

- [Home Page](#) 
[Informazioni](#) 
[Aiuto](#) 

Corso Intermedio - Lezione 4

http://www.vbsimple.net/intermed/int_04.htm

- [L'interfaccia utente di base.](#)
- [La generalizzazione del codice.](#)
- [Le funzioni Len e LenB.](#)
- [Le stringhe a lunghezza fissa.](#)

In questa lezione finalmente procederemo con l'implementazione vera e propria del progetto discusso nelle lezioni precedenti. Cominceremo naturalmente dal posizionare i controlli sulla superficie del form definendo l'interfaccia utente.

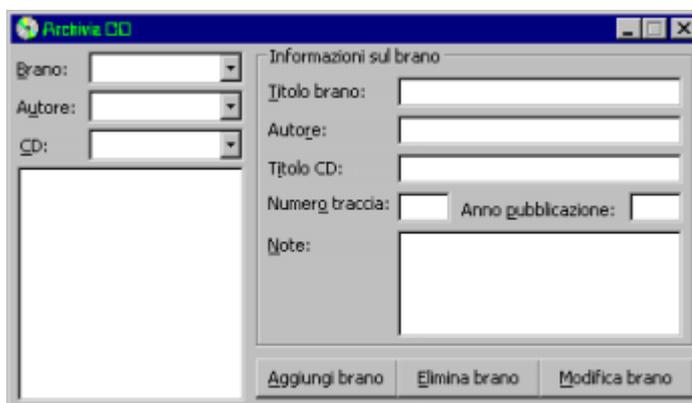
Rivediamo nella figura a fianco il nostro form come dovrà apparire.

La sezione sinistra conterrà una *Label*  di nome **lblFiltroBrani** e la corrispondente *ComboBox*  di nome **cboFiltroBrani**.

Seguono in maniera del tutto simile la *Label* di nome **lblFiltroAutori** e la

ComboBox di nome **cboFiltroAutori** ed infine la coppia **lblFiltroCD** e **cboFiltroCD**.

Tutte le *ComboBox* avranno la proprietà *Style* impostata a **0 - Dropdown Combo** in modo da consentire all'utente la digitazione di dati o la ricerca dall'elenco.



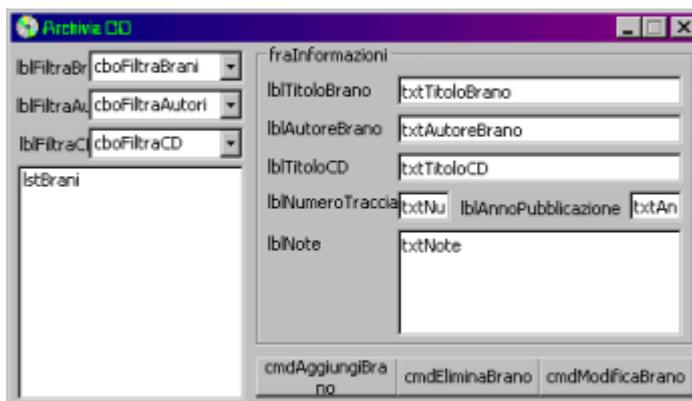
Possiamo notare sin da subito che abbiamo iniziato ad usare le [convenzioni di denominazione](#) dei controlli (e più avanti nelle variabili) al fine di semplificare la scrittura e soprattutto la successiva riletura del codice del progetto. È infatti assolutamente normale a distanza di giorni o settimane dimenticare il corretto funzionamento del progetto. Le convenzioni di denominazione ci aiuteranno in questa ricomprensione.

Sotto il terzetto di *Label* e *ComboBox* avremo una *ListBox*  di nome **lstBrani** che conterrà l'elenco dei brani recuperati dall'archivio dati.

La sezione a destra conterrà invece una serie di coppie *Label* - *TextBox* all'interno di un *Frame*  di nome **fraInformazioni**. Tali coppie hanno i nomi di **lblTitoloBrano** e **txtTitoloBrano**, **lblAutoreBrano** e **txtAutoreBrano**, **lblTitoloCD** e **txtTitoloCD**, **lblNumeroTraccia** e **txtNumeroTraccia**, **lblAnnoPubblicazione** e **txtAnnoPubblicazione**. L'ultima coppia si chiama **lblNote** e **txtNote**; la differenza dalle precedenti coppie sta nel fatto che la *TextBox* ha la proprietà *MultiLine* impostata su **True** per permettere l'inserimento di più righe di testo nella stessa. Questa proprietà non è modificabile durante la [fase di esecuzione](#).

In fondo alla superficie del form sono presenti tre *CommandButton* di nome **cmdAggiungiBrano**, **cmdEliminaBrano** e **cmdModificaBrano**.

Viene illustrata a solo scopo dimostrativo la superficie del form con tutti i nomi dei controlli per semplificarne le posizioni.



Al fine di prevenire errori durante l'esecuzione del programma limiteremo l'inserimento dei dati in base alla lunghezza di ogni membro del record. In altre parole abbiamo definito la lunghezza fissa del campo **Autore del brano** a 30 caratteri. Limiteremo quindi l'inserimento nella corrispondente casella di testo a 30 caratteri. Tale limite potrà essere impostato sia durante la [fase di progettazione](#) che durante [la fase di esecuzione](#).

Una buona regola durante la scrittura di un programma è quella di **generalizzare il funzionamento del codice**. In ragione di quanto detto al paragrafo precedente dovremmo applicare delle limitazioni nell'inserimento dei dati. Abbiamo la possibilità di farlo sia durante la progettazione che durante l'esecuzione.

Abbiamo preferito farlo durante la fase di esecuzione per un banale motivo. Se volessimo modificare l'ampiezza dei campi del record della base dati non dovremmo occuparci di rintracciare quei controlli la cui corrispondente dimensione viene alterata.

In maniera analoga non disabilitiamo alcun controllo durante la fase di progettazione. Sarà più agevole e flessibile farlo durante la fase di esecuzione in funzione delle diverse situazioni. Un'altra buona regola è quella di impostare durante la prima fase soltanto quelle proprietà non modificabili durante la fase di esecuzione.

Effettueremo la limitazione nell'inserimento dei caratteri in base all'ampiezza definita nel campo del record, valutazione che effettueremo mediante l'uso della funzione **Len**, una delle più utili e flessibili funzioni presenti in Visual Basic.

Essa è in grado di restituire l'esatta ampiezza di una stringa o di un dato di altro genere. Nel caso venisse utilizzata con una stringa essa riporterà il numero di caratteri di cui si compone la stringa; per tutti gli altri dati (variabili e costanti) di tipo standard o UDT riporta il numero di [bytes allocati](#) per quel dato, almeno in linea teorica generale perché per certi dati lo spazio di allocazione è maggiore di quello dichiarato; vedi anche le [Informazioni aggiuntive sui tipi di dati](#).

Il suo funzionamento è semplice quanto banale: basterà passare come unico argomento una stringa, una variabile o una costante di altro genere e se ne riceverà in ritorno il numero di caratteri (per una stringa) o il numero di bytes (per tutti gli altri tipi di dato).

La sua utilità si rivela in tutti quei casi in cui si ritiene utile valutare la lunghezza di una stringa (ad esempio per verificare che essa non superi un certo numero di caratteri) oppure per conoscere l'ampiezza di un record, come nel nostro caso.

Un parente molto stretto della funzione *Len* è la funzione *LenB* la cui B aggiuntiva indica bytes. La funzione LenB restituisce sempre il numero di bytes occupati da una stringa, una variabile o una costante. Si differenzia dalla precedente nel caso che l'argomento passato sia una stringa; essa infatti non restituisce il numero di caratteri, ma il numero di bytes occupati da tale stringa (sempre in linea generale però).

Poiché Visual Basic memorizza le stringhe sempre nel formato [Unicode](#) (in opposizione al formato [ANSI](#)) con il quale ogni carattere è costituito da 2 bytes, è chiaro che mentre la prima funzione restituisce il **numero di caratteri** presenti nella stringa, la seconda restituisce il doppio di questo numero, ovvero il **numero di bytes** occupati dalla stessa stringa.

È importante capire quando usare la prima e quando sfruttare la seconda funzione. In nove casi su 10 la prima funzione svolge il lavoro per cui è stata invocata. Infatti, sia che si tratti di stringhe che di altro genere di dati, quasi sempre non ci importa sapere quanta memoria è stata allocata a quella variabile, ma soltanto quanti caratteri è lunga la stringa o quanti caratteri dobbiamo riservare per salvare il contenuto di una variabile in un record.

Abbiamo accennato al fatto che Visual Basic memorizza le stringhe in formato Unicode piuttosto che in ANSI. Ebbene questo è vero in parte o per lo meno nelle viscere del sistema operativo. Nel momento in cui utilizziamo, leggiamo o modifichiamo una determinata stringa, la conversione bytes->caratteri avviene automaticamente ed in maniera del tutto invisibile come se stessimo lavorando con il formato ANSI.

Facciamo un esempio pratico: data la stringa "ABCDE" di 5 caratteri, sappiamo che in memoria lo spazio occupato da essa è il doppio del numero dei caratteri ovvero 10 bytes. Beh saperlo non ci cambia per nulla la vita. 😊

Infatti tutte le operazioni su tale stringa avverranno nella maniera classica: il salvataggio della stessa su un file, la lettura e la modifica influenzano soltanto i caratteri della stringa. Se provassimo a scrivere tale stringa su un file di testo verrebbero scritti soltanto 5 caratteri ovvero 5 bytes. Idem si dica nel momento in cui leggessimo la stringa da un file di testo; sarebbero letti soltanto 5 bytes.

È soltanto una questione interna alla memoria e a Visual Basic che il programmatore può tranquillamente ignorare.

Diverso è il caso quando una stringa Visual Basic deve interagire con funzioni scritte in altri linguaggi. In tal caso la *conversione* non avviene in maniera automatica ma sarà compito dell'utente fornire la stringa nel formato richiesto.

La funzione *LenB*, in genere davvero poco utilizzata si rivela un'arma decisiva durante le valutazioni di questo genere, ma si tratta di esempi fuori dal comune e soprattutto fuori dal Visual Basic puro.

Tornando al nostro progetto, utilizzeremo la funzione *Len* per richiedere, durante la fase di esecuzione, il numero di caratteri di ogni stringa del record definito in precedenza al fine di limitare all'utente l'inserimento di un numero di caratteri superiore.

Abbiamo detto in precedenza che la mancata limitazione nell'inserimento di caratteri potrebbe generare degli errori. In realtà non è esattamente così, o meglio dire, lo sarebbe in una situazione differente dalla nostra.

Abbiamo infatti definito ed utilizzeremo un record a larghezza fissa in cui abbiamo specificato il numero di caratteri di ogni stringa al suo interno. Se, ad esempio assegnassimo al campo **strTitoloCD** una stringa lunga 500 caratteri non si genererebbe alcun errore ed il programma proseguirebbe in maniera del tutto normale.

L'aver, infatti, definito una stringa a lunghezza fissa fa sì che essa non possa contenere un numero di caratteri inferiore o superiore. Gli eventuali caratteri in eccedenza sarebbero rimossi e non presi in considerazione.

In maniera del tutto analoga, se assegnassimo al campo **strTitoloCD** una stringa vuota ovvero una stringa di 0 caratteri, la lunghezza della stringa contenuta in tale campo rimarrebbe comunque quella definita nell'UDT cioè 20 caratteri. Gli eventuali caratteri mancanti sarebbero automaticamente riempiti con spazi alla fine della stringa.

Tale genere di dato è detto infatti *stringa a lunghezza fissa* poiché, indipendentemente dal contenuto assegnato alla stringa, la sua lunghezza rimane comunque costante eliminando le parti in eccedenza o riempiendo le mancanze fino al raggiungimento della dimensione specificata durante la fase di progettazione.

Risulterà adesso ancora più chiaro il significato del termine record a dimensione fissa.

La decisione di limitare l'utente nell'inserimento di dati nelle caselle di testo non è pertanto legata a problemi di natura tecnica (ma potrebbe comunque esserlo in caso di una cattiva implementazione del codice) ma soltanto ad un fattore prettamente umano. Riteniamo inutile e spiacevole *ingannare* l'utente dandogli la possibilità di inserire qualunque testo ma poi troncarli i dati da lui inseriti in funzione delle decisioni del programmatore.

La limitazione nell'inserimento, allora, sarà una risposta automatica al tentativo dell'utente di immettere di un numero di caratteri superiori a quello definito dal programmatore. Gli sarà semplicemente impedito e lui stesso si convincerà ad abbreviare le informazioni.

Inizieremo nella prossima lezione lo studio del codice per l'implementazione di base del progetto analizzato fino ad ora per archiviare i brani dei CD.

[Fibia FBI](#)
6 Febbraio 2002



[Torna alla terza lezione](#)

[Vai alla quinta lezione](#)

