



Stringhe API

http://www.vbsimple.net/news/news_10.htm

Si è accennato più di una volta che nei sistemi Windows le stringhe sono archiviate secondo due codifiche: [ANSI](#) ed [Unicode](#). Windows 95/98/ME utilizzano la codifica ANSI secondo cui ad ogni carattere corrisponde un byte. Windows NT utilizza invece la codifica Unicode per la quale ogni carattere della stringa occupa due bytes, ma consente anche l'utilizzo di stringhe ANSI, sempre preferendo, però, le prime.

[Abbiamo anche detto](#) che Visual Basic maneggia le stringhe utilizzando sempre la codifica Unicode. Altresì molti programmatori in linguaggio C (quelli che lavoravano in ambienti a 16 bit) sono abituati al fatto le stringhe non sono altro che [array](#) oppure [puntatori](#) di caratteri e la dimensione della stringa è determinata dalla presenza di un carattere [NULL](#) terminatore. Tutte le funzioni che manipolano stringhe si aspettano infatti che al termine della stringa ci sia tale carattere, con codice [ASCII](#) 0.

È un'usanza comune troncare una stringa inserendo un valore NULL nel punto in cui essa deve essere interrotta, oppure modificare il punto di inizio della stringa spostando il puntatore al carattere da cui far iniziare la stringa. Questo allo scopo di chiarire che in linguaggio C gli array sono normalissimi puntatori.

Nel momento in cui un programma Visual Basic comincia a [comunicare con un altro programma o una libreria esterna](#) si mettono in ballo dei problemi che inizialmente potevano essere sottovalutati.

Infatti la maggior parte delle funzioni API sono progettate per utilizzare la codifica ANSI e quindi funzionare sia su Windows 9x che su Windows NT. Particolare cura deve pertanto essere apportata nel trattamento delle stringhe. La dichiarazione ([prototipo](#)) di una funzione di libreria conterrà quasi sempre la definizione del tipo di stringa che la funzione richiede. Un esempio di queste dichiarazioni è il seguente:

```
Private Declare Function sndPlaySound Lib "winmm.dll" Alias "sndPlaySoundA" (ByVal  
lpszSoundName As String, ByVal uFlags As Long) As Long  
  
BOOL sndPlaySound(LPCSTR lpszSound, UINT fuSound);
```

La prima delle due righe è la dichiarazione Visual Basic mentre la seconda in linguaggio C come definita nel file header.

La funzione `sndPlaySound` punta in realtà alla funzione **`sndPlaySoundA`** all'interno della libreria WINMM.DLL. La A al termine della funzione indica che le stringhe trattate sono codificate in ANSI. La stessa funzione con codifica della stringhe in Unicode prende il nome di **`sndPlaySoundW`**.

Il tipo di dati **`LP(C)STR`** nella seconda definizione indica un Long Pointer (Constant) String ovvero un puntatore ad una costante stringa e corrisponde al formato standard ANSI. Avremmo anche potuto trovare il tipo **`LP(C)WSTR`**: Long Pointer (Constant) Wide

String corrispondente al stesso tipo di dati ma nella codifica Unicode (Wide).

La C sta ad indicare una stringa costante ovvero che non verrà modificata dalla funzione che la utilizza.

Nel momento in cui passeremo una stringa a tale funzione sarà pertanto passato il [puntatore](#) ad un'area dati contenente tanti caratteri, più uno per il terminatore, quando è lunga la stringa nel formato ANSI ed esattamente il doppio nel caso di Unicode.

Ma poiché in linguaggi di alto livello quale Visual Basic è possibile avere delle stringhe contenenti all'interno anche caratteri NULL con codice ASCII 0, è stato deciso con Visual Basic 3 di utilizzare un nuovo tipo di dati denominato **HLSTR**: High Level String ed erano caratterizzate dal fatto che il puntatore non indirizzava direttamente alla stringa in memoria ma ad una struttura (descrittore) contenente all'interno la lunghezza della stringa ed un altro puntatore all'area dati in cui si trovava la stringa. Non era presente alcun carattere terminatore.

Da Visual Basic 4 in poi è cambiato tutto! Il formato **HLSTR** è stato abbandonato in favore di un nuovo tipo di dati: **BSTR**: Basic String, una struttura molto simile alla **LPWSTR**, costituita appunto da un array di caratteri e da un terminatore finale, ma presenta alcune differenze fondamentali.

- La lunghezza della stringa è memorizzata in una variabile di 4 bytes posta prima dell'area dati e non è quindi determinata dalla presenza del terminatore.
- Il terminatore alla fine della stringa è comunque presente ma non fa parte della stringa; è ovvero escluso dalla lunghezza riportata.
- La stringa può contenere al suo interno altri valori NULL.
- Non è consentito alterare, se non entro certi limiti, la struttura della stringa. OLE definisce alcune funzioni per la manipolazione di questa particolare struttura.

Possiamo schematizzare i quattro tipi di stringhe con:

LPSTR	 Il puntatore alla stringa indica il primo carattere dei dati. La stringa è terminata con NULL ed ogni carattere occupa un byte	10 bytes
LPWSTR	 Il puntatore alla stringa indica il primo carattere dei dati. La stringa è terminata con NULL ed ogni carattere occupa 2 bytes	20 bytes
HLSTR	 Il puntatore alla stringa indica una struttura di due membri: la lunghezza della stringa in bytes ed un puntatore all'area dati. La stringa non è terminata con NULL ed ogni carattere occupa 1 byte	13 bytes
BSTR	 Il puntatore alla stringa indica il primo carattere della stringa. I 4 bytes precedenti ad essa indicano la lunghezza della stringa che è terminata con NULL ed ogni carattere	24 bytes

	occupa 2 bytes	
--	----------------	--

È proposto un esempio per dimostrare la presenza dei 4 bytes di una stringa **BSTR**, che indicano la lunghezza in bytes (non in caratteri) e la presenza del carattere terminatore in Unicode.

Articolo basato su due testi di Bruce Mc Kinney:

1. Hardcore Visual Basic - Seconda edizione 1997
2. Extending Visual Basic with C++ DLLs - Aprile 1996

[Fibia FBI](#)

14 Aprile 2002



[Torna all'indice degli Articoli](#)
