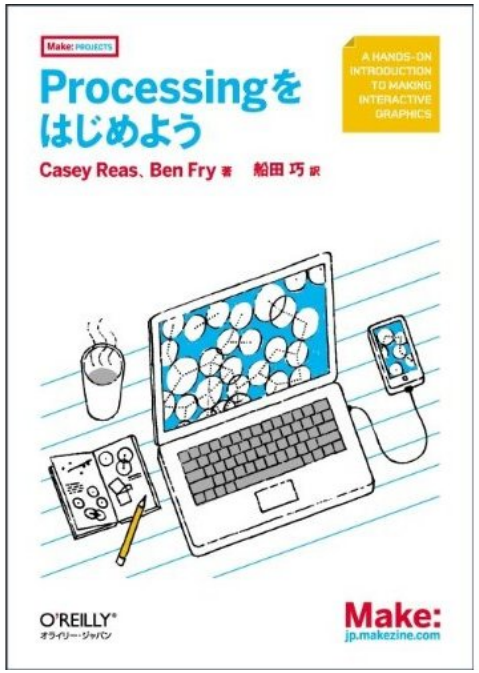
Practice # 2

描画と変数（Processingの基礎）

演習2A　描画の基本



Processingをはじめよう
第二版
（Make: PROJECTS）

オンライン授業　4.21
課題学習　4.28

この本を参考書として使用します（任意）. 以下の資料のなかで、ページ数が書かれてあるものは, 参考書のページに対応しています. 自習復習, 辞書代わりに使用してください.

実行と停止（P13）

Processing を開くと, 自動的にエディタが開きます. ファイルの名前は, デフォルトでは,「sketch_xxxxxxx」となります.（xxは日付）

▼

⌘ + S

で適当なファイル名に変更するとともに, 保存場所を決めます.

▼

「sample2A」として保存すると, 自動的に「sample2A」というフォルダとその内部に「sample2A.pde」という名前のファイルが保存されます. このように, processingのファイルの拡張子は「pde」となります.

▼

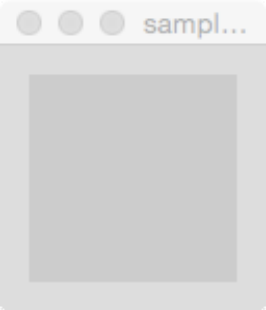
⌘ + R

または, 実行ボタン を押すと,（何もコードを書いていなければ）100 x 100の背景が灰色のウィンドウが表示されます.

▼

⌘ + Q

または, 停止ボタン で終了します.



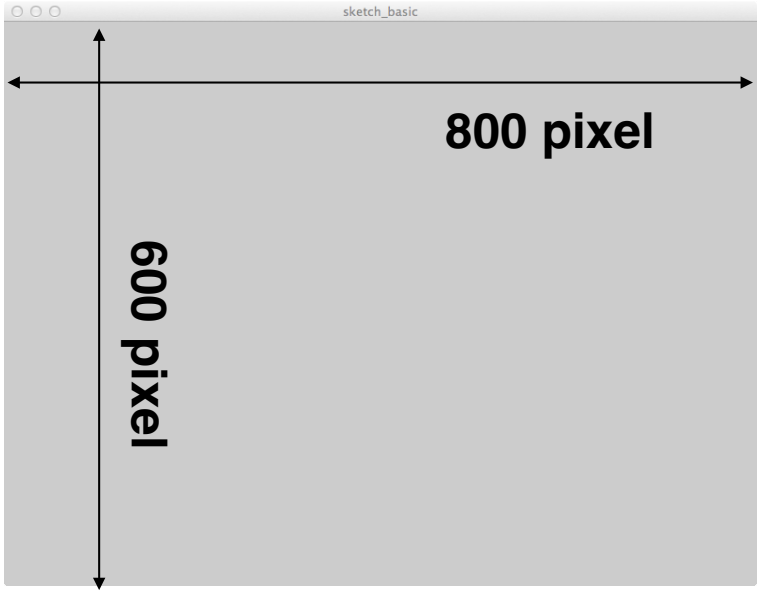
実行アプリケーション

ウィンドウを描く

//800x600のwindowを生成

//デフォルトの背景色はgray

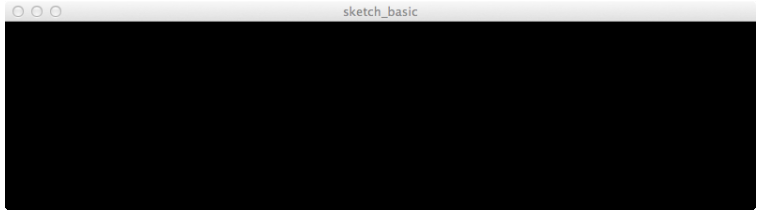
```
size(800,600);
```



//800x200のwindowを生成

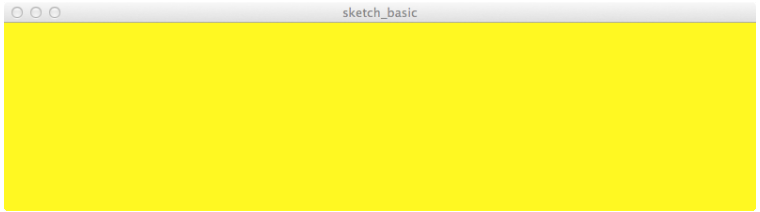
//背景色を黒に

```
size(800,200);
background(0);
```



//背景色を黄色に

```
size(800,200);
background(255,255,0);
```

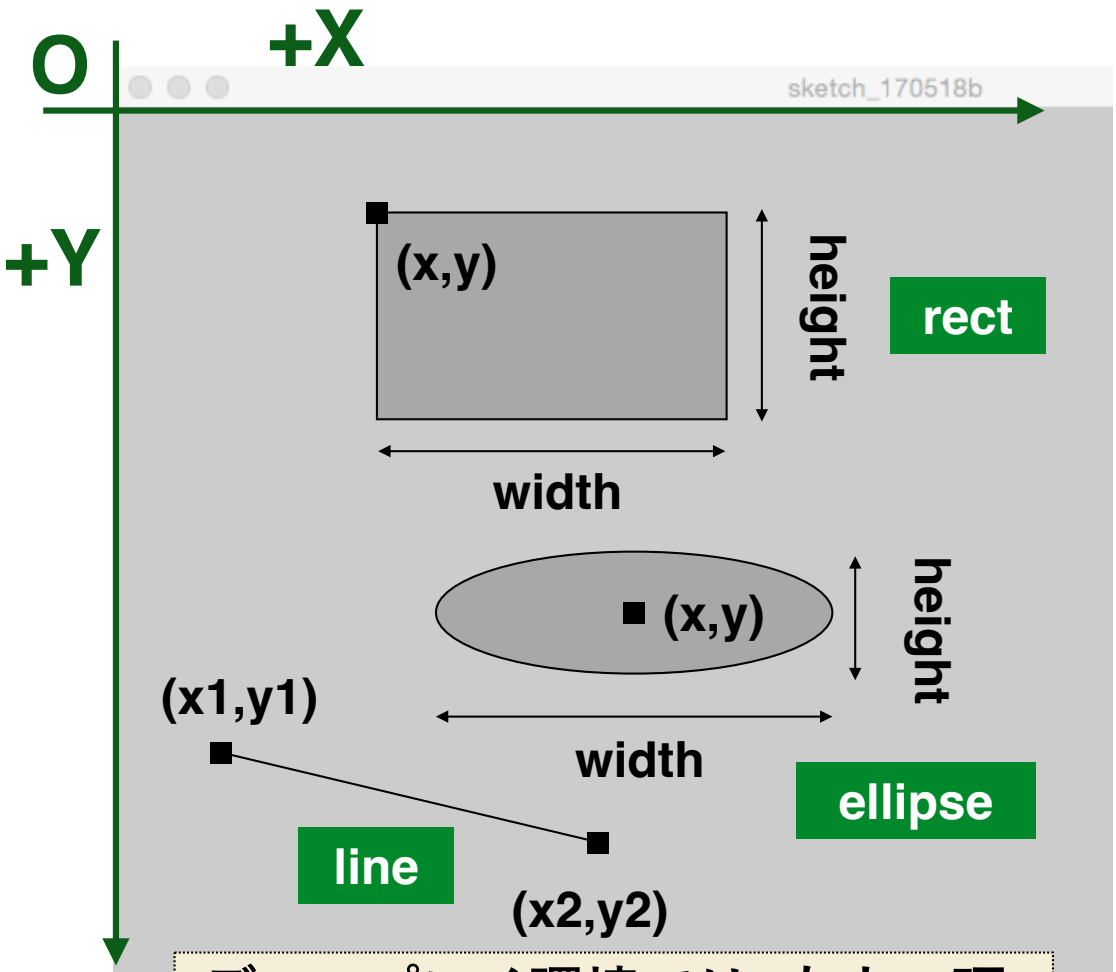


プログラムの命令は、exec(r1, r2, r3, …); という書き方をすると、まずはなんとなく覚えてください。execを「メソッド」、r1, r2, r3を「引数（ひきすう）」と呼びます。引数の数と種類はメソッドによって異なります。引数がない場合、カッコ内は何も書きません。exec(); という書き方となります

幾何図形の描画（P19, P239-240）

主要な幾何学図形の描画

```
//点
point(x,y);

//直線
line(x1,y1,x2,y2);

//三角形
triangle(x1,y1,x2,y2,x3,y3);

//四角形
quad(x1,y1,x2,y2,x3,y3,x4,y4);

//長方形
rect(x,y,width,height);

//円
ellipse(x,y,width,height);

//円弧
arc(x,y,width,height,start,stop);
```



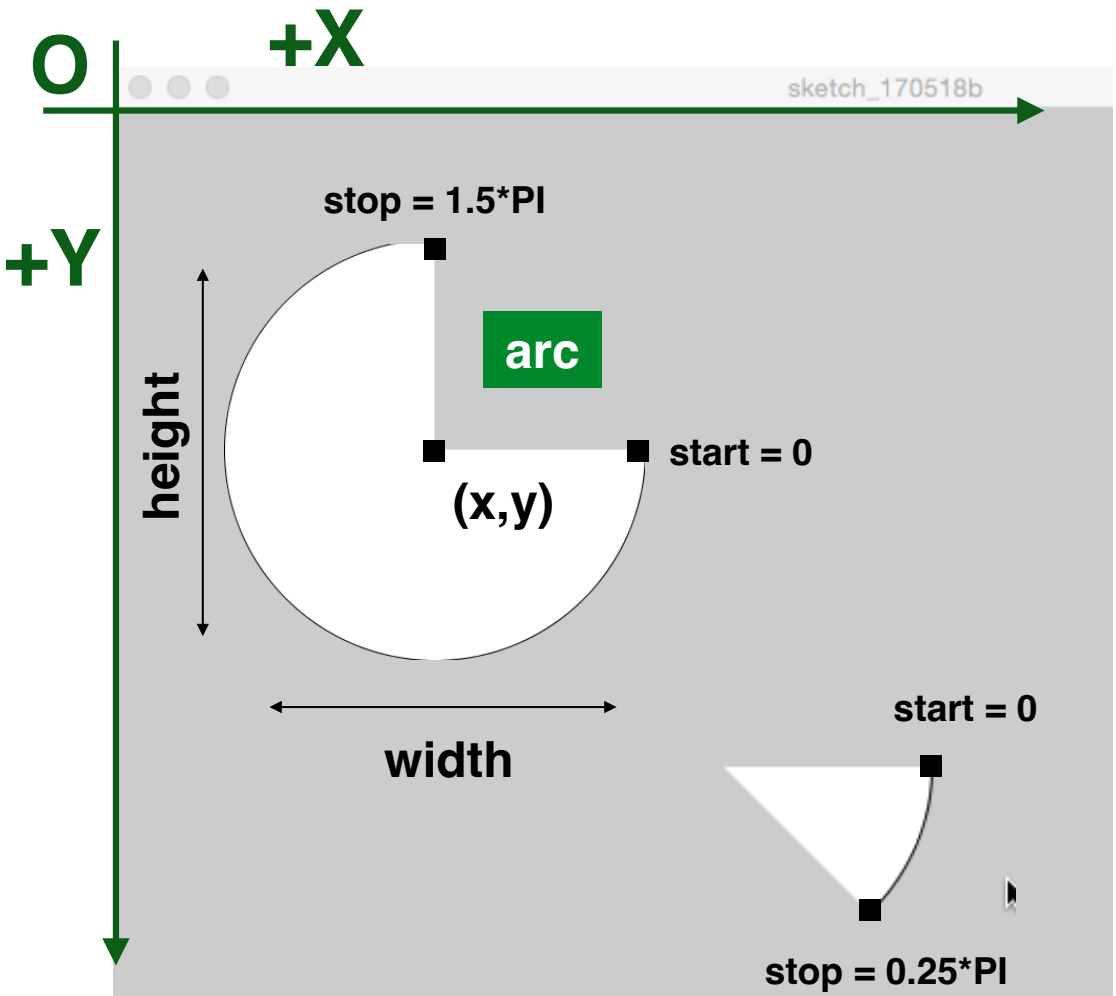
ディスプレイ環境では, 左上の頂点を原点として, 右方向がX軸正方向, 下方向がY軸正方向となることに注意してください.

以上の幾何学図形描画のメソッドのなかで指定する引数の数字は、整数でも小数でもOKです。メソッドによっては、引数に整数（int）しか受け付けないもの、小数（float）しか受付ないものなどがあります。引数は自動販売機の料金投入口のようなものをイメージするとよいです。硬貨投入口にお札を入れることはできないように、それぞれの引数には特定のタイプの入力が想定されています。

幾何図形の描画（P19, P239-240）

主要な幾何学図形の描画

```
//点
point(x,y);

//直線
line(x1,y1,x2,y2);

//三角形
triangle(x1,y1,x2,y2,x3,y3);

//四角形
quad(x1,y1,x2,y2,x3,y3,x4,y4);

//長方形
rect(x,y,width,height);

//円
ellipse(x,y,width,height);

//円弧
arc(x,y,width,height,start,stop);
```



ディスプレイ環境では, 左上の頂点を原点として, 右方向がX軸正方向, 下方向がY軸正方向となることに注意してください.

arc関数における, startは円弧の開始点、stopは終了点を弧度法の角度で表示します. PIと書くことで円周率（π ＝3.1415…）を表すことができます. また、n * PI は nπ を表すものと思ってください.

線と塗りつぶしの色（P26 - 32）

```
void noStroke();
```

線を描かない

```
void strokeWeight(w);
```

線の幅を**w**（float）ピクセルとする.

```
void stroke(r, g, b);
```

線の色をRGB値それぞれの3色の組み合わせ指定. それぞれの大きさは0-255で指定.

```
void stroke(gray);
```

線の色をグレイスケールで指定. 0-255で黒から白まで連続的に変化します.

コード

```
size(600,100);

strokeWeight(5);    rect(20,20,60,60);
strokeWeight(10);   rect(120,20,60,60);
noStroke();         rect(220,20,60,60);
stroke(255, 0 ,0);  rect(320,20,60,60);
noFill();           rect(420,20,60,60);
fill(    0   );     rect(520,20,60,60);
```

```
void noFill();
```

塗りつぶしをしない

```
void fill(r, g, b);
```

```
void fill(gray);
```

塗りつぶしの色を指定. 引数の指定の仕方は, strokeと同様.

実行結果

voidは、対象となるメソッドに「返り値（かえりち）」が無いことを示すもの。現時点では、「取り出し口」の無い自動販売機のようなものを想像してもらえればいいです。

透明度の色（P26 - 32）

```
void stroke(r, g, b, alpha);
```

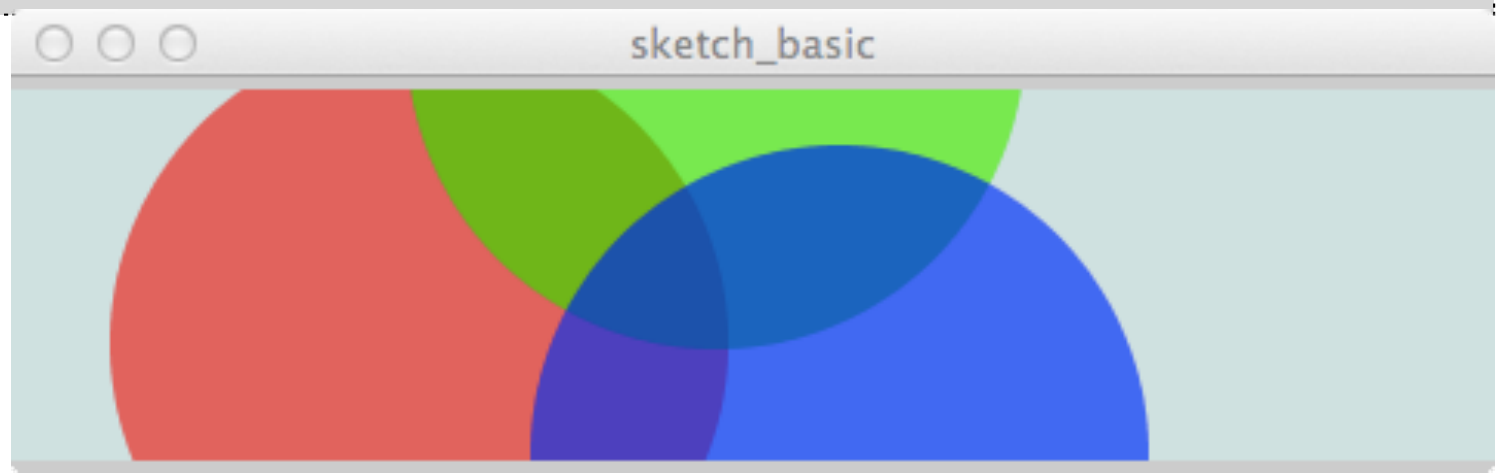
```
void fill(r, g, b, alpha);
```

stroke（線の色）、fill（塗りつぶしの色）の引数が４つの場合、4番目に引数は透明度を表す。0から255で指定する。

コード

```
fill(255,0,0,160);ellipse(132,82,200,200);
fill(0,255,0,160);ellipse(228,-16,200,200);
fill(0,0,255,160);ellipse(268,118,200,200);
```

実行結果



CSSにおける透明度（アルファ値）は、 rgba(268, 118, 200, 0.65) のように, 0.0 - 1.0の小数で指定していましたが, Processingは, アルファ値も0-255で指定していることに注意してください。

Practice # 2

描画と変数（Processingの基礎）

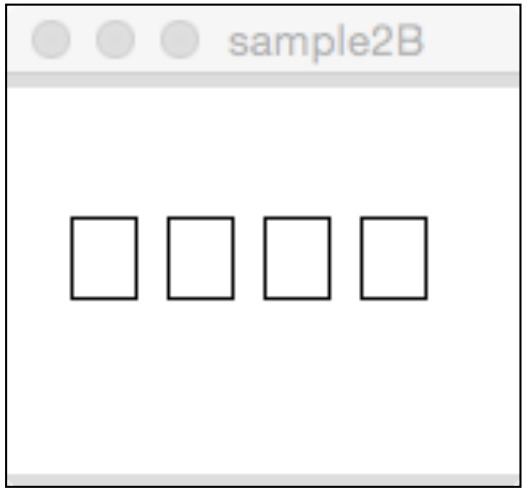
演習2B　変数の概念

以下のように, 変数を使って四角形を四つ並べるプログラムを作成しましょう.

sample2B_1.pde

コード

```
size(160,120);
background(255);

rect(20,40,20,25);
rect(50,40,20,25);
rect(80,40,20,25);
rect(110,40,20,25);
```



実行結果

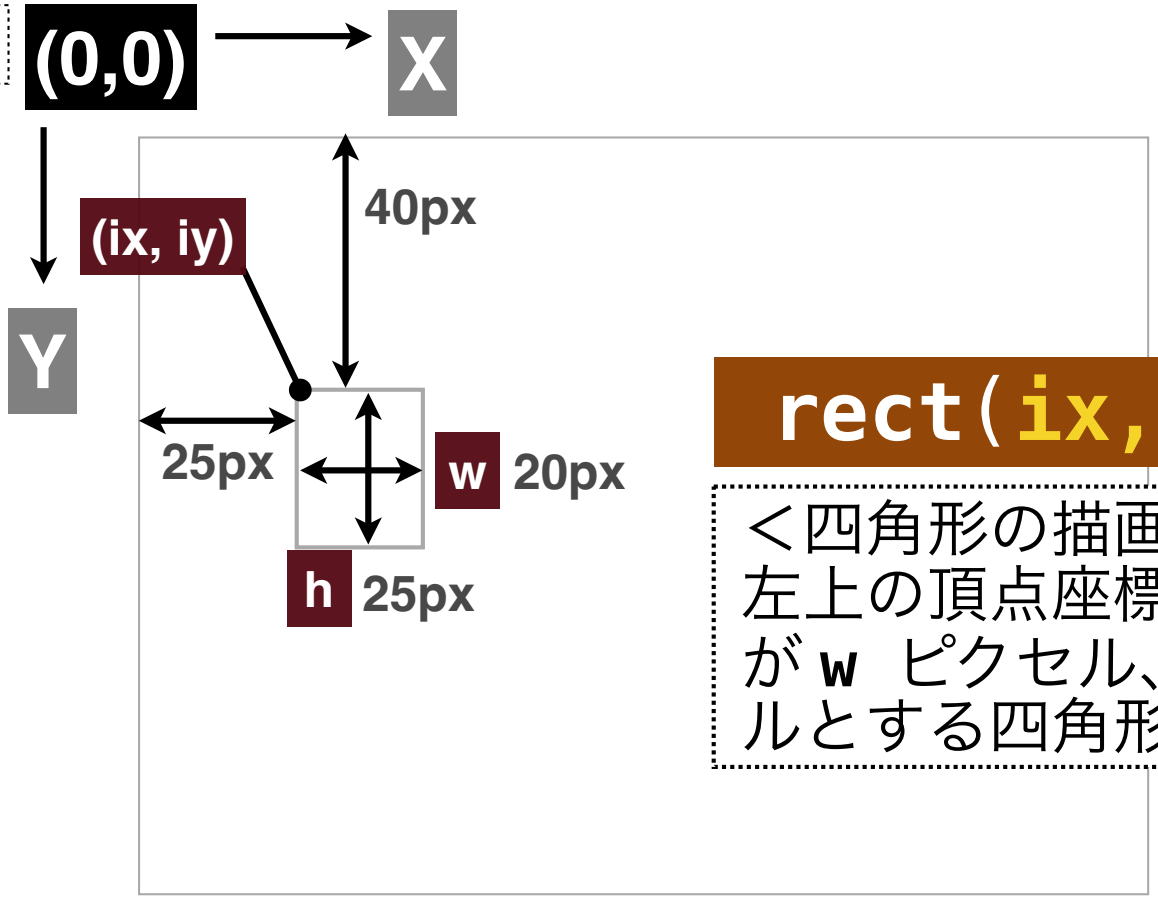
枠線の描画

描画場所, 大きさ

```
rect(25,40,20,25);
rect(50,40,20,25);
...
```

- 四角形を描画せよ. ただし, ...
- 左上頂点の座標が(25,40), 幅が20, 高さが25とする.

実行結果



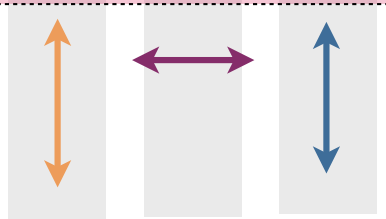

```
rect(ix, iy, w, h);
```

＜四角形の描画＞
左上の頂点座標を**(ix, iy)**、幅が **w** ピクセル、高さが **h** ピクセルとする四角形を描画

枠線の描画

描画場所, 大きさ

```
rect( 20,40,20,25);
rect( 50,40,20,25);
rect( 80,40,20,25);
rect(110,40,20,25);
```

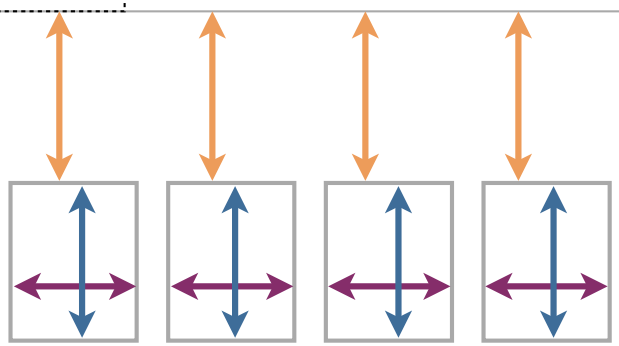


変数を使って,
書き直すと…

```
int w = 20;
int h = 20;
rect( 20,40,w,h);
rect( 50,40,w,h);
rect( 80,40,w,h);
rect(110,40,w,h);
```

変数

実行結果

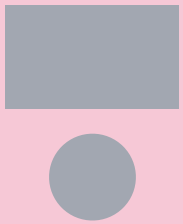


3つの引数（縦の位置, サイズ）で, 同じ数字が繰り返し使われている.

「サイズを変更 → 全てを書き換える」… 冗長??

変数を箱のメタファーで考える.

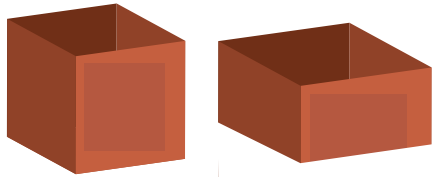
```
int w = 20;
int h = 25;
```



データ型

変数名

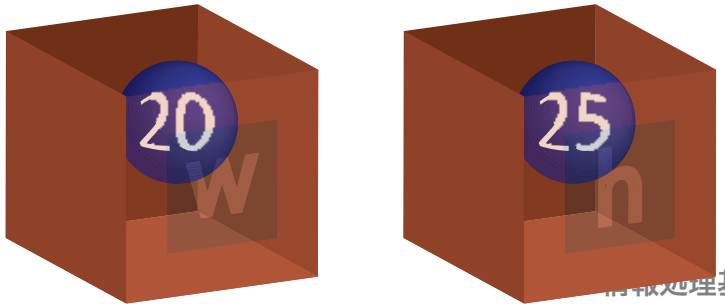
```
rect( 25,40,w,h);
rect( 50,40,w,h);
rect( 80,40,w,h);
rect(110,40,w,h);
```



データ型

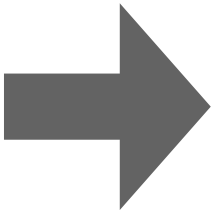
変数名

値



変数を箱のメタファーで考える.

```
int w = 20;
```



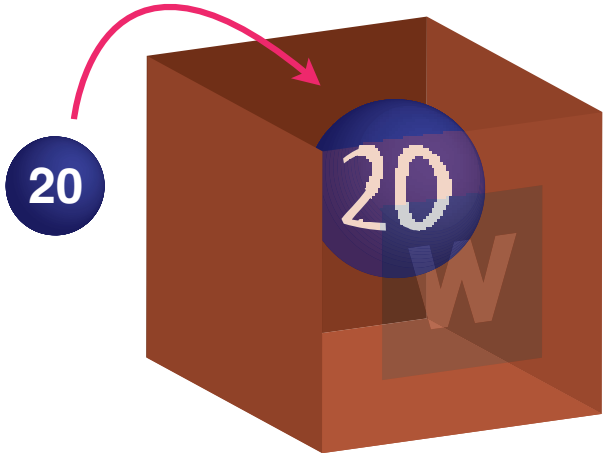
```
int w;    //変数の宣言文
w = 20;   //代入文
```

int（整数）専用の箱に, wのラベルを貼る.

```
int w;    //変数の宣言文
```

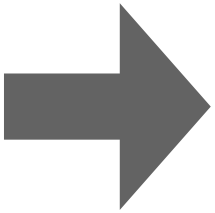
箱に数字（値）を入れる.

```
w = 20;   //代入文
```



変数を箱のメタファーで考える.

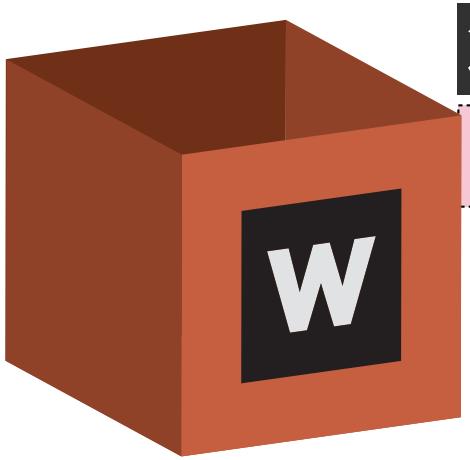
```
int w = 20;
```



```
int w;    //変数の宣言文
w = 20;   //代入文
w = 40;   //代入文2
```

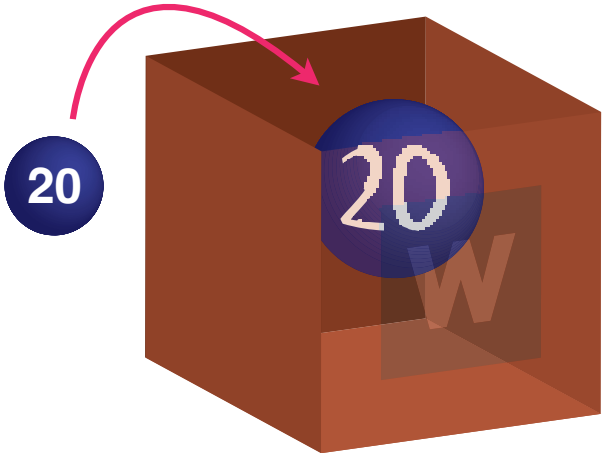
int（整数）専用の箱に, wのラベルを貼る.

```
int w;    //変数の宣言文
```



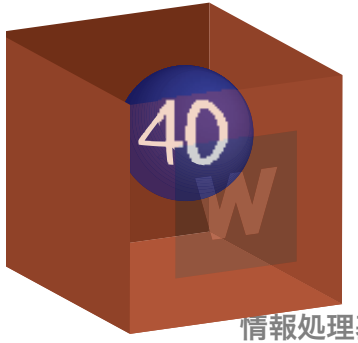
箱に数字（値）を入れる.

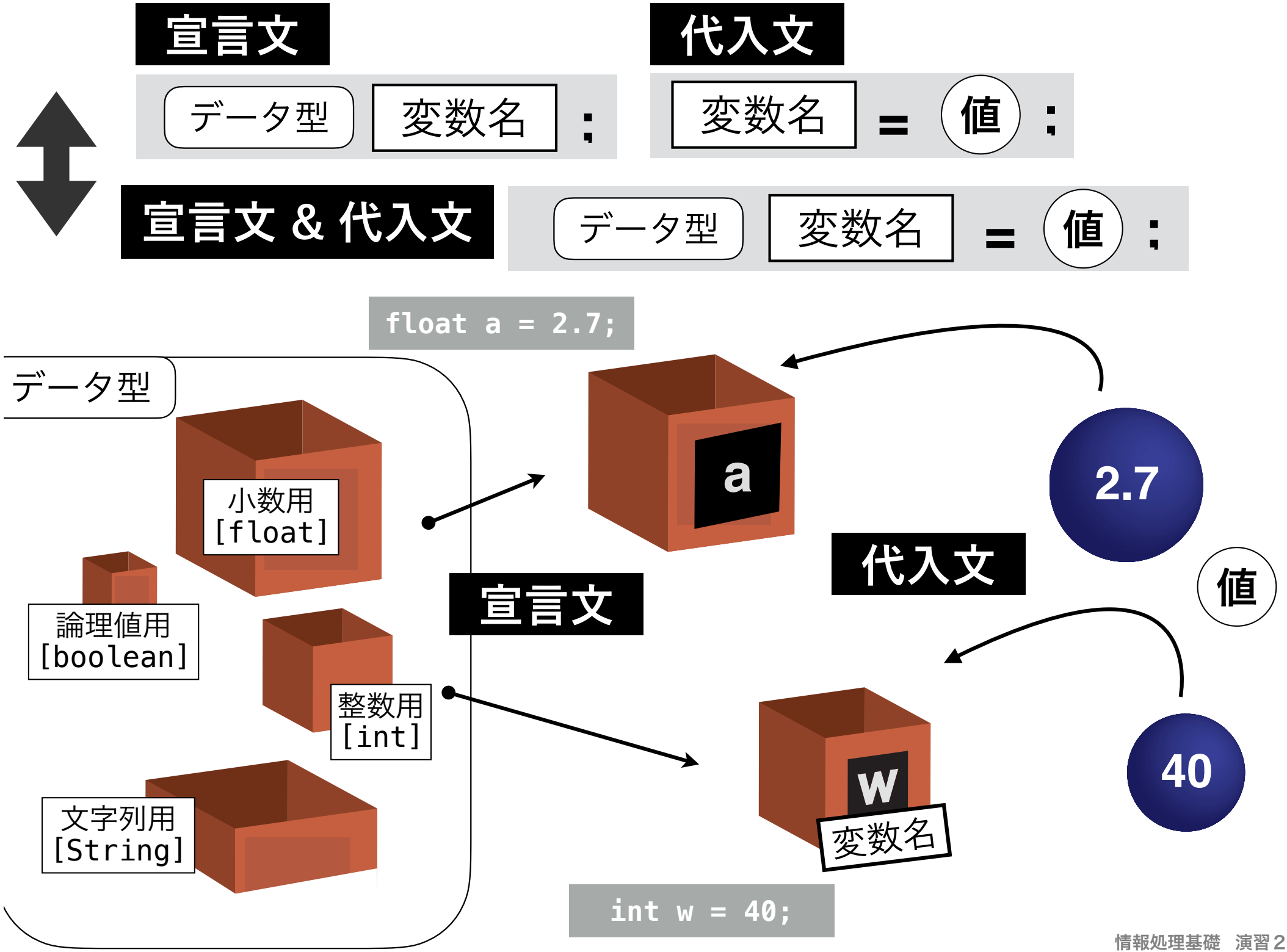
```
w = 20;   //代入文
```



箱の中の値を書き換える.

```
w = 40;   //代入文2
```




宣言文
データ型
変数名
:
代入文
変数名
=
値
:
宣言文 & 代入文
データ型
変数名
=
値
:
float a = 2.7;
データ型
小数用
[float]
論理値用
[boolean]
整数用
[int]
文字列用
[String]
宣言文
a
代入文
2.7
値
w
変数名
40
int w = 40;

変数の名前を決める

宣言文

データ型 変数名 ;

代入文

変数名 = 値 ;

宣言文 & 代入文

データ型 変数名 = 値 ;

#w 2w ×

変数名 **w wi w1 w2 w_1 _w W2** ○

規則

1. 一文字目は英文字（またはアンダーバー）.
2. 二文字目以降は英数字とアンダーバー(_)を使用可.

指針

a. なるべく短い文字数で...
b. 変数の性質を連想させるような文字の選択.

w ← **w**idth　　**h** ← **h**eight

データ型（p227）

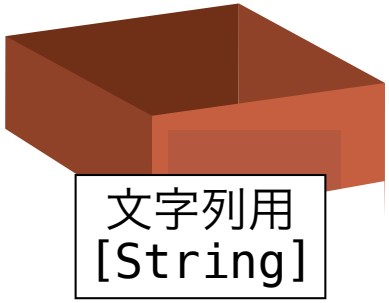
データ型

```
int w1 = 20;

int w2 = 3.7;
int w3 = 20.0;
```

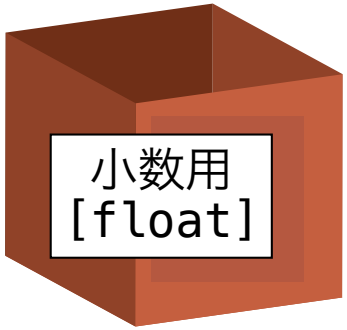
ERROR!!

int型で宣言した変数に小数の値を代入するとエラーになり，実行できません.

```
boolean b1 = true;
boolean b2 = false;
```



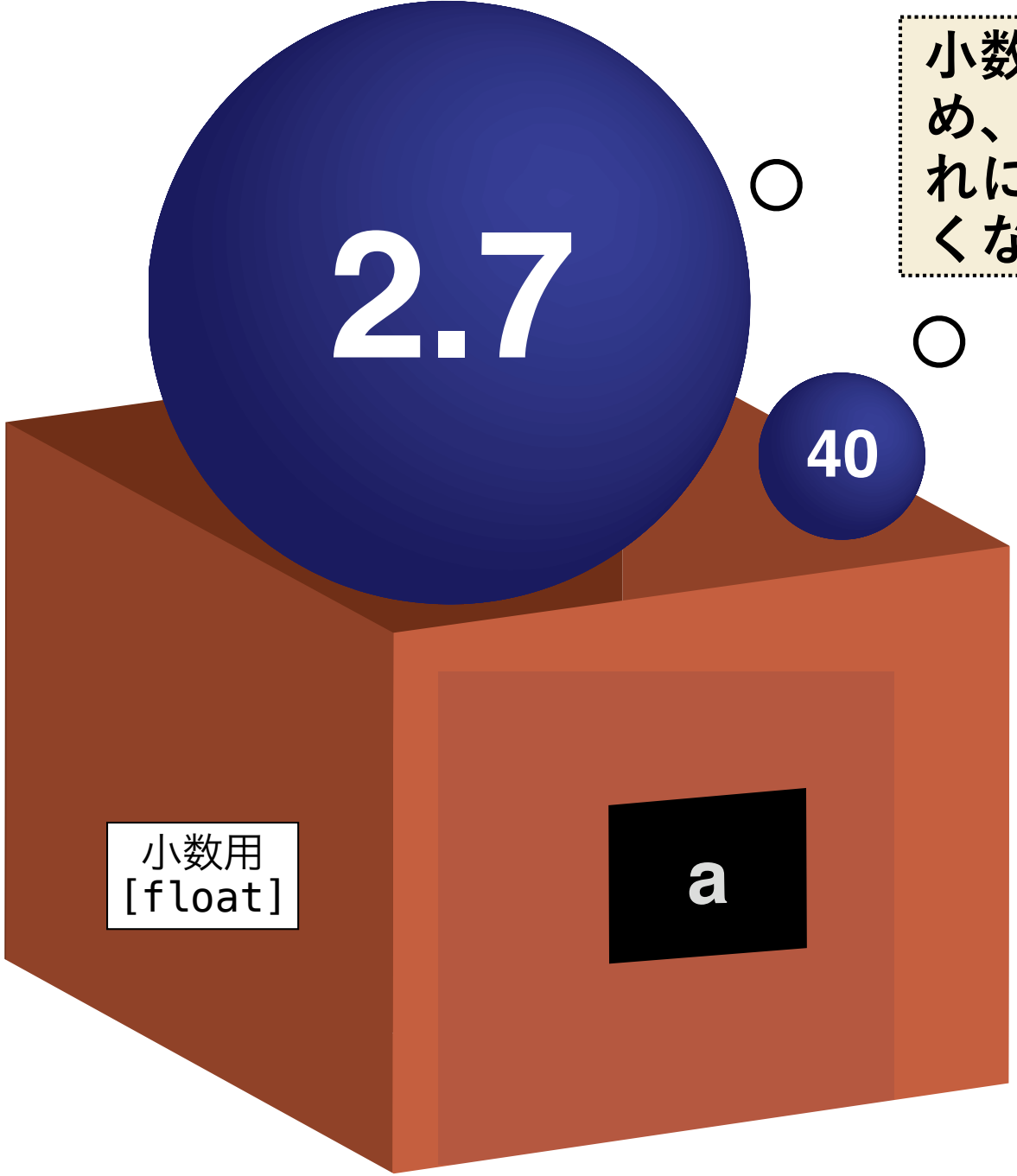

```
String s1 = “hello”;
String s2 = “1”+“2”;
```

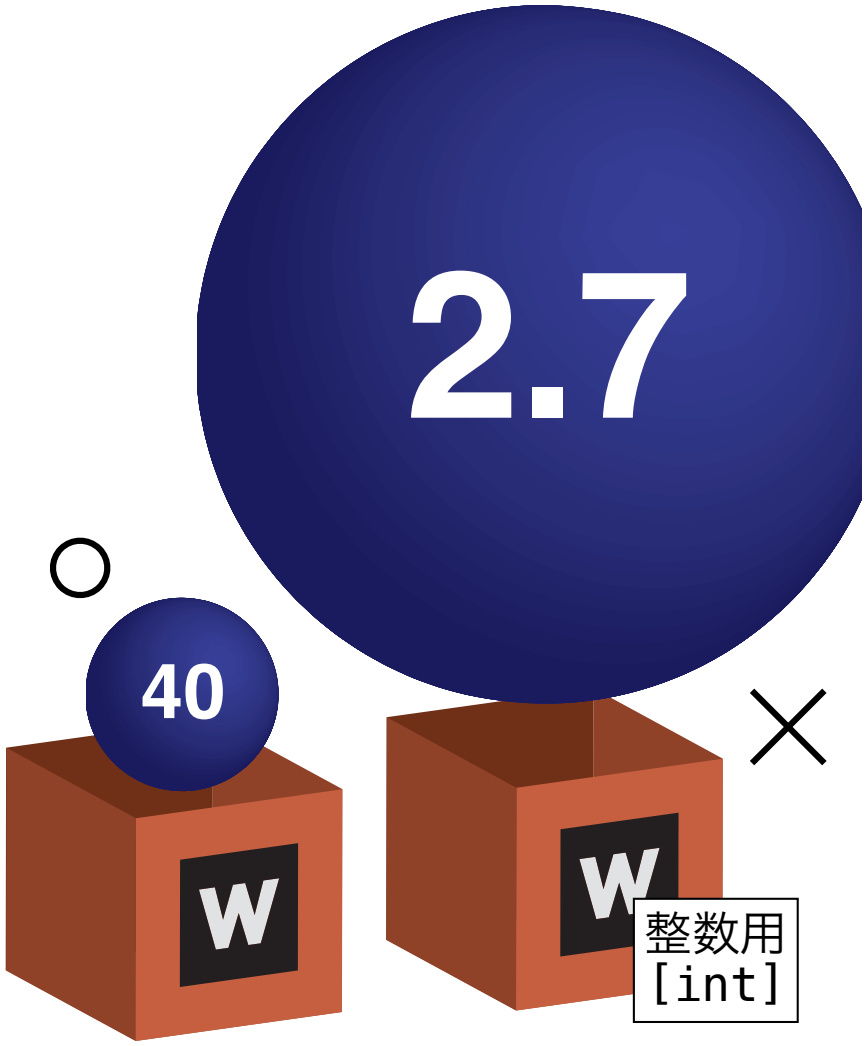



```
float a1 = 2.5;
float a2 = 3;
```

以上のデータ型は、Javaでも同様に適用されます（charは文字1文字用）。このほか、Processing特有のよく使われるデータ型として、color（色）、PImage（画像）、PFont（フォント）を挙げることができます。

小数の方がメモリを多く使うため、ボールのサイズが大きく、それに対応してデータ型の箱も大きくなるイメージを持てば良い。






```
float a = 2.7;○
float a = 40; ○
```

```
int w = 2.7; ×
int w = 40; ○
```

変数を使うと、効率的にプログラムできます.

変数未使用

```
rect( 20,40,20,25);
rect( 50,40,20,25);
rect( 80,40,20,25);
rect(110,40,20,25);
```

→

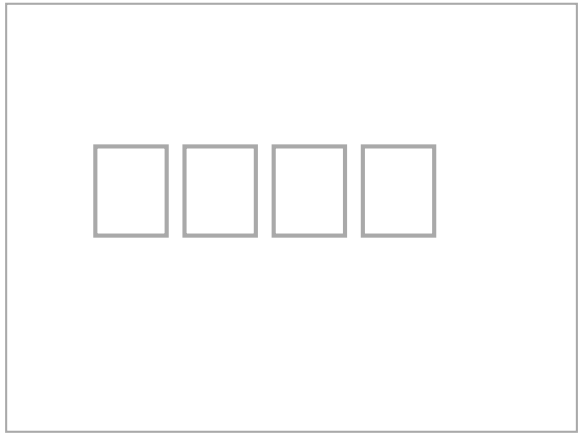
```
rect( 20,40,15,30);
rect( 50,40,15,30);
rect( 80,40,15,30);
rect(110,40,15,30);
```

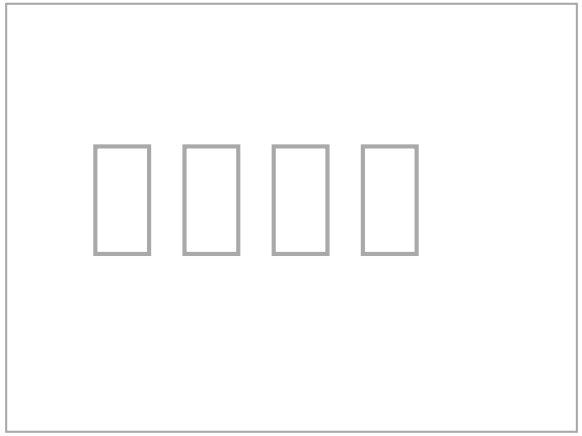
変数使用

```
int w = 20;
int h = 25;
rect( 20,40,w,h);
rect( 50,40,w,h);
rect( 80,40,w,h);
rect(110,40,w,h);
```

→

```
var w = 15;
var h = 30;
rect( 20,40,w,h);
rect( 50,40,w,h);
rect( 80,40,w,h);
rect(110,40,w,h);
```





四角形のサイズをもう少し縦長にしたい....

Practice # 2

描画と変数（Processingの基礎）

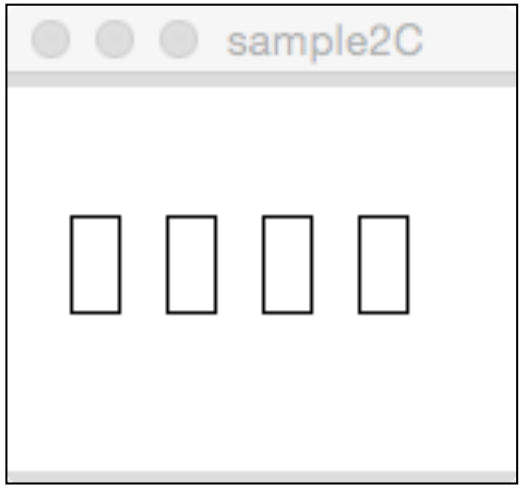
演習2C　プログラムにおける演算

以下のように, 変数を使って四角形を四つ並べるプログラムを作成しましょう.

sample2C_1.pde

```
size(160,120);
background(255);

int ix = 20;
int sp = 30;
int y = 40;
int w = 15;
int h = 30;

rect(ix,y,w,h);
rect(ix+sp,y,w,h);
rect(ix+2*sp,y,w,h);
rect(ix+3*sp,y,w,h);
```

コード



実行結果

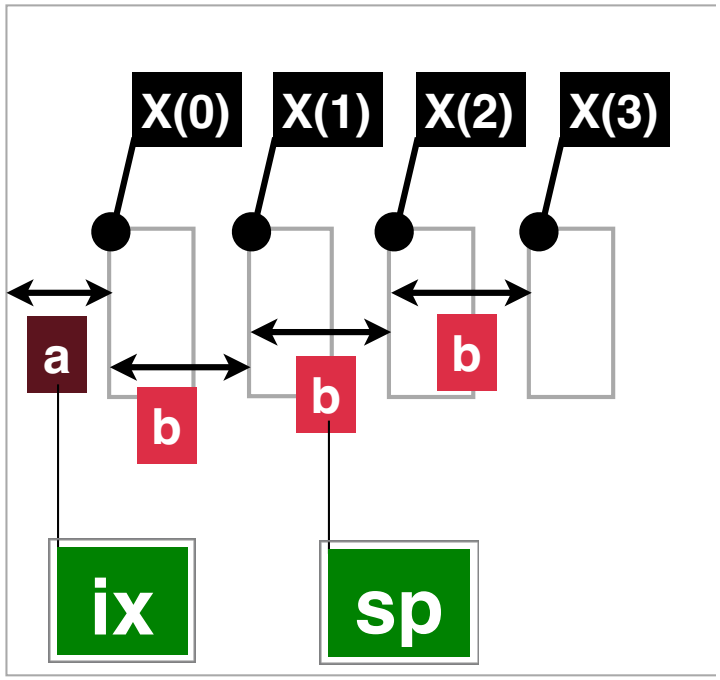
変数と計算（p43-44）

```
int ix = 20;  //四角形の頂点位置(X座標)の始点
int sp = 30;  //四角形の間隔(X方向)
int y = 40;   //四角形の頂点位置(Y座標)
int w = 15;   //四角形の幅を表す変数
int h = 30;   //四角形の高さを表す変数
```

コード

```
rect(ix,y,w,h);
rect(ix+sp,y,w,h);
rect(ix+2*sp,y,w,h);
rect(ix+3*sp,y,w,h);
```

sample2C_1.pde



左からn番目の四角形のx座標は??

X(0) = a

X(1) = a + b

X(2) = a + b + b = a + 2 X b

X(3) = a + b + b + b = a + 3 X b

ix

ix + sp

ix + 2 * sp

ix + 3 * sp

演算の記法

+	加算	a + b	aにbを加える
-	減算	a - b	aからbを引く
*	乗算	a * b	aとbを掛ける
/	除算	a / b	aをbで割る
%	余剰	a % b	aをbで割った余り

x = 8 * 3;	x=24;
x = 4 / 2;	x=2;
x = 5 / 2;	x=2.5;
x = 8 % 3;	x=2;
x = 4 % 2;	x=0;
x = 5 % 2;	x=1;

```
int a = 8 / 3;
float b = 8 / 3;
println("a="+a);
println("b="+b);
```

以上のようにprintln関数を使うと，
計算結果を確認することができます．

```
void print(s);
```

引数の文字列をコンソールに書き出します。引数が数字の場合、文字列として解釈されます。

```
void println(s);
```

print()と同様ですが、最後に改行します。

変数と計算（p43-44）

++	インクリメント	a++;	a=a+1;と同じ
+=		a += b;	a=a+b;と同じ
--	デクリメント	a--;	a=a-1;と同じ
-=		a -= b;	a=a-b;と同じ

```
int x = 24;
x++;
```

x = 25

```
int x = 0;
x--;
```

x = -1

```
int x = 24;
x+=7;
```

x = 31

```
int x = 0;
x-=8;
```

x = -8

四則演算の優先順位

```
int x = 3;
int a = (x+3) * 3;
a = x+3*3;
a= x-3/3;
```

a = 18

a = 12

a = 2

```
int a = (x+3) * 3;
int b = x + 3*3;
println("a="+a);
println("b="+b);
```

以上のようにprintln関数を使うと，計算結果を確認することができます．

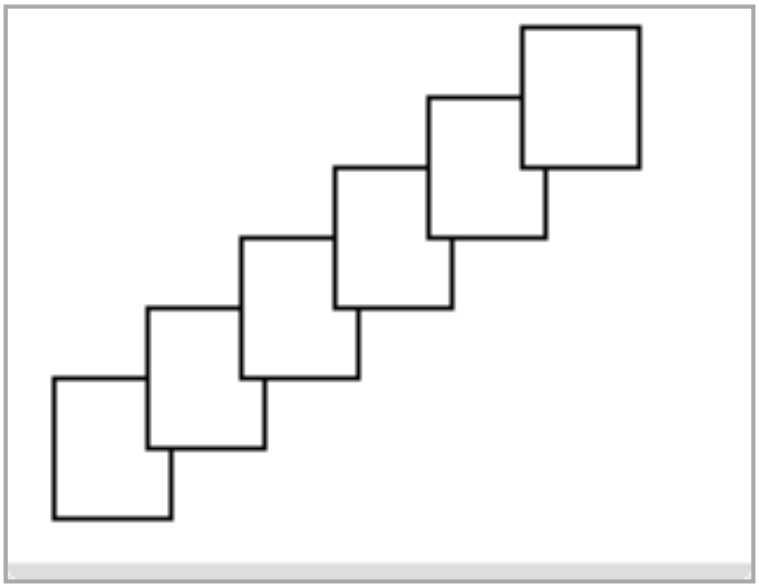
練習

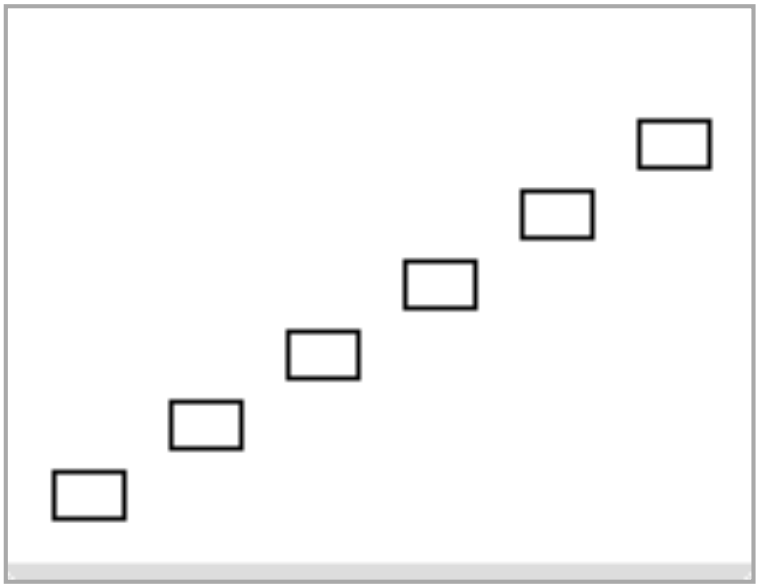
「sample2C_1.pde」を修正し, 6つの長方形が, x軸方向（横方向）だけでなく, y軸方向（縦方向）にも並ぶようにしてください.

このさい, 長方形は右上がりとなるようにしてください. さらに, Y軸方向の四角形同士の間隔を変数（例えばspyとか）で表現すること！！

ファイル名は「sample2C_X.pde」とします.

実行例





sample2C_X.pde

Practice # 2

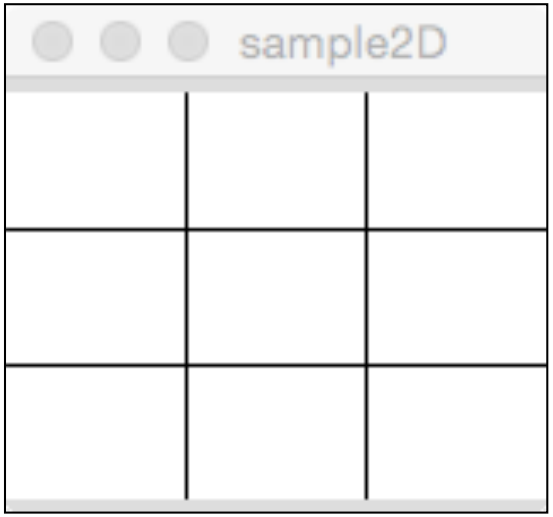
描画と変数（Processingの基礎）

演習2D　特別な変数

sample2D_1.pde

```
size(160,120);
background(255);

float cw = width;  //ウィンドウの幅
float ch = height; //ウィンドウの高さ

float x1 = cw/3;
float x2 = 2*cw/3;

float y1 = ch/3;
float y2 = 2*ch/3;

//縦線を引く
line(x1,0,x1,ch);
line(x2,0,x2,ch);
//横線を引く
line(0,y1,cw,y1);
line(0,y2,cw,y2);
```

コード



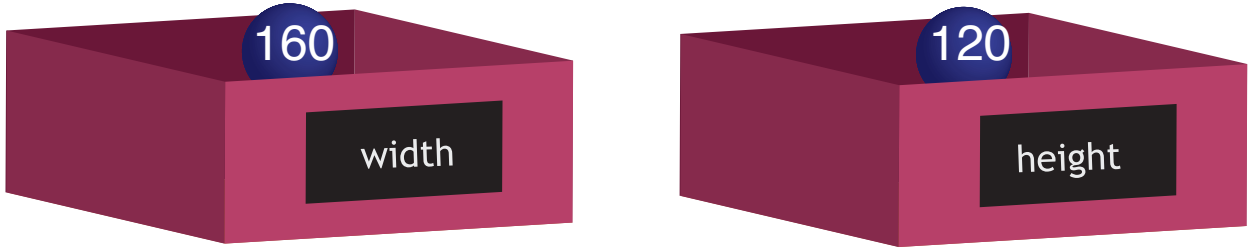
実行結果

width と height（p42）

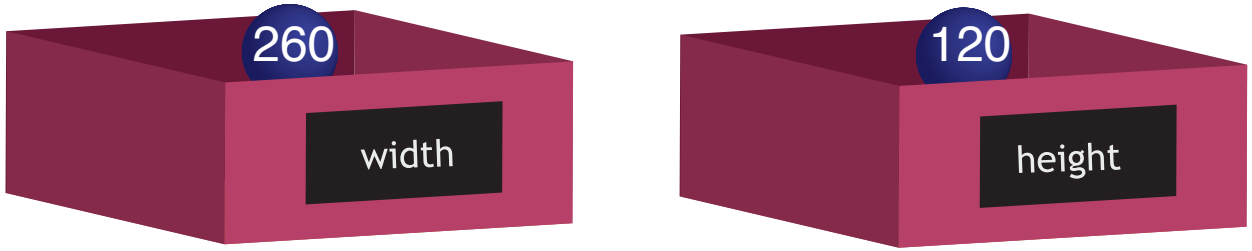
Processingは実行中のプログラムの状態を表す特殊な変数を持っています．現在のウィンドウの幅と高さを記憶している width と height がその一例で，これらは size() 関数を実行した時にセットされます．

width と height は、あらかじめ宣言されているint型の変数（はじめから，ラベルのついた箱が用意されている）と考えればわかりやすいでしょう．

```
size(160,120);
```



```
size(260,120);
```

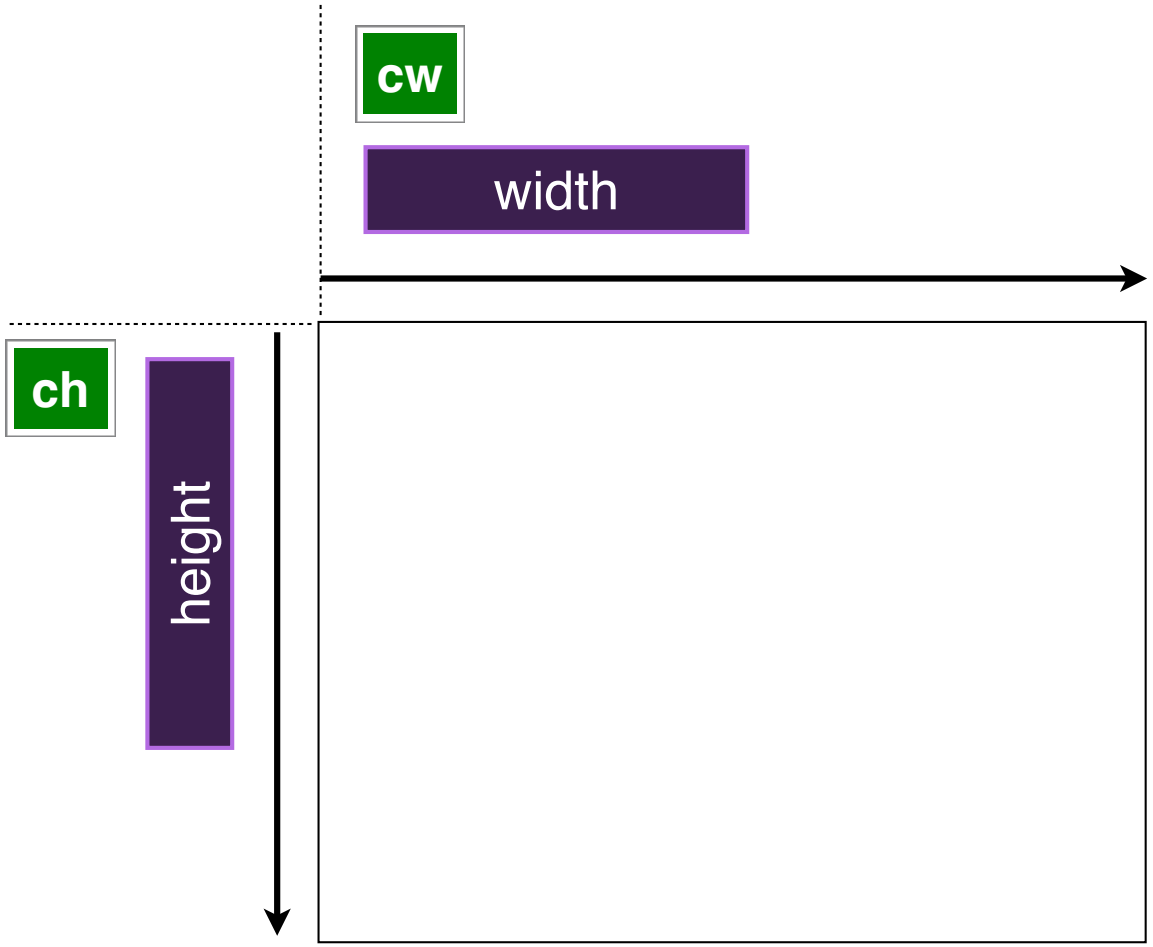


width と height を使いこなす.

単に、widthとheightを、違う名前の変数に置き換えています（文字数を少なくするため）。

sample2D_1.pde

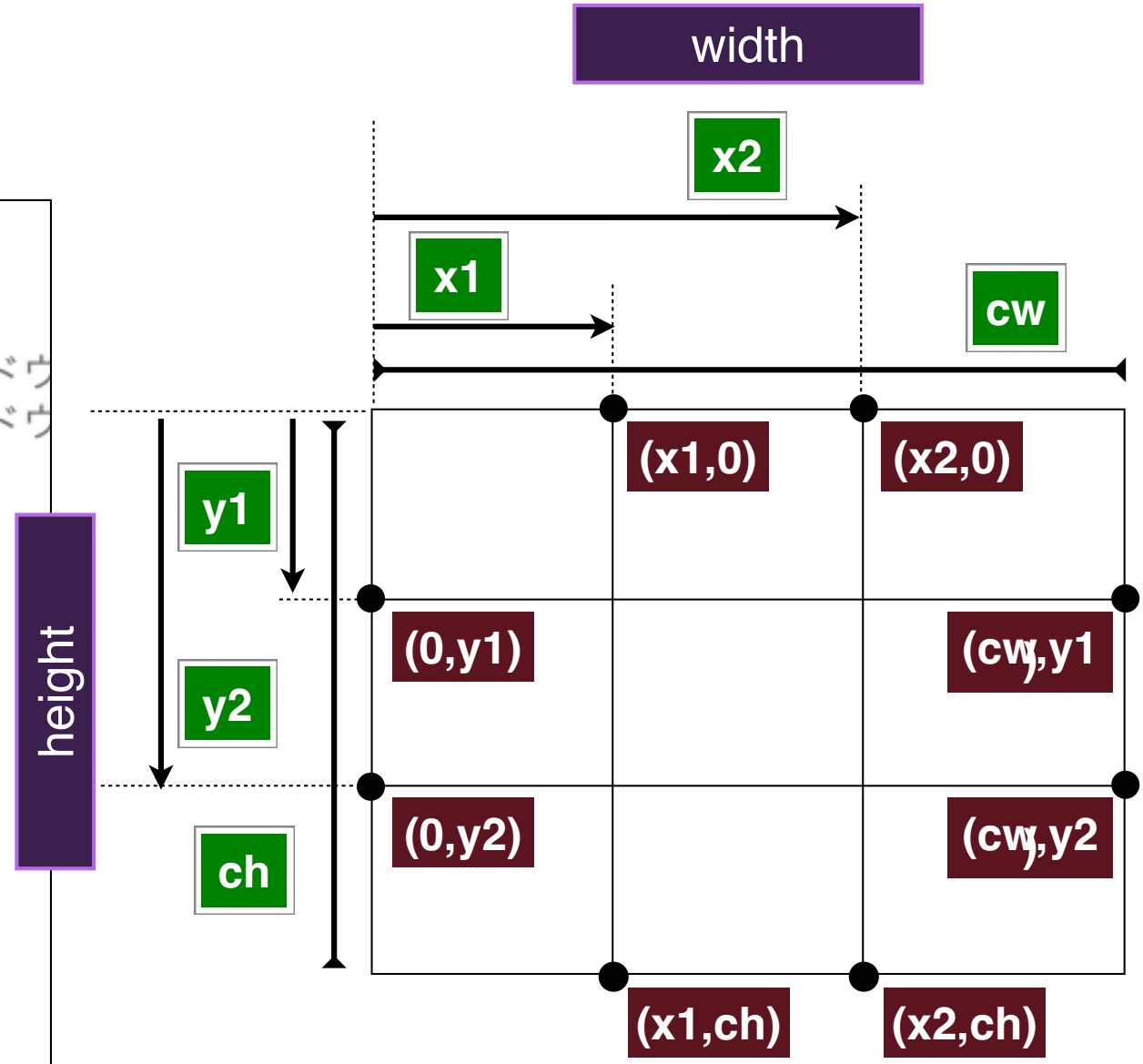
```
size(160,120);
background(255);

float cw = width;   //ウィンドウ
float ch = height;  //ウィンドウ

float x1 = cw/3;
float x2 = 2*cw/3;

float y1 = ch/3;
float y2 = 2*ch/3;

//縦線を引く
line(x1,0,x1,ch);
line(x2,0,x2,ch);
//横線を引く
line(0,y1,cw,y1);
line(0,y2,cw,y2);
```



width と height を使いこなす.

sample2D_1.pde

```
size(160,120);
background(255);

float cw = width;  //ウィンドウ
float ch = height; //ウィンドウ

float x1 = cw/3;
float x2 = 2*cw/3;

float y1 = ch/3;
float y2 = 2*ch/3;

//縦線を引く
line(x1,0,x1,ch);
line(x2,0,x2,ch);
//横線を引く
line(0,y1,cw,y1);
line(0,y2,cw,y2);
```



Practice # ２

描画と変数（Processingの基礎）

- 課題学習２１（必須）
- 課題学習２２（必須）
- 課題学習２３（任意）
- 課題学習２４（任意）
- 課題学習２５（任意）

締め切り　05.11

課題学習２１（必須）

今回学んだ, 四角形（rect）, 円（ellipse）, 線（line）などを使って、自由に人の顔をかいてみてください.

時間に余裕があれば、背景などもどんどん書き足していってしまって構いません。

サイズは自由とします。

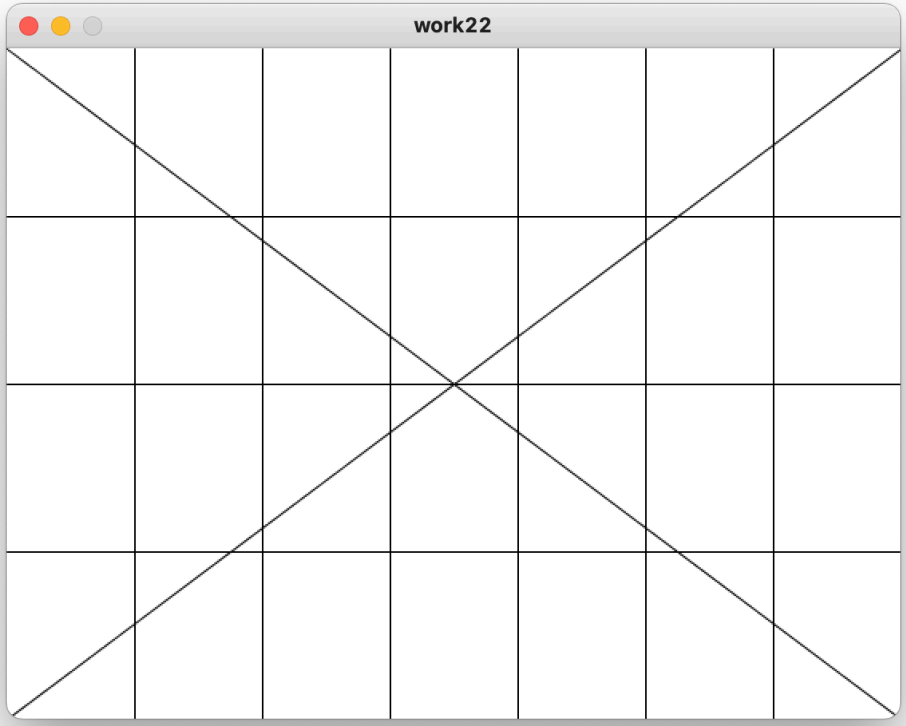
例

work21.pde

課題学習２２（必須）

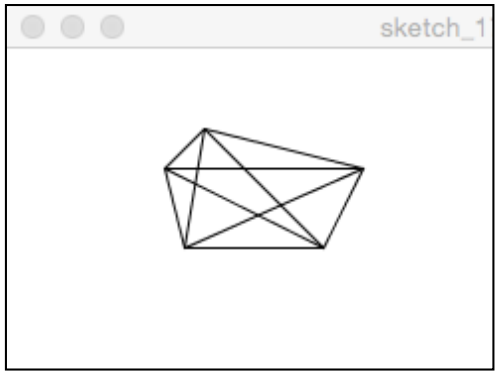
「sample2D_1.pde」を修正して, 任意の描画領域に対して, 横方向を7等分・縦方向を4等分するグリッド線を描くとともに、対角線を新たに加えた新しいプログラムを作成してください. width と height を使うこと.

work22.pde



課題２３（任意）

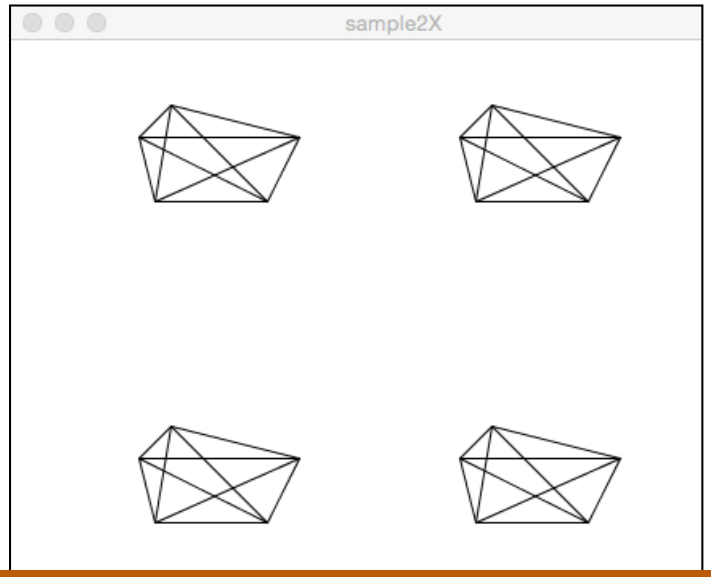
まず、適当に定めた5つの頂点について、すべての頂点間で線が引かれる五角形を描画してください。



ヒント　5つの頂点に対応する10個のint型の変数
(x1, y1), (x2, y2), (x3, y3), (x4, y4), (x5, y5) を作成して、、

この五角形を、右のように平行移動を繰り返して、三つのコピーを複製してください。変数を新たに追加しないこと！！

work23.pde



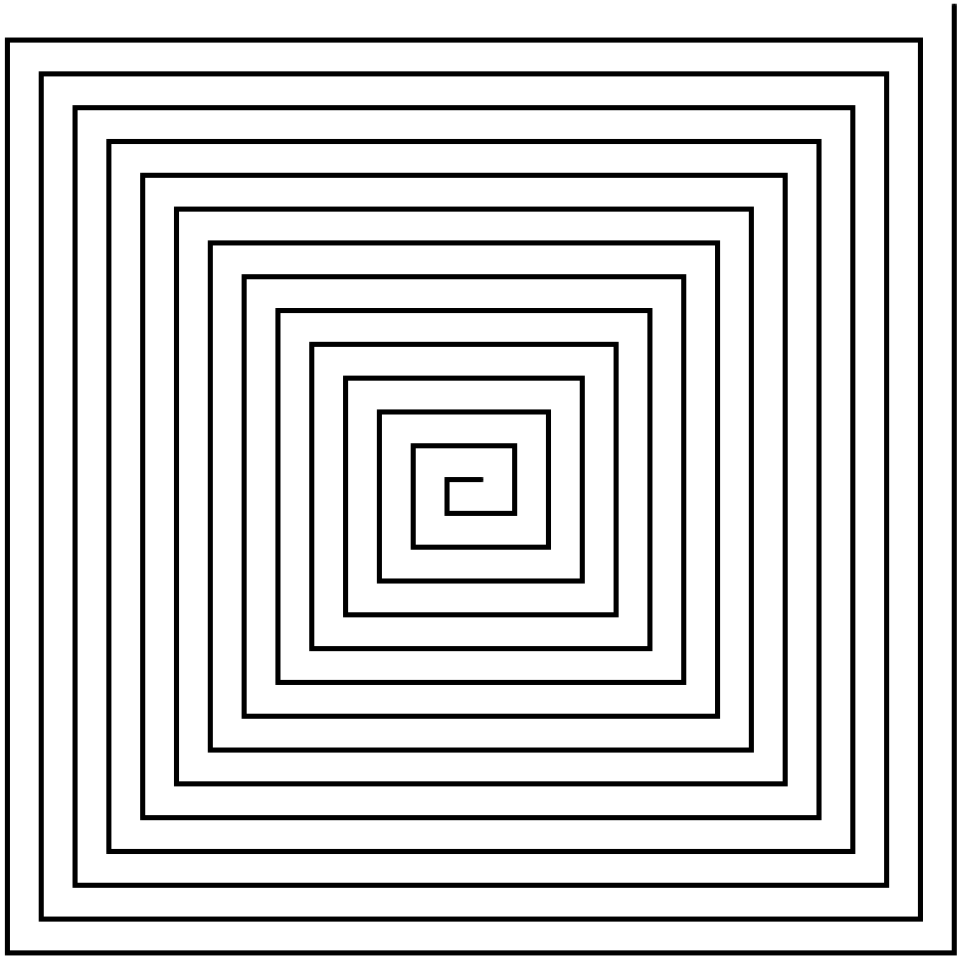
ヒント　インクリメント（+=）、デクリメント（-=）を使いましょう。

課題２４（任意）

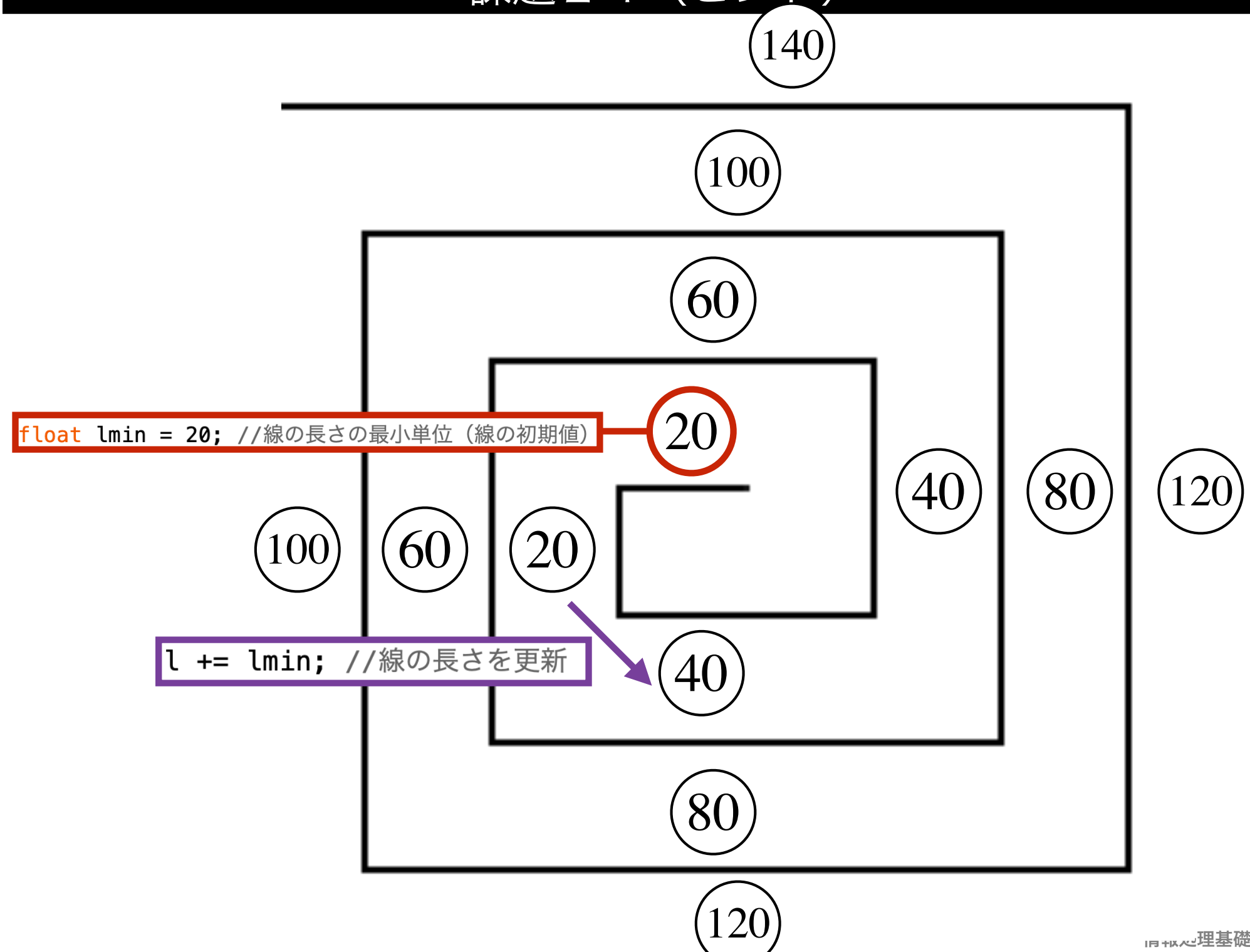
左のコードの続きから, 右の描画を完成させてください.
変数はこれ以上増やさず，繰り返し文（後で習う）の使用も禁止！！
コピペで解決してください。

work24.pde

```
size(800,800);  background(255);
strokeWeight(3); //線の太さを3ピクセル

float x0, y0; //始点
float x1, y1; //終点

//始点の初期値
x0 = 0.5 * width; y0 = 0.5 * height;

float lmin = 20; //線の長さの最小単位（線の初期値）
float l = lmin; //線の長さ

x1 = x0 - l; y1 = y0; //終点を始点の左に設定

line(x0,y0,x1,y1); //線を引く

x0 = x1;  y0 = y1; //終点を始点にコピー
x1 = x0;  y1 = y0 + l; //終点を始点の下に設定

line(x0,y0,x1,y1); //線を引く

l += lmin; //線の長さを更新
```



課題２４（ヒント）

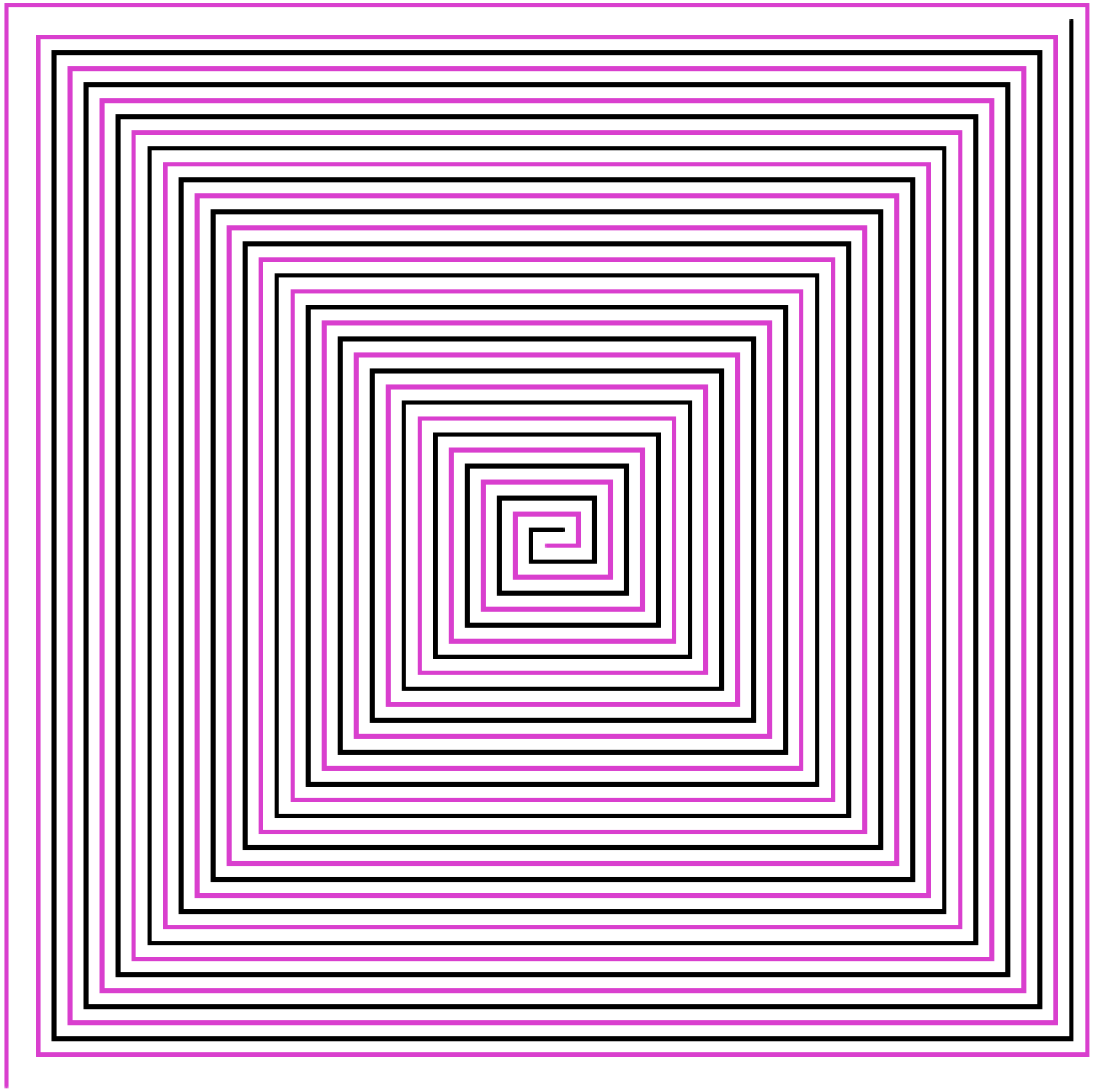

140
100
60
20
40
80
120
100
60
20
40
80
120
float lmin = 20; //線の長さの最小単位（線の初期値）
l += lmin; //線の長さを更新

課題２５（任意）

work25.pde

右の様に、work24の隙間をさらに異なる線で埋めてみましょう。

（紫色の線が新たに追加したものです）



提出方法

ファイル名をwork2x_235XXXとします.

リクエスト時は、名前の先頭に学籍番号を入れるようにしてください。

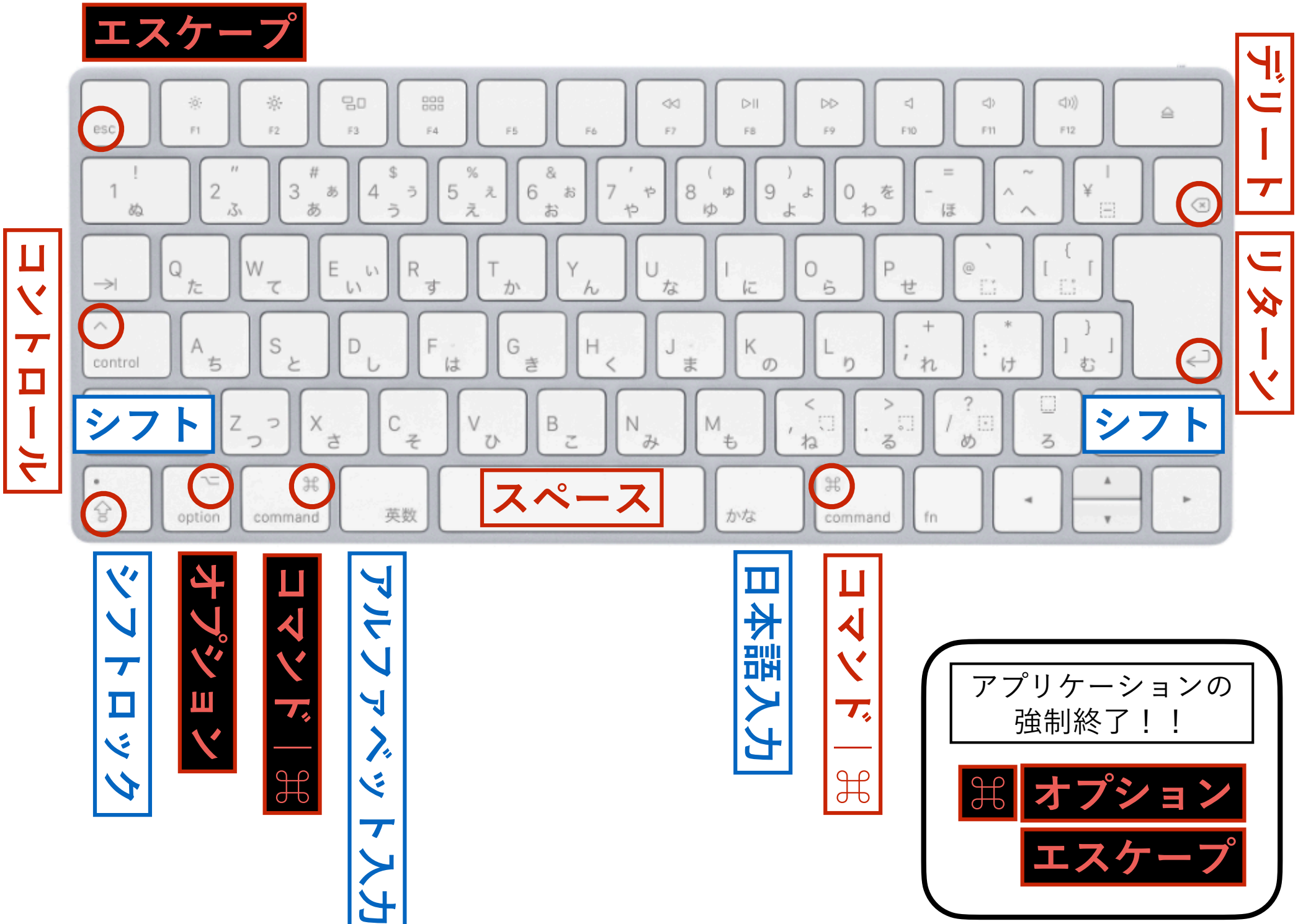
対応する課題のリンクから, フォルダをまるごと提出してください.

課題学習	[YOUTUBE] [課題提出｜WORK21] [課題提出｜WORK22] [課題提出｜WORK23] [課題提出｜WORK24] [課題提出｜WORK25]

https://lab.kenrikodaka.com/univclass/mediabasic2023/

提出方法

リクエスト時は、名前の先頭に学籍番号を入れるようにしてください。


エスケープ
デリート
リターン
コントロール
シフト
シフト
スペース
シフトロック
オプション
コマンド ⌘
アルファベット入力
日本語入力
コマンド ⌘
アプリケーションの
強制終了！！
⌘ オプション
エスケープ

+	加算	a + b	aにbを加える
-	減算	a - b	aからbを引く
*	乗算	a * b	aとbを掛ける
/	除算	a / b	aをbで割る
%	余剰	a % b	aをbで割った余り

シフト