Esercitazione: elaborare immagini bitmap

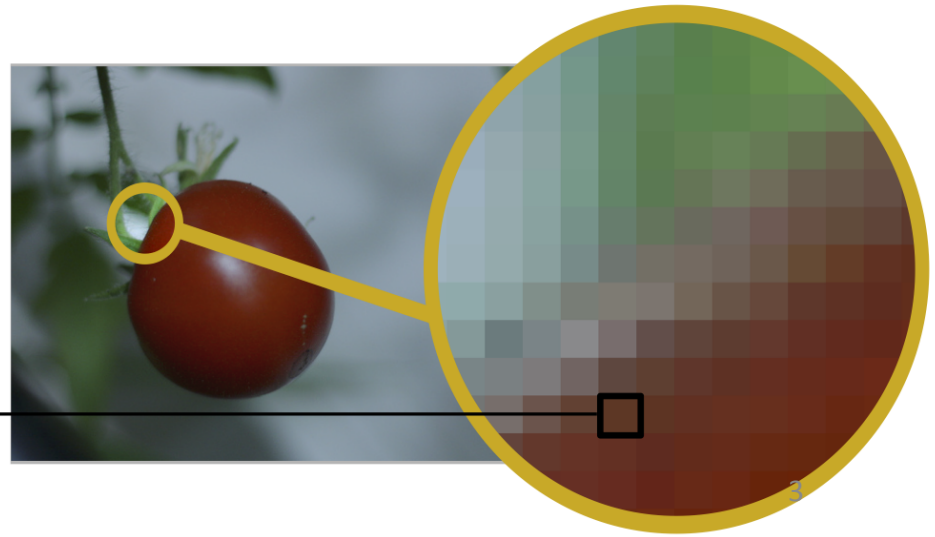
Pierpaolo Loreti

Sommario

- Immagini digitali e pixel
- Formato bitmap
- Lettura e scrittura di bmp
- Rappresentazione del colore
- Modificare i colori
- Convoluzione
- Applicare dei filtri
- …

Immagini Digitali

- Rappresentazione numerica di una foto
  - tipicamente binaria
- La tipica rappresentazione si chiama raster
  - a griglia
  - l'immagine è rappresentata da un insieme finito di elementi elementari: i pixel
    - abbreviazione di Picture ELement
    - i pixel sono gli elementi più piccoli in una immagine
  - ogni pixel rappresenta la "luminosità" di un determinato colore in un punto specifico



Ogni quadratino è un pixel

Definizione e risoluzione

- **Definizione**
  - numero di pixel che costituiscono l'immagine,
  - dimensione informatica
    - numero di colonne x numero di linee
    - Es. 640x480

- **Risoluzione**
  - numero di punti per unità di superficie
  - rapporto tra il numero di pixel e la dimensione reale della sua rappresentazione su un supporto fisico

  - DPI (Dots Per Inch) o PPI (Pixel Per Inch)
    - Inch: il pollice vale 2.54 cm
    - 300 dpi: 300x300 pixel per pollice quadrato
      - 90000 pixel per pollice quadrato
  - Point: 0.353mm
    - deriva dalla risoluzione 72dpi
    - un pixel vale 1/72 di inch

File per immagini

- File con un formato standard idoneo a contenere immagini
  - contengono le informazione per poter visualizzare o stampare l'immagine
  - i dati possono essere non compressi, compressi o vettoriali

- Raster
  - JPEG, PNG, TIFF, GIF, BMP, PSD
- Vettoriali
  - CGM, SVG, AI

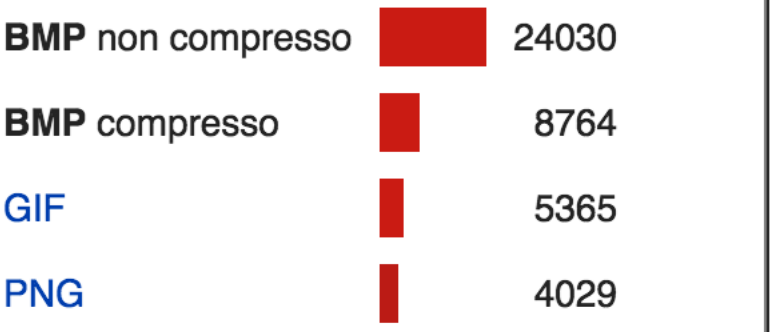
- Formato e compressione
  - BMP: senza compressione (ma anche con)
  - GIF, PNG: formati compressi senza perdita
  - JPEG: compresso con perdita

Peso di un immagine

- La dimensione del file dipende dal numero di pixel e dal come si rappresenta il colore (bit per pixel)
  - profondità del colore
- La dimensione del file dipende anche dalla eventuale compressione e dall'algoritmo di compressione utilizzato

- Senza compressione: 640 * 480 con colori a 24 bit
  - 640 * 480 * 24 = 7,372,800 bits = 921,600 bytes = 900 KiB

Dimensioni in byte della stessa immagine (128 colori) in diversi formati raster



Formato Bitmap

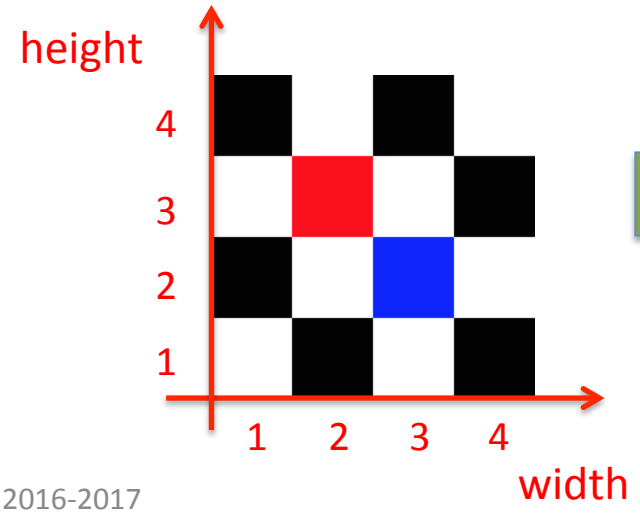
- Formato **Raster** per memorizzazione indipendente dal device
  - fatto dalla **Microsoft**
    - introdotto nel 1990 con windows 3.0
  - pensato per applicazioni in Windows ed OS2
  - noto anche come **device independent bitmap** (DIB)
- Ne esistono tante versioni
  - la più semplice è la V3
  - la V4 e V5 fatte per win 98
    - aggiungono il canale alfa ed altro
    - poco utilizzate
- Supporta immagini
  - dimensione e risoluzione arbitraria
  - monocromatiche o a colori
  - varie profondità di colore
    - 1, 4, 8, 16, 24 o 32 bit per pixel
  - possibile compressione (V4 V5)

File bmp

- Header base
  - **BITMAPFILEHEADER**
  - 14 byte
  - tipo immagine, dimensione, offset
- Info header
  - **BITMAPINFOHEADER**
  - 40 byte
  - dimensione immagine, righe, colonne, ppp
- Palette
- Image Data
  - Array dei pixel

Image data

- La sezione image data contiene la sequenza dei dati relativi ai pixel
- L'immagine è una matrice WxH di numeri
  - W width, H height
- I numeri rappresentano i colori dei pixel



13	14	15	16
9	10	11	12
5	6	7	8
1	2	3	4

ID (W,H)
1 (1,1)
2 (1,2)
3 (1,3)
4 (1,4)
5 (2,1)
6 (2,2)
7 (2,3)
8 (2,4)
9 (3,1)
10 (3,2)
11 (3,3)
12 (3,4)
13 (4,1)
14 (4,2)
15 (4,3)
16 (4,4)

Colore e padding

- Ogni pixel è rappresentato da alcuni byte
  - es. 24 bit (3 byte): FF FF FF
  - nell'header c'è il campo Bits per pixel che specifica quanti byte per pixel sono usati
- Ogni riga dell'array dati deve essere multiplo di 4 byte
  - Se non lo è si fa padding
  - Esempio
    - immagine 8x8 - riga: 3*8=24
    - immagine 10x10 – riga 3*10=30
      - riga da 32 byte padding 2 byte

13	14	15	16	
9	10	11	12	
5	6	7	8	
1	2	3	4	

Leggere una bitmap

```
 9  int main() {
10
11      FILE *fin = fopen("img1.bmp", "rb");
12      int i=0, j=0;
13      unsigned char header[54];
14      unsigned char r,g,b;
15      int width = 4, height = 4;
16      int size;
17
18      fread(header, sizeof(unsigned char), 54, fin);
19
20      for (i=0; i<height; i++ ) {
21          for (int j=0; j<width; j++)
22          {
23              fread(&b,sizeof(b),1,fin);
24              fread(&g,sizeof(g),1,fin);
25              fread(&r,sizeof(r),1,fin);
26
27              //...
28
29          }
30      }
31      fclose(fin);
32      return 0;
33
34  }
```


Leggere una bitmap

```
 9  int main() {
10
11      FILE *fin = fopen("img1.bmp", "rb");
12      int i=0, j=0;
13      unsigned char header[54];
14      unsigned char r,g,b;
15      int width = 4, height = 4;
16      int size;
17
18      fread(header, sizeof(unsigned char), 54, fin);
19
20      for (i=0; i<height; i++ ) {
21          for (int j=0; j<width; j++)
22          {
23              fread(&b,sizeof(b),1,fin);
24              fread(&g,sizeof(g),1,fin);
25              fread(&r,sizeof(r),1,fin);
26
27              //...
28
29          }
30      }
31      fclose(fin);
32      return 0;
33
34  }
```

Leggo i 54 byte dell'header

Leggere una bitmap

```
 9  int main() {
10
11      FILE *fin = fopen("img1.bmp", "rb");
12      int i=0, j=0;
13      unsigned char header[54];
14      unsigned char r,g,b;
15      int width = 4, height = 4;
16      int size;
17
18      fread(header, sizeof(unsigned char), 54, fin);
19
20      for (i=0; i<height; i++ ) {
21          for (int j=0; j<width; j++)
22          {
23              fread(&b,sizeof(b),1,fin);
24              fread(&g,sizeof(g),1,fin);
25              fread(&r,sizeof(r),1,fin);
26
27              //...
28
29          }
30      }
31      fclose(fin);
32      return 0;
33
34  }
```

Loop sulle righe

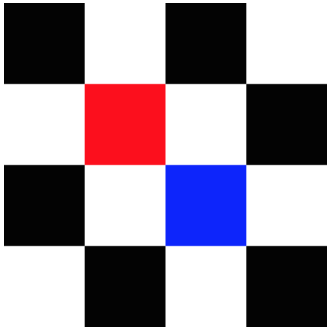
Loop sulle colonne

Leggo i valori di colore del pixel i,j

Esercizio

- Leggere e stampare le 64 triple che rappresentano i colori di ogni bit dell'immagine img1.bmp
  - leggere header 54 byte
  - leggere array immagine 4x4
  - ogni pixel è rappresentato con 3 byte

- Immagine
  - http://ppl.eln.uniroma2.it/finf/img1.bmp



Profondità del colore

- Bianco e nero:
  - immagazzinando un bit per ogni casella, è possibile definire due colori (nero o bianco).
- 16 colori o 16 livelli di grigio:
  - immagazzinando 4 bit in ogni casella, è possibile definire 24 possibilità di intensità per ogni pixel, cioè 16 sfumature di grigio che vanno dal nero al bianco oppure 16 colori diversi.
- Bitmap 256 colori o 256 livelli di grigio:
  - immagazzinando un byte in ogni casella, è possibile definire 28 intensità di pixel, cioè 256 sfumature di grigio che vanno dal nero al bianco oppure 256 colori diversi.
- Paletta dei colori
  - si definisce una tavolozza dei colori che l'immagine contiene ad ognuno dei quali è associato un indice, ad ogni pixel si associa l'indice
    - Con indici a 8 bit è possibile definire 256 colori utilizzabili
- True color
  - questa rappresentazione permette di rappresentare un'immagine definendo i colori componenti (rosso, verde e blu) per ogni pixel.

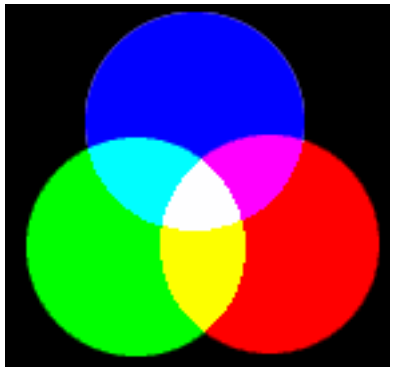
Rappresentazione del colore

- Per poter scambiare informazioni colorimetriche è necessario fornire delle rappresentazioni matematiche
  - associo un numero ad un colore
  - normalmente una tupla di numeri

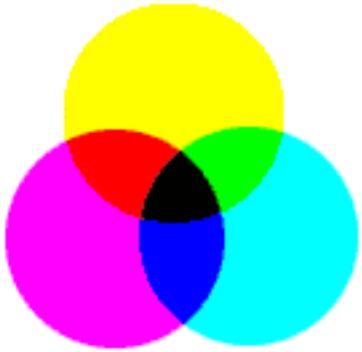
- Le rappresentazioni più note sono
  - RGB (Red, Green, Blue)
  - HSL (Hue, Saturation, Luminance)
  - CMYK (Cyan, Magenta, Yellow and blacK).
  - YUV

Sintesi di colori

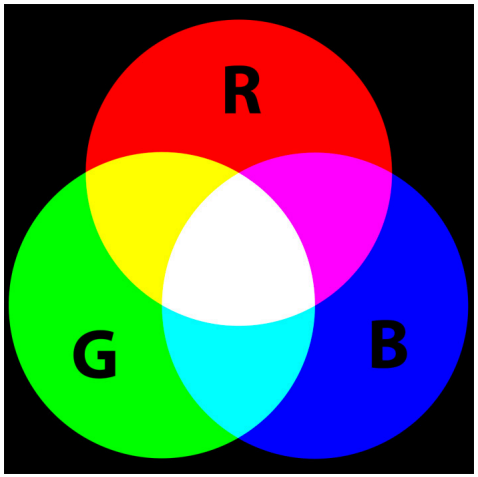
- Sintesi additiva
  - aggiunga delle componenti della luce
    - Le componenti della luce sono direttamente aggiunte all'emissione, è il caso dei monitor o delle televisioni a colori
  - Se aggiungono le tre componenti Rosso, verde, blu (RVB), si ottiene il bianco
  - L'assenza delle componenti da il nero.
  - I colori secondari sono il ciano, il magenta ed il giallo dato che



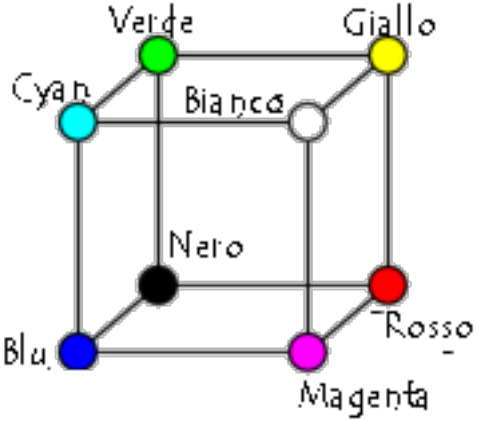
- Sintesi sottrattiva
  - Le componenti sono assorbite dalla materia
  - Questo processo è usato in fotografia e per la stampa dei colori
  - I colori secondari sono il blu, il rosso e il verde

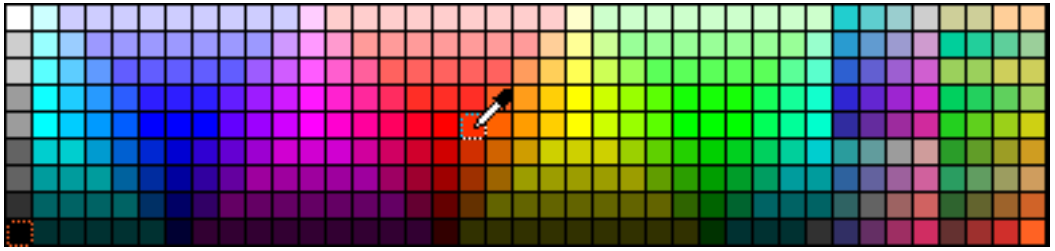


Codifica RGB

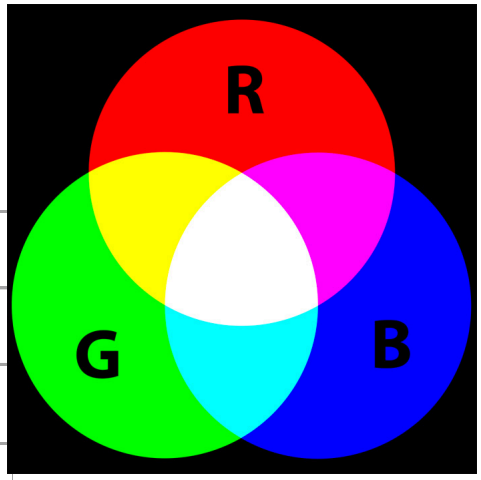


- Codifica Red, Green, Blue
  - elaborato nel 1931 dalla Commissione Internazionale dell'Illuminazione
  - basata sul funzionamento dell'occhio umano che ha recettori distinti per i colori
- Le tre componenti di colore si sommano
  - sintesi additiva
- Codifica su un byte ogni componente di colore
  - 256 per ogni componente di colore
  - 16.777.216 combinazioni possibili
    - l'occhio umano ne distingue circa 2 milioni.





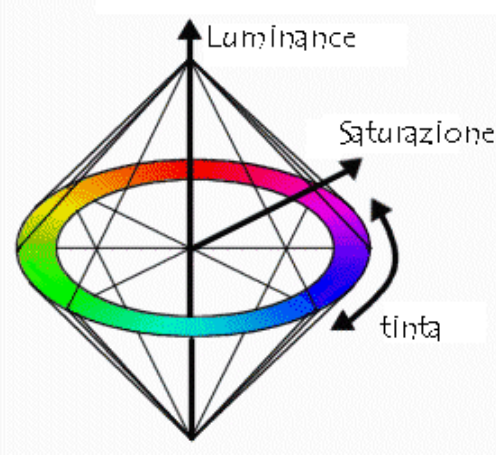
Colori comuni

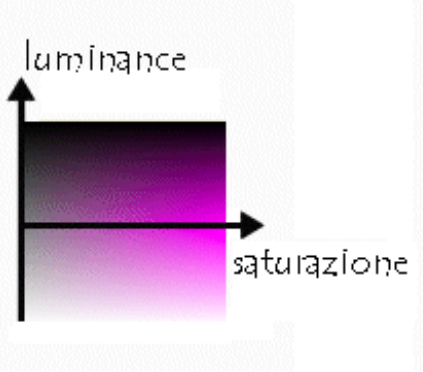


R	G	B	Hex Value	Color
0	0	0	000000	Black
255	0	0	FF0000	Red
0	255	0	00FF00	Green
0	0	255	0000FF	Blue
255	255	0	FFFF00	Yellow
255	0	255	FF00FF	Magenta
0	255	255	00FFFF	Cyan
255	128	128	FF8080	Bright Red
128	255	128	80FF80	Bright Green
128	128	255	8080FF	Bright Blue
64	64	64	404040	Dark Grey
128	128	128	808080	Intermediate Grey
192	192	192	C0C0C0	Bright Grey
255	255	255	FFFFFF	White

Rappresentazione HLS

- **Tonalità** (in inglese Hue)
  - descrive la sensazione del colore, in altre parole se il colore è rosso, giallo, verde, ciano, blu, magenta ...
  - come per lo spettro di luce visibile
  - valore da 0 a 360 gradi
- **Saturazione**
  - grado in cui la tonalità differisce da un grigio neutro
  - 0%, grigio, al 100% colore intenso
  - Quanto più lo spettro della luce è concentrata intorno una lunghezza d'onda, più saturo il colore sarà.
- **Luminosità**
  - illuminazione del colore
  - allo 0% il colore è completamente nero, al 50% il colore è puro, al 100% diventa bianco





Operazioni sui colori

- Negative
  - 255-C
  - Ritorna il colore opposto. Es. bianco diventa nero, rosso diventa ciano
- Darken
  - C/p or C-p
  - Dividere il colore per una costante (maggiore di 1), o sottrarre una costante, per renderlo più scuro.
- Brighten
  - C*p or C+p
  - Moltiplicare il colore per una costante (maggiore di 1), o aggiungere una costante ad esso, per renderlo più luminoso.
- Greyscale
  - (R+G+B)/3
  - Calcolare la media dei 3 canali per ottenere un colore grigio con la stessa luminosità.
- Remove Channel
  - R=0, G=0 e B=0
  - Impostando uno o più canali a 0, rimuovere completamente quel componente colore dalla foto.
- Swap Channels
  - R=G, G=R, ...
  - Scambiare i valori di due canali di colore per ottenere un'immagine con un colore completamente diverso.

Modifiche ai colori

Originale

C/2

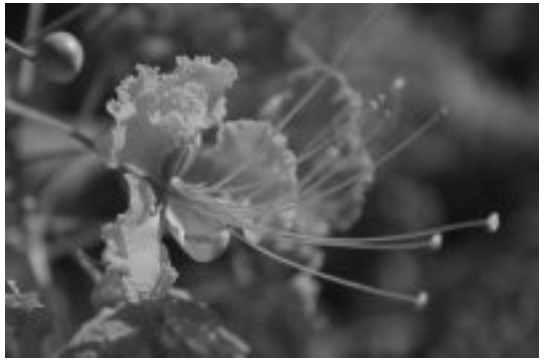
C*2

C-50

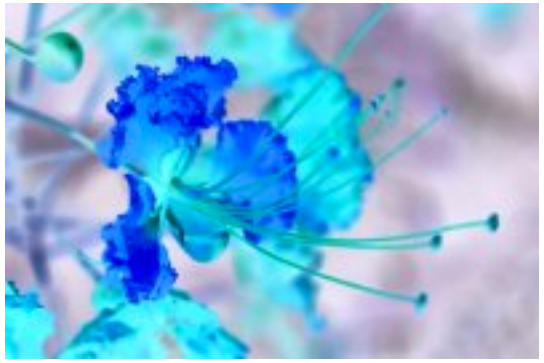
C+50

 Nota: il minimo è 0 ed il massimo 255 – alcuni colori saturano

Modifiche ai colori



Scala di Grigio



Negativo

B=0

G=0

R=0

Copiare un immagine

```
 8  int main() {
 9      int i=0, j=0;
10      unsigned char header[54];
11      unsigned char r,g,b;
12      int width = 200, height = 132;
13      int size;
14
15      FILE *fin = fopen("img2.bmp", "rb");
16      FILE *fout = fopen("img2-mod.bmp", "wb");
17
18      fread(header, sizeof(unsigned char), 54, fin);
19
20      fwrite(header, 54, 1, fout); //Copia header
21
22      for (i=0; i<height; i++ ) {
23          for (int j=0; j<width; j++)
24          {
25              fread(&b,sizeof(b),1,fin);
26              fread(&g,sizeof(g),1,fin);
27              fread(&r,sizeof(r),1,fin);
28              // Scrittura
29              fwrite(&b,1,1,fout);
30              fwrite(&g,1,1,fout);
31              fwrite(&r,1,1,fout);
32          }
33      }
34      fclose(fin);
35      fclose(fout);
36      return 0;
37  }
```


Copiare un immagine

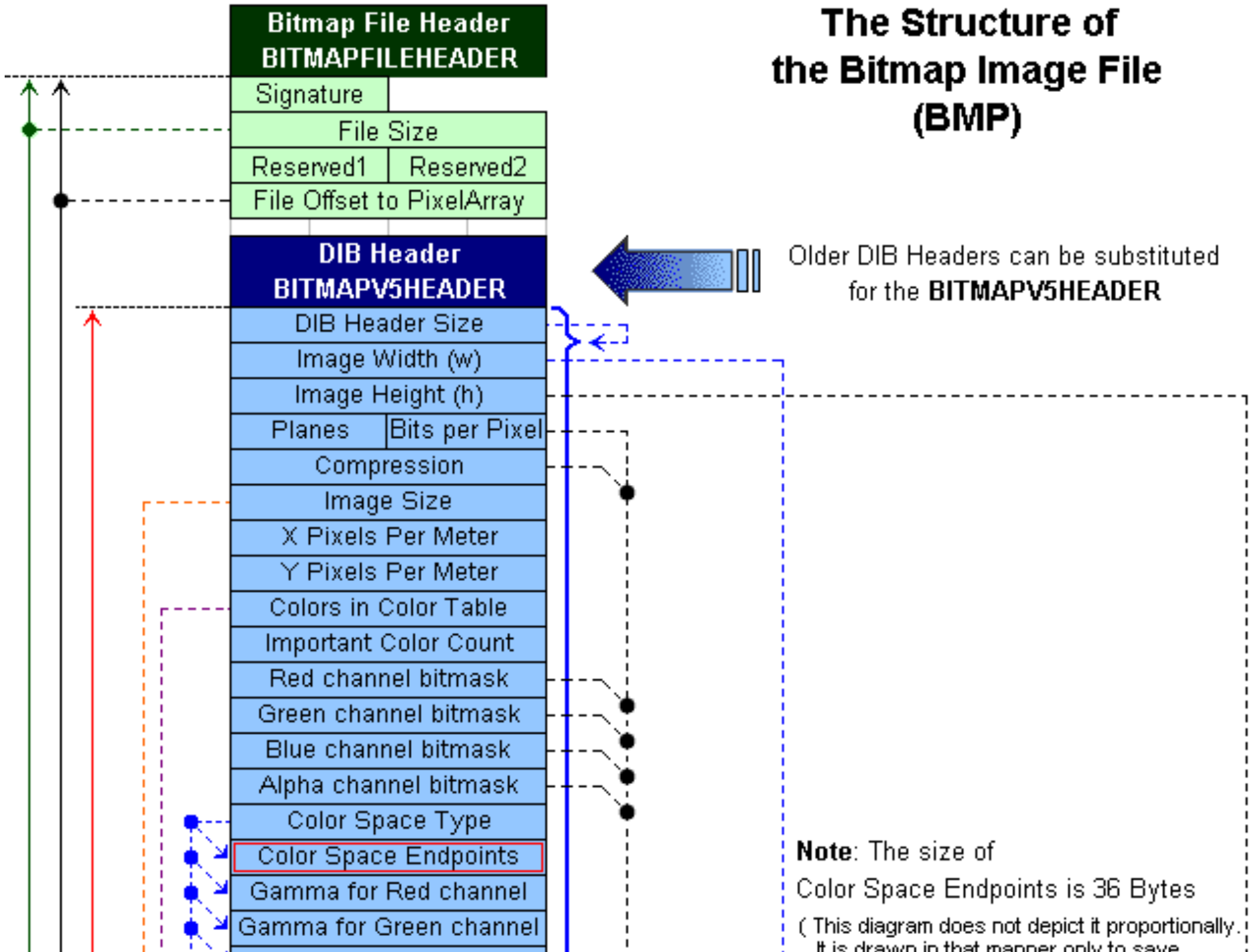
```
 8  int main() {
 9      int i=0, j=0;
10      unsigned char header[54];
11      unsigned char r,g,b;
12      int width = 200, height = 132;
13      int size;
14
15      FILE *fin = fopen("img2.bmp", "rb");
16      FILE *fout = fopen("img2-mod.bmp", "wb");
17
18      fread(header, sizeof(unsigned char), 54, fin);
19
20      fwrite(header, 54, 1, fout); //Copia header
21
22      for (i=0; i<height; i++ ) {
23          for (int j=0; j<width; j++)
24          {
25              fread(&b,sizeof(b),1,fin);
26              fread(&g,sizeof(g),1,fin);
27              fread(&r,sizeof(r),1,fin);
28              // Scrittura
29              fwrite(&b,1,1,fout);
30              fwrite(&g,1,1,fout);
31              fwrite(&r,1,1,fout);
32          }
33      }
34      fclose(fin);
35      fclose(fout);
36      return 0;
37  }
```

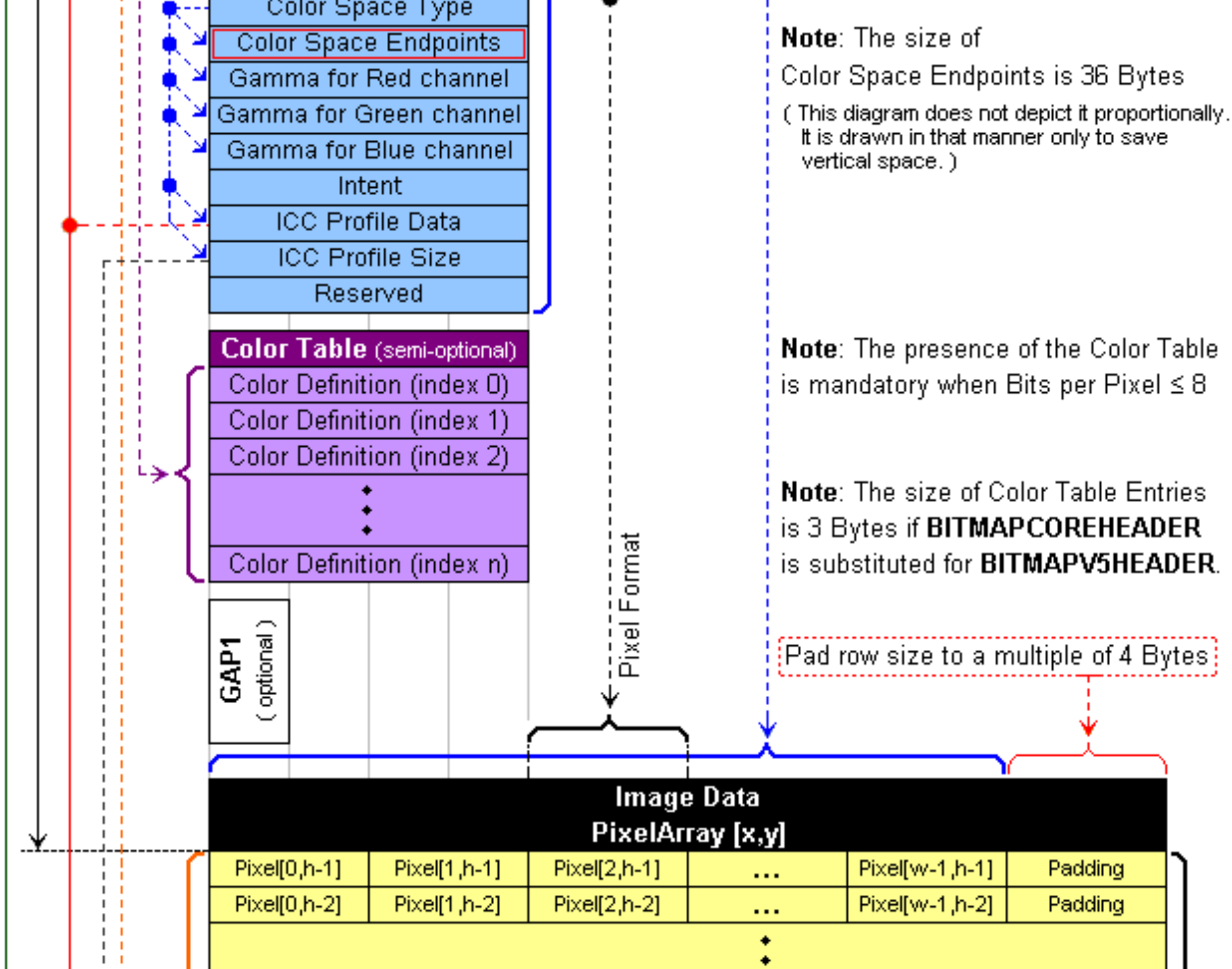
Esercizio

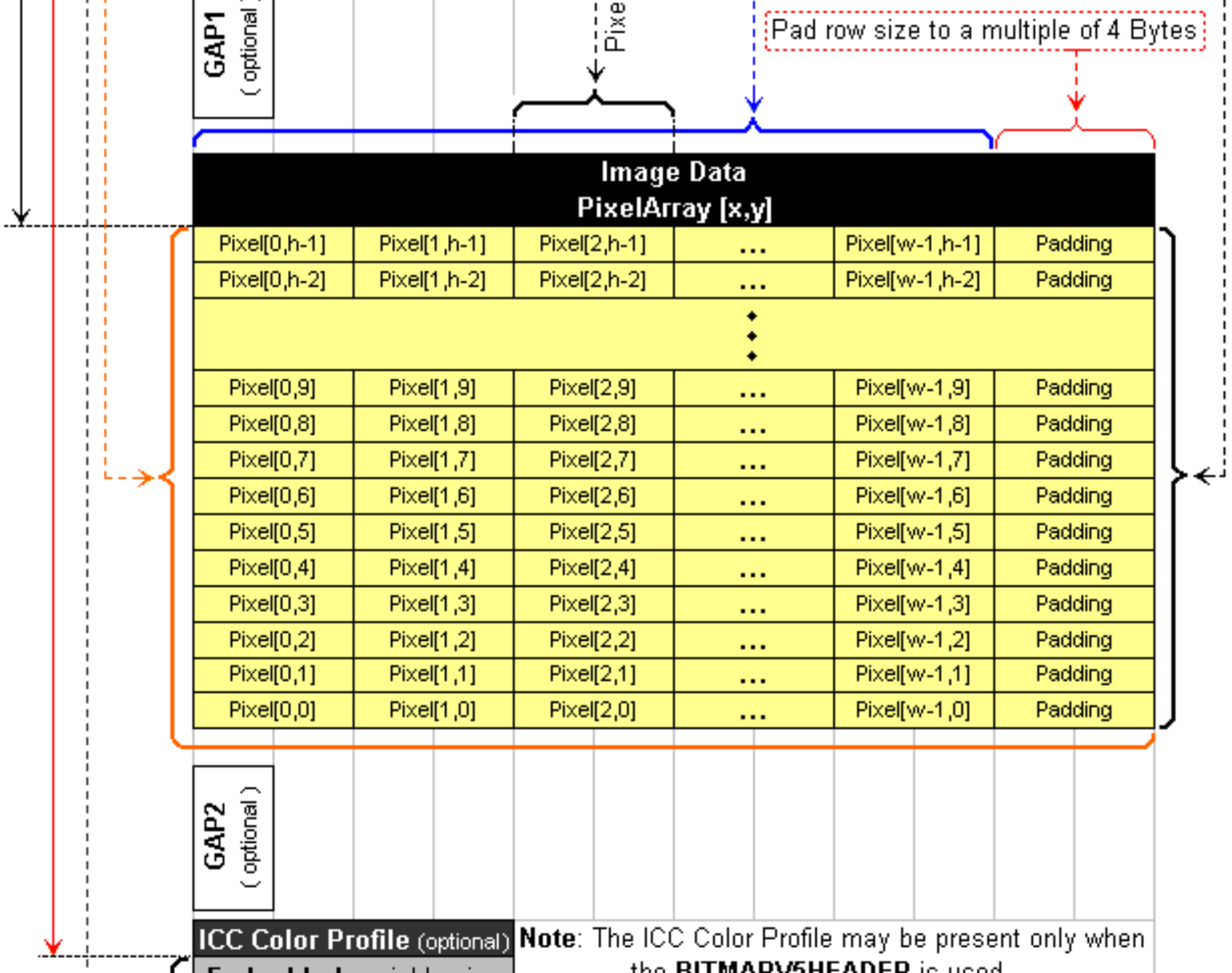
- Leggere l'immagine img2.bmp (200x132) e salvare le seguenti immagini modificate
  - Scala di grigio
  - Immagine negativa
  - Immagine senza in canale rosso

- Immagine
  - http://ppl.eln.uniroma2.it/finf/img2.bmp

The Structure of the Bitmap Image File (BMP)



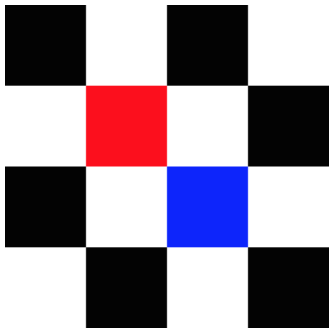

Color Space Type
Color Space Endpoints
Gamma for Red channel
Gamma for Green channel
Gamma for Blue channel
Intent
ICC Profile Data
ICC Profile Size
Reserved
Color Table (semi-optional)
Color Definition (index 0)
Color Definition (index 1)
Color Definition (index 2)
Color Definition (index n)
GAP1 (optional)
Pixel Format
Note: The size of Color Space Endpoints is 36 Bytes
(This diagram does not depict it proportionally. It is drawn in that manner only to save vertical space.)
Note: The presence of the Color Table is mandatory when Bits per Pixel ≤ 8
Note: The size of Color Table Entries is 3 Bytes if BITMAPCOREHEADER is substituted for BITMAPV5HEADER.
Pad row size to a multiple of 4 Bytes
Image Data
PixelArray [x,y]
Pixel[0,h-1]
Pixel[1,h-1]
Pixel[2,h-1]
...
Pixel[w-1,h-1]
Padding
Pixel[0,h-2]
Pixel[1,h-2]
Pixel[2,h-2]
...
Pixel[w-1,h-2]
Padding


GAP1
(optional)
Pixe
Pad row size to a multiple of 4 Bytes
Image Data
PixelArray [x,y]
Pixel[0,h-1] Pixel[1,h-1] Pixel[2,h-1] ... Pixel[w-1,h-1] Padding
Pixel[0,h-2] Pixel[1,h-2] Pixel[2,h-2] ... Pixel[w-1,h-2] Padding
Pixel[0,9] Pixel[1,9] Pixel[2,9] ... Pixel[w-1,9] Padding
Pixel[0,8] Pixel[1,8] Pixel[2,8] ... Pixel[w-1,8] Padding
Pixel[0,7] Pixel[1,7] Pixel[2,7] ... Pixel[w-1,7] Padding
Pixel[0,6] Pixel[1,6] Pixel[2,6] ... Pixel[w-1,6] Padding
Pixel[0,5] Pixel[1,5] Pixel[2,5] ... Pixel[w-1,5] Padding
Pixel[0,4] Pixel[1,4] Pixel[2,4] ... Pixel[w-1,4] Padding
Pixel[0,3] Pixel[1,3] Pixel[2,3] ... Pixel[w-1,3] Padding
Pixel[0,2] Pixel[1,2] Pixel[2,2] ... Pixel[w-1,2] Padding
Pixel[0,1] Pixel[1,1] Pixel[2,1] ... Pixel[w-1,1] Padding
Pixel[0,0] Pixel[1,0] Pixel[2,0] ... Pixel[w-1,0] Padding
GAP2
(optional)
ICC Color Profile (optional)
Note: The ICC Color Profile may be present only when

File bmp: img1.bmp

img1.bmp

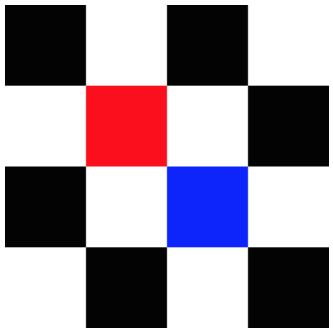
```
 0 424D6600 00000000 00003600 00002800 00000400 00000400 00000100
28 18000000 00003000 0000120B 0000120B 00000000 00000000 0000FFFF
56 FF000000 FFFFFF00 00000000 00FFFFFF FF0000FF FFFFFFFF FF0000FF
84 FFFFFF00 00000000 00FFFFFF 000000FF FFFF
```



File bmp: img1.bmp

img1.bmp

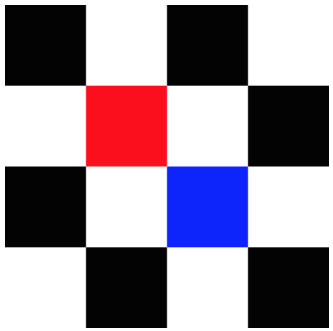
```
 0 424D6600 00000000 00003600 00002800 00000400 00000400 00000100
28 18000000 00003000 0000120B 0000120B 00000000 00000000 0000FFFF
56 FF000000 FFFFFF00 00000000 00FFFFFF FF0000FF FFFFFFFF FF0000FF
84 FFFFFF00 00000000 00FFFFFF 000000FF FFFF
```



File bmp: img1.bmp

img1.bmp

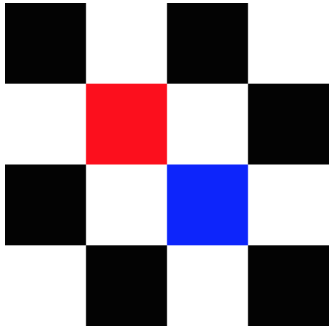
```
 0 424D6600 00000000 00003600 00002800 00000400 00000400 00000100
28 18000000 00003000 0000120B 0000120B 00000000 00000000 0000FFFF
56 FF000000 FFFFFF00 00000000 00FFFFFF FF0000FF FFFFFFFF FF0000FF
84 FFFFFF00 00000000 00FFFFFF 000000FF FFFF
```



File bmp: img1.bmp

img1.bmp

```
 0 424D6600 00000000 00003600 00002800 00000400 00000400 00000100
28 18000000 00003000 0000120B 0000120B 00000000 00000000 0000FFFF
56 FF000000 FFFFFF00 00000000 00FFFFFF FF0000FF FFFFFFFF FF0000FF
84 FFFFFF00 00000000 00FFFFFF 000000FF FFFF
```



Esempio di lettura header

```
FILE* fin = fopen("img3.bmp","rb");
unsigned int width,height;
unsigned char header[54];
fread(header,sizeof(unsigned char),54,fin);
memcpy(&width,&header[18],4);
memcpy(&height,&header[22],4);

fseek(fin,18,SEEK_SET);
fread(&width,sizeof(unsigned char),4,fin);
fseek(fin,22,SEEK_SET);
fread(&height,sizeof(unsigned char),4,fin);
```

https://en.wikipedia.org/wiki/BMP_file_format

Esercizio

- Leggere l'header e i dati delle seguenti immagini

- Immagine
  - http://ppl.eln.uniroma2.it/finf/img4.bmp
  - http://ppl.eln.uniroma2.it/finf/img5.bmp
    - Necessita padding

Convoluzione

Immagine

0	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0
0	1	2	2	1	1	0	0
0	1	2	2	3	2	2	1
0	1	2	3	2	2	1	1
0	1	2	2	1	2	1	0
0	1	1	1	1	2	1	0
0	0	0	1	1	1	0	0

Kernel

4	0	0
0	0	0
0	0	-4

Immagine risultante

Convoluzione

Immagine

0	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0
0	1	2	2	1	1	0	0
0	1	2	2	3	2	2	1
0	1	2	3	2	2	1	1
0	1	2	2	1	2	1	0
0	1	1	1	1	2	1	0
0	0	0	1	1	1	0	0

Kernel

4	0	0
0	0	0
0	0	-4

Immagine risultante

	-8						

4*0+0*1+0*1+
0*0+0*1+0*1+
0*0+0*1+(-4*2)
=-8

Convoluzione

Immagine

0	1	1	0	0	0	0	0
0	1	1	1	1	0	0	0
0	1	2	2	1	1	0	0
0	1	2	2	3	2	2	1
0	1	2	3	2	2	1	1
0	1	2	2	1	2	1	0
0	1	1	1	1	2	1	0
0	0	0	1	1	1	0	0

Kernel

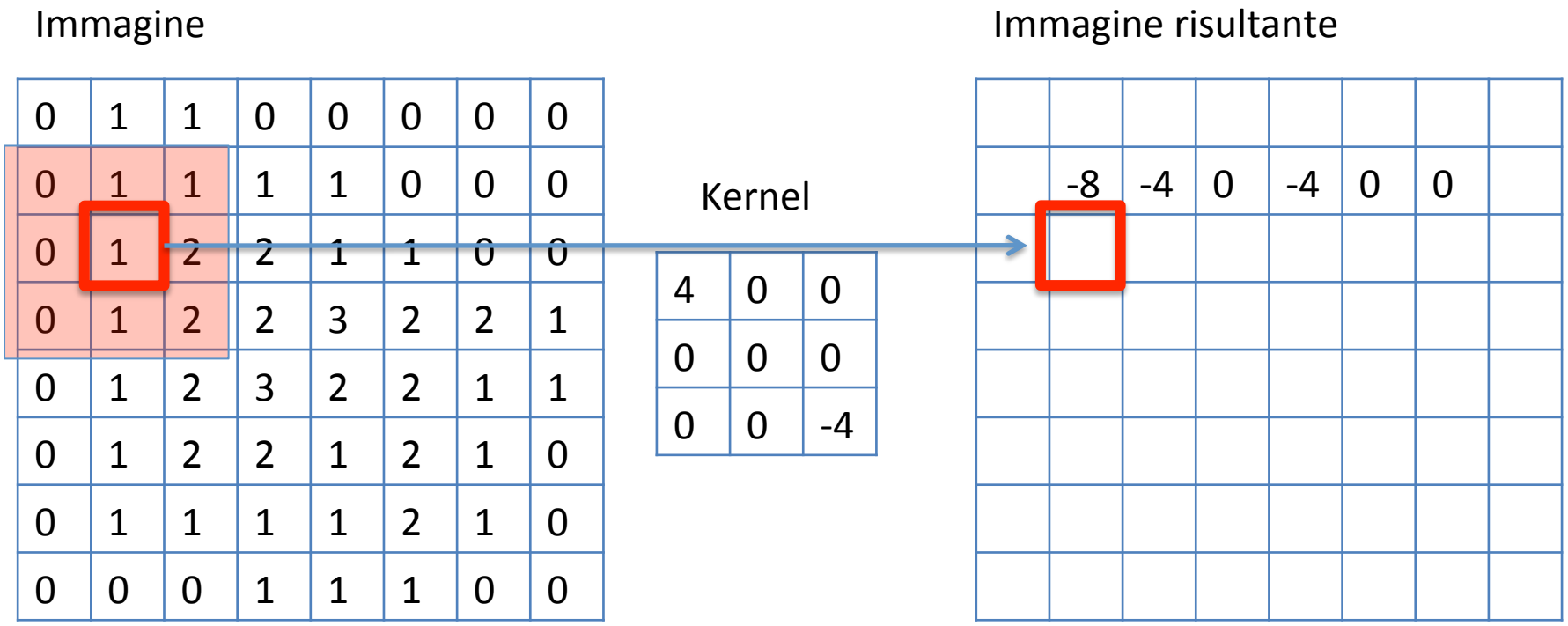
4	0	0
0	0	0
0	0	-4

Immagine risultante

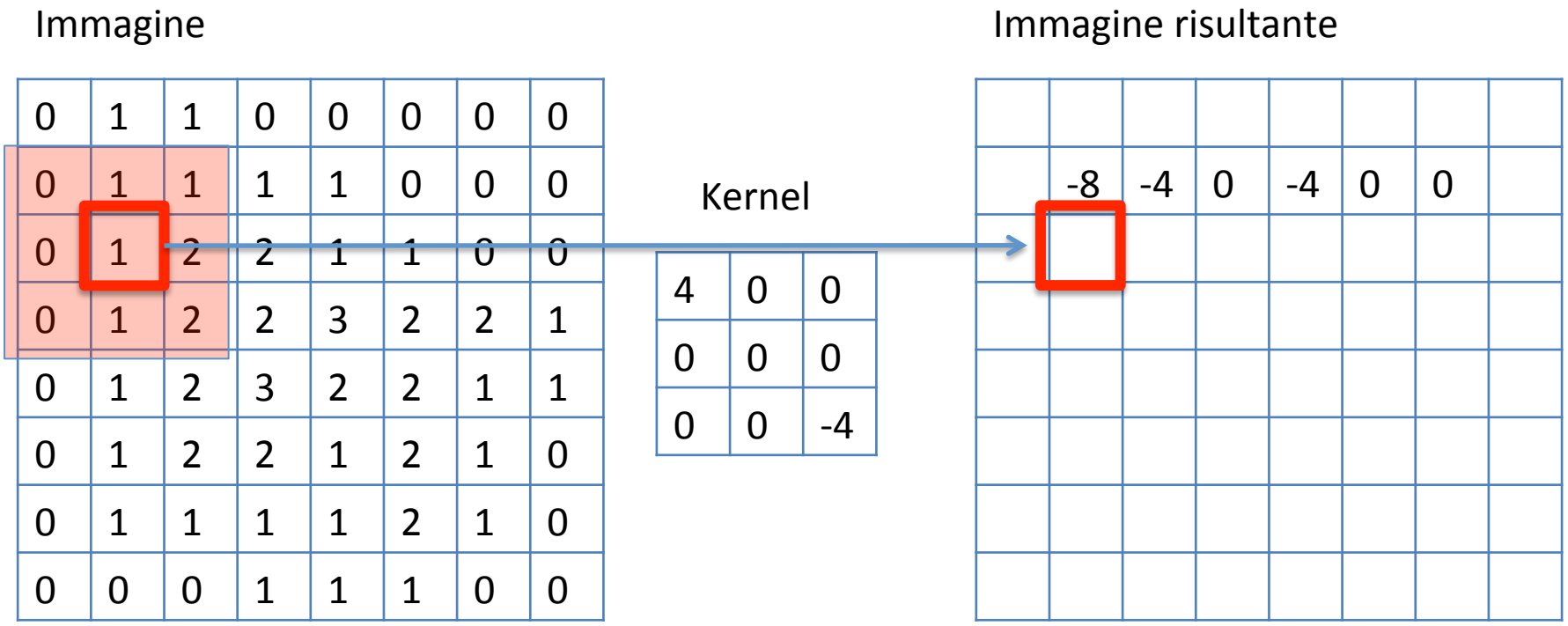
		-8	-4				

4*1+0*1+0*0+
0*1+0*1+0*1+
0*1+0*2+(-4*2)
=-4

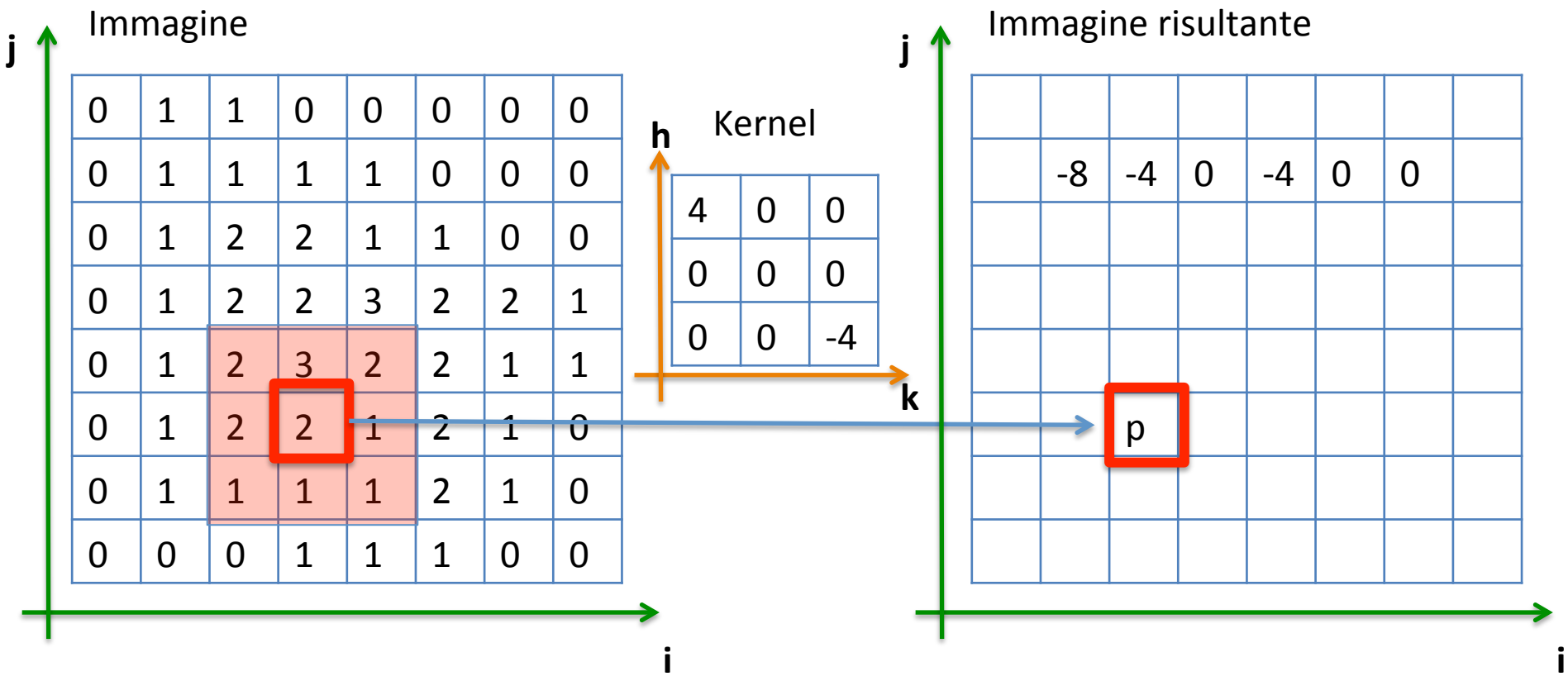
Convoluzione



Convoluzione



Convoluzione



$$p(i,j) = \sum_{u=0}^{2}\sum_{v=0}^{2} c(i-1+u, j-1+v) * k(u,v)$$

Normalizzazione e correzione

$$p(i,j) = \sum_{u=0}^{2}\sum_{v=0}^{2} c(i-1+u, j-1+v) * k(u,v)$$

Normalizzazione

$$\hat{p}(i,j) = p(i,j) * factor - bias$$

Limitazione

$$p(i,j) = \min(\max(\hat{p}(i,j), 255), 0)$$

Filtri

Originale

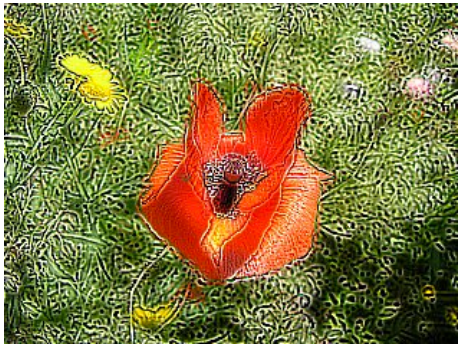
BLUR

0	0.2	0
0.2	0.2	0.2
0	0.2	0

Sharpen

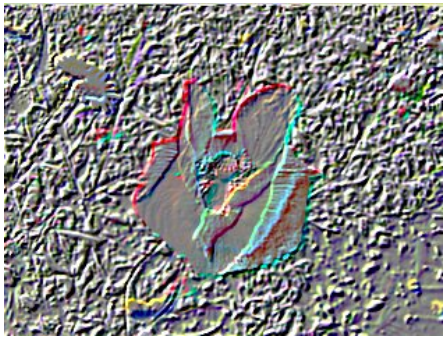
-1	-1	-1
-1	9	-1
-1	-1	-1

Edge detection



1	1	1
1	-7	1
1	1	1

Emboss



-1	-1	0
-1	0	1
0	1	1

bias=128

Dove non indicato: factor =1, bias=0

Esercizio

- Leggere l'immagine img3.bmp e salvare le seguenti immagini
  - Immagine con applicato il filtro di blur
  - Applicare altri filtri
  - ...
  - Immagine capovolta verticalmente

- Immagine
  - http://ppl.eln.uniroma2.it/finf/img3.bmp
  - http://ppl.eln.uniroma2.it/finf/filtimgst.c

```c
int main() {
    int i=0, j=0,im,jm;
    unsigned char header[54];
    unsigned char r,g,b;
    int width = 320, height = 240;
    unsigned char rm[height][width],gm[height][width],bm[height][width];

    FILE *fin = fopen("img3.bmp", "rb");

    fread(header, sizeof(unsigned char), 54, fin);
    for (i=0; i<height; i++ ) {
        for (int j=0; j<width; j++)
        {
            fread(&b,sizeof(b),1,fin);
            fread(&g,sizeof(g),1,fin);
            fread(&r,sizeof(r),1,fin);
            bm[i][j]=b;
            rm[i][j]=r;
            gm[i][j]=g;

        }
    }
    fclose(fin);

    FILE *fout = fopen("img3-mod-st.bmp", "wb");
    fwrite(header, 54, 1, fout);
    for (i=0; i<height; i++ ) {
        for (int j=0; j<width; j++)
        {
            // applicare il fltro
            b=bm[i][j];
            g=gm[i][j];
            r=rm[i][j];
            // salvo il pixel
            fwrite(&b,1,1,fout);
            fwrite(&g,1,1,fout);
            fwrite(&r,1,1,fout);
        }
    }

    fclose(fout);
    return 0;
}
```


```
bv=0;
rv=0;
gv=0;
for(jm=0;jm<5;jm++){
    for(im=0;im<5;im++){
        bv+=bm[i-2+im][j-2+jm]*filter[im][jm];
        rv+=rm[i-2+im][j-2+jm]*filter[im][jm];
        gv+=gm[i-2+im][j-2+jm]*filter[im][jm];
    }
}
b=min(max((int) (bv*factor+bias),0),255);
r=min(max((int) (rv*factor+bias),0),255);
g=min(max((int) (gv*factor+bias),0),255);
```