配列　扱う変数が多い時は配列を利用する

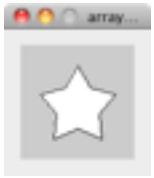
配列の宣言方法

int [] data = new int[5];
→data[0]からdata[4]までの配列を作る
int [] x = new {19,20,1,50,125,10,10,0};
→new[0]からnew[7]までの配列を作り、それぞれに値を代入する。

配列の使用方法

data[0] = 562;
→配列data[0]に562を代入する。

[arrays_sample01]　配列を使用した座標の指定



```
int [] x = { 50,61,83,69,71,50,29,31,17,39 };
int [] y = { 18,37,43,60,82,73,82,60,43,37 };

beginShape();
for (int i = 0; i < x.length; i++){
  vertex(x[i], y[i]);
}
endShape(CLOSE);
```

解説　x.lengthはxの配列の個数を表す。
このプログラムでは、”x.length”は10である。

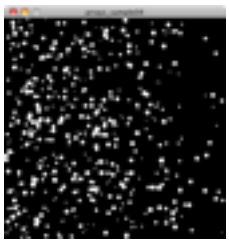
[arrays_sample02]　マウスポインタの位置を配列に記録し、残像として表示

```
int num = 50;
int[] x = new int[num]; //配列xを50個定義
int[] y = new int[num]; //配列yを50個定義

void setup(){
  size(200,200);
  noStroke();
  smooth();
  fill(255, 102);
}

void draw(){
  background(0);
  for(int i = num-1; i>0; i--){ //配列を一つ後ろにずらす
    x[i] = x[i-1];
    y[i] = y[i-1];
  }
  x[0] = mouseX;
  y[0] = mouseY;
  for(int i = 0;i<num;i++){
    ellipse(x[i],y[i],i/2.0,i/2.0);
  }
}
```

[arrays_sample03]　クリックの座標を配列に記録し、その位置に図形を表示

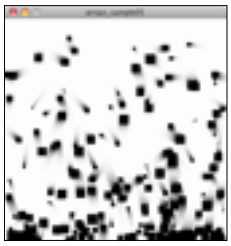
```
int num = 10;
int[] x = new int[num];
int[] y = new int[num];
int count = 0;

void setup(){
  size(400,400);
  noStroke();
  smooth();
  fill(255);
}

void draw(){
  background(0);

  for(int i = 0;i<num;i++){
    ellipse(x[i],y[i],10,10);
  }
}

// クリックによって配列xとyに現在のマウスの座標を代入
void mousePressed() {
  y[count] = mouseY;
  x[count] = mouseX;
  count++;

  if(count>=num){
    count = 0;
  }

}
```

[arrays_sample04]
配列によって複数の図形を描写（その1）



```
int ballCount = 500;
int ballSize = 8;
int ballSpeed = 3;
float[]xspeed = new float[ballCount];
float[]yspeed= new float[ballCount];
float[]xpos = new float[ballCount];
float[]ypos = new float[ballCount];
float[]wdth = new float[ballCount];
float[]ht = new float[ballCount];

void setup(){
  size(400, 400);
  background(0);

  //initialize values for all balls
  for (int i= 0; i< ballCount; i++){

    // set varied ball speed
    xspeed[i] = random(1, ballSpeed);
    yspeed[i] = random(-ballSpeed, ballSpeed);

    // ball varied ball sizes
    wdth[i]= random(1, ballSize);
    ht[i]= wdth[i];

    // set initial ball placement
    xpos[i] = width/2+random(-width/3,  width/3);
    ypos[i] = height/2+random(-height/3,  height/3);
  }

  noStroke();
  frameRate(30);
}

void draw(){

  background(0);

  for (int i=0; i<ballCount; i++){

    //draw balls
    ellipse(xpos[i], ypos[i], wdth[i], ht[i]);

    //upgrade position values
    xpos[i]+=xspeed[i];
    ypos[i]+=yspeed[i];

    /*conditionals:
     detects ball collision with sketch window edges
     accounting for ball thickness.
     */
    if (xpos[i]+wdth[i]/2>=width || xpos[i]<=wdth[i]/2){
      xspeed[i]*=-1;
    }
    if (ypos[i]+ht[i]/2>=height || ypos[i]<=ht[i]/2){
      yspeed[i]*=-1;
    }
  }
}
```

[arrays_sample05]
配列によって複数の図形を描写（その2）



```
// Bouncing Rectangles with Simple Physics III
int shapes = 200;
float[]w = new float[shapes];
float[]h = new float[shapes];
float[]x = new float[shapes];
float[]y = new float[shapes];
float[]xSpeed = new float[shapes];
float[]ySpeed = new float[shapes];
float[]gravity = new float[shapes];
float[]damping = new float[shapes];
float[]friction = new float[shapes];
//controls rate rects are born
float shapeCount;
float birthRate = .25;
// control width of spray when rects are born
float sprayWidth = 5;

void setup(){
  size(400, 400);
  noStroke();
  //initialize arrays with random values
  for (int i=0; i<shapes; i++){
    x[i] = width/2.0;
    w[i] = random(2, 17);
    h[i] = w[i];
    xSpeed[i] = random(-sprayWidth, sprayWidth);
    gravity[i] = .1;
    damping[i] = random(.7, .98);
    friction[i] = random(.65, .95);
  }
}

void draw(){
  //fade background
  fill(255, 100);
  rect(0, 0, width, height);
  fill(0);

  // shapeCount births rects over time
  for (int i=0; i<shapeCount; i++){
    rect(x[i], y[i], w[i], h[i]);
    x[i]+=xSpeed[i];
    ySpeed[i]+=gravity[i];
    y[i]+=ySpeed[i];

    //collision detection
    if (y[i]>=height-h[i]){
      y[i]=height-h[i];
      // bounce
      ySpeed[i]*=-1.0;
      // slow down vertical motion on ground collision
      ySpeed[i]*= damping[i];
      // slow down lateral motion on ground collision
      xSpeed[i]*=friction[i];
    }
    if (x[i]>=width-w[i]){
      x[i]=width-w[i];
      xSpeed[i]*=-1.0;
    }
    if (x[i]<=0){
      x[i]=0;
      xSpeed[i]*=-1.0;
    }
  }
  if (shapeCount<shapes){
    shapeCount+=birthRate;
  }
}
```

時間を扱う関数

millis()
プログラムがスタートしてからの時間を返す。単位はミリセカンド。（1000ミリ秒=1秒）

second() ：現時刻の秒を返す

minute() ：現時刻の分を返す

hour() ：現時刻の時間を返す

day() ：日付を返す

month() ：月を返す

year() ：西暦を返す

[time_sample01] プログラムを開始してから4秒経過するとアニメーションを開始

```
int x = 10;

void setup(){
  size(200,200);
}

void draw(){
  background(255);
  //プログラムを開始してから4秒以上経過場合
  if(millis() > 4000){
    x++;
  }
  line(x, 0, x, 200);
}
```

[time_sample02] 時間の経過によりアニメーションを変化させる

```
float x = 0;

void setup(){
  size(200,200);
  smooth();
}

void draw(){
  background(255);

  if(millis() < 5000){ //5秒未満の場合
    x=x+0.1;
  }else if(millis() < 7000){ //5秒以上7秒未満の場合
    x++;
  }else{  //7秒以上の場合
    x=x+0.1;
  }

  line(x, 0, x, 200);
}
```

[time_sample03] 日時を扱う関数を使用し

```
void setup(){
  size(600,600);
  smooth();
}

void draw(){
  background(255);

  float s = map(second(), 0, 60, 0, 600);
  float m = map(minute(), 0, 60, 0, 600);
  float h = map(hour(), 0, 24, 0, 600);

  line(s, 0, s, 200);
  line(m, 200, m, 400);
  line(h, 400, h, 600);
}
```

map()関数について

ある値の範囲を他の値の範囲に変換する時にmap()関数を使用します。

例)
data = map(x, a, b, c, d);
aからbまでの範囲内でのxの位置をcからdまでの値の範囲内に置き換える。