



## Invertire una stringa

[http://www.vbsimple.net/howto/ht\\_029.htm](http://www.vbsimple.net/howto/ht_029.htm)

- [Prima versione \(lenta, senza l'utilizzo dell'API\)](#)
- [Seconda versione \(lenta, senza l'utilizzo dell'API\)](#)
- [Terza versione \(veloce, con l'utilizzo dell'API\)](#)
- [Confronto tra le tre versioni](#)

In alcune situazioni può essere necessario invertire una stringa, dal primo all'ultimo carattere. Il suo scopo si rivela utile nelle operazioni di ricerca partendo dalla fine. Ben lo sapevano i programmatori della [Microsoft](#), tanto che dalla versione 6 tutte le [edizioni di Visual Basic](#) includono la funzione di inversione di una stringa chiamata **StrReverse**.

Ma come sviluppare questa funzione con Visual Basic 5?

Sono proposte tre versioni: una semplicissima ma molto lenta, una di confronto con la prima, sempre lenta, ed una più complessa che utilizza l'[API](#), più orientata al linguaggio C che al Visual Basic.

---

### Prima versione (lenta, senza l'utilizzo dell'API)

Difficoltà: 1 / 5

Per effettuare l'inversione di una stringa sarà necessario leggere ogni carattere della stringa partendo dall'ultimo carattere e memorizzarlo nella nuova stringa.

La prima soluzione consiste nell'utilizzare l'istruzione *Mid* per estrarre un carattere partendo da una posizione specifica.

```
1. Public Function SlowStrReverse(ByVal Stringa As String) As String
2.     Dim Conta As Integer
3.     For Conta = Len(Stringa) To 1 Step -1
4.         SlowStrReverse = SlowStrReverse & Mid(Stringa, Conta, 1)
5.     Next Conta
6. End Function
```

La funzione è semplicissima: il suo nome è **SlowStrReverse** (ovvero StrReverse lenta) e richiede come parametro la stringa da invertire.

Al suo interno abbiamo un ciclo che parte dall'ultimo carattere della stringa e procede fino al primo. All'interno del ciclo viene estratto di volta in volta un carattere e viene aggiunto alla stringa *SlowStrReverse*, che restituirà alla fine della funzione, la stringa invertita.

Questa procedura, tuttavia, risulta molto lenta perché l'istruzione *Mid*, di volta in volta, deve estrarre un carattere specifico.

Inoltre la funzione richiede che il valore (la stringa) sia [passata per valore](#). Sembra un

particolare da poco, ma ogni volta che la funzione viene richiamata deve essere effettuata una copia della stringa da invertire nella variabile-parametro Stringa. Ciò comporta un ulteriore rallentamento. Pertanto una [chiamata per riferimento](#) risulta di fatto, in alcuni casi, nettamente più veloce, poiché la funzione riceve soltanto un [puntatore](#) e non una copia della stringa da invertire.

Per ulteriori informazioni consultare la sezione [Informazioni aggiuntive sulle funzioni](#).

---

## Seconda versione (lenta, senza l'utilizzo dell'API)

Difficoltà: ► 1 / 5

Questa seconda versione rappresenta una variante della versione precedente, a dimostrare l'inefficienza del metodo risultativo. Questa funzione si differenzia dalla precedente per il fatto che ogni volta che un carattere viene estratto dalla stringa, viene di fatto eliminato dalla stringa originale.

```
1. Public Function Slow2StrReverse(ByVal Stringa As String) As String
2.     While Len(Stringa) > 0
3.         Slow2StrReverse = Slow2StrReverse & Right(Stringa, 1)
4.         Stringa = Left(Stringa, Len(Stringa) - 1)
5.     Wend
6. End Function
```

Alla riga 3 viene estratto l'ultimo carattere della stringa ed alla riga esso stesso viene eliminato, cosicché ogni volta la stringa diventi sempre più corta.

Apparentemente potrebbe sembrare un'ottimizzazione del metodo precedente in quanto la stringa da trattare diventa regolarmente più breve; ma in realtà non è affatto così. Per spiegare questo bisogna risalire al linguaggio C con cui l'ambiente e il compilatore di Visual Basic sono scritti.

In linguaggio C le stringhe sono memorizzate come [vettori](#) di caratteri, ed i riferimenti ai vettori sono [puntatori](#). Pertanto per il compilatore, estrarre il carattere in posizione n anziché m è esattamente la stessa cosa; infatti la richiesta di estrarre un carattere dalla posizione n si trasforma in una richiesta di lettura dall'area di memoria x, senza che il compilatore di fatto debba analizzare tutta la stringa.

Pertanto l'operazione di accorciamento della stringa si trasforma in un ciclo inutile che rallenta ulteriormente l'operazione di inversione della stringa.

Altresì non sarà possibile effettuare la chiamata alla funzione per riferimento, in quanto essa lavora direttamente sul parametro Stringa. L'utilizzare la chiamata ByRef opera la distruzione dei dati all'uscita della funzione di inversione.

Le due versioni saranno paragonate più avanti, ma le motivazioni appena accennate, da sole, dovrebbero bastare a dimostrare che conoscere il funzionamento delle stringhe nel linguaggio C può di fatto, evitare inutili rallentamenti.

---

## Terza versione (veloce, con l'utilizzo dell'API)

Difficoltà:  3 / 5

L'ultima versione che vedremo sarà la più efficiente, sia perché utilizzeremo funzioni [API](#), notoriamente più veloci delle funzioni fornite dal linguaggio Visual Basic, sia perché riporteremo le stringhe al loro stato naturale, ovvero [vettori](#) di caratteri.

Il codice che sembra assomiglia molto ad una soluzione di linguaggio C anziché di Visual Basic, ma si dimostra estremamente efficiente.

```

1. Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" (pDst As Any,
   pSrc As Any, ByVal ByteLen As Long)
2.
3. Public Function FastStrReverse(ByVal Stringa As String) As String
4.     Dim Buffer1() As Byte
5.     Dim Buffer2() As Byte
6.     Dim Conta1 As Integer
7.     Dim Conta2 As Integer
8.     Conta1 = Len(Stringa)
9.     ReDim Buffer1(Conta1)
10.    ReDim Buffer2(Conta1 - 1)

```

La funzione si chiamerà **FastStrReverse** (StrReverse veloce).

Il [tipo di dati](#) che meglio rappresenta i caratteri è il semplice Byte. In questa funzione sfrutteremo il fatto che le operazioni su un singolo numero sono molto più veloci delle operazioni effettuate sopra un stringa. Infatti in linguaggio C ogni stringa, anche di un solo carattere, deve contenere un codice ASCII 0 che identifica la sua fine. Quindi l'estrazione di un solo carattere da una stringa comporta oltre alla ricerca della posizione, anche l'[allocazione](#) di un'area di memoria per la stringa e la copia della stringa in tale area.

La funzione utilizza una funzione API di nome *RtlMoveMemory*, ma ridefinita da molti, per comodità, **CopyMemory**. Essa richiede tre parametri: un puntatore all'area di memoria di destinazione (*pDst*), uno all'area di partenza da cui leggere (*pSrc*) ed un valore indicante il numero di bytes da copiare (*ByteLen*).

Questa funzione lavora direttamente sulla memoria, ignorando di fatto i tipi di dati, i limiti e la congruenza d'essi. Ciò comporta quindi notevoli rischi in caso di errori di calcolo.

Alle righe 4 e 5 abbiamo dichiarato due vettori di caratteri (Byte) senza dimensione fissa, di nome **Buffer1** e **Buffer2**. Essi serviranno per contenere i dati della stringa da invertire e lavoreremo sopra d'essi anziché sopra le stringhe.

La dimensione dei vettori viene determinata in fase di esecuzione mediante l'istruzione *Redim*. Questo evita di dichiarare vettori troppo grandi o troppo piccoli per contenere i dati della stringa da invertire.

Nota però che il secondo vettore è di un carattere più piccolo del primo vettore. Questo perché la funzione API lavora con le convenzioni C che richiedono alla fine un codice ASCII 0 di terminazione.

```

11.    CopyMemory Buffer1(0), ByVal Stringa, Conta1
12.    Conta2 = 0
13.    For Conta1 = Conta1 - 1 To 0 Step -1
14.        Buffer2(Conta2) = Buffer1(Conta1)
15.        Conta2 = Conta2 + 1
16.    Next Conta1
17.    FastStrReverse = StrConv(Buffer2, vbUnicode)
18. End Function

```

Alla riga 11 abbiamo la chiamata alla funzione *CopyMemory*. Essa copierà i dati da Stringa al vettore Buffer1. La dimensione del buffer da copiare è indicata in Conta1. Nota che la funzione richiede che sia passato come primo argomento un valore, non una matrice; pertanto dovremo utilizzare come indirizzo di destinazione il primo valore del vettore.

Mediante quest'operazione, potremo richiamare i caratteri della stringa direttamente senza utilizzare l'istruzione *Mid* e l'elaborazione lavorerà su numeri interi, anziché su stringhe.

Nel ciclo alle righe 13-16 effettueremo l'estrazione di un valore dal primo vettore e lo inseriremo nel secondo vettore. Quest'operazione risulta particolarmente veloce poiché non sono necessarie [allocazioni](#) di memoria temporanee.

Così, all'uscita dal ciclo, il vettore Buffer2 conterrà la nostra stringa invertita, ma in formato [ANSI](#) invece che [UNICODE](#). Si rende pertanto necessaria un'ultima conversione di formato prima della restituzione del risultato finale (riga 17).

## Confronto tra le tre versioni

Difficoltà: ► 1 / 5

La prova delle tre versioni verrà effettuata mediante cronometrazione di un certo numero di inversioni di una stringa a scelta dell'utente.

Inseriamo sopra un form una *TextBox* `ab1` di nome Testo con il testo **"Stringa da invertire"** ed una *TextBox* `ab2` di nome Ripetizioni con il testo **"5000"**.



Inseriamo anche una matrice di *CommandButton* `ab3` di nome **Inverti**, con indici da 0 a 2. Le caption dei tre pulsanti sono: **"Inverti (Lento)"**, **"Inverti (Lento 2)"** e **"Inverti (Veloce)"**. Aggiungiamo anche una *Label* `ab4` di nome **TempoImpiegato**.

Tutto il codice si riassume in una sola routine (sono naturalmente escluse le tre funzioni viste in precedenza):

```

1. Private Declare Function GetTickCount Lib "kernel32" () As Long
2.
3. Private Sub Inverti_Click(Index As Integer)
4.     Dim Tempo As Long
5.     Dim Conta As Long
6.     Dim Risultato As String
7.
8.     Tempo = GetTickCount
9.     For Conta = 1 To Val(Ripetizioni.Text)
10.        Select Case Index
11.            Case 0: Risultato = SlowStrReverse(Testo.Text)
12.            Case 1: Risultato = Slow2StrReverse(Testo.Text)
13.            Case 2: Risultato = FastStrReverse(Testo.Text)
14.        End Select
15.    Next Conta

```

```
16.     Tempo = GetTickCount - Tempo
17.     TempoImpiegato.Caption = "Tempo Impiegato: " & CStr(Tempo) & " msec"
18.     Me.Caption = Risultato
19. End Sub
```

Alla prima riga abbiamo dichiarato la funzione API `GetTickCount` che restituisce il [numero di millisecondi trascorsi dall'avvio di Windows](#). Essa ci servirà come cronometro per testare la velocità delle tre funzioni.

Nella routine che gestisce l'[evento](#) ⚡ Click sopra i tre pulsanti, inseriremo il nostro codice cronometrato. Abbiamo dichiarato la variabile **Tempo** che servirà per effettuare i tempi di calcolo delle funzioni; la variabile **Conta** servirà per eseguire più volte le funzioni di inversione di stringa; infine la variabile **Risultato** conterrà la stringa invertita.

Alla riga 8 viene letto il tempo iniziale dalla pressione del pulsante. Segue un ciclo che si ripete tante volte quanto il numero scritto nella casella di testo **Ripetizioni**. All'interno di tale ciclo sarà necessario controllare quale pulsante è stato premuto, mediante controllo del parametro *Index*. Ad ogni tasto premuto corrisponderà una funzione di inversione. Il testo da invertire è specificato nella proprietà Text della casella **Testo**.

Solo dopo che la funzione è stata richiamata il numero di volte richiesto, la funzione potrà terminare, ma prima verrà calcolato il tempo di elaborazione (riga 16) ed esso sarà mostrato nella Label **TempoImpiegato** (riga 17).

In chiusura della funzione, sarà cambiata la Caption del form con il risultato dell'inversione.

Come già anticipato, la funzione più veloce è la terza (*FastStrReverse*), mentre si dimostra più lenta la seconda versione (*Slow2StrReverse*).



Comunque, la vera velocità della terza versione si dimostra nelle richieste di inversioni di stringhe superiori a 20 caratteri.

In questo esempio, è vano richiedere che il parametro Stringa per la prima funzione (*SlowStrReverse*) sia passato per riferimento, anziché per valore, in quanto non è possibile passare una proprietà di un'oggetto per riferimento.

[Fibia FBI](#)  
21 Febbraio 2001

 [Torna all'indice degli HowTo](#)