



## Calcolo del CRC32 di un testo

[http://www.vbsimple.net/activity/act\\_18.htm](http://www.vbsimple.net/activity/act_18.htm)

**Richiesta di:** [Claudio Gucchierato](#) - 23 Settembre 2001

**Difficoltà:**  4 / 5

*Avrei la necessità (e la curiosità) di sapere come si può calcolare il CRC di un file. Immaginando che la cosa possa interessare anche altre persone (in rete non si trovano esempi documentati in italiano) vi suggerirei, se lo ritenete possibile, di dedicare un nuovo argomento nella sezione Richieste dei lettori.*


Detto, fatto! La richiesta è alquanto interessante sia come problema pratico (calcolare il CRC32 di un file) sia come argomento didattico (elaborazione di dati binari).

Il **CRC**, *Cyclic (o Circular) Redundancy Check* ovvero **Controllo a ridondanza ciclica**, è un algoritmo di calcolo utilizzato per verificare la correttezza dei dati durante la trasmissione o la rilettura degli stessi; tali algoritmi sono due: il *CRC16* che utilizza un valore a 16 bit, quindi una precisione di  $2^{16}$ , ed il *CRC32* che utilizza un valore a 32 bit, quindi una precisione di  $2^{32}$ .

L'algoritmo del CRC si basa sull'analisi dei singoli caratteri del testo da analizzare. Ognuno di questi viene analizzato bit per bit fino alla generazione di un codice di hash rappresentativo del testo analizzato. Il risultato del CRC32 è infatti un numero a 32 bit compreso tra 0 e oltre 2 miliardi.

L'utilizzo di Visual Basic per calcoli binari di questo genere non è proprio ottimale e la scelta di un altro linguaggio quale il C produrrebbe sicuramente risultati migliori. Tralascieremo, pertanto, la pura efficienza del codice e ci concentreremo sullo studio dell'applicazione dell'algoritmo. Si raccomanda la consultazione delle [informazioni aggiuntive sui sistemi di numerazione](#) e [sull'algebra booleana](#). Questo articolo tratterà l'algoritmo a 32 bit e si snoda lungo quattro sezioni:

1. Calcolare la tabella utilizzata per il calcolo del CRC32;
2. Calcolare il CRC32 di un testo;
3. Come applicare il CRC32;
4. Ulteriori applicazioni del CRC32.

Per comodità abbiamo inserito il codice per il calcolo del CRC32 all'interno di un modulo  standard e lasciato nel form soltanto il codice per la lettura del file.

### **Calcolare la tabella utilizzata per il calcolo del CRC32**

Il primo passo verso il calcolo del CRC32 consiste nel generare una tabella di 256 elementi, ognuno dei quali contenente un valore a 32 bit. Esiste una particolare procedura per generare questi valori, alquanto complessa da spiegare. Vedremo subito il codice che ci chiarirà le idee:

```
1. Option Explicit
2.
3. Private blnTabellaCalcolata As Boolean
4. Private lngTabellaCRC(255) As Long
5.
6. Private Const lngSemeCRC As Long = &HEDB88320
7. Private Const lngDefaultCRC As Long = &HFFFFFFFF
8.
```

Poiché l'elaborazione della tabella del CRC32 può richiedere un certo tempo utilizzeremo una variabile di tipo *Boolean* (riga 3) che informi la funzione che effettua il calcolo del CRC32 che la tabella è già stata calcolata e pertanto non ha bisogno di essere elaborata nuovamente.

Alla riga 4 è dichiarata una [matrice](#) di 256 elementi ognuno dei quali è un [numero Long](#) [ovvero un intero a 32 bit](#).

Alle righe 6 e 7 sono dichiarate due costanti  utilizzate più avanti come valori di default.

```
9. Private Sub InizializzaTabella()
10.   Dim intBytes As Integer
11.   Dim intConta As Integer
12.   Dim lngTmpCRC As Long
13.   Dim lngValoreCRC As Long
14.
```

La funzione di calcolo della tabella è chiamata *InizializzaTabella* ed è [dichiarata Private](#) per renderla inaccessibile alle altre parti del codice. Sarà la funzione che calcola il CRC ad eseguirla in base al valore della variabile **blnTabellaCalcolata**.

Tralasciando per il momento le altre variabili, quella di nome **lngValoreCRC** conterrà il valore che assumerà ogni elemento della tabella.

```
15.   For intBytes = 0 To 255
16.     lngValoreCRC = intBytes
17.     For intConta = 0 To 7
18.       lngTmpCRC = (lngValoreCRC And &HFFFFFFFE) \ 2 And &H7FFFFFFF
```

Sarà eseguito un calcolo per ogni elemento della matrice (quindi 256 volte). Il valore iniziale di **lngValoreCRC** corrisponderà all'indice di ogni elemento della tabella (riga 16).

Segue un ciclo che verrà ripetuto 8 volte. Nessuna attenzione al valore della variabile **intConta**: essa non sarà utilizzata in nessuna parte del codice.

Merita invece una particolare attenzione la riga successiva (riga 18): essa indica l'esecuzione di uno shift a destra di 1 bit senza segno, detto anche *ShiftRight*. Per una spiegazione sul concetto del BitShifting si rimanda alla sezione [Articoli - News](#). In particolare essa svolge le seguenti operazioni:

- Imposta il primo bit del numero lngValoreCRC a 0  
(lngValoreCRC And &HFFFFFFFE)
- Divide il numero ottenuto per 2 ( $2^1$  bit)


- Imposta l'ultimo bit di tale risultato a 0  
And &H7FFFFFFF
- Assegna il risultato dell'espressione alla variabile **lngTmpCRC**

```

19.         If lngValoreCRC And &H1 Then
20.             lngValoreCRC = lngTmpCRC Xor lngSemeCRC
21.         Else
22.             lngValoreCRC = lngTmpCRC
23.         End If
24.     Next intConta
25.     lngTabellaCRC(intBytes) = lngValoreCRC
26. Next intBytes
27. blnTabellaCalcolata = True
28. End Sub
29.

```

Alla riga 19 viene verificato se il primo bit (quello all'estrema destra) del numero **lngValoreCRC** - che inizialmente viene posto uguale al valore indicato da **intBytes** - è impostato su 1.

Nel caso esso lo sia, il valore del numero **lngTmpCRC** ottenuto alla riga 18 viene mischiato mediante l'[operatore XOR](#) con il valore seme (*salt*) standard indicato dalla costante  **lngSemeCRC**.

L'algoritmo standard del CRC32 prevede l'uso del valore di seme 3.988.292.384 corrispondente al valore esadecimale EDB8 8320. Tutti gli algoritmi di calcolo del CRC32 utilizzano tale valore e per questa ragione producono sempre lo risultato analizzando il medesimo file. In un'implementazione propria del CRC32 è possibile utilizzare un valore di seme differente per generare quindi un valore di CRC32 differente. In situazioni normali viene utilizzato il valore indicato dalla costante **lngSemeCRC**.

Tornando all'analisi del codice, se invece il primo bit è uguale a 0 il valore del numero **lngTmpCRC** non viene mischiato con alcun numero. In entrambi i casi il valore risultante viene assegnato alla variabile **lngValoreCRC**.

Fatto questo il ciclo procede fino all'esecuzione di 8 volte. Quel valore **lngValoreCRC** che alla riga 16 conteneva l'indice della matrice, dopo la prima esecuzione conterrà un numero differente, risultante dall'operazione di RightShift di 1 bit e l'eventuale mix con il valore seme eseguita per 8 volte consecutive prima di essere assegnato come valore della matrice **lngTabellaCRC** alla riga 25.

Terminata l'elaborazione di tutti i valori della tabella, dopo 256 iterazioni, il valore della variabile **blnTabellaCalcolata** viene impostato su **True** per impedire una successiva rielaborazione che richiederebbe ulteriore tempo. La tabella verrà infatti elaborata soltanto una volta.

## Calcolare il CRC32 di un testo

Il procedimento per il calcolo del CRC32 vero e proprio è relativamente semplice e si riassume in un'unico calcolo rappresentabile in linguaggio C come:

```
CRC = (CRC >> 8) XOR CRCTABLE[INPUT XOR (CRC AND FFh)]
```

Il calcolo rappresenta 3 operazioni:

- RightShift del CRC precedente di 8 bit;
- Mix mediante operatore XOR tra il codice [ASCII](#) del carattere in esame e la parte bassa (compresa tra 0 e 255) del CRC precedente;
- Mix tramite operatore XOR del primo valore trovato con il valore contenuto nella tabella all'indice indicato dal secondo valore trovato.

Beh... forse non è poi così semplice come sembrava. 😊

Vediamo allora passo dopo passo di cosa si tratta.

```
30. Public Function CalcolaCRC(ByVal Buffer As String, Optional ByVal CRCPrecedente As
    Long = lngDefaultCRC, Optional UltimoCRC As Boolean = False) As Long
31.     Dim Carattere As Byte
32.     Dim Conta As Integer
33.     Dim Temp As Long
34.     Dim IndiceTabellaCRC As Long
35.
```

La funzione che calcola il CRC prende il nome di (ovvio) *CalcolaCRC* e richiede un parametro obbligatorio corrispondente alla stringa di cui calcolare il CRC. È possibile passare altri due parametri opzionali: **CRCPrecedente** indica il valore di CRC ottenuto in precedenti elaborazioni, utile per effettuare elaborazioni parziali di testi di grosse dimensioni; se esso non viene fornito viene assunto il valore di default indicato dalla costante *lngDefaultCRC*. L'altro parametro opzionale è un valore booleano di nome **UltimoCRC** che indica se il calcolo del codice CRC è terminato e pertanto può essere applicata l'ultima trasformazione; il valore predefinito è *False*.

All'interno della funzione sono dichiarate 4 variabili: la prima, di nome **Carattere**, identifica il codice [ASCII](#) del carattere in esame nella stringa da elaborare; la seconda, di nome **Conta**, verrà utilizzata soltanto per effettuare un ciclo su tutti i caratteri della stringa da elaborare; le ultime due variabili: **Temp** ed **IndiceTabellaCRC** rappresentano i primi due passaggi descritti nella spiegazione del precedente codice in linguaggio C, ovvero il RightShift di 8 bit ed il mix mediante operatore XOR, operazioni che saranno descritte più avanti.

```
36.     If blnTabellaCalcolata = False Then InizializzaTabella
37.     CalcolaCRC = CRCPrecedente
38.     For Conta = 1 To Len(Buffer)
39.         Carattere = Asc(Mid$(Buffer, Conta, 1))
40.         Temp = (CalcolaCRC And &HFFFFFF00) \ &H100 And &HFFFFFF
41.         IndiceTabellaCRC = Carattere Xor (CalcolaCRC And &HFF)
42.         CalcolaCRC = Temp Xor lngTabellaCRC(IndiceTabellaCRC)
43.     Next Conta
```

Innanzitutto prima di procedere con l'elaborazione del CRC32 è fondamentale che la tabella **lngTabellaCRC** sia elaborata. Se il valore della variabile **blnTabellaCalcolata** è *False* la funzione di inizializzazione della tabella non è stata richiamata in elaborazioni precedenti e pertanto l'operazione dovrà essere svolta ora, prima di procedere con il codice. Nel caso che la tabella fosse già stata elaborata in precedenza sarebbe superfluo ed impegnativo elaborarla nuovamente.

Il valore iniziale del CRC è dato dal parametro **CRCPrecedente** fornito oppure ottenuto per default. Segue un ciclo eseguito per ogni singolo carattere della stringa **Buffer** (riga 38). Al suo interno vengono eseguite le quattro operazioni descritte in precedenza:

- Viene estratto il codice ASCII del carattere in esame e memorizzato nella variabile **Carattere** (riga 39);
- L'operazione di RightShift del CRC di 8 bit viene memorizzata nella variabile **Temp** (riga 40);
- Viene recuperato e memorizzato nella variabile **IndiceTabellaCRC** l'indice della matrice effettuando il mix del valore di Carattere con la parte bassa (gli ultimi 8 bit) del CRC precedente (riga 41);
- Il nuovo valore di CRC viene generato mischiando il valore indicato dalla variabile **Temp** con il valore contenuto nella tabella alla posizione indicata da **IndiceTabellaCRC** (riga 42).

Tale passaggio è ripetuto per ogni carattere della stringa di cui calcolare il CRC32.

```
44.   If UltimoCRC = True Then CalcolaCRC = Not CalcolaCRC
45. End Function
```

Prima di concludere la funzione viene effettuato un ultimo controllo: se il valore del parametro **UltimoCRC** è **True** sarà necessario effettuare un'ultimo aggiustamento al valore del CRC32 invertendone tutti i bit (riga 44).

L'operazione è necessaria SOLTANTO per l'ultimo pezzo di CRC da calcolare.

## Come applicare il CRC32

Vediamo un semplice utilizzo del modulo appena sviluppato effettuando il calcolo del codice CRC32 di un file di testo a scelta dell'utente. È importante considerare che il file scelto dall'utente può essere di qualunque dimensione: è pertanto errato provare a caricarlo interamente in memoria e calcolare il CRC32 in un'unica operazione. Il file pertanto sarà letto a blocchi di 2 KB per volta ed ogni volta sarà calcolato il CRC32 di ogni blocco. La funzione *CalcolaCRC* infatti permette un calcolo segmentato, mediante l'utilizzo del parametro **CRCPrecedente**.

Il nostro form si comporrà di tre soli e semplici controlli: una Label descrittiva di nome **lblNomeFile**, una TextBox in cui

l'utente immetterà il percorso del file di cui calcolare il CRC32, di nome **txtNomeFile** ed un semplice pulsante di nome **cmdCalcolaCRC32**.

Il codice si compone di un'unica routine:



```
1. Private Sub btnCalcolaCRC32_Click()
2.   Dim FileNR As Integer
3.   Dim Buffer As String
4.   Dim CRC32 As Long
5.   txtNomeFile.Text = Trim$(txtNomeFile.Text)
6.   If (Dir(txtNomeFile.Text, vbNormal) = "") Or (txtNomeFile.Text = "") Then
7.     MsgBox "Il file specificato non esiste!", vbCritical Or vbOKOnly
8.   Else
9.     If (GetAttr(txtNomeFile.Text) And vbDirectory) <> vbDirectory Then
```

```
10.      Me.MousePointer = vbHourglass
11.      FileNR = FreeFile
12.      Open txtNomeFile.Text For Binary Access Read As FileNR
```

Alla riga 6 viene verificata l'esistenza del file specificato nella casella **txtNomeFile**; vedi anche l'[HowTo dedicato alla verifica dell'esistenza di un file](#) ed alla riga 9 viene appurato che non si tratti di una cartella.

Con la certezza che il file esiste, alla riga 12 viene effettuata l'apertura in modalità binaria dello stesso. Vedi anche l'[HowTo dedicato alla lettura del contenuto di un file](#).

```
13.      CRC32 = CalcolaCRC("")
14.      Buffer = String$(2048, 0)
15.      Do
16.          Get FileNR, , Buffer
17.          If EOF(FileNR) Then Buffer = Left$(Buffer, 2048 - Loc(FileNR) + LOF
            (FileNR))
18.          CRC32 = CalcolaCRC(Buffer, CRC32, False)
19.      Loop Until EOF(FileNR)
```

La prima operazione da effettuare è l'inizializzazione del valore di CRC32, effettuata passando semplicemente alla funzione *CalcolaCRC* una stringa vuota (riga 13). Come specificato in precedenza, il contenuto del file verrà letto a blocchi di 2 KB ciascuno; a tale scopo, alla riga 14 viene preparata la variabile **Buffer**.

Segue un ciclo che si ripeterà fino a quando non verrà letto tutto il file, sempre a blocchi di 2 KB ciascuno. L'unico controllo necessario riguarda la fine del file (riga 17). L'ultimo blocco del file infatti non sarà grande 2 KB esatti ma la sua dimensione risulterà dal resto delle letture precedenti. Ad esempio durante la lettura di un file di 5 KB i primi due blocchi avranno dimensione normale 2 KB ma l'ultimo farà eccezione con 1 KB. Viene pertanto verificato se ci troviamo alla fine del file ed in tal caso il buffer viene ridimensionato.

Ogni pacchetto del file letto viene inviato alla funzione *CalcolaCRC*, fornendo anche il valore del CRC precedente. Il terzo parametro indicherà che il pacchetto inviato non è l'ultimo ma si richiedono ulteriori elaborazioni. Il valore di ritorno della funzione *CalcolaCRC* viene memorizzato nella variabile **CRC32** per riutilizzarlo nell'elaborazione del blocco successivo. Infatti **CRC32** è sia valore di ritorno che parametro inviato alla funzione.

```
20.      Close FileNR
21.      CRC32 = CalcolaCRC("", CRC32, True)
22.      MsgBox "Il CRC32 del file è " & Hex(CRC32), vbInformation Or vbOKOnly
23.      Me.MousePointer = vbDefault
24.  End If
25. End If
26. End Sub
```

Alla fine del ciclo di lettura ed elaborazione del CRC32 viene chiuso il file (riga 20) ed infine effettuata l'**ultima elaborazione** per richiedere il valore finale del CRC passato come parametro (riga 21).

Ottenuto l'ultimo dato esso sarà mostrato tramite una MessageBox (riga 22).



## Ulteriori applicazioni del CRC32


Si consideri sempre il CRC32 come un metodo di verifica dei dati e mai come sistema di protezione contro le modifiche dei dati. L'algoritmo infatti è standard nell'elaborazione; quello che può cambiare è il suo campo di applicazione oppure il valore di seme utilizzato.

Possiamo infatti memorizzare nei nostri programmi (e dove lo vedremo subito dopo) il CRC32 di ogni singolo file, oppure valore di singoli segmenti del file in analisi. Quello che si rivela utile controllare sono i dati critici alla corretta elaborazione di un processo.

Potrebbe essere utile verificare ad esempio che un certo blocco di dati non sia stato modificato poiché in tal caso il programma fornirebbe dati importanti ma fondamentalmente errati, ad esempio in un file di dati di un bilancio aziendale.

Resta da sciogliere un ultimo punto: dove e come memorizzare i valori di CRC32 senza che questi vadano a modificare il contenuto del file la cui elaborazione produrrebbe un CRC32 differente?

Le soluzioni più semplici consistono in un file di dati di appoggio. Tutti i CRC32 possono essere scritti al suo interno ed il programma che ne richiede la verifica si occuperà di estrarli e confrontarli con il valore CRC32 reale ottenuto dall'elaborazione.

Un'altra soluzione potrebbe essere quella di scriverli come [file di risorse](#)  all'interno di un file di [libreria \(DLL\)](#). Un'ultima soluzione, un po' drastica a dire il vero, consiste nell'aggiungere i dati alla fine del file eseguibile stesso, avendo cura che l'elaborazione del CRC32 di questo escluda quei 4 bytes contenenti il CRC32 stesso.

In ragione della struttura dell'implementazione adottata l'algoritmo di calcolo potrà essere utilizzato per calcolare il valore CRC32 non solo dell'intero file, ma anche per elaborazioni parziali di dati al fine di velocizzare l'esecuzione di calcolo.

Qualunque sia la soluzione applicata è fondamentale tenere a mente che il CRC32 non può essere considerato un algoritmo di protezione. Se qualcuno vuole utilizzarlo come tale abbia almeno la minima cura di criptare i dati rappresentativi del valore di CRC.

Si tenga anche conto che il presente codice è sviluppato in puro Visual Basic e come tale è soggetto ad una pesante lentezza, in parte legata alla soluzione del codice ed in parte legata ai limiti del linguaggio stesso.

[Fibia FBI](#)

26 Gennaio 2002

Un ringraziamento particolare a [Detonate](#)



[Torna all'introduzione delle Richieste dei lettori](#)

---