

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

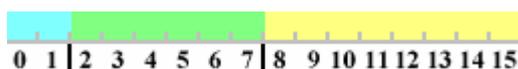
## Analisi di un Variant

[http://www.vbsimple.net/news/news\\_11.htm](http://www.vbsimple.net/news/news_11.htm)

Uno dei [tipi di dati](#) più usati (ed a volte abusati) nei progetti scritti in Visual Basic, tanto che in VBScript si tratta dell'unico tipo di dato, è il **Variant**.

Si tratta di una tipologia di dati mista: una variabile dichiarata come Variant può assumere vari tipi di dati: numeri interi, decimali a singola e doppia precisione, stringhe, booleani, oggetti e persino errori. È il tipo di dati predefinito se non specificato diversamente, tramite un'istruzione `As Tipo` oppure `DefTipo`. Altresì una variabile non dichiarata, qualora non [venga usato Option Explicit](#), viene dichiarata automaticamente con Variant.

Le variabili di tipo Variant occupano 16 bytes di memoria più eventuale spazio aggiuntivo per i dati puntati, nei sottotipi *String* ed *Object*. La loro struttura può essere scissa in tre parti:



**Figura 1**

- I bytes 0 e 1 indicano il sottotipo dei dati
- I bytes da 2 a 7 sono in genere utilizzati fuorché per il sottotipo Decimal
- I bytes da 8 a 15 contengono il valore

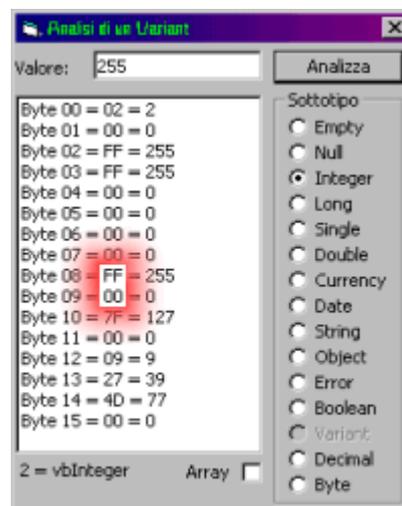
Il sottotipo dei dati è sempre contenuto nel primo byte del Variant: il valore di tale byte sarà una costante dell'[enumerazione VbVarType](#). Il secondo byte (alla posizione 1) conterrà il valore 0 nel caso di una variabile normale, mentre in caso di un [array](#) il secondo byte conterrà il valore [esadecimale](#) 20, corrispondente alla costante `vbArray (&H2000)`.

I bytes alle posizioni 2-7 sono quasi sempre inutilizzati e contengono dei valori non importanti, fuorché nel caso di variabili di sottotipo Decimal.

Gli ultimi 8 bytes conterranno il valore della variabile Variant e saranno utilizzati soltanto quelli necessari alla riproduzione di un valore del sottotipo richiesto. Ad esempio una variabile con sottotipo Integer utilizzerà solo i primi due bytes (16 [bit](#)), mentre con un valore Long saranno utilizzati i primi 4 bytes (32 bit).

È fondamentale ricordare che il byte meno significativo è quello alla posizione 8 mentre quello più significativo si trova alla posizione 15.

Per dimostrare l'analisi di una variabile Variant è stato creato un progetto in grado di scomporre il contenuto di una variabile Variant e mostrarne il contenuto come singoli bytes separati.



Nella figura a fianco è riprodotto il numero 255 utilizzando il sottotipo Integer. La variabile viene analizzata, scomposta ed il suo contenuto è mostrato nell'elenco sulla sinistra.

Poiché le variabili Integer occupano 16 bit, il valore 255 sarà contenuto nei bytes alle posizioni 8 e 9. Il valore verrà quindi riprodotto in esadecimale con **00FF**. Il sottotipo dei dati è indicato nel byte alla posizione 0 e nel nostro caso indica il valore 2, corrispondente alla costante **vbInteger**. Il byte alla posizione 1 indicherà invece il valore 0 poiché si tratta di una variabile semplice e non di un array.

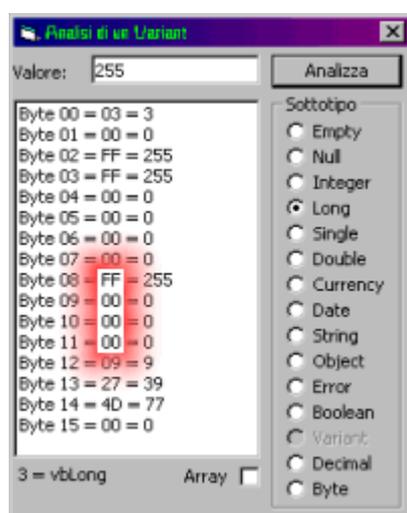


Figura 3

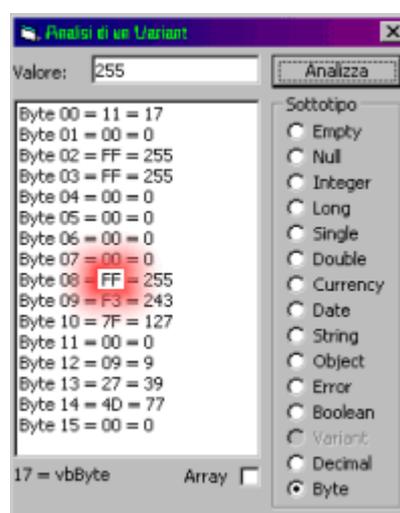


Figura 4

Altresì le variabili di sottotipo Long, che si compongono di 32 bit, contengono i loro dati nei byte 8-11 e nella Figura 3 il valore 255 è mostrato come **000000FF** esadecimale. Nella Figura 4, poiché il sottotipo scelto è il Byte, i dati sono reclusi al byte 8, con il valore esadecimale **FF**.

Quattro casi un po' particolari sono le variabili con sottotipo String, Object, Error e Decimal; vediamo in dettaglio il funzionamento di tali tipologie di dati. Cominciamo analizzando il contenuto di una variabile Variant/String. È fondamentale ricordare che le stringhe sono rappresentate come [puntatori](#) ad una zona contigua di caratteri. Vedi anche l'articolo dedicato alle [stringhe API](#).

Utilizzando il progetto relativo a quest'articolo, inseriamo una stringa nel campo **Valore**, selezioniamo il sottotipo *String*, disattiviamo la CheckBox **Array** e premiamo il pulsante **Analizza**. Utilizzando la funzione **StrPtr**, apparirà in basso un'etichetta con il puntatore alla stringa generata, lo stesso puntatore contenuto nei bytes alle posizioni 8-11; i puntatori, infatti, sono valori a 32 bit. Nell'esempio sottostante il puntatore fa riferimento a **0061E574**.

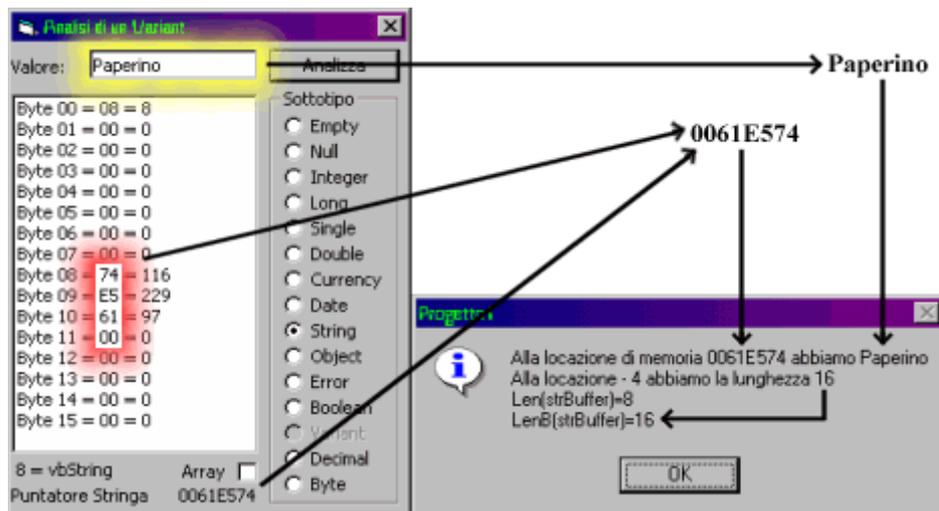


Figura 5

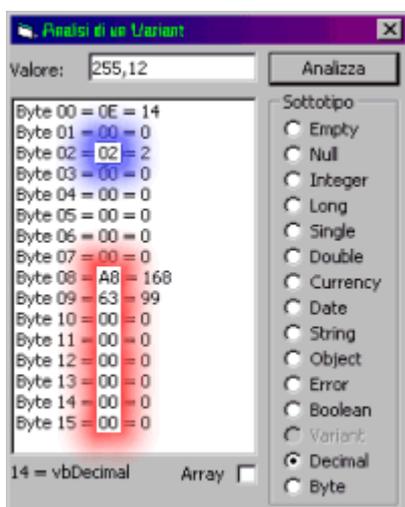
Un click sopra l'etichetta con il puntatore alla stringa rivelerà una serie di informazioni su tale locazione di memoria. Apparirà una MessageBox con il contenuto dell'area di memoria puntata e verrà letta la lunghezza della stringa alla locazione indicata dal puntatore meno 4 bytes. Lo ricordiamo, le variabili stringa **BStr** contengono un riferimento alla lunghezza della stringa esattamente 4 bytes prima del puntatore relativo alla stringa.

La lunghezza riportata sarà esattamente uguale alla lunghezza riportata dall'istruzione [LenB](#) e doppia rispetto all'istruzione [Len](#) sulla stringa.

Le variabili di sottotipo Object si comportano in maniera quasi uguale a quelle di sottotipo String. I bytes alle posizioni 8-11 conterranno infatti un puntatore ad un'area dati. Il puntatore indicato sull'etichetta in basso sarà il medesimo riportato dalla funzione **ObjPtr**. A differenza delle stringhe non sarà facile analizzare il contenuto dell'oggetto puntato.

Un caso molto più semplice sono le variabili di sottotipo Error. Come per i sottotipi precedenti, anche il sottotipo Error mostrerà un valore nell'etichetta in fondo al Form.

Il valore indicato non sarà un puntatore ma piuttosto un codice di errore. Un click sopra l'etichetta mostrerà una finestra di avviso con il codice e la descrizione dell'errore generato.



L'analisi di una variabile con sottotipo Decimal rivelerà qualcosa di differente rispetto tutti gli altri sottotipi di dati analizzati fino ad ora.

Innanzitutto è bene ricordare che il Decimal non è un tipo di dati nativo di VB. Non è pertanto possibile dichiarare variabili di tipo Decimal. L'unico modo per utilizzare un valore Decimal consiste nel richiamare la funzione *CDec* con una variabile di tipo Variant.

Nell'esempio a fianco abbiamo inserito il valore **255,12** nella casella apposita, scelto il sottotipo Decimal e premuto il tasto

Analizza. Il numero sarà quindi contenuto nei bytes alle posizioni 8-15. Tuttavia se analizziamo bene il contenuto di tali dati noteremo cosa realmente accade. Nel nostro esempio il numero è stato riprodotto con **00 00 00 00 00 00 63 A8**; una rapida conversione in decimale dimostrerà che il numero indicato è **25512**, ovvero il nostro numero specificato ma moltiplicato per 100 o per meglio dire privato della sua virgola.

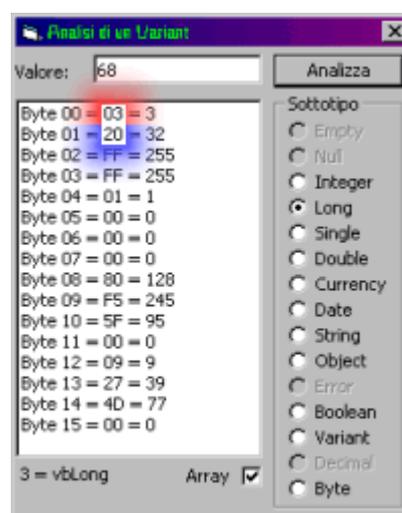
Pertanto i dati riportati alle posizioni 8-15 sono la rappresentazione intera del numero originale. Per memorizzare il numero originale verrà utilizzato il byte alla posizione 2, inutilizzata da tutti gli altri sottotipi di dati. Nel nostro esempio infatti troviamo in tale posizione il valore **02** che rappresenta il moltiplicatore in potenza di dieci del numero intero ( $10^2 = 100$ ).

Per riottenere il numero originale sarà pertanto necessario dividere il numero intero, riportato alle posizioni 8-15 (**25512**) per 10 elevato alla potenza indicata alla posizione 2 (**100**). Nel nostro esempio infatti  $25512 / 100$  restituisce 255,12.

Prima di concludere con l'argomento Variant diamo uno sguardo a cosa accade assegnando ad una variabile Variant un array.

Nella nostra applicazione di esempio abbiamo fatto in modo che i pulsanti di scelta Empty, Null, Error e Decimal venissero bloccati in caso di scelta di un Array; questo perché non è possibile creare vettori di tale tipo di dati, in quanto non nativi in VB.

Nella figura a fianco abbiamo specificato il valore 68, scelto il sottotipo Long ed attivata la CheckBox Array. I dati che verranno rappresentati sono alquanto incomprensibili e rappresentano un puntatore alla struttura creata in memoria per contenere l'array.



Ciò che dovrebbe maggiormente incuriosirci sono i primi due bytes: **03 20**. Il primo di questi due sappiamo indica il sottotipo della variabile Variant; il secondo byte, corrispondente al valore esadecimale 2000 (**vbArray**), indica che tale dato è appunto un [vettore](#) in memoria.

In sostanza quindi i primi due bytes sono dati dalla combinazione mediante [operatore Or](#) delle [costanti](#) **vbArray** e **vbLong** (o altro sottotipo dati).

Fibia FBI  
9 Agosto 2002

 [Torna all'indice degli Articoli](#)