ARTICLE

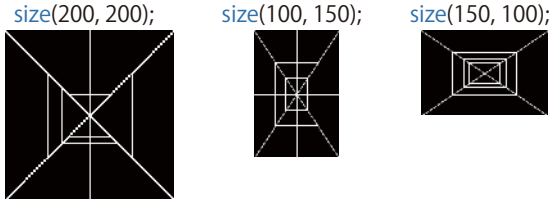
# Processing でプログラミングに挑戦！ —第 3 回 繰り返しを使いこなそう—

杉浦　学
鎌倉女子大学

## 宿題の解説

前号では，ウィンドウのサイズに合わせて形が変化する，クモの巣の模様(図 -1)を描く宿題を出題しました．

### ● 前号の宿題




四角形は小さい順に，画面の縦・横の長さのそれぞれの四分の一，三分の一，二分の一になるように割り算で計算します

図 -1　クモの巣の模様

この宿題の解答例をスケッチ 1 に示します．width と height の変数と，計算を組み合わせることで，2 行目の size 関数で指定するウィンドウのサイズに合わせてクモの巣の形が変形します．

```
1  //描画の準備
2  size(200,200);
3  background(0);
4  stroke(255);
5
6  //直線を描く
7  line(0,0,width,height);
8  line(width,0,0,height);
9  line(0,height/2,width,height/2);
10 line(width/2,0,width/2,height);
11
12 //四角形を描く
13 rectMode(CENTER);
14 noFill();
15 rect(width/2,height/2,width/2,height/2);
16 rect(width/2,height/2,width/3,height/3);
17 rect(width/2,height/2,width/4,height/4);
```

スケッチ 1　クモの巣を描く（宿題の解答例）

## 繰り返しのパターンを見つける

変数と計算を使って，繰り返している処理を同じ形に変形できれば，それらの処理を効率良く記述することができます．まずは複数の斜線を描くスケッチ 2 を見てみましょう．

```
1 size(480,120);
2 strokeWeight(8);
3 line(20,40,80,80);
4 line(80,40,140,80);
5 line(140,40,200,80);
6 line(200,40,260,80);
7 line(260,40,320,80);
8 line(320,40,380,80);
9 line(380,40,440,80);
```

スケッチ 2　複数の斜線を描くスケッチ（文献 1）P.45 より）

スケッチ 2 を実行すると，図 -2 のような模様が表示されます．斜線の始点と終点の x 座標を 60 ずつ増やし（右にずらしながら），斜線を合計で 7 本描いています．

図 -2　複数の線を描く

スケッチ2に変数を導入したものがスケッチ3です．

```
1  size(480,120);
2  strokeWeight(8);
3
4  int x=20;
5
6  line(x,40,x+60,80);
7  x+=60;
8
9  line(x,40,x+60,80);
10 x+=60;
11
12 line(x,40,x+60,80);
13 x+=60;
14
15 line(x,40,x+60,80);
16 x+=60;
17
18 line(x,40,x+60,80);
19 x+=60;
20
21 line(x,40,x+60,80);
22 x+=60;
23
24 line(x,40,x+60,80);
25 x+=60;
```

スケッチ3　変数を導入した斜線を描くスケッチ

スケッチ3では，斜線の始点のx座標を変数xに保存し，線を1本描くごとに60だけ増やしています．変数xの値を現在の値より60増やしたい場合は，x=x+60; と書くところを短く x+=60; と書くことができます．終点のx座標は始点のx座標のxに60を加えた「x+60」になります．変数を導入するとスケッチの行数は増えてしまいましたが，斜線を1本描く処理に注目してみると，まったく同じ処理が7回登場していることを発見できました（図-3）．なお25行目は省略しても描画の結果は同じになります．

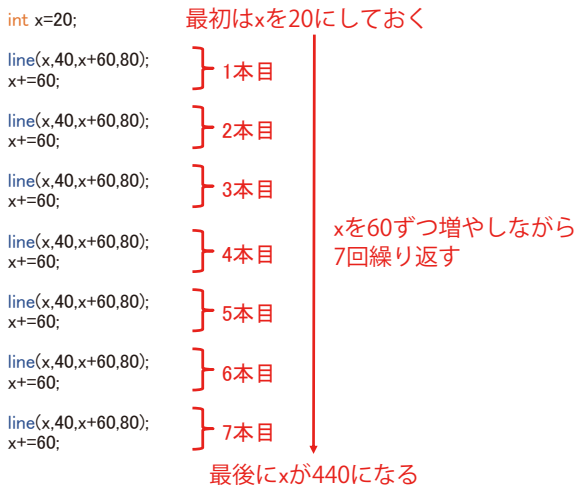



図-3　同じ処理の繰り返し

## forループによる繰り返し

図-3に示したように，最初に変数xを宣言して20を代入したら，xに60を加えながら，line(x,40,x+60,80) ; というまったく同じ処理を7回繰り返せばよいことになります．こうした繰り返し処理を記述するには「forループ」を使います．forループを使ってスケッチ3を書き直したものがスケッチ4です．

```
1 size(480,120);
2 strokeWeight(8);
3 for(int x=20; x<440; x+=60){
4   line(x,40,x+60,80);
5 }
```

スケッチ4　forループを使って斜線を描くスケッチ

forループの文法を整理しておきます（図-4）．繰り返しをしたい処理は「{」と「}」で囲みますが，囲まれている処理が分かりやすいように段下げ（インデント）をするようにしましょう．編集メニューから自動フォーマットを選択すれば，自動でインデントをすることができます．forの後の括弧には，図-4に示したように3種類の処理を記述します．慣れないうちは，最後の処理（この場合は x+=60）に「;」をつけてしまうことが多いので注意してください．

青：繰り返しを始める前に一回だけする処理
緑：繰り返しを継続する条件
赤：繰り返しが一回終わるたびにする処理

```
for (int x=20;  x<440;  x+=60) {
  line(x,40,x+60,80);
}
```

繰り返したい処理を{と}で挟む

図 -4　for ループの文法

繰り返しの継続に関する条件を記述するためには，比較演算子（表 -1）という記号を利用します．繰り返しが一度終わるたびにこの条件を調べ，条件が成立する場合は繰り返しを継続します．今回は「x が 440 より小さい」ときに繰り返しを継続するので，合計で 7 回繰り返しが行われます（8 回目で x が 440 になるので，8 回目の繰り返しは行われません）．

表 -1　代表的な比較演算子

| | |
|---|---|
| > | より大きい |
| < | より小さい |
| >= | 以上 |
| <= | 以下 |
| == | 等しい |
| != | 等しくない |

プログラミングに慣れてくれば，スケッチ 2 のような繰り返しの処理をすぐに for ループを使って記述することもできるでしょう．慣れないうちはまずは for ループを使わないで描いてみて，繰り返しのパターンを見つけてから，for ループに書き換えるという手順を踏むとうまくいきます．

### ❑ 練習問題

前回の宿題の解答例（スケッチ 1）の 3 つの四角形を描いている部分を，for ループを使って書き換えてみましょう．繰り返しの回数を数える変数の名前は i としてください（繰り返しに用いる変数の名前として，i がよく使われます）．また，i の値を 1 だけ増やす処理「i=i+1;」は「i++;」と短く書くことができます．この短い書き方も練習してみましょう．

＜解説＞

for ループを使ったクモの巣のスケッチの例
（該当部分のみ抜粋）

```
//四角形を描く
rectMode(CENTER);
noFill();
for(int i=2;  i<5;  i++){
  rect(width/2,height/2,width/i,height/i);
}
```

## デバッガ

スケッチを書いて実行した際に，1 回で想定どおりの結果が得られる場合もありますが，意図と異なる結果となることも多くあります．そうした場合は不具合（バグ）の原因を突き止めてスケッチを修正（デバッグ）する必要があります．変数や繰り返しを含んだ少し複雑なスケッチを効率良くデバッグするには，「デバッガ」を使うと便利です．デバッガを使うことで，スケッチの実行を一時停止し，変数の値を確認しながら 1 行ずつ実行をすることができます．

デバッガを使うには，エディタの右上にあるボタンをクリックします．同じ操作は「デバッグ」メニューの「デバッガを有効化」からも行えます．ここではスケッチ 4 の変数 x の値がどのように変化するか確認してみましょう．変数の値を確認したり，スケッチの実行の流れを検証したりするために実行を一時停止したい行を「ブレークポイント」と呼びます．行番号の部分をクリックすると，その行をブレークポイントに設定することができます．今回は for ループで繰り返しを行う 4 行目にブレークポイントを設定します（図 -5）．なお，ブレークポイントを解除するには，もう一度その行をクリックします．


図 -5　デバッガの有効化とブレークポイントの設定

デバッガを有効化すると，エディタ上部のボタンが 4 つに増えます．デバッグ（▶）ボタンを押すと，ブレー

クポイントを設定した行の直前で実行が停止します．変数の値を一覧できる小さなウィンドウに表示されている変数 x の値は 20 になっているはずです(図 -6)．

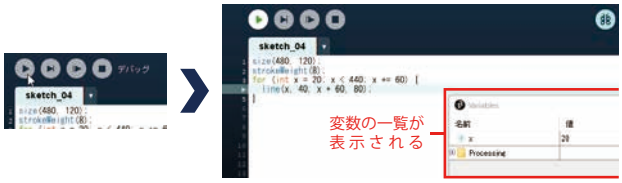



図 -6　デバッグの実行

繰り返しを進めて，変数の値の変化を観察してみましょう．「ステップ(▶|)」ボタンを押すと，1 行ずつ実行することができます．なお「続行 (|▶)」ボタンは，次のブレークポイントの直前まで実行を進めます．今回はステップボタンを押してみましょう．次に実行される行の行番号が▶の印に変わり，行が薄青色になります．今回の場合は，2 回目の繰り返しが実行されるので，3 行目の for ループに▶が表示されます．もう一度ステップボタンを押すと，4 行目に▶が移動して，3 行目の x+=60 が実行された結果が反映されて，変数 x の値が 80 に増えます(図 -7)．

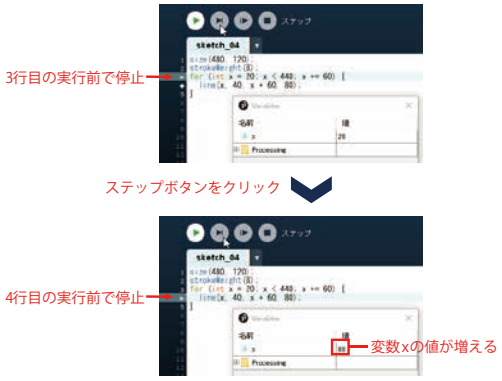


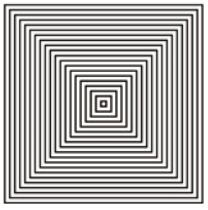

図 -7　ステップ実行

連続してステップボタンをクリックしていくと，3 行目に▶が戻り，4 行目に▶が進むことを繰り返す様子が確認できます．変数 x の値が 60 ずつ増えていくことを確認してみましょう．変数 x は最終的に 440 まで増えて繰り返しが終了しますが，デバッガで確認できるのは繰り返しが終了する直前の値の 380 までです．デバッガによる実行を停止したい場合は，「停止」(■) ボタンをクリックします．

デバッグが終わったら，デバッガを無効化しておきます．デバッガを有効化したときに押したエディタの右上のボタンを押すか，「デバッグ」メニューの「デバッガを無効化」を選択します．

### ❑ 練習問題

以下のお手本を参考に，for ループを使って複数(20 個)の正方形を使ったイラストを描いてみましょう．ウィンドウの大きさなどの描画の準備については，スケッチの例を示しました．正方形の中心の座標は(250, 250)に設定し，一番小さな正方形の 1 辺の長さは 10 ピクセルで，一番大きなものは 390 ピクセルとしてください．それぞれの正方形の大きさは 20 ピクセルの差があります．一番小さなものから描いてもよいですし，大きなものから描いてもよいでしょう．できれば 2 通りの方法を考えてみましょう．うまくいかない場合は，デバッガを使ってみてください．



```
1 //描画の準備
2 size(500,500);
3 background(255);
4 rectMode(CENTER);
5 strokeWeight(3);
6 noFill();
```

ヒント：描画の準備部分のスケッチ

＜解説＞

小さな正方形から描く例(該当部分のみ抜粋)

```
for(int i=10; i<=390; i+=20){
  rect(250,250,i,i);
}
```

大きな正方形から描く例(該当部分のみ抜粋)

```
for(int i=390; i>=10; i-=20){
  rect(250,250,i,i);
}
```

## 乱数

次は for ループによる繰り返しと「乱数」を組み合わせてみましょう．乱数を使うとプログラムが実行されるたびに異なる値を作り出すことができます．random 関数は設定した範囲内の float 型の乱数を作り出します．random 関数のパラメータを 1 つ指定した場合は乱数の上限，2 つ指定した場合は下限と上限を指定します．たとえば，random(256) とした場合は 0 以上 256 未満の乱数が生成されます．

この乱数を使ってスケッチ 4 を改造し，実行されるたびに異なるカラフルな色の斜線を生成するように改造したものがスケッチ 5 です．スケッチ 5 を実行した結果の一例が図 -8 です．

```
1 size(480,120);
2 strokeWeight(8);
3 for(int x=20; x<440; x+=60){
4   stroke(random(256),random(256),random(256));
5   line(x,40,x+60,80);
6 }
```

スケッチ 5　カラフルな斜線を描くスケッチ

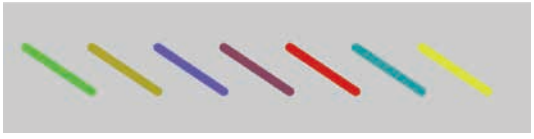


図 -8　カラフルな斜線を描く

斜線を描く前の 4 行目で，stroke 関数の 3 つのパラメータに乱数を入れ込み，異なる線の色を設定しています（図 -9）．これにより，毎回異なる 3 つの数値が色に設定されて，カラフルな斜線が描ける仕組みです．

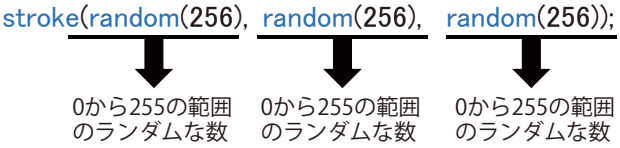



図 -9　カラフルな斜線を描く仕組み

### ● 宿題

スケッチ 5 を応用して，お手本のようなカラフルな円の模様を描くスケッチを書いてみましょう．

それぞれの円の直径は 35 ピクセルです．ヒントのスケッチのコメントの部分（9 行目）に複数の処理を追加して完成させてください．画面の縦横に円を敷き詰めるために，for ループの中に for ループを入れ込んだスケッチになっています．変数 x と変数 y の値を，デバッガを使って調べながら取り組んでみましょう．

```
1  //描画の準備
2  size(480,480);
3  background(0);
4  noStroke();
5
6  //カラフルな円を敷き詰める
7  for(int y=0; y<=height; y+=40){
8    for(int x=0; x<=width; x+=40){
9      //ここに処理を追加して宿題を完成させる
10   }
11 }
```

宿題のヒント


参考文献

1)　Reas, C., Fry, B. 著，船田　巧 訳：Processing をはじめよう 第 2 版，オライリージャパン (2016)．



(2020 年 1 月 4 日受付)



**杉浦　学（正会員）**　manabu@kamakura-u.ac.jp

鎌倉女子大学家政学部家政保健学科准教授．慶應義塾大学大学院政策・メディア研究科後期博士課程修了．博士（政策・メディア）．プログラミング教育をはじめとした情報教育に関する研究に取り組む．中高生向けの著書に『Scratch ではじめよう！ プログラミング入門 Scratch3.0 版』（日経 BP 社）など．