JPEGのアルゴリズムについて

JPEGアルゴリズムの確認

[Reference]
http://en.wikipedia.org/wiki/Jpeg

2006/10/21 平野拓一（東京工業大学）
2006/10/21 Takuichi Hirano (Tokyo Institute of Technology)

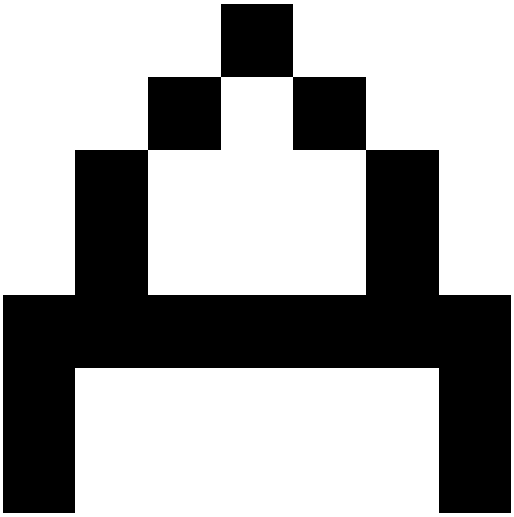
BMPファイルの読み込み

```
In[1]:= SetDirectory["d:/hira2/public_html/hobby/edu/jpeg"]
Out[1]= d:\hira2\public_html\hobby\edu\jpeg

        SetDirectory["d:/Home/takuichi/hira2/public_html/hobby/edu/jpeg"]

In[2]:= g = Import["a.bmp"]
Out[2]= - Graphics -

In[3]:= Show[g]

Out[3]= - Graphics -
```

InputForm[g]

```
In[4]:= ImgList = g[[1, 1]];  (* {R,G,B} の2次元リスト *)

        ImgListRed = Map[(#[[1]] &), ImgList, {2}];  (* R (0-255) の2次元リスト *)
        ImgListGreen = Map[(#[[2]] &), ImgList, {2}];  (* G (0-255) の2次元リスト *)
        ImgListBlue = Map[(#[[3]] &), ImgList, {2}];  (* B (0-255) の2次元リスト *)

        nx = Length[ImgList[[1]]];
        ny = Length[ImgList];

In[9]:= MakeRGBRasterImage[ImgListRed_, ImgListGreen_, ImgListBlue_] := Module[{},
          Graphics[Raster[Table[{ImgListRed[[j, i]],
              ImgListGreen[[j, i]], ImgListBlue[[j, i]]}, {j, 1, ny}, {i, 1, nx}],
            {{0, 0}, {nx, ny}}, {0, 255}, ColorFunction → RGBColor]]
        ];
```

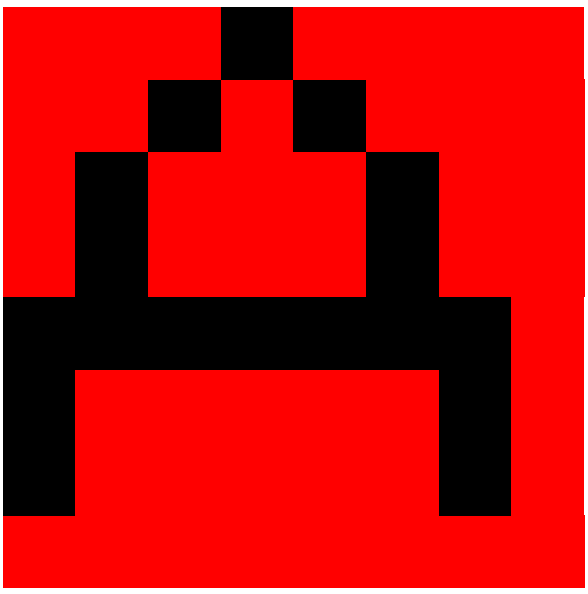
```
In[10]:= Show[MakeRGBRasterImage[ImgListRed, ImgListGreen, ImgListBlue],
          AspectRatio → Automatic]
```



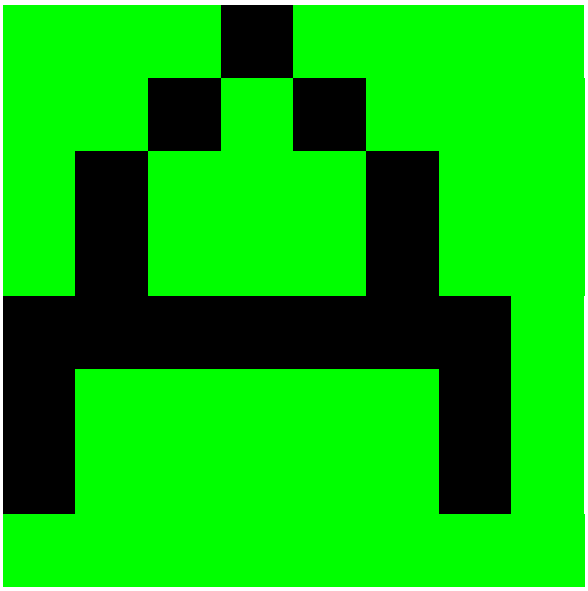
```
Out[10]= - Graphics -
```

```
In[11]:= ZeroTable = Table[0, {n, 0, ny - 1}, {m, 0, nx - 1}];
         Print["**** Red Component ****"];
         Show[MakeRGBRasterImage[ImgListRed, ZeroTable, ZeroTable],
           AspectRatio → Automatic];
         Print["**** Green Component ****"];
         Show[MakeRGBRasterImage[ZeroTable, ImgListGreen, ZeroTable],
           AspectRatio → Automatic];
         Print["**** Blue Component ****"];
         Show[MakeRGBRasterImage[ZeroTable, ZeroTable, ImgListBlue],
           AspectRatio → Automatic];
```

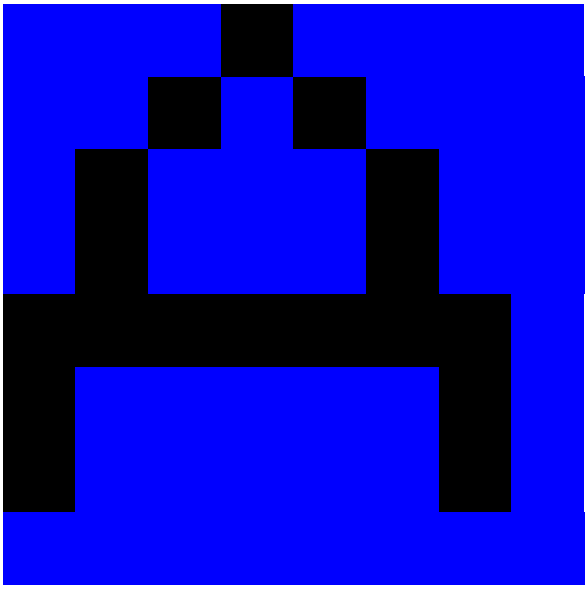
```
**** Red Component ****
```



**** Green Component ****



**** Blue Component ****



(R,G,B) → (Y, Cb, Cr)
JPEGではRGBでなく、YCbCr(またはYUVとも言われる)でそれぞれの成分を処理する。YUVはテレビ放送のPAL方式で採用されている色表現。

http://en.wikipedia.org/wiki/YCbCr

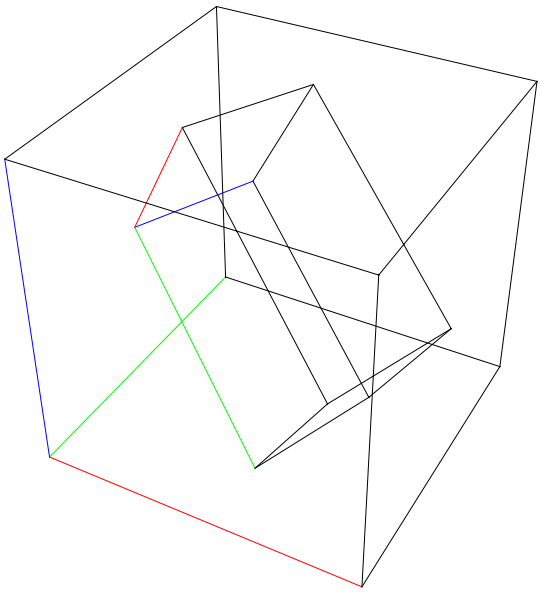
The Y component represents the brightness of a pixel
The Cb and Cr components together represent the chrominance

The human eye can see more detail in the Y component than in Cb and

Cr. Using this knowledge, encoders can be designed to compress images more efficiently.

In[18]:=
```
YCbCrFromRGB[{r_, g_, b_}] := {0.299 * r + 0.587 * g + 0.114 * b,
   128 - 0.168736 * r - 0.331264 * g + 0.5 * b,
   128 + 0.5 * r - 0.418688 * g - 0.081312 * b};
RGBFromYCbCr[{y_, cb_, cr_}] :=
  {-179.45579155073017` - 1.2175717664097974`*^-6 * cb + 1.4019995886573404` * cr + y,
   135.45879483222961` - 0.3441356788389385` * cb - 0.7141361555818125` * cr + y,
   -226.81606046360054` + 1.7720000660738162` * cb + 4.062980628562637`*^-7 * cr + y};
```

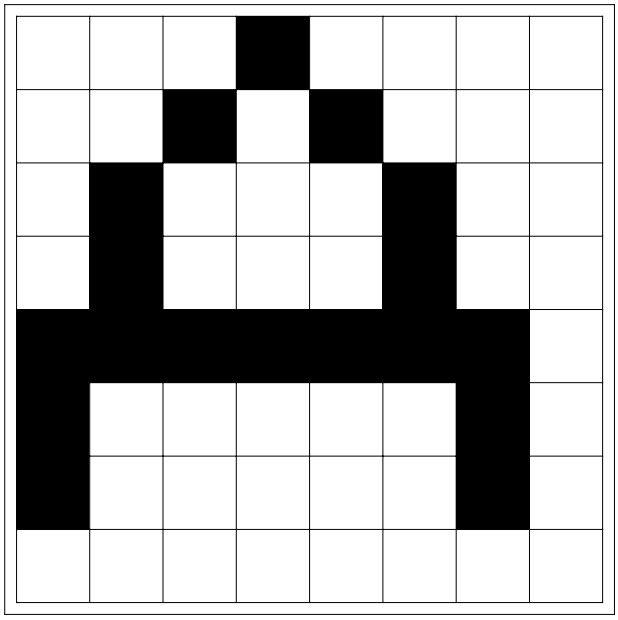
In[20]:=
```
p[0] = {0, 0, 0};
p[1] = {255, 0, 0};
p[2] = {255, 255, 0};
p[3] = {0, 255, 0};
p[4] = {0, 0, 255};
p[5] = {255, 0, 255};
p[6] = {255, 255, 255};
p[7] = {0, 255, 255};
pcubeRGB = {{p[0], p[1]}, {p[1], p[2]}, {p[2], p[3]}, {p[3], p[0]},
   {p[4], p[5]}, {p[5], p[6]}, {p[6], p[7]}, {p[7], p[4]},
   {p[0], p[4]}, {p[1], p[5]}, {p[2], p[6]}, {p[3], p[7]}};
pcubeYCbCr = Map[YCbCrFromRGB, pcubeRGB, {2}];
pcubeRGB2 = {{RGBColor[1, 0, 0], Line[{p[0], p[1]}]},
   {RGBColor[0, 1, 0], Line[{p[0], p[3]}]}, {RGBColor[0, 0, 1], Line[{p[0], p[4]}]}};
pcubeYCbCr2 = {{RGBColor[1, 0, 0], Line[{YCbCrFromRGB[p[0]], YCbCrFromRGB[p[1]]}]},
   {RGBColor[0, 1, 0], Line[{YCbCrFromRGB[p[0]], YCbCrFromRGB[p[3]]}]},
   {RGBColor[0, 0, 1], Line[{YCbCrFromRGB[p[0]], YCbCrFromRGB[p[4]]}]}};
Show[{Graphics3D[{Map[Line, pcubeRGB, 1], pcubeRGB2}],
   Graphics3D[{Map[Line, pcubeYCbCr, 1], pcubeYCbCr2}]},
  Boxed → False];
```



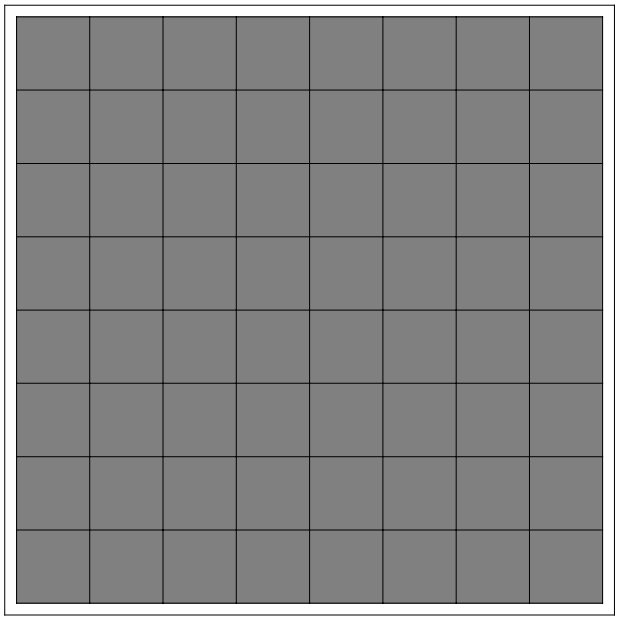
In[34]:=
```
ImgListY = Map[(YCbCrFromRGB[#][[1]] &), ImgList, {2}];   (* Yの2次元リスト *)
ImgListCb = Map[(YCbCrFromRGB[#][[2]] &), ImgList, {2}];   (* Cbの2次元リスト *)
ImgListCr = Map[(YCbCrFromRGB[#][[3]] &), ImgList, {2}];   (* Crの2次元リスト *)

ImgListY = ImgListY - 128;  (* Yは128を引いて-128-127 の範囲にする *)
```

In[35]:=
```
Print["**** Y Component ****"];
ListDensityPlot[ImgListY,
  FrameTicks → False,
  PlotRange → {-128, 127}];
Print["**** Cb Component ****"];
ListDensityPlot[ImgListCb,
  FrameTicks → False,
  PlotRange → {0, 255}];
Print["**** Cr Component ****"];
ListDensityPlot[ImgListCr,
  FrameTicks → False,
  PlotRange → {0, 255}];
```

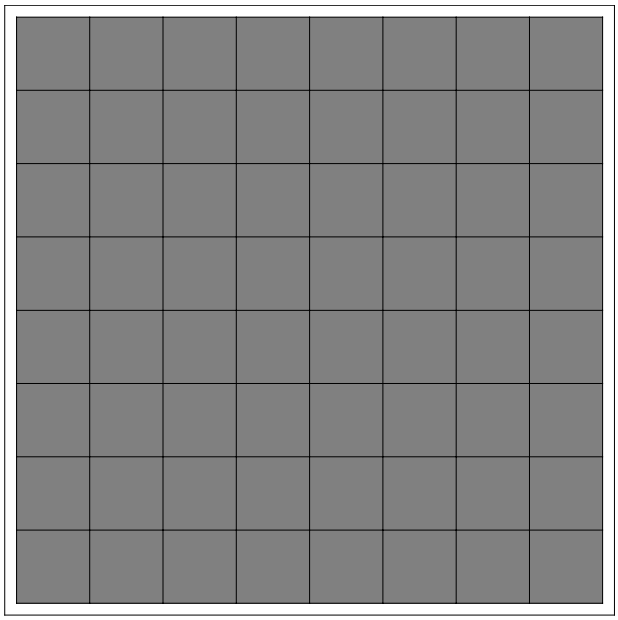
```
**** Y Component ****
```



```
**** Cb Component ****
```



**** Cr Component ****

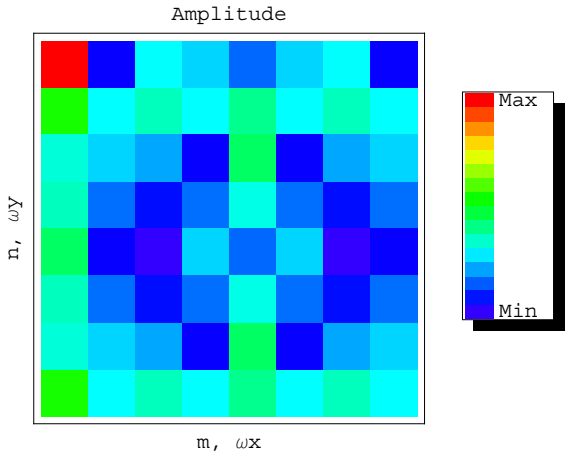


DFT (Discrete Fourier Transform)
JPEGではDCTが使われるので、これは関係ないが、基本的なDFTと比較するために計算してる。

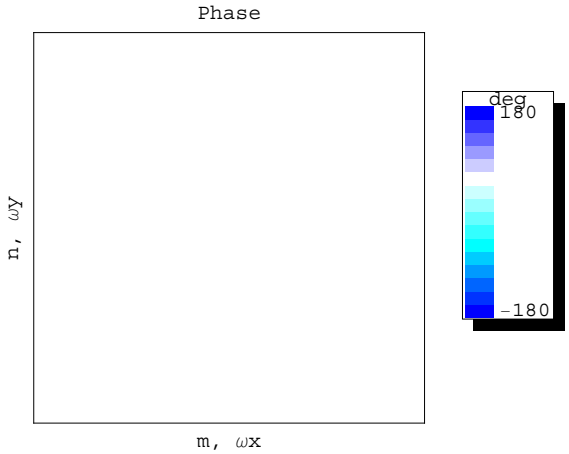
```
In[44]:= FImgListY = Fourier[ImgListY];
         FImgListCb = Fourier[ImgListCb];
         FImgListCr = Fourier[ImgListCr];
```

In[47]:=

```
<< Graphics`Legend`;
ShowLegend[
  ListDensityPlot[Abs[ Reverse[FImgListY] ],
   Mesh → False,
   PlotRange → Automatic,
   PlotLabel → "Amplitude",
   FrameLabel → {"m, ωx", "n, ωy"},
   FrameTicks → None,
   ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &),
   DisplayFunction → Identity]
  , {(Hue[0.7 * (#), 1, 1] &), 16, "Max", "Min",
   LegendPosition → {1.1, -.4}, LegendSize → {0.4, 1.}, LegendLabel → ""}
 ];

colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);

ShowLegend[
  ListDensityPlot[Arg[Reverse[FImgListY]] * 180/π,
   Mesh → False,
   PlotRange → {-180, 180},
   PlotLabel → "Phase",
   FrameLabel → {"m, ωx", "n, ωy"},
   FrameTicks → None,
   ColorFunction → colfun,
   DisplayFunction → Identity]
  , {colfun[-# + 1] &, 16, "180", "-180",
   LegendPosition → {1.1, -.4}, LegendSize → {0.4, 1.}, LegendLabel → "deg"}
 ];
```



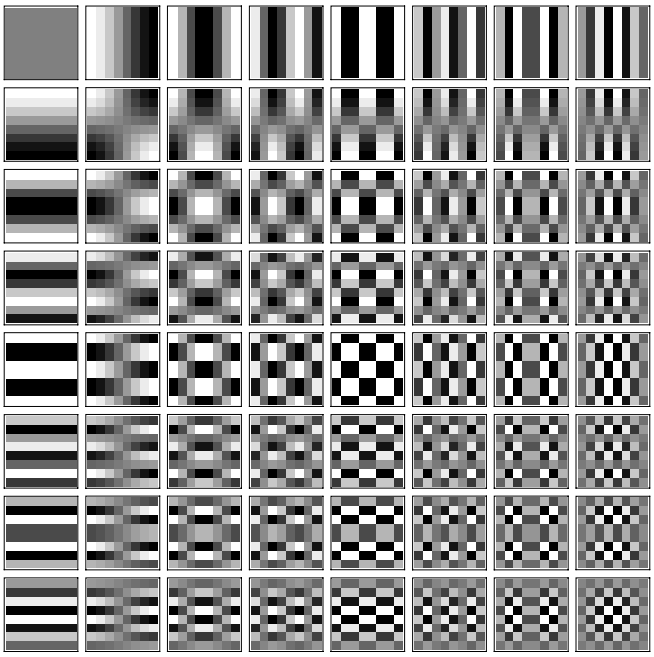

```
Arg::indet : 不定式Arg[0. + 0. i]が見付かりました。 詳細
Arg::indet : 不定式Arg[0. + 0. i]が見付かりました。 詳細
DensityGraphics::zval : 非数値Interval[{-180, 180}]が第1引数の位置{4, 3}で見付かりました。 詳細
DensityGraphics::zval : 非数値Interval[{-180, 180}]が第1引数の位置{4, 7}で見付かりました。 詳細
```



DCT (Discrete Cosine Transform; **離散コサイン変換**)

基底の描画（計算と無関係。教育用）

In[53]:=
```
c[p_] := If[p == 0, 1/Sqrt[2], 1];
b[u_, v_, m_, n_] := 2/nn * c[u] * c[v] * Cos[(2*m + 1)*u*π/(2*nn)] * Cos[(2*n + 1)*v*π/(2*nn)];
nn = 8;
buv[u_, v_] :=
  ListDensityPlot[Reverse[Table[b[u, v, m, n], {m, 0, nn - 1}, {n, 0, nn - 1}]],
   Mesh → False,
   Frame → True,
   FrameTicks → None,
   DisplayFunction → Identity];
Show[GraphicsArray[Table[buv[u, v], {u, 0, nn - 1}, {v, 0, nn - 1}]],
 ImageSize → {320, 320}]
```



Out[57]= - GraphicsArray -

２次元リストにDCTを行う関数の定義

[**参考**]
貴家仁志：よくわかるディジタル画像処理、CQ出版社、p.123

In[58]:=
```
DCT[lis_] := Module[{c, g, nn},
   c[p_] := If[p == 0, 1/Sqrt[2], 1];
   nn = Length[lis];
   g[u_, v_] := 2/nn * c[u] * c[v] * Sum[lis[[m + 1, n + 1]] *
       Cos[(2*m + 1)*u*π/(2*nn)] * Cos[(2*n + 1)*v*π/(2*nn)], {m, 0, nn - 1}, {n, 0, nn - 1}];
   Table[g[u, v], {u, 0, nn - 1}, {v, 0, nn - 1}]
  ];
```

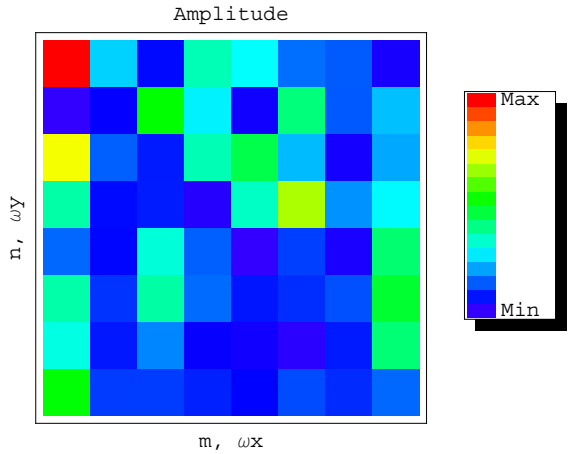
```
In[59]:= FImgListY = DCT[ImgListY];
         FImgListCb = DCT[ImgListCb];
         FImgListCr = DCT[ImgListCr];

         Print[Round[FImgListY] // MatrixForm];
         Print[Round[FImgListCb] // MatrixForm];
         Print[Round[FImgListCr] // MatrixForm];
```

$$\begin{pmatrix} 442 & -108 & 24 & -157 & 128 & -66 & 59 & 11 \\ 0 & 19 & -231 & -121 & -15 & -181 & 57 & -100 \\ 333 & 60 & 32 & 158 & -201 & -99 & -13 & -90 \\ 163 & 25 & 32 & 5 & 151 & 302 & -81 & 125 \\ -64 & -23 & 142 & 61 & 0 & 47 & 10 & 185 \\ -162 & -42 & 163 & 66 & -30 & 40 & -54 & -212 \\ 138 & 31 & -77 & -18 & 14 & -4 & 32 & 183 \\ 230 & 46 & -46 & 35 & -22 & 52 & -38 & -64 \end{pmatrix}$$

$$\begin{pmatrix} 1024 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1024 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

```
In[65]:= << Graphics`Legend`;
         ShowLegend[
           ListDensityPlot[Abs[ Reverse[FImgListY] ],
            Mesh → False,
            PlotRange → Automatic,
            PlotLabel → "Amplitude",
            FrameLabel → {"m, ωx", "n, ωy"},
            FrameTicks → None,
            ColorFunction → (Hue[-0.7 * (# - 1), 1, 1] &),
            DisplayFunction → Identity]
           , {(Hue[0.7 * (#), 1, 1] &), 16, "Max", "Min",
            LegendPosition → {1.1, -.4}, LegendSize → {0.4, 1.}, LegendLabel → ""}
          ];

         colfun[x_] := RGBColor[0, 3 * x, 1] /; (x ≤ 1 / 3);
         colfun[x_] := RGBColor[3 * (x - 1 / 3), 1, 1] /; (1 / 3 < x ≤ 2 / 3);
         colfun[x_] := RGBColor[-3 * (x - 2 / 3) + 1, -3 * (x - 2 / 3) + 1, 1] /; (2 / 3 < x);
```



Quantization (量子化)

量子化テーブル

In[70]:=
$$qty = \begin{pmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{pmatrix};$$ (* Y成分用 *)

$$qtcbcr = \begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix};$$ (* Cb,Cr成分用 *);

In[76]:=
```
text = Table[{Text[StyleForm[
       ToString[Transpose[Reverse[qty]][[i, j]]], FontSize → 16, FontWeight → "Bold",
       FontColor → Hue[0.7 * (Transpose[Reverse[qty]][[i, j]] / 255.), 1, 1]],
      {i, j}]}, {i, 1, 8}, {j, 1, 8}];
Show[Graphics[{text}],
  PlotRange → {{0.5, 8.5}, {0.5, 8.5}},
  AspectRatio → Automatic];
```

16 11 10 16 24 40 51 61

12 12 14 19 26 58 60 55

14 13 16 24 40 57 69 56

14 17 22 29 51 87 80 62

18 22 37 56 68 109 103 77

24 35 55 64 81 104 113 92

49 64 78 87 103 121 120 101

72 92 95 98 112 100 103 99

量子化する
sは品質（大きいほど良い）

In[85]:=
```
s = 1;

FImgListY2 = Round[FImgListY / (qty / s)];
FImgListCb2 = Round[FImgListCb / (qtcbcr / s)];
FImgListCr2 = Round[FImgListCr / (qtcbcr / s)];

Print[FImgListY2 // MatrixForm];
Print[FImgListCb2 // MatrixForm];
Print[FImgListCr2 // MatrixForm];
```

$$\begin{pmatrix} 28 & -10 & 2 & -10 & 5 & -2 & 1 & 0 \\ 0 & 2 & -17 & -6 & -1 & -3 & 1 & -2 \\ 24 & 5 & 2 & 7 & -5 & -2 & 0 & -2 \\ 12 & 1 & 1 & 0 & 3 & 3 & -1 & 2 \\ -4 & -1 & 4 & 1 & 0 & 0 & 0 & 2 \\ -7 & -1 & 3 & 1 & 0 & 0 & 0 & -2 \\ 3 & 0 & -1 & 0 & 0 & 0 & 0 & 2 \\ 3 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

$$\begin{pmatrix} 60 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

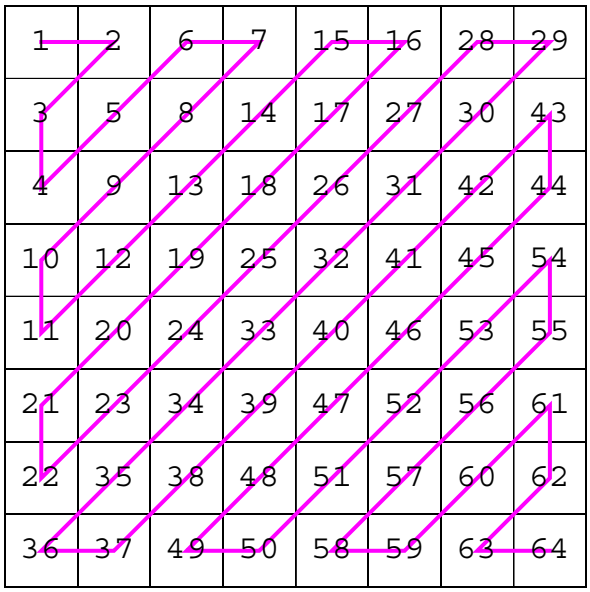
$$\begin{pmatrix} 60 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

本来ならばここでジグザグスキャンを行った後、エントロピー符号化（ハフマン符号化または算術符号化）が入り、JPEGファイルが生成される。

In[92]:=
```
dx = dy = 1.;
ggrid = Table[{Line[{{dx * (i - 1), 0}, {dx * (i - 1), dy * 8}}],
    Line[{{0, dy * (j - 1)}, {dx * 8, dy * (j - 1)}}]}, {i, 1, 9}, {j, 1, 9}];
```

$$\text{zigzagmat} = \begin{pmatrix} 1 & 2 & 6 & 7 & 15 & 16 & 28 & 29 \\ 3 & 5 & 8 & 14 & 17 & 27 & 30 & 43 \\ 4 & 9 & 13 & 18 & 26 & 31 & 42 & 44 \\ 10 & 12 & 19 & 25 & 32 & 41 & 45 & 54 \\ 11 & 20 & 24 & 33 & 40 & 46 & 53 & 55 \\ 21 & 23 & 34 & 39 & 47 & 52 & 56 & 61 \\ 22 & 35 & 38 & 48 & 51 & 57 & 60 & 62 \\ 36 & 37 & 49 & 50 & 58 & 59 & 63 & 64 \end{pmatrix};$$

```
mat = Transpose[Reverse[zigzagmat]];
zigzaglin = Line[Table[Position[mat, i][[1]] - {dx / 2, dy / 2}, {i, 1, 64}]];
text =
  Table[{Text[StyleForm[ToString[mat[[i, j]] ], FontSize → 16, FontWeight → "Bold"],
     {i - dx / 2, j - dy / 2}]}, {i, 1, 8}, {j, 1, 8}];
Show[Graphics[{ggrid, {RGBColor[1, 0, 1], AbsoluteThickness[2], zigzaglin}, text}],
  AspectRatio → Automatic];
```



ここから先はJPEGファイルの展開過程である。

Inverse Quantization (**逆量子化**)

In[105]:=

```
FImgListY3 = Round[FImgListY2 * (qty / s)];
FImgListCb3 = Round[FImgListCb2 * (qtcbcr / s)];
FImgListCr3 = Round[FImgListCr2 * (qtcbcr / s)];

Print[FImgListY3 // MatrixForm];
Print[FImgListCb3 // MatrixForm];
Print[FImgListCr3 // MatrixForm];
```

$$\begin{pmatrix} 448 & -110 & 20 & -160 & 120 & -80 & 51 & 0 \\ 0 & 24 & -238 & -114 & -26 & -174 & 60 & -110 \\ 336 & 65 & 32 & 168 & -200 & -114 & 0 & -112 \\ 168 & 17 & 22 & 0 & 153 & 261 & -80 & 124 \\ -72 & -22 & 148 & 56 & 0 & 0 & 0 & 154 \\ -168 & -35 & 165 & 64 & 0 & 0 & 0 & -184 \\ 147 & 0 & -78 & 0 & 0 & 0 & 0 & 202 \\ 216 & 0 & 0 & 0 & 0 & 100 & 0 & -99 \end{pmatrix}$$

$$\begin{pmatrix} 1020 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\begin{pmatrix} 1020 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

IDCT (Inverse Discrete Cosine Transform; **逆離散コサイン変換**)

In[112]:=

```
IDCT[lis_] := Module[{c, f, nn},
   c[p_] := If[p == 0, 1/Sqrt[2], 1];
   nn = Length[lis];
   f[m_, n_] := 2/nn * Sum[c[u] * c[v] * lis[[u + 1, v + 1]] *
       Cos[(2 * m + 1) * u * π/(2 * nn)] * Cos[(2 * n + 1) * v * π/(2 * nn)], {u, 0, nn - 1}, {v, 0, nn - 1}];
   Table[f[u, v], {u, 0, nn - 1}, {v, 0, nn - 1}]
  ];
```

In[116]:=

```
IFFImgListY = IDCT[FImgListY3];
IFFImgListCb = IDCT[FImgListCb3];
IFFImgListCr = IDCT[FImgListCr3];
```

(Y, Cb, Cr)→(R,G,B)

(Y, Cb, Cr)→(R,G,B)の式の導出（教育用）

In[119]:=
```
Clear[y, cb, cr, r, g, b];
Solve[{y == 0.299 * r + 0.587 * g + 0.114 * b,
   cb == 128 - 0.168736 * r - 0.331264 * g + 0.5 * b,
   cr == 128 + 0.5 * r - 0.418688 * g - 0.081312 * b}, {r, g, b}] // Simplify
```

Out[120]=
$$\{\{r \to -179.456 - 1.21889 \times 10^{-6}\, cb + 1.402\, cr + 1.\, y,\ g \to 135.459 - 0.344136\, cb - 0.714136\, cr + 1.\, y,\ b \to -226.816 + 1.772\, cb + 4.06298 \times 10^{-7}\, cr + 1.\, y\}\}$$

(Y, Cb, Cr)→(R,G,B)
を計算

In[121]:=
```
IFFImgListY = IFFImgListY + 128;  (* Yは128を足して0-255 の範囲にする *)
ImgList2 = Table[{IFFImgListY[[i, j]],
    IFFImgListCb[[i, j]], IFFImgListCr[[i, j]]}, {i, 1, nx}, {j, 1, ny}];
```

In[124]:=
```
ImgListR2 = Map[(RGBFromYCbCr[#][[1]] &), ImgList2, {2}] // Round;
(* Rの2次元リスト *)
ImgListG2 = Map[(RGBFromYCbCr[#][[2]] &), ImgList2, {2}] // Round;
(* Gの2次元リスト *)
ImgListB2 = Map[(RGBFromYCbCr[#][[3]] &), ImgList2, {2}] // Round;
(* Bの2次元リスト *)
```

画像を表示

In[125]:=

```
Show[MakeRGBRasterImage[ImgListR2, ImgListG2, ImgListB2],
  AspectRatio → Automatic];
```