Un file di bitmap indipendente è composto da un *TBitmapFileHeader* seguito da un record *TBitmapInfo* e dai dati attuali della bitmap.

TBitmapFileHeader

```
TBitmapFileHeader = record
  bfType: Word;
  bfSize: Longint;
  bfReserved1: Word;
  bfReserved2: Word;
  bfOffBits: Longint;
end;
```

Definisce l'intestazione di un file di bitmap indipendente dal dispositivo che contiene i dati che definiscono il tipo, la dimensione e la disposizione del file di bitmap.

bfType
Indica il tipo del file, deve essere di tipo BM.

bfSize
Indica la dimensione del file in blocchi di 4 byte.

bfReserved1 e bfReserved2
Come indicato dal nome sono riservati da Windows.

bfOffBits
Indica l'inizio delle informazioni attuali della bitmap nel file.

TBitmapInfo

```
TBitmapInfo = record
  bmiHeader: TBitmapInfoHeader;
  bmiColors: array[0..0] of TRGBQuad;
end;
```

Contiene le informazioni della dimensione e dei colori per una bitmap indipendente dal dispositivo per Windows 3.0.
L'attuale bitmap è definita come un array di byte che rappresentano i pixel della bitmap.

bmiHeader
Contiene il record *TBitmapInfoHeader* che definisce la dimensione e la formattazione dei colori per la bitmap.

bmiColors
Array di record *TRGBQuad* che definiscono i colori della bitmap.
Il numero di ingressi nell'array è determinato dal valore del campo *biBitCount* del record *bmiHeader*.

TRGBQuad

```
TRGBQuad = record
  rgbBlue: Byte;
  rgbGreen: Byte;
  rgbRed: Byte;
  rgbReserved: Byte;
end;
```

Mantiene i dati di colore RGB per le bitmap, come nel campo *bmiColors* del record *TBitmapInfo*.

rgbBlue, rgbGreen e rgbRed
Rappresentano rispettivamente le intensità del blu, del verde e del rosso nell'ingresso del pixel della bitmap.

rgbReserved
Non usato: deve essere 0.

TBitmapInfoHeader

```
TBitmapInfoHeader = record
  biSize: Longint;
  biWidth: Longint;
  biHeight: Longint;
  biPlanes: Word;
  biBitCount: Word;
  biCompression: Longint;
  biSizeImage: Longint;
  biXPelsPerMeter: Longint;
  biYPelsPerMeter: Longint;
  biClrUsed: Longint;
  biClrImportant: Longint;
end;
```

I record *TBitmapInfoHeader* vengono utilizzati dai record *TBitmapInfo* per definire le dimensioni e la formattazione dei colori di un bitmap indipendente dal dispositivo per Windows 3.0.

biSize
Indica la dimensione in byte del record.

biWidth e biHeight
Indicano rispettivamente la larghezza e l'altezza in pixel della bitmap.

biPlanes
Indica il numero di piani per il dispositivo di destinazione, deve essere impostato a uno.

biBitCount
Indica il numero di bit richiesti per descrivere ogni pixel nella bitmap.
Se *biBitCount* è 1, la bitmap è monocromatica, la tabella dei colori deve contenere due ingressi e ogni bit nella bitmap rappresenta un pixel; un bit a zero rappresenta il primo colore nella tabella, un bit a uno rappresenta il secondo colore.
Se *biBitCount* è 4, la bitmap ha fino a 16 colori numerati da 0 a 15, ogni pixel nella bitmap richiede quattro bit per indicarne il colore; la tabella dei colori contiene 16 ingressi, ogni byte nella bitmap rappresenta due pixel, il primo nel mezzo byte superiore e il secondo nel mezzo byte inferiore.
Se *biBitCount* è 8, la bitmap ha fino a 256 colori, ogni byte rappresenta un pixel, quindi ogni byte nella bitmap rappresenta un indice tra 0 e 255 nella tabella dei colori.
Se *biBitCount* è 24, la bitmap ha fino a 2^{24} colori, la tabella dei colori non esiste e ogni pixel è rappresentato da una terna di byte che indicano l'intensità del rosso, del verde e del blu nel pixel.

biCompression
Indica il tipo di compressione utilizzato per la bitmap; può essere una delle costanti *bi_*:

bi_RGB	La bitmap non è compressa.
bi_RLE8	La bitmap utilizza il formato codificato in lunghezza con 8 bit per pixel.
bi_RLE4	La bitmap utilizza il formato codificato in lunghezza con 4 bit per pixel.

biSizeImage
Indica la dimensione in byte dell'immagine della bitmap.

biXPelsPerMeter e biYPelsPerMeter
Indicano rispettivamente la risoluzione orizzontale e verticale del dispositivo di destinazione per la bitmap.

biClrUsed
Specifica il numero di ingressi nella tabella dei colori attualmente utilizzati dalla bitmap.
Il valore di *biBitCount* determina il massimo numero di ingressi; 0 in *biClrUsed* indica che viene utilizzato in numero massimo.
Un valore tra 1 e 23 per *biClrUsed* indica l'attuale numero di colori richiamati.
Se *biBitCount* è 24, *biClrUsed* è la dimensione della tabella dei colori di riferimento utilizzata da Windows per ottimizzare le prestazioni della tavolozza dei colori.

biClrImportant
Indica il numero di colori importanti per la visualizzazione della bitmap; il valore zero indica che tutti i colori sono importanti.

Poiche' i dati della bitmap sono allineati a 4 byte, ogni tanto e' possibile incontrare qualche bit in piu'. Questi bit "strani" possono assumere qualsiasi valore, cosi' spesso possiamo fare finta che non ci siano ed elaborare anche loro. Ovviamente questo e' possibile per operatori puntuali come il threshold, ma non locali come la deconvoluzione. Per sapere esattamente quanti byte codificano una riga dell'immagine basta usare la formula:

```
  RowBytes := ((biWidth * biBitCount + 31) Shr 5) Shl 2;
```

Cosi' se biWidth = 33 e biBitCount = 1 le righe vengono codificate con 8 byte (sprecando gli ultimi 31 bit "strani"), anziche' con esattamente 33 bit.

A causa di una errata descrizione (ufficiale!) delle precedenti versioni del formato BMP, non tutti i campi dell'header sono sempre riempiti correttamente. Quando si carica una immagine e' meglio usare meno informazioni possibili... Per prima cosa bisogna calcolare esattamente la lunghezza del file e memorizzarla in bfSize. Poi bisogna accertarsi che il valore di bfOffBits sia compreso tra

```
  bfSize - RowBytes * biHeight
```

e

```
  SizeOf(TBitmapFileHeader) + SizeOf(TBitmapInfoHeader) + SizeOf(TRGBQuad) * (1 Shl biBitCount)
```

altrimenti il file e' certamente corrotto. Infine si deve definire

```
  biSizeImage := RowBytes * biHeight;
```

In fase di scrittura e' meglio cancellare tutti i chunk che eventualmente precedono e seguono i dati della bitmap, definire

```
  bfOffBits := bfSize - RowBytes * biHeight;
```

e aggiornare bfSize. Questa operazione e' necessaria perche' durante il caricamento alcune applicazioni (come ph*t*sh*p) assumono erroneamente che l'immagine sia codificata subito dopo la palette, prescindendo dal valore di bfOffBits.