



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

Si, no, forse... 
http://www.vbsimple.net/weird/wrd_06.htm

Difficoltà:  3 / 5

Nel corso base abbiamo [introdotta la logica booleana](#), secondo la quale tutte le espressioni ricevono un risultato Vero o Falso ed abbiamo potuto approfondire l'[algebra booleana nelle informazioni aggiuntive](#). È noto quindi a tutti che se un'espressione restituisce valore Vero, combinando tale espressione con l'operatore **NOT** otteniamo il valore Falso.

Tutto vero se non esistessero le leggi di Murphy a capovolgere tutte le regole esistenti e conosciute nell'universo. 😊

L'esempio trattato utilizza la funzione [API SHFileOperation](#) che necessita a sua volta di una struttura di nome **SHFILEOPSTRUCT**.

La definizione di tale struttura è stata, purtroppo, molte volte errata anche da parte di Microsoft. La documentazione ufficiale MSDN infatti dichiara la struttura  con:

```
1. typedef struct _SHFILEOPSTRUCT{
2.     HWND hwnd;
3.     UINT wFunc;
4.     LPCSTR pFrom;
5.     LPCSTR pTo;
6.     FILEOP_FLAGS fFlags;
7.     BOOL fAnyOperationsAborted;
8.     LPVOID hNameMappings;
9.     LPCSTR lpszProgressTitle;
10. } SHFILEOPSTRUCT, FAR *LPSHFILEOPSTRUCT;
```

E la trasporta quindi in Visual Basic come:

```
1. Private Type SHFILEOPSTRUCT
2.     hwnd As Long
3.     wFunc As Long
4.     pFrom As String
5.     pTo As String
6.     fFlags As Integer
7.     fAnyOperationsAborted As Boolean
8.     hNameMappings As Long
9.     lpszProgressTitle As String
10. End Type
```

[[Tratto da How To Delete a File into the Win95 Recycle Bin \(Q154005\)](#)]

Si tratta effettivamente di una svista di qualche sbadato programmatore, che è stata corretta successivamente con una postilla come questa, aggiunta in altri articoli sullo stesso argomento: *"NOTE: The 32-bit version of Visual Basic aligns structures at double word boundaries, but some API functions, such as the SHFileOperation, expect the data to arrive as byte aligned"*.

Tuttavia lo stesso errore è stato commesso da tantissimi altri programmatori che

sottovalutavano il comportamento di Visual Basic nell'uso di tipi di dati definiti dall'utente, cosicché la precedente definizione di **SHFILEOPSTRUCT** si è diffusa notevolmente facendo passare l'errore inosservato.

Svilupperemo un semplicissimo progetto per dimostrare come un errore di questo genere può facilmente condurre a dubitare sulla correttezza delle regole di base dell'algebra booleana.

```

1. Option Explicit
2.
3. Private Type SHFILEOPSTRUCT
4.     hWnd As Long
5.     wFunc As Long
6.     pFrom As String
7.     pTo As String
8.     fFlags As Integer
9.     fAnyOperationsAborted As Boolean
10.    hNameMaps As Long
11.    sProgress As String
12. End Type
13.
14. Private Const FO_DELETE = &H3
15.
16. Private Declare Function SHFileOperation Lib "shell32.dll" Alias
    "SHFileOperationA" (lpFileOp As SHFILEOPSTRUCT) As Long
17.

```

Le dichiarazioni riprendono la definizione precedente di **SHFILEOPSTRUCT** e vi aggiungono la dichiarazione della funzione API *SHFileOperation* che utilizza tale struttura.

```

18. Private Sub Form_Load()
19.     Dim udtFileOp As SHFILEOPSTRUCT
20.     Open "c:\blabla.txt" For Output As #1
21.     Write #1, "ciao ciao"
22.     Close #1

```

All'avvio del programma sarà creato un file di testo di nome BLABLA.TXT e posto nella radice del disco C:. Questo non è il modo migliore di scrivere un file di testo ma è stata utilizzata la soluzione più semplice possibile. A tale scopo si consulti l'[articolo riguardante la scrittura di un file di testo](#).

```

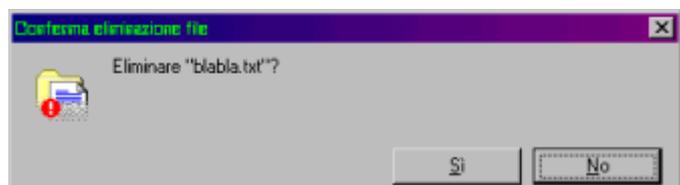
23.     With udtFileOp
24.         .hWnd = Me.hWnd
25.         .pFrom = "c:\blabla.txt"
26.         .wFunc = FO_DELETE
27.         Call SHFileOperation(udtFileOp)

```

Alla riga 27 è richiamata la funzione *SHFileOperation* con la variabile **udtFileOp** che conterrà il percorso del file da eliminare (riga 25) e l'operazione richiesta è la cancellazione (riga 26).

Quando viene richiesto di eliminare il file è fondamentale rispondere **No**.

La funzione al suo ritorno dovrebbe



pertanto indicare che l'utente ha annullato l'operazione ed è questo lo scopo del [membro](#) **fAnyOperationsAborted** della struttura e qui appare la stranezza.

```
28.     MsgBox ".fAnyOperationsAborted=" & .fAnyOperationsAborted
29.     MsgBox "Not .fAnyOperationsAborted=" & (Not .fAnyOperationsAborted)
```

Le due righe 28 e 29 dovrebbero dati risultati opposti. Mentre la prima, come detto restituisce valore Vero, la seconda dovrebbe riportare il risultato Falso, in ragione dell'operatore **Not** usato.

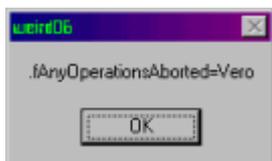


Figura 2

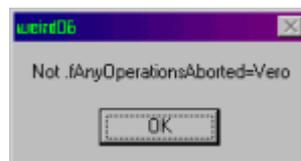


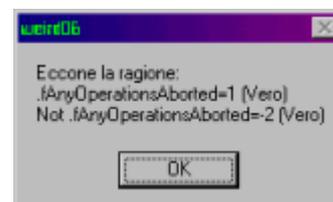
Figura 3

Purtroppo, come mostrano le due figure sovrastanti, in entrambi i casi il risultato è Vero. Sembra assurdo a dirsi ma il contrario di Vero è Vero.

Esiste una spiegazione di questo comportamento a dir poco assurdo ed è legato all'errata dichiarazione della struttura **SHFILEOPSTRUCT**.

```
30.     MsgBox "Eccone la ragione: " & vbNewLine & ".fAnyOperationsAborted=" & CLng
      (.fAnyOperationsAborted) & " (" & CBool(.fAnyOperationsAborted) & ")" & vbNewLine &
      "Not .fAnyOperationsAborted=" & CLng(Not .fAnyOperationsAborted) & " (" & CBool
      (Not .fAnyOperationsAborted) & ")"
31.     End With
32.     Unload Me
33. End Sub
```

L'aver dichiarato il membro come Boolean limita le risposte a Vero e Falso che la variabile restituisce. Tuttavia l'operatore **Not** non valuta l'espressione prima di invertirla ma il linguaggio segue un percorso differente.



L'operatore **Not** in realtà inverte tutti i bit della variabile in questione (*.fAnyOperationsAborted*) ed il risultato è riportato come espressione da valutare.

Una variabile booleana contiene il valore Falso quando tutti i bit che la compongono sono 0 e contiene valore Vero quando almeno uno dei suoi bit sia 1. Solitamente una variabile booleana contiene il valore **0** (0000 0000 0000 0000 = Falso) e la negazione produce quindi il risultato **-1** (1111 1111 1111 1111 = Vero).

Tuttavia, come abbiamo già detto, e come nel nostro caso, se la variabile non contiene il valore 0, il risultato booleano sarà comunque Vero. Nell'esempio precedente infatti la variabile *.fAnyOperationsAborted* assume valore 1 se l'utente ha annullato l'operazione; pertanto la negazione (l'inversione dei bit) del valore **1** (0000 0001) produce il valore **-2** (1111 1110), la cui valutazione booleana corrisponde sempre a Vero. L'unico caso in cui il risultato è Falso avviene quando **tutti i bit** della variabile sono uguali a 0 e l'inversione produce il risultato contrario.



Dopo aver terminato l'uso dell'esempio mostrato in precedenza, ricordarsi di

eliminare manualmente oppure mediante il programma stesso il file **C:\BLABLA.TXT**.

Questo esempio voleva dimostrare come un semplice errore nella dichiarazione può generare una serie di errore difficili da tracciare perché sintatticamente tutto appare corretto.

Per concludere, la dichiarazione corretta della struttura **SHFILEOPSTRUCT** utilizza dei valori Long sia per il membro *fFlags* sia per il successivo *fAnyOperationsAborted*. Si raccomanda quindi di non fidarsi ciecamente delle dichiarazioni API copiate da qualche esempio ma, in caso di errori insoliti, verificare il valore di ciascun membro delle strutture API.

[Fibia FBI](#)

27 Dicembre 2002



[Torna all'indice delle Stranezze](#)
