コンピュータリテラシ＃11 – グラフィクス/図と表

久野 靖 (電気通信大学)

2017.5.19

1　今回の目標

今回の目標は次の通りです。

- 画像の基本概念、ピクセル画像とベクター画像について理解する— 画像は実験の写真や図などの形で多数使うので知っておくことで後で有効に作業できます。
- PPM、EncapsulatedPostScript などのファイル形式について学ぶ— 画像ファイル形式は多数ありますがピクセルとベクターの例を 1 つずつ詳しく知っておくと他に応用できます。
- LaTeX 文書における図や表の扱いを理解する — レポートに図や表を入れられることは必須のスキルです。

2　ピクセルグラフィクス

2.1　画像の表現 exam

インターネット上で最も多く使われている情報の形式はテキスト情報ですが、2 番目に多いのは**画像** (イメージ) ですね。画像の存在しないネットはほとんど考えられないと思います。ここでは画像とその表現について取り上げます。

画像を入力する装置の代表は**デジタルカメラ**や**スキャナ**、画像を出力する装置の代表は**ディスプレイ**と**プリンタ**ということになるでしょう。ではデジタルカメラやスキャナはどのような形で画像を取り込み、ディスプレイやプリンタはそれをどのようにして復元しているのでしょうか。

まず**モノクロ**画像から考えます。画像は平面的な広がりを持つ、空間的に連続したものです。これをデジタル化して取り込むには、まず平面を縦横のます目に十分細かく区切ります。続いて、それぞれのます目の明るさを電圧や電流に変換する素子を使って測り、その明るさを決まった範囲の整数値として取り込みます (**AD 変換** — Analog-Digital 変換)。つまり、画像は縦横に並んだ多数の点の集まりであり、それぞれの点ごとに、ます目の範囲内の元画像の明るさを**サンプリング** (sampling、計測し代表値を取ること) および**量子化** (段階にあてはめ何番目かの値にすること) した値を持つわけです。この「点」のことを**ピクセル** (pixel) と呼びます (図 1)。

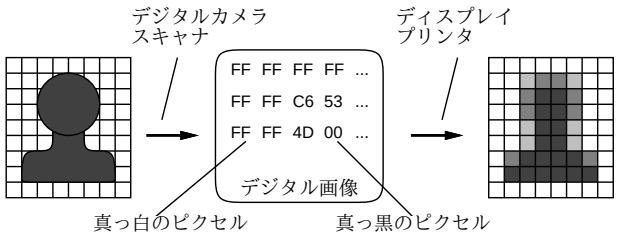

図 1: デジタル画像の原理

ディスプレイはこの点の集まりを画面に表示し、それぞれのピクセルごとにその値に応じた明るさ/暗さで光らせます。また、プリンタは紙の上にそれぞれの点の集まりを色素を使って定着させますが、ピクセルごとにその値に応じた量の色素を定着させます。そこで、人間がこれらを見ると元の画像 (をデジタル化して復元したもの) が見られるわけです。

カラー画像の場合も画像がピクセルの集まりであることは同じですが、色の情報を取り込むために、各ピクセルごとに赤 (Red)、緑 (Green)、青 (Blue) の光の 3 原色のフィルタを通して 3 つのサンプル値を取り込みます。

つまり、各ピクセルは 3 つの値の組 (**RGB 値**) として表現します。広く使われるのは、RGB 値の各色ごとに 8 ビット (0〜255 の値) を使い、1 ピクセルあたり 24 ビット (3 バイト) でカラー画像を表す方法です。これを **24 ビットカラー**といいます。カラーディスプレイは各ピクセルごとに RGB の 3 つの光る点を制御し、それぞれを RGB 値に応じた明るさで光らせますし、プリンタは各ピクセルごとに 3 色 [1] の色素を配合して各ピクセルが RGB 値に対応する色になるよう制御します。

2.2 ピクセルグラフィクスの得失 exam

画像は多数のピクセルの集まりですから、画像を加工することは、個々のピクセルごとにその RGB 値を変化させてやることで行えます。また、マウスやタブレットペンなどを使って「お絵描き」をする場合も、描画範囲上でマウスポインタがなぞった部分のピクセルの色を変化させることで「インク」のようにそこの部分の色を変えることができます。

このように、画像を構成するピクセルを直接取り扱うようなグラフィクスの方式を一般に**ピクセルグラフィクス**と呼びます。ピクセルグラフィクスに基づく作画ソフトのことを**ペイントソフト** (比較的簡単な絵を描く場合に使う) や**フォトレタッチソフト** (写真などの細密な絵を加工するような場合に使う) と呼びます。代表的なフォトレタッチソフトとしては Photoshop、Gimp などがあります。

ピクセルグラフィクスに基づく処理には、次のような利点があります。

○ 画面の表示能力に見合う細かさと色数のデータならその画面で表せるすべての画像が表現可能。

○ 「ぼかし」「にじみ」などの効果が使え、中間的な色合いや独特のタッチを持った絵が作れる。

その一方で、次のような弱点もあります。

△ ピクセル数を大きくすると (これは絵を大きくする場合だけでなく、点の取り方を細かくする場合も含まれる)、ファイルも巨大になりやすい。

△ 描いた絵を拡大したり回転するなどの加工に弱く、大きくするとぎざぎざが目立ったりする。

△ ある場所に絵を「描いてしまう」と、そこのピクセルの色を設定してしまうので、後から消したり動かしたりが難しい。(消しゴムで消すというのは結局「背景で塗っている」のと同じ。そして動かすと後が「空白」になる。) [2]

最後の弱点に対しては、画像を複数の**レイヤ** (層) に分けて扱うことである程度対処可能です。レイヤ機能を持つソフトでは、透明なシート (レイヤ) を複数使ってそれぞれに絵を描き、それらを全て重ねて眺めたものが最終的な絵になります。重ねる順番や位置などは描いた後でも変更できますし、描き損なった場合はそのレイヤだけ消してやり直せば済みます。

ここまではソフトの機能を中心に説明してきましたが、作成した画像は最終的にはファイルに出力します。ピクセルグラフィクスのファイル形式は多数あり、またソフトごとにそのソフトで扱いやすい独自形式を持ったりしますが、共通に使われる代表的なものを挙げておきます。

- **BMP**(Windows Bitmap) — Windows 固有の、圧縮のないピクセル画像ファイル。あまり使われることはない。

[1] 正確には、印刷の場合は「真っ黒」を表現するため 4 色目として黒の色素を持たせます。また 3 色も印刷の性質上、シアン、マゼンタ、イエロー (色の三原色) の組合せを使います。

[2] ただし、間違って塗った場合には、元の状態を記憶しておくことである程度戻すことは可能。大量には難しい。

- **PBM**(Portable Bitmap) — Unix 文化で普及している、圧縮のないピクセル画像ファイル。形式が簡単なので後の実習で使う。
- **GIF** WWW で最初に使われた圧縮のある画像ファイル形式。色数が最大 256 色という制約があるが、アイコン等には使いやすい。
- **JPEG** — WWW で GIF に続いて普及した、写真などの画像に適した、損失のある圧縮を主に用いる画像形式。[3]
- **PNG** — WWW で最も新しく普及した形式。GIF の 256 色という制約をなくし、JPEG と異り損失のない圧縮を用いている。

Web で画像ファイルを使う場合は、最後の 3 つ (GIF、JPEG、PNG) のどれかの形式を使います (どのブラウザでも対応しているため)。

2.3 PBM 画像を作成する exam

PBM 形式の画像には「テキスト形式」つまり普通の文字だけで記述する形式が用意されています。ここで画像の原理を実感していただくために、テキストエディタでこの形式を打ち込んでみます。最初は一番簡単な **2 値画像** (モノクロで真っ白と真っ黒しかない) を作ってみます。sol 上で Emacs を起動し、`t1.pbm` という名前で次の内容から成るファイルを作成し保存してください。

```
P1 6 6          ←P1 は「2 値画像」、6 6 は幅と高さ
1 1 0 0 1 1     ←ピクセルの値 (1 か 0) が 36 個必要
1 1 0 0 1 1     ←値の間には空白か改行が必要
0 0 0 0 0 0
0 0 0 0 0 0
1 1 0 0 1 1
1 1 0 0 1 1
```

次に「`gimp t1.pbm &`」により Gimp を起動します。窓が複数開きますが、小さい画像が表示されている窓を選び、拡大率を最大にしてください。図 2 のように確かに白黒の画像になっていることが分かります。

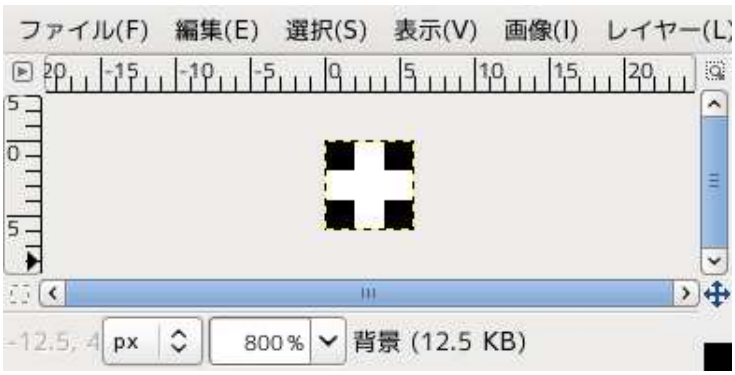

図 2: 2 値画像を表示する

しかし色がないとつまらないですね。そこで次はカラー画像にしてみましょう。ファイル名としてこんどは「`t2.ppm`」を指定し、次の内容を打ち込んでください。こんどは RGB で指定するので数値の数が 1 ピクセルあたり 3 個になります (全部で幅×高さ× 3 個の数値が必要)。RGB の値は 0〜255 の範囲で指定します。

[3]損失のある圧縮とは、圧縮したものを展開したときに完全に元のデータとは一致しないような圧縮方式をいう。その代わり、圧縮率を高くしやすい。JPEG では圧縮率を指定することで「ファイルサイズが小さいが品質が落ちる」「ファイルサイズが大きいが品質が高い」などの制御ができる。

```
P3  6  6  255   ← P3 はカラー画像、255 は RGB 値の最大数
0  0  255  0  0  255  0  0  255  0  0  255  255  0  0  255  0  0
0  0  255  0  0  255  0  0  255  0  0  255  255  0  0  255  0  0
0  0  255  0  0  255  0  0  255  0  0  255  255  0  0  255  0  0
0  0  255  0  0  255  0  0  255  0  0  255  255  0  0  255  0  0
0  0  255  0  0  255  0  0  255  0  0  255  255  0  0  255  0  0
0  0  255  0  0  255  0  0  255  0  0  255  255  0  0  255  0  0
```

こんどは Gimp で開いてみると、色がついている。元データを見ると分かるように、左側の 4 ピクセルは (0,0,255) つまり真っ青、右側の 2 ピクセルは (255,0,0) つまり真っ赤になっている (左から RGB の順なので)。

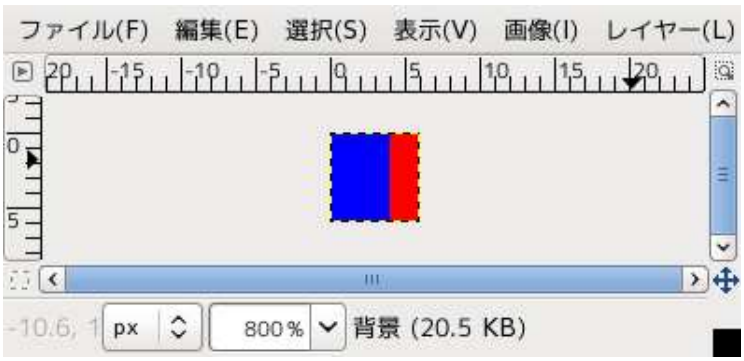

図 3: カラー画像を表示する

演習 1 上の 2 つの例を打ち込んで Gimp(フリーソフトなので Windows 用や Mac 用も無料で入手可能) で表示してみなさい。うまくいったら、次の演習をやってみなさい。

a. 白黒でもカラーでもよいので、もう少し大きな画像に挑戦してみる (PBM のテキスト形式では数値は並んでいる順だけに意味があるので、扱いやすくするために好きなところで行をかえてよい)。ただし、最初にどのような画像にするか紙にスケッチし、その形を再現するように作ること。

b. 「色が徐々に変わって行く」「図形のふちがぼけている」「半透明な図形が重なっている」などの効果から 1 つ選び、カラー画像として作成してみよ。ただし、最初にどのような画像にするか紙にスケッチし、その形を再現するように作ること。

c. (自由課題) テキスト形式の白黒またはカラー画像を作るやり方で、自分の好きな画像を作ってみなさい。どのような画像であるかはあなたに任されるが、例題で示したものよりは「美しい」ことが期待される。エディタで打ち込む以外に、プログラムを書いて出力するなどの方法を用いてもよい。

3 ベクターグラフィクス

3.1 ベクターグラフィクスとその特徴 exam

ピクセルグラフィクスとは全く違う絵の表し方として、図形などの位置や輪郭を数値的/数式的に覚えておき、絵が必要になる瞬間にその式に応じて絵を生成して表示するという方式があります。このようなモデルを (位置、方向などの「ベクトル」を用いて絵を表すことから) **ベクターグラフィクス**と呼びます (図 4)。ベクターグラフィクスでは、絵は円、直線、矩形などの比較的単純な図形の集

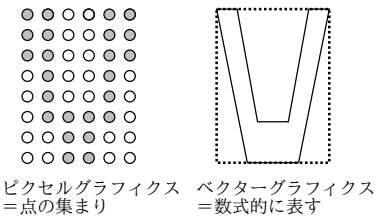

図 4: ベクターグラフィクスとピクセルグラフィクス

まりで表すのが普通ですが、高度なソフトになると 3 次曲線、ベジェ曲線などの数式に基づく曲線を活用してもっと柔軟な形を取り扱うこともできます。

ベクターグラフィクスで絵を描くのは、「無限に伸び縮み可能な針金で作った図形にスクリーントーンを貼って好きな順に重ねて行く」ようなものだと思えばよいでしょう。針金ですから、あとで自由に置き場所や大きさを調整することができるわけです。ベクターグラフィクスに基づく作画ソフトで代表的なものは Illustrator ですが、より簡便なものはドローソフト、ドローツールと呼ばれます。たとえば PowerPoint 等の中の作画機能はドローツールになっています。これは、いちど描いた図を後で動かしたりしたいのでドロー系が使われているわけです。

ベクターグラフィクスの得失はだいたいピクセルグラフィクスの裏返しと考えればよいでしょう。

- ○ 図形の拡大・縮小・回転・重なり順の変更などは単にその変更に基づいて絵を表示し直すだけなのでいくらでも自由に行なえる。
- ○ 絵は数式的に表されているので、拡大してもぎざぎざになることはない。
- ○ 絵の情報は座標や形などの情報なので、ファイルの大きさは小さくて済むし、拡大/縮小してもファイルサイズは変わらない。
- △ 絵の細かさはソフトに用意されている階調機能や模様機能などで決まってしまい、細かい色合いは使いにくい。
- △ ぼかし、にじみなどの効果は使えない。[4]

ベクターグラフィクスのファイル形式としては、**PostScript** があります (TeX の出力に使いましたね)。PostScript はもともとはプリンタで出力するページを記述するための言語であり、手で書くこともできます。現在広く使われている **PDF**(Portable Document Format) は PostScript の技術を土台に、よりコンパクトになるように設計されていて、印刷形式の文書を配布するのに有用です。このほか、WWW などのページ内容としてベクターグラフィクスを記述できるように設計された言語として **SVG**(Scalable Vector Graphics) があります。

3.2 PostScript — ベクターグラフィクス記述言語

前述のように、PostScript はベクターグラフィクスのファイル形式ですが、正確には「言語」でもあります。今回は、PostScript ファイルを次のような形で作ります。長さの単位はすべて pt(ポイント、1pt = $\frac{1}{72}$inch) です。

```
%!PS-Adobe-2.0              ← PostScript であることを表す
%%BoundingBox: 0 0 400 300 ←絵の範囲を表す。
                                (今回は 400x300 にした。)
…(ここにさまざまな図形記述を入れる)…
showpage      ←プリンタに送った場合にページを出力する
```

線を引くには次のコマンドを使います。

[4]図形を塗りつぶすときに、階調 (グラデーション) や模様 (テクスチャ) などを使うことはできます。

- `newpath` — 新しい線引きを開始する。
- X Y `moveto` — ペンを指定した座標 (X, Y) に移動。
- X Y `lineto` — 座標 (X, Y) までの線を登録しながら移動。
- `stroke` — `lineto` で指定した線引きを一気に実行する。
- W `setlinewidth` — 線の太さを Wpt にする。

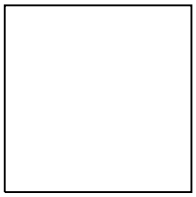

図 5: 正方形を描く

では正方形を描くという簡単な例を挙げておきます (図 5)。これをたとえば sol で `t3.eps` というファイルに Emacs で打ち込み、「`gv t3.eps &`」とすると見ることができます (gv は PostScript を画面で見るためのツールです)。また、PostScript プリンタに送るとそのままプリントされます。

```
%!PS-Adobe-2.0
%%BoundingBox: 0 0 400 400
newpath 100 100 moveto 100 200 lineto
200 200 lineto 200 100 lineto 100 100 lineto stroke
showpage
```

ところで、PostScript は言語だと書いたのはどういうことでしょう? それは、たとえば「ループ」を使って先の正方形を繰り返し描いたりできます。同じ場所に描いてもつまらないので、次のコマンドも知っておいてください。

- T_x T_y `translate` — 絵を描くときの原点を X 軸方向に T_x、Y 軸方向に T_y だけずらす。
- S_x S_y `scale` — 絵を描くときの大きさを X 方向に S_x 倍、Y 方向に S_y 倍する。
- D `rotate` — 絵を描くときの角度を原点のまわりに (反時計回りに)D 度回転する。

繰り返し自体は次のようにします。

- N { 動作列 } `repeat` — 「動作列」を N 回繰り返す。

では、これを使って正方形を 5 回ずらして描いてみます (図 6)。

```
%!PS-Adobe-2.0
%%BoundingBox: 0 0 400 400
5 {
  newpath 100 100 moveto 100 200 lineto
  200 200 lineto 200 100 lineto 100 100 lineto stroke
  30 -15 translate
} repeat
showpage
```

最後に文字について説明しましょう。文字の表示には、次の 2 つのステップが必要です。

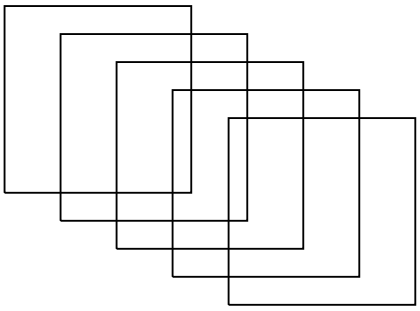

図 6: ループで正方形を描く

- /フォント名 `findfont` S `scalefont setfont` — 指定した名前のフォントを探してきて、サイズ (ポイント数)S に設定し、使用するフォントとしてセットする。
- X Y `moveto`（文字列）`show` — 指定位置 (X, Y) に文字列を表示する。

一度フォントをセットしたら、別のものに変更しないで繰り返し `show` を使って構いません。文字列を丸かっこで囲んで指定するというのはちょっと変わっていますね。もし文字列の中に丸かっこを入れたければ、「`\(`」「`\)`」のようにバックスラッシュを前につけてください。

図 7: さまざまなフォント

例 (図 7) では「灰色の濃さ」や「色」を変更しているので、その説明もしておきます。

- G `setgray` — 灰色の明るさ (0.0〜1.0、大きいほど明るい) を設定。
- R G B `setrgbcolor` — 色を RGB で指定 (どれも 0.0〜1.0、大きいほどその色が強い)。

```
%!PS-Adobe-2.0
%%BoundingBox: 0 0 400 300
/Helvetica findfont 24 scalefont setfont
20 20 moveto (This is Helvetica 24pt.) show
/Times-Roman findfont 36 scalefont setfont
20 60 moveto (Times\(-Roman\) 36pt.) show
/Courier-Bold findfont 12 scalefont setfont
20 110 moveto (This is Courier-Bold 12pt.) show
0.5 setgray
```

```
/Ryumin-Light-EUC-H findfont 40 scalefont setfont
20 140 moveto（これは明朝体です。）show
5 rotate 0.8 0.3 0.2 setrgbcolor
/GothicBBB-Medium-EUC-H findfont 48 scalefont setfont
20 180 moveto（ゴシック体です。）show
showpage
```

なお、日本語については必ず EUC コードを使用してください。[5]Emacs であれば「Ctrl-X [RET] f euc-jp [RET]」でファイルの文字コードを EUC に設定できます。

ところで、PostScript のフォントは**アウトラインフォント**、つまり数式で輪郭を表現したフォントを使っていることが特徴で、このため任意のポイント数に大きさを設定できます。今日では他のソフトでもこれが普通ですが、PostScript がこれを始める前は、フォントはすべて「ある大きさで」デザインされていたので、自由にさまざまなサイズを指定することはできませんでした。実際にさまざまな大きさにフォントを変更してみてください。

演習 2 PostScript の例題を打ち込んで動かしてみなさい。様子が分かったら、以下の小課題から 1 つ以上やってみなさい。

a. 単純な図形を繰り返し表示させて模様を作りなさい (拡大縮小や回転も使うとよい)。
b. 複数サイズの文字や線を組み合わせて自分の名刺を作ってみなさい。
c. (自由課題) 自分が描きたいと思う好きなものを描いてみなさい。絵の内容についてはあなたに任されますが、例題よりは「美しい」ことが期待されます。

4 LaTeX 文書における図や表の扱い

4.1 LaTeX 文書に画像を入れる exam

画像が作れるようになったので、これを LaTeX 文書に入れる方法を説明しておきます。

1. TeX に取り込むには、ファイル形式は **EPS** 形式 (Encapsulated — PostScript `%%BoundingBox:` の指定された PostScript) であることが必要です。[6] 先に作った PostScript の例題は既にそのようにしてありましたね。それ以外の形式のファイルであれば、次のようにして変換できます (または Gimp の機能として画像を EPS で保存もできます)。

   ```
   convert test.ppm fig1.eps
   ```

2. LaTeX では`\documentclass` と`\begin{document}`の間に次のような行を追加します。

   ```
   \usepackage{graphicx}
   ```

   そして本文中の図を入れたい場所に次のようなコマンドを入れます。

   ```
   \begin{center}
   \includegraphics[scale=0.5]{fig1.eps}
   \end{center}
   ```

   見て分かる通り、「0.5」は縮小比率、「`fig1.eps`」は PS ファイル名です。scale の代わりに `width=8cm` のように出来あがりの幅を指定もできます (中央そろえは必須ではないです)。

あとはこれまで通りに `platex` で整形します。`platex` コマンド使用時だけでなく、`pxdvi` や `dvipdfmx` を使うときも、画像ファイルが同じ場所に置いてある必要があります。

[5]日本語フォントとして EUC フォントを指定しているためです。
[6]最近は 1 ページの PDF が取り込める LaTeX 整形系も増えています。

4.2 図や表の扱い exam

ここまでで LaTeX の文書に図や表が入れられることは分かりましたが、この資料などを見ていただくと分かるように、実際の文書では図や表の扱いが少し違いますね? 具体的には次のようになっています。

(1) 図や表の位置は「ページの上端」とか図表が多いときは「別のページ」など、本文とは別に配置されていることが多い。

(2) 図や表にはタイトル (**キャプション**) と図表番号がついていて、本文では「図 1」などのようにその番号で参照される。

ここではこれらの扱いについて説明しましょう。[7]

上記 (1) や (2) のためには、図や表のここまでに述べた「本体部分」を figure 環境、table 環境で囲み、caption コマンドでキャプションをつけます。全体がどのような感じになるかを見て頂きましょう。

```
\begin{figure}[htbp]
\begin{center}
\includegraphics[scale=0.7]{FIGS/c7-ps01.eps}
\end{center}
\caption{正方形を描く}\label{fig-rectangle}
\end{figure}

\begin{table}[htbp]
\caption{AND 演算の結果}\label{tbl-bitwiseand}
\begin{center}
\begin{tabular}{c|ll}
AND 演算 & 0 & 1 \\
\hline
0       & 0 & 0 \\
1       & 0 & 1 \\
\end{tabular}
\end{center}
\end{table}
```

\begin{table}や\begin{figure}の後の [htbp] とは何でしょうか? それは、LaTeX に「この図や表は次の方針で配置してください」という希望を指定するものです。

- h(here) — なるべくこの table 環境や figure 環境を置いた場所の近くに配置してほしい
- t(top)/b(bottom) — なるべく現在のページの一番上/一番下に配置してほしい
- p(page) — 本文とは別に図表のページを作ってそこに集めてほしい

これを「希望する順」に指定しているので、上の例だと「できるだけここに、だめならページの上、だめならページの下、それでもだめなら別ページ」となります。実際にやってみると希望を述べても思い通りにならないことが多いのですが…

次にキャプションは表では表の「上に」、図では図の「下に」配置するのが通例ですので、そのようにしてください。さて、キャプションの後ろに\label{fig-rectangle}などとあるのは何でしょうか。それは次の節でまとめて紹介します。

[7] とくに (1) のように、位置が本文とは別に配置されるもののことを、LaTeX では**フロート** (float) と呼びます。

4.3 ラベルと参照 exam

図や表は「図 1」のように番号で参照すると述べましたが、それを手で管理するのは大変です。文書を改訂していると図表が増えたり位置を入れ換えたりとかはよくあるので。

そこで、上で示したように図や表の箇所で「好きな名前」を指定した label コマンドを置きます。そのうえで、参照する箇所には

```
…図\ref{fig-rectangle}や表\ref{tbl-btiand}のように…
```

のように、ref コマンドを使って参照を指定します。すると、この ref コマンド全体が「1」「2」のように図や表の番号に置き換わって整形されます。それは LaTeX にとっては簡単なことで、整形しているときに順番に出て来る図や表に連番を割り当てていけばよいだけですから。

ただ 1 つ注意する点があります。それは「下の方にある図表の番号を手前の方で参照する」ときは、最初に整形した時には (まだその図表まで到達していないので)LaTeX には番号が分からないことです。そのときは ref コマンドのところには「?」が現れますので、「もう 1 回」整形を実行してください。図表番号の情報は 1 回実行するごとに .aux ファイルに書き出され、次の処理でこれを読むため、2 回やれば先の方の図表番号も正しく分かります。ただし、図表の入れ換えなどがあると番号が変わりますが、そのときは「変わってからあとにもう 1 回」整形する必要があります。

そのほか「undefined reference」というメッセージが出ることもありますが、それは「ref コマンドで指定した名前のものが見付からない」で、だいたいはタイプミスによります。対応する label コマンドと同じ名前になっているか確認してください。

なお、label コマンドを置くのは図表だけに限定されません。節や章の番号も参照したいことがありますが、そのために次のように section などの箇所に label を指定しておくとよいわけです。

```
\setion{ラベルと参照}\label{sec-labelref}
```

こうしておけば「第\ref{sec-labelref}節を見てください」のようにして ref で参照できるわけです。

4.4 文献の参照

文献もやはり、番号などで参照し、その管理がややこしいものの 1 つです。文献については LaTeX まわりに様々なツールがあるのですが、ここでは一番簡単なものを説明しておきます。

```
\begin{thebibliography}{99}
\bibitem{dolbook} 兼宗，久野，ドリトルで学ぶプログラミング、
イーテキスト研究所，2008.
\bibitem{wing} Wing，Jannet M.:~Computational Thinking，
CACM，vol.~49，no.~3，pp.~33-35，2006.
\end{thebibliography}
```

このように (通常文書の末尾に)thebibliography 環境を置き、その中で bibitem コマンドでラベルを指定した項目を区分すると、文献に連番が振られます。参照するときは文献は「\cite{dolbook}」のように cite コマンドを使用します。するとその箇所に対応するラベルを持つ bibitem の番号が埋められます。

演習 3 図や表を 1 つ以上含んだ LaTeX 文書を作成しなさい。本文中から図表番号を参照すること。

本日の課題 11A

本日の課題は「演習 3」ですが、その内容として「演習 1」「演習 2」に含まれる小問 (合計で 6 個) から 1 つ以上を選択し、結果をレポートとして報告するものとしてください。これらは図を作る課題なので、それらを LaTeX 文書の図として入れてください。LaTeX による整形結果を PDF ファイルにして、LMS の「レポート＃ 11」の箇所からアップロードしてください。以下の内容がこの順に含まれるようにしてください。

- 題名「コンピュータリテラシレポート＃ 11」、学籍番号と氏名、提出日付を書く。(グループでやったものはグループのメンバー全員の氏名も脚注などで別途書く。)
- 課題の再掲を書く (どんな課題であるかをレポートを読む人が分かる程度に要約する)。
- レポート本体の内容 (やったこととその結果) を書く。
- 考察 (課題をやった結果自分が新たに分かったことや考えたこと) を書く。
- 以下のアンケートに対する回答。

  Q1. ピクセルグラフィクスとベクターグラフィクスについてどれくらい知っていましたか。新たに知って面白かったことは何ですか。

  Q2. LaTeX での図表の扱いや参照機能についてどう思いましたか。

  Q3. リフレクション (今回の課題で分かったこと)・感想・要望をどうぞ。

なお、課題はグループでやって構いません。その場合も、(メンバー氏名を明記した上で) レポートは必ず各自で執筆してください。レポート文面が同一 (コピー) と認められた場合は同一であると認めた全員について点数にペナルティを科すことがあります。