza del record è data dalla somma delle dimensioni dei singoli campi.

Una tabella di record è un Insieme finito di record aventi lo stesso formato**.** E' quindi fondamentale definire come per gli array il numero degli elementi. Un record in una tabella può essere individuato:

1) per chiave
2) per posizione

la chiave è quella parte del record che serve per distinguerlo da tutti gli altri. Può essere costitui ta da uno o più campi del record.
Individuare un record per chiave significa ricercare all'interno della tabella il record avente la chiave specificata;
Individuare un record per posizione implica la conoscenza a priori della posizione del record all'interno della tabella
Terminata l'esecuzione del programma che fa uso della tabella, vengono perse tutte le informazioni che conteneva: questo accade perché la tabella di ecord viene costruita e gestita in memoria centrale.

Generalità sugli archivi

Un archivio è un insieme di informazioni tali che:

tra esse esista un nesso logico (ovvero si riferiscano ad uno stesso argomento);

siano rappresentate secondo un formato che ne renda possibile l'interpretazione (ad esempio elenco telefonico);

siano registrate su un supporto su cui è possibile scrivere e rileggere informazioni anche a distanza di tempo;

la loro organizzazione ne permetta una facile consultazione (ad esempio elenco telefonico ordinato per città e per ogni città in ordine alfab etico).

Un Archivio è un insieme di record, aventi tutti lo stesso Tracciato, ciascuno dei quali è costituito da un insieme di informazioni dette attributi (o *campi).* Tali informazioni descrivono le ***proprietà caratteristiche*** di un *oggetto* o ***ENTITÀ*** della realtà che si sta studiando e consentono di d istinguere i vari esemplari delle entità in esame. Ogni campo è caratterizzato da un **nome**, che permette di fare riferimento ad una proprietà comune a tutte le enti tà (es. campo COGNOME) e da un **valore** che specifica la proprietà del singolo oggetto (es. ROS SI).
Ad esempio l'anagrafica dei clienti di un'azienda è composto da schede aventi tutte la stessa dimensione e contenenti lo stesso tipo di informazione. Ogni scheda rappresenta un record. In figura 1 è riportato un semplice esempio di archivio relativo ad un insieme di persone.

Cognome	***Nome***	***Data Di Nascita***	***Professione***	***Comune_residenza***
Annese	Gemma	5/12/64	Impiegato	Bari
Bianchi	Anna	25/6/86	Studente	Taranto
Rossi	Carlo	1/11/61	Avvocato	Taranto
Rossi	Giuseppe	15/3/63	Insegnante	Brindisi
Vergaro	Alfredo	3/12/85	Studente	Bari

Figura 1 Archivio *PERSONE.*

Per creare un archivio occorre specificare:
- Nome dell'archivio, di solito con riferimento al suo contenuto (es. FORNITORI, CLIENTI, MAGAZZINO…);
- Tracciato record, cioè quali informazioni compongono il record;
- Per ogni campo del record oltre al nome occorre specificare il tipo e la dimensione
- Supporto da usare per archiviare i dati (es. fogli di carta, nastri magnetici, dischi magnetici…)
- Tipo di Organizzazione che verrà utilizzata per la registrazione e la ricerca delle informazioni.

Uno degli attributi (o un gruppo di attributi) di un record viene detto **CHIAVE**, e serve per individuare un record all'interno dell'Archivio.

Una **CHIAVE PRIMARIA** è una chiave che individua in modo univoco un record fra quelli contenuti nell'archivio: il suo valore può essere p osseduto da un solo record. La chiave può essere composta da uno o più campi, in base al tipo di applicazione e di uso che se ne vuole fare.
Si può scegliere come chiave una campo informativo, ad es. il Codice Fiscale di una persona oppure un CODICE FORMALIZZATO, ovvero che non sia un campo informativo. Ad esempio:

a) Codice progressivo numerico : 01, 02, 03, …
b) Codice progressivo alfanumerico : A001, A002, …A010 , B001, ….

Nell'esempio dell'archivio PERSONE una chiave del g enere potrebbe essere un nuovo campo chiamato MATRICOLA, contenente ad es. un codice progressivo alfanumerico.

Nell'archivio AGENDA, avente ad es. il seguente tracciato record:

Data	Ora	Note

una chiave primaria potrebbe esse composta dai campi DATA e ORA

Una chiave non primaria è detta **chiave secondaria**: il suo valore individua, in generale, più di un record. Ad esempio, con riferimento all'archivio *PERSONE*, si può osservare che *Professione e Comune_residenza* sono due possibili chiavi secondarie.
Una chiave secondaria selettiva è una chiave cui è associato un numero relativamente basso di record (ad esempio Professione nell'archivio PERSONE è una chiave secondaria selettiva).
Possono esistere più chiavi primarie, ma nel seguito faremo spesso l'ipotesi che esista un particolare attributo con funzione di chiave primaria. Del resto questa ipotesi è molto realistica: si pensi ad attributi come CodiceArticolo, NumeroMatricola, CodiceFiscale, CodiceCliente ecc.
Le informazioni contenute in un file non devono essere necessariamente di tipo omogeneo. Esiste infatti la possibilità di creare dei file struttura ti, in grado cioè di contenere elementi di pari caratteristiche, oppure composti da informazioni di natura diversa.

Gli Archivi in VISUAL BASIC

Il modo in cui i dati sono organizzati, memorizzati e recuperati è chiamato *organizzazione del file.* Due tipi di organizzazioni comuni sono quella *sequenziale* e quella ad *accesso casuale*.

Per elaborare file di dati sono necessari tre passaggi:

a) ***Aprire*** il file. Prima che qualsiasi dato possa essere scritto sul disco o letto da esso, il file deve essere aperto. In genere un file viene aperto nella routine Form_Load.
b) ***Leggere* o *scrivere*** i record di dati. Si potranno compiere queste operazioni nella routine associata al form per l'immissione dei dati.
c) ***Chiudere*** il file. La chiusura dovrebbe essere sempre l'ultima operazione in un progetto che gestisce file di dati. La posizione migliore è in genere la routine del comando Esci nel form di avvio.

Formato dell'istruzione OPEN

Open *"NomeFile"* **For** {Input | Output | Append | Random} **As** *#NumeroFile* [Len= *LunghezzaRecord]*

Gli elementi tra parentesi graffe sono la **modalità** di accesso, che indica il modo con cui è possibile accedere al fIle. Le parentesi graffe indicano che è possibile effettuare una scelta, e che la voce però non può essere omessa. Le prime tre scelte sono usa te per i file di tipo sequenziale. Il numero di file #NumeroFile può essere tra 1 e 511. La lunghezza de l record può raggiungere i 32.767 caratteri.

Esempi dell'istruzione OPEN

```
Open App.Path &"\FileDati.Dat" For Output As #1
Open "C:\Paghe\Dipendenti.Dat" For Input As #2
```

Il primo esempio apre un file chiamato *FileDati.Dat* come file di output, indicandolo come file #1. Il file viene aperto facendo riferimento ad un percorso relativo, ovvero nella cartella dove si trova l'applicazione che lo deve usare. Il secondo esempio apre un file nell'unità C:\Paghe chiamato *Dipendenti.dat* come file per l'input, chiamandolo #2. In questo caso si fa riferimento ad un percorso assoluto, indicando in modo esplicito la cartella dove si trova il file.

Modalità di accesso	Descrizione
Output	I dati di output del progetto sono scritti su memoria di massa. Eventuali nuovi dati sono scritti all'iniziodel file, sovrascrivendo qualunque dato

	esistente
Input	I dati che sono forniti al progetto dalla memoria di massa. Questa modalità legge dati precedentemente memorizzati.
Append	I dati di output del progetto sono scritti su memoria di massa. In questo caso, i nuovi dati vengono aggiunti alla fine del file.
Random	I dati possono essere letti o scritti e i record possono essere elaborati in un ordine qualsiasi

Ricordare che i file di dati devono sempre essere aperti prima di essere usati. Quando un file di dati viene aperto, sono compiute le seguenti azioni:

1. Viene controllato nella directory se il nome del file indicato esiste. Se il nome del file non esiste, viene creata una voce di directory per questo file, tranne nella modalità Input, che genera un messaggio di errore se il file non esiste.
2. Viene creato un **puntatore** di **file** e impostato all'inizio del file per tutte le modalità eccettuata Append, nella quale la posizione è la fine del file. Il puntatore viene sempre aggiornato per indicare la sua posizione corrente nel file.
3. Al file viene assegnato un **numero** di **file** per potervi fare riferimento. A ciascun file usato in un progetto deve essere assegnato un numero univoco; dopo che il file è stato chiuso, il numero può essere riutilizzato per un file diverso.

Formato dell'istruzione CLOSE

Close [#Numerofile]

Esempi di istruzione CLOSE

Close *#1*

Close #1,#2

Close

L'istruzione Close viene usata per terminare l'elaborazione. di un file su disco. Quando essa è usata senza un numero di file, vengono chiusi tutti i file aperti. L'istruzione Close svolge varie operazioni:

1. Scrive un indicatore di fine del file **(EOF)** alla fine del file.
2. Rilascia il numero del file.

Tenere presente che l'esecuzione di un'istruzione End chiuderà automaticamente tutti i file aperti. Ciò non va però usato come criterio di programmazio ne. Una buona regola è chiudere (con l'istruzione Close) esplicitamente ogni file che èstato aperto nel progetto.

I file ad accesso sequenziale

Le strutture più elementari sono i file ad accesso sequenziale, il cui uso è in genere riservato alla memorizzazione delle informazioni testuali.
L'organizzazione dei dati, tuttavia, fa sì che le strutture di questo tipo non siano adatte a contenere una grande quantità di informazioni, a causa del di latarsi dei tempi di accesso dovuto al fatto che la ricerca di un particolare dato può comportare la le ttura dell'intero file.
I file sequenziali contengono elementi di dati che sono memorizzati uno dopo l'altro in sequenza. In fase di lettura da disco, i dati devono essere letti con la stessa sequenza con cui sono stati scritti. Per leggere un particolare elemento di dati, devono essere stati letti tutti i dati precedenti.

Apertura di un file sequenziale

Un file, di qualsiasi tipo esso sia, per poter essere utilizzato deve necessariamente essere aperto. È possibile effettuare ciò per mezzo del comando *Open*, la cui sintassi, per le strutture ad accesso sequenziale, è la seguente:

```
Open <percorso_file>
[For <modalità>]
As [#]<numero_identificatore>
```

<percorso_file> è una stringa che identifica il percorso del file che si desidera creare o leggere. Il carattere separatore è in tutti i casi costituito dalla barra retroversa "\".
<modalità> indica la maniera con cui si desidera accedere al file. La scelta è fra *Input*, *Output* e *Append*.
Nel primo caso, il file è aperto in lettura, ovvero è utilizzato esclusivamente per reperire delle informazioni senza effettuare su di esse alcuna modifica. È evidente che l'apertura di un file in questa modalità richiede che esso sia stato in prec edenza creato sul disco, altrimenti provoca la notifica di un errore.
Diversamente accade invece per i file aperti in output. In questo caso, infatti, si provoca la generazione di una nuova struttura e la sovrascrittura del file eventualmente già presente sul disco con il nome specificato dopo la parola chiave *Open*. Com'è facile intuire, l'accesso in output ha il fine di permettere la scrittura delle informazioni sui file.
In alcuni casi, tuttavia, risulta necessario inserire all'interno di uno di essi dei nuovi dati senza distruggere quelli inseriti in una o più sessioni precedenti. La modalità da utilizzare per raggiunger e tale scopo è denominata *Append*.

< numero identificatore>
Dopo aver dichiarato l'uso che si desidera fare del file, è necessario assegnare ad esso un numero, che ne costituisce un identificatore univoco. Tale valore deve essere un intero compreso fra 1 e 511 e va specificato dopo la parola chiave *As*. Per le strutture non destinate alla condivisione con altre applicazioni, è opportuno utilizzare i numeri compresi fra 1 e 255, riservando i rimanenti ai file che devono essere resi accessibili da più processi nello stesso momento. Si noti che, per compatibilità con il linguaggio BASIC classico, il numero di identificazione si può far precedere dal carattere *#*.

Scrittura di dati su un file sequenziale

Usare l'istruzione Write # per inserire dati in un file sequenziale. Per poter eseguire l'istruzione Write #, il file deve essere stato aperto in modalità Output o Append. L'alternativa migliore è di solito la modalità Append, perché può essere usata per creare un file e per aggiungervi dati alla fine. Ricordare che in modalità Append il puntatore del f ile viene collocato alla fine del file. Se il file non contiene record, l'inizio del file coincide con la sua fine. L'unico caso in cui viene usata la modalità Output è quando si desidera scrivere sopra vecchi dati.
L'elenco dei campi da scrivere può contenere espres sioni stringa, espressioni numeriche o entrambe, e può essere separata da virgole o punti e virgola.

Formato dell'istruzione WRITE

Write *#NumeroFile, LtstaDeiCampi*

Esempi dell'istruzione WRITE

Write *#1,* txtConto.Te.xt, txtDescrizione.Text, txtPrezzo.Text

Write #*2.* stConto. stDescriztone. cPrezzo

L'istruzione Write # invia i campi di dati sul disco. Gli elementi sono separati da virgole, le stringhe sono racchiuse tra virgolette, e dopo l'ultimo elemento vengono inseriti i codici di ritorno a capo e avanzamento riga.

I file ad accesso casuale

I file ad accesso casuale sono caratterizzati dall'organizzazione molto rigida in strutture dette *record*. Un file è quindi composto da un numero variabile di record accodati. Al fine di comprendere meglio il concetto, si pensi a uno schedario, quale ad esempio l'anagrafica dei clienti di un'azienda. L'archivio è composto da schede aventi tutte la stessa dimensione e contenenti lo stesso tipo di informazione. Ogni scheda rappresenta un record.
Si deve quindi creare un nuovo tipo di dati, il RECORD. Tale operazione è eseguita per mezzo della struttura *Type*, la cui sintassi è:

```
Type <nomeRecord>
<nome_campo_1> As <tipo_1>
<nome_campo_2> As <tipo_2>
…
<nome_campo_n> As <tipo_n>
End Type
```

All'interno della struttura devono essere dichiarati gli elementi che la compongono, che sono denominati *campi*. Per ognuno di essi deve essere specificato il tipo di dati che lo caratterizza. I campi possono essere anche di tipo totalmente diverso. Si possono infatti definire dei record contenenti contemporaneamente delle informazioni di tipo testuale, numerico e logico. Ad esempio, la seguente dichiarazione è corretta:

```
Type Auto
  Codice As String*10
  Marca As String*50
  Modello As String*50
  Cilindrata As Integer
  Diesel As Boolean
End Type
```

Si noti che sono stati dichiarati due campi di tipo alfanumerico, uno di tipo logico e uno numerico intero. La struttura è denominata *Auto*. In questo modo si è provveduto ad aggiungere un nuovo tipo di dati a quelli standard previsti da Visual Basic. È quindi possibile dichiarare la variabile *WorkAuto*, di tipo *Auto* digitando:

```
Dim WorkAuto As Auto
```

Le variabili definite come record prevedono, data la propria conformazione, una modalità di assegnamento dei valori leggermente diversa rispetto a quella prevista dalle strutture convenzionali. In genere, si esegue un'operazione di assegnamento per ogni campo, secondo la sintassi:

```
<variabile>.<campo> = <valore>
```

Ad esempio, volendo assegnare il valore 2499 al campo *Cilindrata* della variabile *WorkAuto*, definita in precedenza, occorre scrivere:

```
WorkAuto.Cilindrata = 2499
```

L'apertura di un file ad accesso casuale

Come per le strutture sequenziali, l'apertura di un file ad accesso casuale avviene per mezzo dell'istruzione *Open* che, in questo caso, assume la forma:

```
Open <percorso_file> For Random As [#]<identificatore>  Len = lunghezza_record>
```

dove *percorso_file* indica il percorso completo del file che si desidera aprire, mentre *identificatore* costituisce un numero utilizzato per contraddistinguere in modo univoco tale struttura e pertanto va

fornito come parametro a tutti i comandi che sono destinati a gestirla. Si noti che la modalità di accesso è indicata per mezzo della parola chiave *Random*, che deve essere obbligatoriamente specificata, indipendentemente dal tipo di operazione che si desidera effettuare sul file, sia essa di lettura o scrittura. Si noti altresì che anche il parametro *Len* è obbligatorio. Esso deve contenere l'esatta dimensione del record che costituisce l'un ità di informazione memorizzata nel file. Qualora essa non sia nota, può essere determinata per mezzo della funzione *Len*. Ad esempio, volendo assegnare alla variabile *DimRec* la dimensione del record *WorkAuto*, è necessario digitare:

DimRec = Len(WorkAuto)

Il contenuto della variabile *DimRec* costituisce il valore da passare come ultimo parametro all'istruzione *Open* per consentire la gestione di un file composto da elementi di tipo *Auto*. Analogamente ai file sequenziali, anche le strutture ad accesso casuale devono essere chiuse dopo l'uso per mezzo dell'istruzione *Close*.

La lettura di un record

La lettura di un record contenuto in un file ad accesso casuale avviene per mezzo dell'istruzione *Get*, caratterizzata dalla seguente sintassi:

Get [#]<identificatore>, <posizione>, <record>

dove <*identificatore*> rappresenta il numero che univocamente identifica il file oggetto dell'operazione di lettura e <*variabile*> è il nome della variabile in cui i dati letti devono essere posti. Il parametro <*posizione*> indica la posizione del record da leggere. Si tratta di un valore intero compreso fra 1 e il numero dei record presenti nel file. Ad esempio, si supponga di voler accedere al quarto record presente nel file di identificatore 1 e di voler porre il suo contenuto nella variabile di tipo record *WorkAuto*. Ciò è possibile per mezzo della riga:

Get #1, 4, WorkAuto

Solo in un caso il valore del parametro <*posizione*> può essere omesso; ciò si verifica in occasione dell'effettuazione di operazioni di lettura in sequenza; l'assenza del numero indicante la posizione provoca infatti l'accesso al record successivo a quello corrente. Non possono tuttavia essere omesse le virgole di separazione. Ad esempio, la sequenza

```
Get #1, 4, WorkAuto
```

```
Get #1,, WorkAuto
```

provoca la lettura del quarto e del quinto record del file identificato dal numero 1.

A proposito di identificatore

Per evitare di dover assegnare personalmente un identificatore ad ogni file da gestire, esiste in Visual Basic l'istruzione **FreeFile** che assegna ad una variabile di tipo integer il primo numero di identificatore libero tra quelli ancora da assegnare. Così, se nel programma sono stati già aperti 3 file con identificatori risp. 1, 2, 3, il seguente procedimento aprirà il file Magaz.dat con identificatore 4.

Id_file=FREEFILE

Open App.Path & "\Magaz.dat" For RANDOM as #Id_file Len=LEN(WORKMAG)

dove WORKMAG è il record di lavoro.

La scrittura di un record

Per scrivere il contenuto di un record in un file ad accesso casuale è possibile utilizzare l'istruzione *Put*, la cui sintassi è pressoché identica a quella delcomando *Get*:

Put [#]<identificatore>, <posizione>, <record>

In questo caso, la variabile indicata come terzo parametro è il record contenente i dati da scrivere. Ad esempio, la riga

Put #1, 5, WorkAuto

scrive il contenuto della variabile *Dato* nel quinto elemento del file identificato dal numero 1. Il valore assunto dal parametro <*posizione*> assume un'importanza fondamentale, in quanto determina se è aggiunto un nuovo record all'archivio o se ne è sovrascritto uno già esistente. Quest'ultima evenienza si verifica quando è indicata una posizione già occupata da un elemento. Per aggiungere un nuovo record al file, invece, è necessario indicare un valore pari al numero totale dei record incrementato di un'unità.
Per conoscere il numero totale dei record contenuti in un File si assegna ad una variabile di tipo contatore il risultato della divisione tra la lunghezza espressa in byte del file interessato e la lunghezza del record. Per ottenere la lunghezza in byte del file si usa l'istruzione LOF(identificatore) dove LOF significa Length Of File. Ad es. l'istruzione:

CONTA = LOF(1) / LEN(WorkAuto)

Fornirà al programma il numero di record contenuti nel file con identificatore 1. Se il file è stato appena creato, CONTA avrà valore 0.

La cancellazione logica di un record

Il metodo più semplice per cancellare un record consiste nel sovrascriverlo con un elemento vuoto. In questo modo, tuttavia, non è possibile recuperare lo spazio da esso occupato sul disco. Si tratta cioè di una cancellazione *logica*, non *fisica*, in quanto Visual Basic non dispone di un'istruzione in grado di rimuovere un record e di recuperare automaticamente lo spazio da esso occupato. È possibile sfruttare a proprio vantaggio la possibilità di effettuare solo una cancellazione logica dei record contenuti in un file per fare in modo che degli elementi eventualmente eliminati per sbaglio possano essere agevolmente recuperati. Ciò è possibile aggiungendo un campo booleano alla struttura dei record e facendo in modo che il programma che accede all'archivio consideri cancellati tutti gli elementi caratterizzati dal contenere il valore logico *False* all'interno di questo campo. L'eliminazione di un record comporta quindi la semplice variazione del valore di un suo campo. Analogamente, è possibile recuperare un elemento cancellato per errore impostando nuovamente al valore *True* il campo booleano. La struttura *Auto* può pertanto essere modificata come segue:

```
Type Auto
 Codice As String*10
 Marca As String*50
 Modello As String*50
 Cilindrata As Integer
 Diesel As Boolean
 Flag As Boolean
End Type
```

La cancellazione fisica di un record

Quando la quantità di informazioni da gestire diven ta elevata, la necessità di recuperare lo spazio occupato dai record cancellati diventa evidente, sia per evitare lo spreco di spazio sul disco, sia per non ridurre drasticamente i tempi di accesso alle informazioni costringendo il programma a leggere dei dati inutili. Come già osservato in precedenza, Visual Basic non dispone di un'istruzione in grado di provvedere automaticamente alla cancellazione fisica di un record. Tuttavia, la scrittura di una simile procedura non presenta un livello di difficoltà elevato. Essa deve solo creare un nuovo file e copiare al suo interno tutti i record non cancellati. Successivamente, deve eliminare il primo file ed assegnare il suo nome alla nuova struttura. È ciò che fa la procedura di seguito descritta, ch e riceve come parametro il nome del file da compattare, che si suppone composto da record di tipo *Auto*:

```
Sub CompattaFile()
Dim ID_old As Integer
Dim ID_new As Integer
Dim WorkAuto As Auto
ID_old = FreeFile
Open App.Path & "\Automobili.dat" For Random As ID_old Len = Len(WorkAuto)
ID_new = FreeFile
Open App.Path & "\Temp.dat" For Random As ID_new Len = Len(WorkAuto)
Do While Not EOF(ID_old)
Get ID_old, , WorkAuto
If WorkAuto.Flag= true Then
Put ID_new, , WorkAuto
End If
Loop
Close ID_old, ID_new
Kill App.Path & "\Automobili.dat"
Name App.path & "\Temp.dat" As App.Path &"\Automobili.dat"
End Sub
```

Si noti l'uso della funzione *FreeFile*, che restituisce un numero adatto ad essere utilizzato come identificatore di file ed evita così il rischio di utilizzare degli identificatori già in uso in altre parti del programma. La procedura provvede a leggere in modo sequenziale il file *NomeFile.dat* e a copiare in un file denominato *Temp.dat* tutti i record per i quali il campo *Flag* assume il valore True. Si noti che le operazioni di lettura e scrittura sono eseguite sequenzialmente, in quanto è stato omesso il valore indicante la posizione nelle istruzioni *Get* e *Put*. Il ciclo di copiatura termina quando sono esauriti i record da leggere. Quando ciò avviene, la funzione EOF (*End Of File*), già descritta nella scorsa lezione, restituisce il valore *True*. Dopo aver copiato tutti i record non cancellati logicamente, la procedura provvede a chiudere entrambi i file. Si noti che a tal fine utilizza un'unica istruzione *Close*, in cui gli identificatori dei file da chiudere sono separati da una virgola. Il passo successivo consiste nel sostituire il file originale con quello creato. Ciò comporta l'esecuzione di due operazioni: la cancellazione del file di origine e la ridenominazione di quello generato dalla procedura. L'eliminazione avviene per mezzo dell'istruzione *Kill*, la cui sintassi è

```
Kill <Nome_file>
```

Il file *Temp.dat* è quindi rinominato per mezzo dell'istruzione *Name*, che è caratterizzata dalla seguente sintassi:

```
Name <Vecchio_nome> As <Nuovo_nome>
```