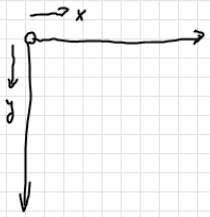
ÜBUNG: Graustufenreduktion

Geben Sie einen Algorithmus (in C++) an, welcher die Graustufen eines 8-bit Grauwertbildes (Name: `pic`) auf 16 Graustufen reduziert.

Übung: Graustufenreduktion

```
// Bild "pic" (channel8-Typ) anlegen und laden
...
int y_size, x_size;
...
y_size = pic.rows();
x_size = pic.columns();
...
for (y=0; y < y_size; y++) {
    for (x=0; x < x_size; x++) {
        pic[y][x] = pic[y][x] & 0xF0;
    }
}
```



unsigned char

bitweise AND

0xF0 ≙ 1111 0000

ÜBUNG: Glättungsfilter "gleitender Mittelwert"

Geben Sie einen Algorithmus (in C++) an, welcher ein 8-bit Grauwertbild dadurch glättet, indem der Grauwert eines Bildpunktes (x,y) durch den Mittelwert über seine 8-er Nachbarschaft ersetzt wird (**f**=Quellbild, **g**=Zielbild).

$$g(x,y) = \frac{1}{9} \cdot [f(x,y) + f(x-1,y) + f(x-1,y-1) + \ldots\ldots + f(x-1,y+1)]$$

Zu beachten: - Randbehandlung
- Zahlbereich nicht überschreiten

```
// Quellbild "f" laden
int y_size, x_size, sum;
y_size = f.rows(); x_size = f.columns();
:
channel8 g(y_size, x_size); // Zielbild in der Größe des Quellbildes anlegen

for(y=1; y<y_size-1; y++){
    for(x=1; x<x_size-1; x++){
        sum = f[y-1][x-1] + f[y-1][x] + ..........
        g[y][x] = sum / 9.0 + 0.5; // mit Rundung
    }
}
```

ÜBUNG: Histogrammberechnung

Geben Sie einen Algorithmus (in C++) an, das Histogramm eines Bildes berechnet.

Hinweise:

Die Bildgröße eines Bildes "pic" in x- und y-Richtung erhält man mit den Befehlen

```
y_size = pic.rows();        // Integer-Format
x_size = pic.columns();
```

Den Grauwert eines Bildpunktes an der Stelle (x,y) erhält bzw. setzt man mit

```
pic[y][x] = Grauwert;       // unsigned char
Grauwert  = pic[y][x];
```

Histogrammberechnung:

```
unsigned char Grauwert;
unsigned int histogram[256];

// Initialisierung
for (i=0; i<256; i++)
    histogram[i] = 0;

:

for (y=0; y<y_size; y++) {
    for (x=0; x<x_size; x++) {
        Grauwert = pic[y][x];
        histogram[Grauwert]++;
    }
}
```

ÜBUNG: Operationen mit Look-up-Tabellen

Geben sei ein 4-bit-Grauwertbild
→ Graustufen 0….15.

Geben Sie die Look-up-Tabellen für folgende Operationen an:

a) Binarisierung des Bildes bei Grauwert 10.

b) Reduktion auf 4 Grauwerte (0, 5, 10, 15) in 4 äquidistanten Stufen.

c) Anhebung des Kontrastes im dunklen Grauwertbereich auf Kosten des hellen Bereiches.

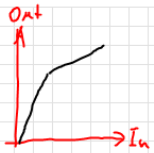
d) Hervorheben von Bitebene 0.

In	Out
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	15
11	15
12	15
13	15
14	15
15	15

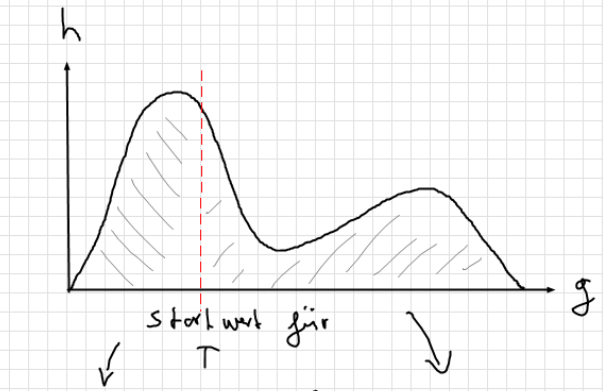
In	Out
0	0
1	0
2	0
3	0
4	5
5	5
6	5
7	5
8	10
9	10
10	10
11	10
12	15
13	15
14	15
15	15

In	Out
0	0
1	2
2	4
3	6
4	8
5	10
6	11
7	11
8	12
9	12
10	13
11	13
12	14
13	14
14	15
15	15

In	Out
0	0
1	15
2	2
3	15
4	4
5	15
6	6
7	15
8	8
9	15
10	10
11	15
12	12
13	15
14	14
15	15



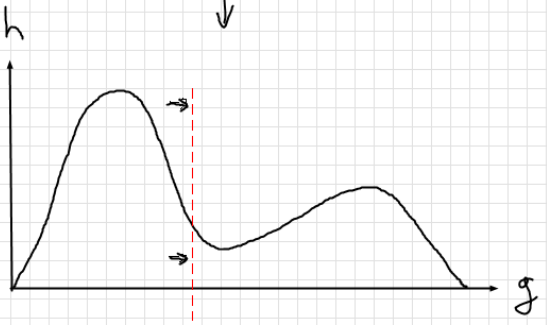
Adaptive Schwellwertberechnung



Berechnung von
μ_1, σ_1, P_1

Berechnung von
μ_2, σ_2, P_2

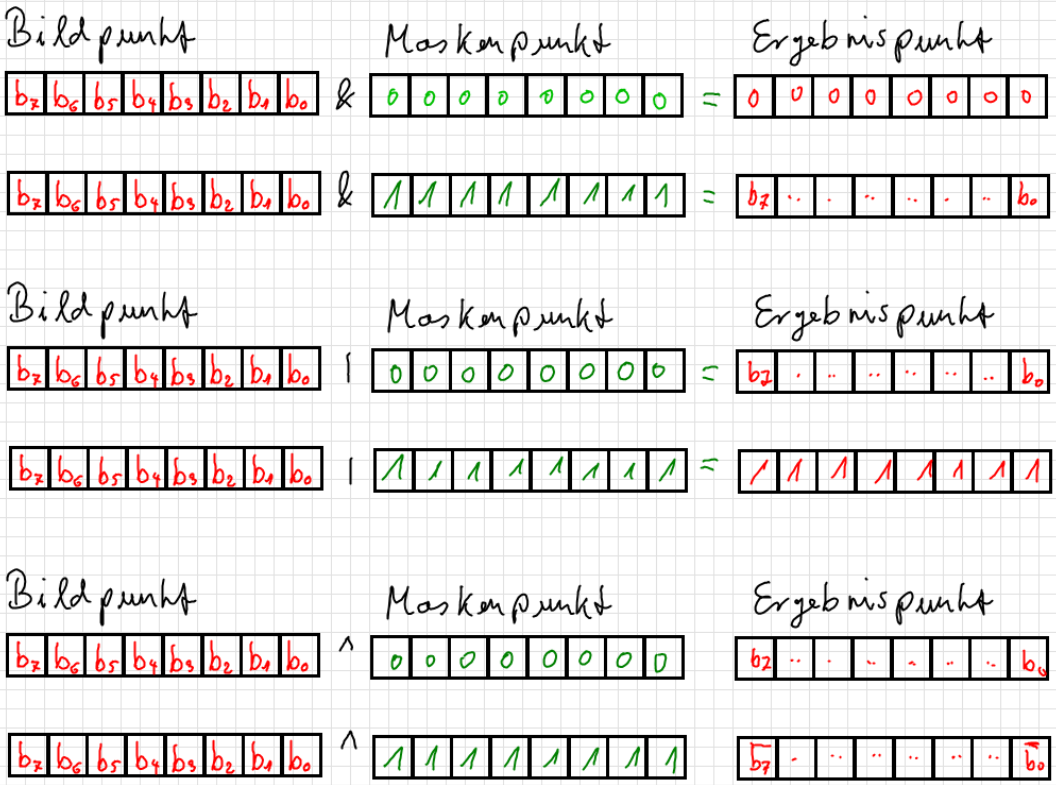
Mit Schwellwertformel
das verbesserte $\bar{T}$ berechnen



... wiederholen, bis sich $\bar{T}$ nicht mehr ändert!

Bitweise log. Operationen zwischen Bildern

⇒ Maskieren mit 0 bzw. 255



Histogrammausgleich

```
/*---------------------*/
/* images & channels   */
/*---------------------*/
image     img;  // normalized (color) image
channel8  src;  // source picture       // 8-bit-image (source)
channel8  dst;  // destination picture  // 8-bit-image (source)

splitImageToHSI splitter;

/*****************************************/
/*    the program                        */
/*****************************************/

// load the source image
loader.load("Flur2.bmp",img);

// extract the intensity channel only
splitter.getIntensity(img,src);

// determine image size
const int rowSize    = src.rows();
const int columnSize = src.columns();

// set destination size to source size
dst.resize(rowSize,columnSize,0,false,true);

int       Hg[256];
int       AHg[256];

const int gmax = 255;

// init. Histogramm
for(int g=0; g<=gmax; g++){
   Hg[g]=0;
}

// berechne Histogramm
for(int y=0; y<rowSize; y++){
   for(int x=0; x<columnSize; x++){
      Hg[src[y][x]]++;
   }
}

// berechne akkumuliertes Histogramm
AHg[0] = Hg[0];
for(int g=1; g<=gmax; g++){
   AHg[g] = AHg[g-1] + Hg[g];
}

// berechne Histogrammausgleich
for(int y=0; y<rowSize; y++){
    for(int x=0; x<columnSize; x++){
       dst[y][x] = (double)(AHg[src[y][x]] -  AHg[0]) / (AHg[gmax] -  AHg[0]) * gmax ;
   }
}

// view pictures
view.show(src);
viewTransformed.show(dst);
```