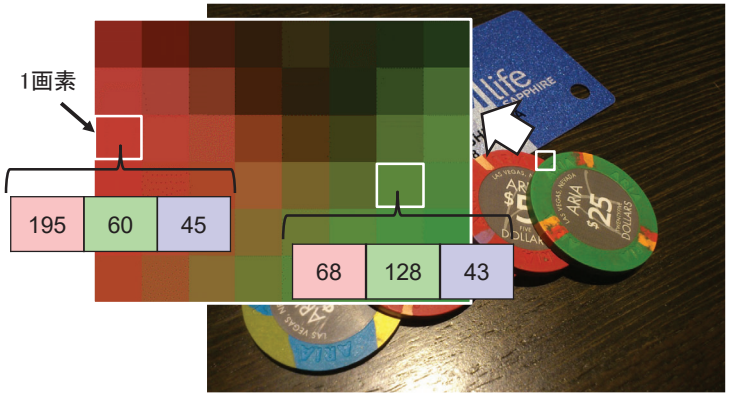
画素値の基本（復習）

ディジタル画像データの基本

- 画素ごとに赤，緑，青の3色の情報を持っている
- 赤，緑，青（RGB）の組み合わせで色を表現



画素値の例

- 一般的には0から255（8 bit）で表現
- 全部で16,777,216色を表現可能

R	G	B	色
0	0	0	
255	0	0	
0	255	0	
0	0	255	
255	255	0	
0	255	255	
255	0	255	

R	G	B	色
10	10	10	
128	128	128	
200	200	200	
255	255	255	
255	192	0	
202	254	207	
235	189	173	

画素値の範囲について

- 0から255なので，unsigned char に格納
- 型変換やオーバーフロー/アンダーフローに注意

```
typedef struct {
  unsigned char r;
  unsigned char g;
  unsigned char b;
} PIXEL;

typedef struct {
  int    xsize;
  int    ysize;
  int    level;
  PIXEL **pBuffer;
} IMAGE;
```

例：

```
pImage->pBuffer[0][0].r = 0;     /* rは0 */
pImage->pBuffer[0][0].r = 100; /* rは100 */
pImage->pBuffer[0][0].r = 300; /* rは44 */
pImage->pBuffer[0][0].r = 255 - 256; /* rは255 */
```

ビット演算を利用した
減色処理

減色処理

- RGB各色の数値の組み合わせを減らしつつ，原画像の色に近い色に割り当て

原画像
RGB各色　0から255
色数：256^3 = 16,777,216色

減色画像の例
RGB各色　0, 64, 128, 192のみ
色数：4^3 = 64色
下位6ビットを0にしている

減色処理の実装方法

- If文で切り分ける
  - 例：0以上64未満の画素値は全て0にする，64以上128未満の画素値は全て64にする・・・
- 割り算
  - 例：画素値を64で割ってから小数点を切り捨てて，64倍する
- ビット演算を使うと最も簡単

ビット演算

- 2進表記の各桁で論理演算を行う
  - AND(&)，OR(|)，右ビットSHIFT(>>)の例

```
unsigned char A = 113;   /* 0111 0001 */
unsigned char B = 31;    /* 0001 1111 */
unsigned char D;

D = A & B;               /* AND:    0001 0001 (17) */
D = A | B;               /* OR:     0111 1111  (127) */
D = A >> 3;              /* SHIFT: 0000 1110  (14) */
```

他にもXOR(^)，左ビットSHIFT(<<)，NOT(~)などがある

右シフト演算の補足

- 符号なし変数:
  - 隙間は0で埋められる (論理シフト)
- 符号あり変数のシフト
  - 隙間は最上位ビットで埋められる* (算術シフト)

```
unsigned char A = 113;   /* 0111 0001 */
char C = -64;            /* 1100 0000 */
char D;

D = A >> 3;              /* 0000 1110 (14) */
D = C >> 3;              /* 1111 1000 (-8) */
```


AND演算のマスクと16進数表記

- AND演算を利用すると，所望のビットを残して，他を切り捨てることができる
  - Bにおいて0の桁は，ANDをとると必ず0
  - Bにおいて1の桁は，ANDととると必ずAと一致
  - このような処理をビットマスクとよぶ

```
unsigned char A = 113;   /* 0111 0001 */
unsigned char B = 31;    /* 0001 1111 */
unsigned char D;

D = A & B;               /* AND:  0001 0001 (17) */
```

上位3ビットを0にするマスク

必ず0　Aのビットと一致

マスクBの計算が面倒であれば

- 10進表記ではなく16進表記を利用すると楽
  - 2進数の下位4ビットは16進数の下一桁目，2進数の上位4ビットは16進数の次の桁に相当
  - 0001→1，1111→F

```
unsigned char A = 113;   /* 0111 0001 */
unsigned char B = 0x1F;  /* 0001 1111 */
unsigned char D;

D = A & B;               /* AND:  0001 0001 (17) */
```

上位3ビットを0にするマスク

必ず0　Aのビットと一致

【演習課題4_4_1】

- 画像の各画素の下位nビットを0でビットマスクすることで，減色画像を作成する
  - n = (自身の学籍番号の下一桁 % 7) + 1 とする
  - img_procにipBitMask関数として実装すること
  - プロトタイプ宣言も忘れずに追加
  - main.cでは，ipCopy関数の代わりにipBitMask関数を呼び出す

演習課題の目安: 10分