



# I file di dati

---



# 1) I file sequenziali

---

- Utili per la memorizzazione di informazioni testuali
- Si tratta di strutture organizzate per righe e non per record
- Non sono adatte per grandi quantità di dati per il dilatarsi dei tempi d'accesso alle singole informazioni



## 1.1 Apertura e chiusura di file sequenziali

---

- Prima di poter effettuare qualsiasi operazione sui file è necessario **“aprire il file”**
- Al termine del suo utilizzo è necessario **“chiudere il file”**
- Apertura file:
  - Un file sequenziale può essere aperto in 3 modi diversi:
    - **Input** → per operare in lettura
    - **Output** → per operare in scrittura dall’inizio del file (se vi sono altre informazioni sul file queste vengono sovrascritte)
    - **Append** → per operare in scrittura ponendo i nuovi dati in coda a quelli esistenti



## 1.1) Apertura e chiusura di file sequenziali

---

### ○ Sintassi per apertura file:

**Open**<Nome> [**For**<Modalità>] **As**[#]<Identificatore> [**Len=**<DimBuffer>]

**<Nome>** → Stringa contenente il nome del file compreso path  
(prova.txt; ..\prova.txt; c:\...\prova.txt; Dati\prova.txt)

**<Modalità>** → Input o Output o Append (qualora si dovesse omettere la modalità di apertura, il file verrà aperto in accesso casuale che vedremo in seguito)

**<Identificatore>** → Un numero intero da 1 a 255 con il quale verrà identificato il file all'interno del codice

**<DimBuffer>** → Numero di byte riservati in memoria centrale per l'accesso al file. Non può superare i 32767 byte. (Di solito si omette e si lascia al sistema la libertà di usare la dimensione più adatta)



## 1.1) Apertura e chiusura di file sequenziali

---

- Esempio di apertura:
  - **Open "Prova.txt" For Append As #1**
  - **Open "C:\Documenti.txt" For Input As #2**
  - **Open "..\Pippo.txt" For Output As #3**
- Note:
  - Se si apre un file in **Input** e questo **non esistere** → **viene generato un errore.**
  - Se si apre un file in modalità **Append o Output** e questo **non esistere** → **viene creato automaticamente.**
  - Se si apre un file in modalità **Output** ed il file **già esisteva** → questo **viene sovrascritto** con la perdita dei precedenti dati.



## 1.1) Apertura e chiusura di file sequenziali

---

### ○ Sintassi per chiusura file:

**Close** [#]<Identificatore>

<**Identificatore**> → Il numero intero usato per quel file in apertura

### ○ Esempio di chiusura:

- **Close #1**
- **Close #2, #3**



## 1.2) Il controllo sull'esistenza di un file

---

### ○ Sintassi:

- **Dir** (<Nome>)

<Nome> → Stringa contenente il nome del file compreso path (prova.txt; ..\prova.txt; c:\....\prova.txt; Dati\prova.txt)

- La funzione restituisce una stringa vuota ("") nel caso in cui il file non dovesse esistere



## 1.3) Le operazioni di lettura e scrittura su un file sequenziale

---

### ○ Scrittura su file

- E' possibile operare in scrittura su di un file precedentemente aperto utilizzando:
  - **Print** # <Identificatore>, <Dato>
  - **Write** # <Identificatore>, <Dato1>, <Dato2>, <Dato3>....., <DatoN>
- Con "Print" si scrive nel file e **non** si ha lo **spostamento a capo linea**, inoltre non si può scrivere più di **un dato per volta**.
- Con "Write" si scrive nel file **un pacchetto di dati** e si ottiene lo **spostamento a capo linea**
- Nota: Tutti i dati che aggiungiamo in scrittura si accodano ai dati eventualmente presenti.



## 1.3) Le operazioni di lettura e scrittura su un file sequenziale

---

- Lettura da file
  - E' possibile operare la lettura da un file precedentemente aperto utilizzando:
    - **Line Input** # <Identificatore>, <Dato>
    - **Input** # <Identificatore>, <Dato1>, <Dato2>, <Dato3>....., <DatoN>
  - **"Line Input"** si usa se in scrittura si era usato **"Print"**
  - **"Input"** si usa se in scrittura si era usato **"Write"**
  - Nota: Tutti i dati che leggiamo sono quelli successivi a quelli eventualmente già letti



## 1.3) Le operazioni di lettura e scrittura su un file sequenziale

---

- Per verificare l'eventuale fine della lettura di un file si utilizza la seguente funzione booleana:
  - EOF (<Identificatore>)
    - Restituisce True se si è raggiunto la fine del file
    - Restituisce False in caso contrario

Nota: L'identificatore non è preceduto del cancelletto "#"



## 2 I file ad accesso casuale (random)

---

- I file di questo tipo sono costituiti da un numero variabile di **record** accodati
- Si può immaginare di avere un archivio di schede ognuna delle quali rappresenta un record



## 2.1) La definizione dei record

---

- Il **record** non è un tipo predefinito di Visual Basic ma deve essere definito dal programmatore mediante l'istruzione **Type** da inserire esclusivamente all'interno di un modulo di codice standard (file .bas)
- I tipi di dati definiti con l'istruzione Type fanno parte dei **tipi personalizzati** o **tipi definiti dall'utente**
- **Sintassi** per definire un tipo record:
  - [Private | Public] **Type** <Nome tipo>  
    <Nome Campo1> As <Tipo1>  
    <Nome Campo2> As <Tipo2>  
    <Nome CampoN> As <TipoN>

**End Type**



## 2.1) La definizione dei record

---

- Esempio di definizione tipo record:
  - **Type** RecStudente  
Cognome **As String** \*30  
Nome **As String** \*30  
Classe **As String** \*10  
Credito **As Byte**  
**End Type**
- Una volta definito il nuovo tipo chiamato **RecStudenti** è possibile dichiarare variabili del tipo personalizzato appena creato:
  - **Public** Studente1 **As RecStudente**
  - **Public** Studente2 **As RecStudente**
- L'assegnamento del singolo campo della struttura avviene con la notazione punto(.)
- Esempio:
  - Studente1.Credito=12
  - Studente1. Nome="Marco"



## 2.2) L'apertura dei file ad accesso casuale. Le funzioni Len, FreeFile e LOF

---

- L'apertura di un file ad accesso casuale avviene con la stessa sintassi vista per l'apertura di un file ad accesso sequenziale con l'unica differenza nella Modalità di accesso richiesto (**Random**):

Esempio:

**Open**<Nome>[**For Random**]**As**[#]<Identificatore> **Len=**<DimRecord>

- L'unica modalità di accesso a questi tipi di file è "Random" ed è inoltre la modalità di default.



## 2.2) L'apertura dei file ad accesso casuale. Le funzioni Len, FreeFile e LOF

---

- La dimensione **Len=** è invece obbligatoria e deve coincidere con la dimensione in byte del singolo record che si vuole leggere o registrare.
- Per stabilire la dimensione di un record si usa la funzione **Len()** alla quale viene passata il nome di una variabile precedentemente dichiarata come record:
  - Esempio:  
`DimRecord = Len (Studente1)`



## 2.2) L'apertura dei file ad accesso casuale. Le funzioni Len, FreeFile e LOF

---

- Esempio di apertura di un file "Scuola.dat"

```
DimRecord=Len(Studente1)
```

```
Open "C:\Scuola.dat" For Random As #1 Len=DimRecord
```

- Analogamente si potrebbe:

```
Open "C:\Scuola.dat" For Random As #1 Len=Len(Studente1)
```



## 2.2) L'apertura dei file ad accesso casuale. Le funzioni Len, FreeFile e LOF

---

- Funzioni utili nella gestione dei file:
  - La costante di sistema "FreeFile" ci consente di avere il primo riferimento numerico disponibile per non rischiare di utilizzare riferimenti già usati nel programma:


`NumeroFile=FreeFile`

- La funzione "LOF(<Identificatore>)" ci restituisce la dimensione in byte di un file:

`DimFile=LOF(1)`

Nota: L'identificatore non è preceduto del cancelletto "#"

Se dividiamo la dimensione di un file con la dimensione di un singolo record otteniamo il numero di record registrati



## 2.3) Le operazioni di lettura e scrittura su un file ad accesso casuale

---

- Per effettuare la lettura di un record contenuto in un file ad accesso casuale viene utilizzata l'istruzione "Get" con la seguente sintassi:

`Get[#]<Identificatore>, [<Posizione>], <VariabileRecord>`

- Identificatore → Numero identificatore del file
- Posizione → Numero intero che rappresenta la posizione del record all'interno del file che si vuole leggere.
- VariabileRecord → Variabile in cui vengono trasferiti i dati letti



## 2.3) Le operazioni di lettura e scrittura su un file ad accesso casuale

---


### ○ Esempio di lettura:

- Supponiamo di voler accedere al terzo record del file "Scuola.dat" e porre il contenuto del record all'interno della variabile "Studente" si scriverà:

Get #1, 3, Studente

- Qualora si dovesse omettere il numero del record da leggere, il sistema leggerà il record successivo a quello eventualmente già letto o scritto o posizionato:

Get #1, , Studente




## 2.3) Le operazioni di lettura e scrittura su un file ad accesso casuale

---

- Per effettuare la scrittura di un record all'interno di un file ad accesso casuale viene utilizzata l'istruzione "Put" con la seguente sintassi:

Put[#]<Identificatore>, [<Posizione> ],<VariabileRecord>

- La variabile "VariabileRecord" in questo caso contiene il record da scrivere.
- Si tenga presente che il valore di "Posizione" è in questo caso fondamentale per evitare di sovrascrivere involontariamente record già presenti nel file.



## 2.3) Le operazioni di lettura e scrittura su un file ad accesso casuale

---

- Esempio di Scrittura:


- Supponiamo di voler scrivere un record i cui valori sono contenuti nella variabile "Studente" al quinto posto del file "Scuola.dat" si scriverà:

Put #1, 5, Studente

(se nella posizione 5 c'era un record precedentemente caricato, questo viene sovrascritto)

- Qualora si dovesse omettere il numero del record da scrivere, il sistema scriverà il record successivo a quello eventualmente già letto o scritto o posizionato:

Put #1, , Studente



## 2.3) Le operazioni di lettura e scrittura su un file ad accesso casuale

---

- Esempio di Scrittura:
  - Se si desidera aggiungere in coda al file "Scuola.dat" un record, sarà necessario stabilire prima il numero di record presenti nel file e successivamente scrivere nella posizione successiva:

`DimRecord=Len(Studiante)`

`PosizioneLibera=LOF(1) \ DimRecord+1`

`Put #1, PosizioneLibera, Studiante`



## 2.4) Il posizionamento all'interno di un file ad accesso casuale

---

- L'Istruzione "Seek" imposta la posizione, all'interno di un file aperto, in cui avrà luogo la successiva operazione di lettura o scrittura.

- Sintassi:

Seek[#]<Identificatore>,<Posizione>

- Esempio di lettura del 15° studente:

Seek #1, 15

Get #1, , Studente



## 2.4) Il posizionamento all'interno di un file ad accesso casuale

---

- La funzione "Seek()" restituisce un numero (Long) che specifica la posizione di lettura/scrittura corrente, ossia il numero del record su cui avrà effetto la successiva operazione di lettura/scrittura (se non viene indicato un numero di record nelle istruzioni Get e Put).

- Sintassi:

Seek(<Identificatore>)

- Esempio:

Numero=Seek (1)

` Se Numero assume valore 21 allora la prossima  
` istruzione opera sul record 21 all'interno del file

Get #1, , Studente



## 2.4) Il posizionamento all'interno di un file ad accesso casuale

---

- La funzione "Loc()" opera come la funzione Seek() ma restituisce un numero (Long) che specifica la posizione dell'ultimo record letto o scritto.

- Sintassi:

Loc(<Identificatore>)

- Esempio:

```
Seek#1,71
```

```
i=1
```

```
Do
```

```
    Get #1, ,Classe5A[i]
```

```
    i=i+1
```

```
Loop While Loc(1)=90
```



## 2.5) La cancellazione logica e la cancellazione fisica

---

- All'interno di un file random, i record non possono essere cancellati fisicamente a meno di ricopiare l'intero file in un nuovo file escludendo i record che si vogliono cancellare fisicamente.
- Si ricorre quindi all'artificio di effettuare una **cancellazione logica** dei record che non interessano e successivamente si effettua una **cancellazione fisica** con la tecnica della riscrittura di un nuovo file.



## 2.5) La cancellazione logica e la cancellazione fisica

---

- Cancellazione Logica:
  - Nella definizione del record da usare, viene inserito un campo "Cancellato" di tipo booleano usato per contrassegnare i record che si vogliono cancellare logicamente pur lasciandoli all'interno del file. Il programma sarà scritto in modo tale da considerare:
    - **cancellati** tutti i file che conterranno il campo "Cancellato" a True
    - **non cancellati** i record che conterranno il campo "Cancellato" a False.



## 2.5) La cancellazione logica e la cancellazione fisica

---

- Esempio:

```
Type RecStudente
  Cognome As String *30
  Nome As String *30
  Classe As String *10
  Credito As Byte
  Cancellato As Boolean
```

```
End Type
```

```
Public Stud As RecStudente
```

```
Get #1,12,Stud
```

```
  Studente.Cacellato=True
```

```
Put #1,12,Stud
```

- Si è operata la cancellazione logica dello studente posto nella posizione 12 all'interno del file (senza eliminarlo fisicamente)