コンピュータ基礎演習

第10回

理工学部 情報科学科 隅田 康明

sumida@ip.kyusan-u.ac.jp

対面/遠隔での質問受付体制

- 7/13から教室で授業時間に対面でも質問受付開始予定
  - 本来の時限の受講者のみ入室可
    - 入室前に手洗い、消毒、座席表への記入
    - 私語厳禁、座席は完全固定
    - 指示に従わない場合には減点や欠席措置
  - 質問は遠隔でも受け付ける
    - Zoomの画面を教室のプロジェクターに投影
    - Zoomでの個別対応は難しくなるかも知れません

詳細はK‘sLifeで連絡の後に、Moodleに掲載

今後の予定

- 第10回：繰り返し(2)
  - 配列を使って沢山の図形を動かす
  - レポートとは別に、制作課題で何を作るか、を第11回開始までに提出する
- **第11回～第13回：制作課題の作成**
  - 3回分かけてプログラムとレポートを作成
  - 第10回までのプログラムをアレンジして提出
  - 独自性の高いプログラムを作りたい場合は、設計書を提出すること

レポートの締め切り (減点なしで受け付ける期間)

- 第9回：第10回授業日の16時まで
- 第10回：第10回の週の金曜日18:00まで
- 第10回(制作課題の内容)：第10回の週の金曜日18:00まで
  - ベースにする回を書くだけ（第4回と第10回を合わせる、など）
    - これはMoodleで提出。未提出の場合は制作課題の最高点を制限。
  - **制作課題設計書：7/9(木)の23:59まで**
    - 独自性の高い作品に挑戦する人のみ。Moodle＋メール提出。
- 制作課題：最終回授業日の18:00まで
- 追加課題：7月24日 18:00まで

- 遅れ提出の締め切り：7/24 18:00まで
  - Moodleでの提出も打ち切り
  - 以降は特別に認めた場合を除いて受け付けない
    - 余程の事情がない限りは認めない
  - 最終的なレポート締め切りと考えて問題ない

制作課題について

- 第11回～13回の3回でプログラムとレポートを作成
- 第10回までのプログラムのアレンジ
  - 追加課題のアレンジでも可
    - ただし、追加課題をアレンジする場合は、それぞれが別の作品とすること
  - 提出済みのレポートのアレンジでも可、
    ただし、元のプログラムからの変更点が少ない場合は、元のレポートを0点とした上で評価する
- 独自性の高いプログラム
  - 事前の設計書を必ず提出すること
    - 事前に設計書を提出すれば、事前にアドバイスを返す
  - 作れないものを作ろうとすると、単位を落とす危険がある
  - 最終的には自己責任だが、出来ないものを作ろうとしている場合は止めるので、その時は指示に従うこと

何を作るか？

- 基本的には、これまでのアレンジで問題ない
  - ただし、最低でも変数、条件分岐、繰り返しを使った工夫をすること（全てでなくても良い）
  - 最低3回分の作業時間をかけたレポートを提出すること
    - 当たり前だが、少しアレンジして終わり、などだと点数はほとんどでない
- 既に出したレポートの続きでも問題ない
  - 第8回のゲームをさらに改良する、など
  - ただし、同じものを出してきた場合には、元のレポートは0点とする(点数も当然低くなる)

自分で作ったプログラムの再利用

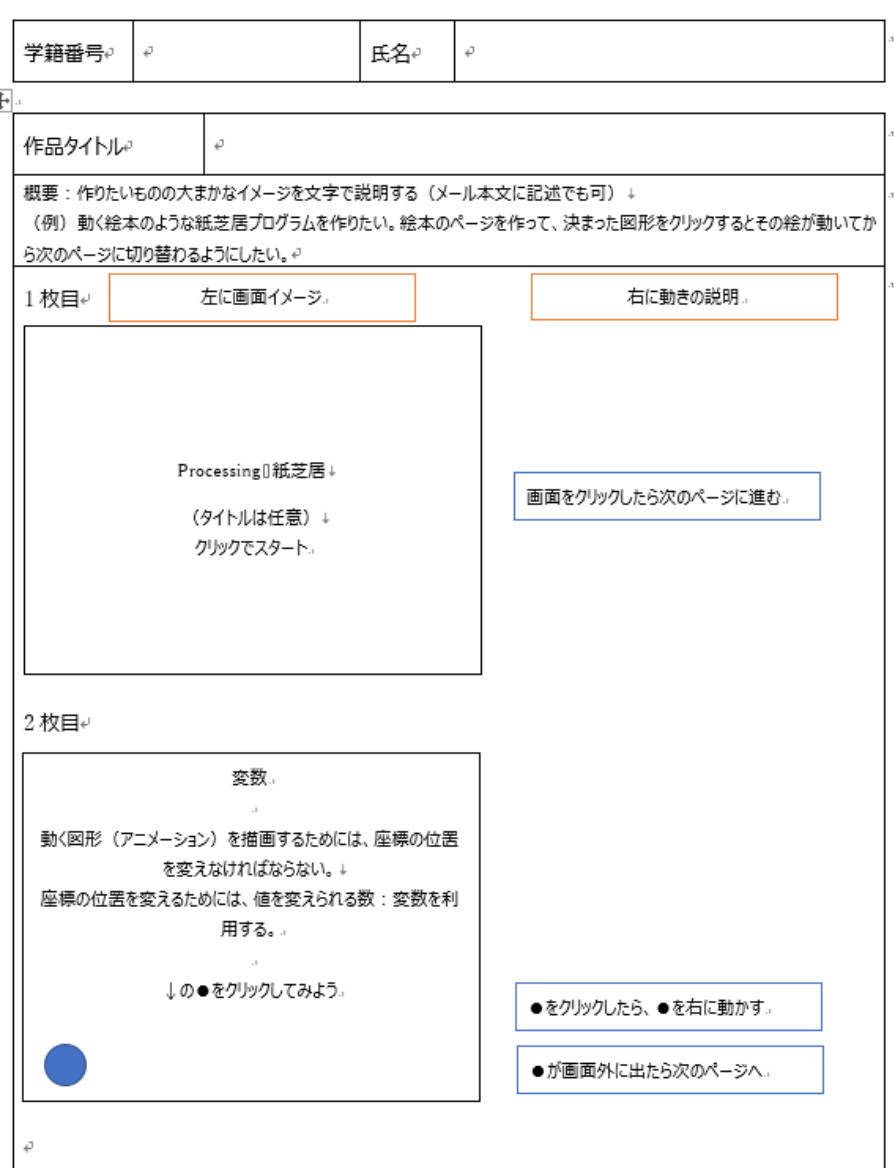
- 自分で書いたプログラムは積極的に利用してよい
  - ゲームの障害物に利用
  - 背景として利用
  - 動く図形を増やすときに利用など

- ただし、テキトウにコピーして貼り付けても上手くは動かない
  - 横着はせずに、必要な部分を必要なだけコピーして貼り付けること！

制作課題の設計書

- 手書きの紙を撮影した写真か、PDFファイルで提出
  - 写真を提出する場合は、なるべく歪みがないようにすること
    - 読めないものを送らない
  - PDFを作成するものは何でも良い
    - PDFにコメントを付けるので、編集制限は書けないこと
  - 設計書を提出＋設計書のイメージに近い作品を提出で、評点に+5点（それ自体が工夫点になるので）
    - 10回までのアレンジでも満点は取れます
      - ただし、工夫点を多くしないと難しい（独自作品でも同じですが）
- 設計書はメールで提出
  - 「PC基礎：制作課題設計書」の件名で送ること
    - 当日までに返信がなければ催促すること
      - メールで反応がなければOpenChatやZoomで

設計書の例



コンピュータ基礎演習□制作課題設計書

学籍番号		氏名	

作品タイトル	

概要：作りたいものの大まかなイメージを文字で説明する（メール本文に記述でも可）
（例）動く絵本のような紙芝居プログラムを作りたい。絵本のページを作って、決まった図形をクリックするとその絵が動いてから次のページに切り替わるようにしたい。

1 枚目　左に画面イメージ　右に動きの説明

Processing□紙芝居
（タイトルは任意）
クリックでスタート

画面をクリックしたら次のページに進む

2 枚目

変数

動く図形（アニメーション）を描画するためには、座標の位置を変えなければならない。
座標の位置を変えるためには、値を変えられる数：変数を利用する。

↓の●をクリックしてみよう

●をクリックしたら、●を右に動かす

●が画面外に出たら次のページへ

- これはWordで作った例
  - Wordのテンプレートは Moodleで配布
- 手書きでもOK
- PowerPointの場合
  - 1枚目：コンピュータ基礎演習制作課題設計書
  - 2枚目：作品タイトルと概要（説明文）
  - 3枚目～：画面のイメージと動きの説明

独自性の高い作品作りに挑戦する場合

- 作れそうにないものを作ろうとしない事
  - 完成しなければ、レポートを出せない　＝ 単位を落とす

- ある程度動く状態にまず持っていくことを優先する
  - 未完成でも一通りの動作が出来ていれば評価できる
  - こまめにバックアップを取っておくこと
    - Moodleに途中経過のレポートを出しておくと安全

- 妥協することも大事
  - まずはレポートを期限内に提出すること
  - こだわりたいなら、レポートを出してからでも良い
    - 2つ目を出しても評価はする

制作課題の評価

- 基本点(減点式)＋工夫点(加点式)で評価
- 基本点：最低基準を満たしていれば付く点数
  - 必要項目の不足や提出物の不足で減点する
- 工夫点：次のいずれかの工夫ごとに加点（レポートで説明）
  - プログラムの工夫
    - 変数、乱数、条件分岐、繰り返し、メソッド、座標変換、クラス、三角関数等の数式
  - 画像処理の工夫
    - アプリなどを使って画像を加工
      - どのアプリを使って、どんな工夫をしたのかを明記。どういった仕組みで加工されているのか、 Processingでも実現出来そうかの考察が必要
  - レポート自体の工夫
    - アニメーション、スライドマスタ、スライド背景にProcessingで作った画像を使うなど

工夫点の評価

- プログラムに書いただけでは点数は付かない
  - いくらすごいプログラムを書いても、
    説明できないなら点数は高くならない
- レポートで工夫点として説明したものを評価する
  - 自分で描いたプログラムを自分で説明する
  - 作った作品をアピールするつもりで工夫点を書いていこう

- レポートを作るのは第13回
  - まずはレポートのもとになるプログラムを作っていく

作成するプログラムについての注意

- 単位を落とす例
  - 不正（まず間違いなくばれます）
    - インターネット上のプログラムや、Processingの参考書のプログラムをコピー
      - ただし、事前に相談があればアレンジを認めることもある
    - 友人と(殆ど)同じプログラムを提出
      - 全員0点にします
  - 手抜き
    - 雛型やサンプルと同じプログラムを提出
      - 当然、0点です
    - レポートと同じプログラムを提出
      - その回のレポートに多くの時間をかけたのであれば、それなりの点数は取れるが、元のレポートが手抜きなら単位を落とす
    - そもそも、プログラムが短い
      - 最低でも200分程度の努力が見えない場合は0点
      - 短いプログラムでも、すごい工夫をしてきた場合は除く
        - まず無いケース。もしあれば追加で説明を求めます

何を作ったらいいか決められない場合

- 下記のどれかを選ぶこと

- **第6回：ジェネラティブアート入門**
  - 第7回の条件分岐を取り入れる
    - 角度や時間経過で形や色を変化
- **第8回：ゲーム作成**
  - 普通にゲーム性を高めていけば工夫点が増える
    - 敵を増やす、点数アップの仕組みを増やすなど
  - 雛型Aでレポートを出したなら、雛型Bを選ぶなども可
- **第9回：繰り返しで図形を並べる**
  - 円形模様の種類を増やす
    - それぞれ違う条件で変化させれば工夫点が増える
    - 自作メソッドも増えるので工夫点としやすい

注意：
ベース回のレポートを確認しておくこと

- Moodleでレポートを再確認しておくこと
  - 自分の端末上での確認だけではダメ

- 例えば、第4回と第6回のレポートをベースにする場合、採点するときには、当然その2回分を再確認する

- Moodle上からレポートを確認できなくなっていた場合には、そのレポートの評価を取り消すので、レポートが消えていないか確認しておくこと

小テストについての注意

- 評価方法を、最新の評価　→　最高点に変更
  - 復習のために最新評価としていましたが、
    新しい内容がなくなるために、第11回から変更

- 小テストは必ず全て受験し、点数を取っておくこと
  - 答えも出ているので、何度も解けば必ず正解出来る
  - 全て満点で10点になる
  - 必ず10点分の評点を稼いでおくこと
    - 最低でも6点は取っておかなければ単位が危ない
      - 最後のレポート6割、小テスト6割、他のレポート提出6回で何とか単位が習得できる点数（レポートが出来ていない場合は別ですが）

遠隔授業期間中の質問

- **困ったら早めに質問・相談！！**
  - 学生側から質問されないと、
    誰が困っているのか、何が分からないのか、分かりません
- メール：やり取りに時間はかかるが一番確実
- Zoom：授業時間中限定
  - 時間は限られるが、作業中の画面を見ながら教えられるので、問題を短時間で解決出来る可能性が高い
- Line OpenChat：授業時間中限定
  - 文字だけのやり取りに限定（画像アップロードは禁止）
  - 質問内容が他の学生にも分かるので注意すること
    - プログラムの全文貼り付け等は厳禁！

授業についての質問メールについて

[授業名(曜日時限)]についての質問	件名
～先生	誰宛か
[授業名(曜日時限)]を受講しています、 20AA999の九産太郎です。	何者か
（質問内容） --	用件
20AA999 九産太郎 九州産業大学　芸術学部　○○学科 1年	署名

2日返信がなければもう一度送って下さい（なるべく見落としたくない）

今回の授業内容

- 配列：繰り返しで沢山の図形を動かす
  - 100個の図形を動かすプログラム

- レポート内容
  - 雛型プログラムのアレンジ
  - これまでの内容を応用して、多くの図形の動きを変える

for文：繰り返し

- 構文

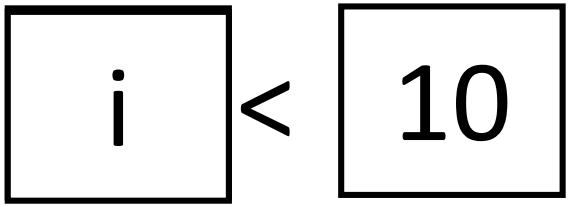
```
for(ループ変数の初期化; 条件; 更新) {
    繰り返したい処理;
}
```

- ループ変数の初期化：変数を宣言し、初期値を代入
  - 例）int i = 0; //int型の変数iを宣言し0を代入
- 条件：繰り返す条件を論理式で書く
  - 例） i < 10; // iが10より小さい間繰り返す
- 更新：ループ変数を増やす、減らすなど
  - 例）i++ //iを1増やす

繰り返しの例

- i を0から10まで 1 ずつ増やして表示する

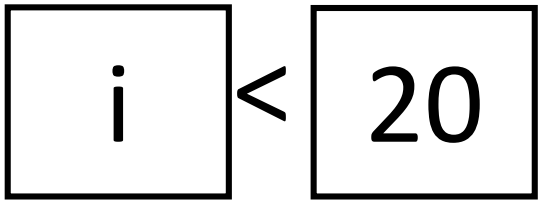
```
for(int i = 0; i < 10; i++) {
  print(i + " ");
}
```



i と 10を比較

- x を0から20まで 2 ずつ増やして表示する

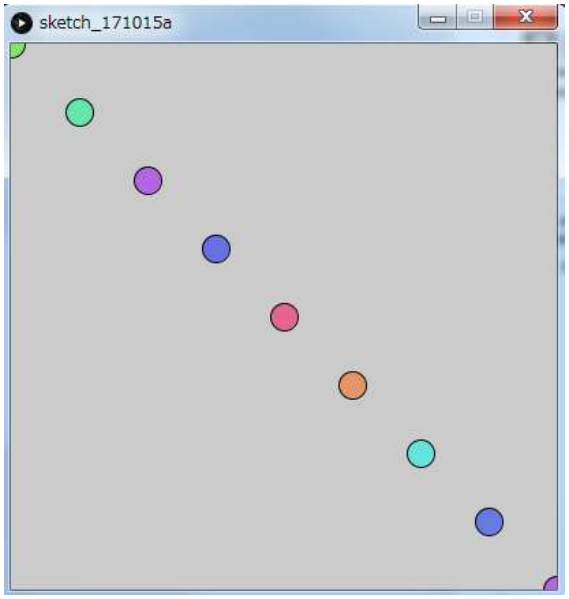
```
for(int x = 0; x < 20; x+=2) {
 print(x + " ");
}
```

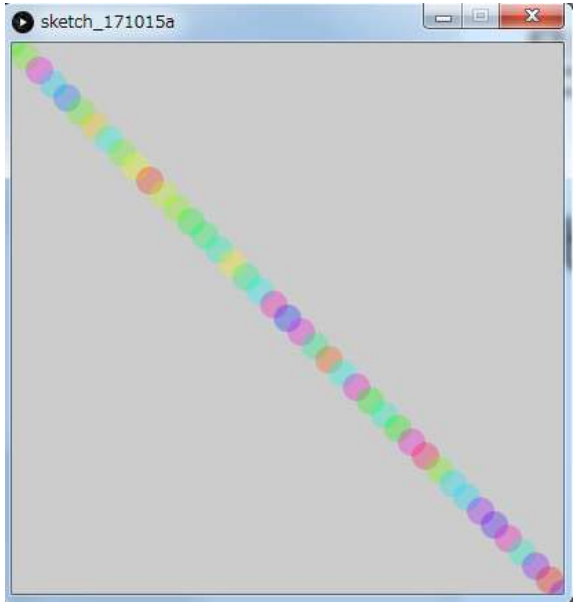


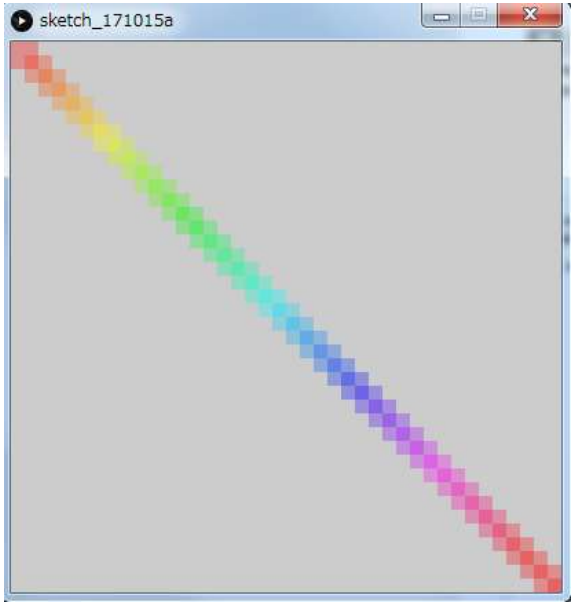
i と 20を比較

問題：斜めに図形を並べてみる

- ヒント
  - y座標が固定されていたら、
    縦方向の座標はいつもに同じ
  - y座標にもx座標と同じ値を使ってみよう
- 小テストの問題：自分でも実行して試してみよう





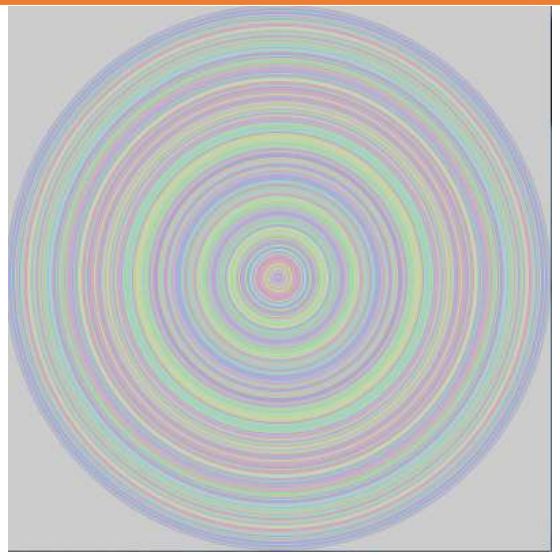


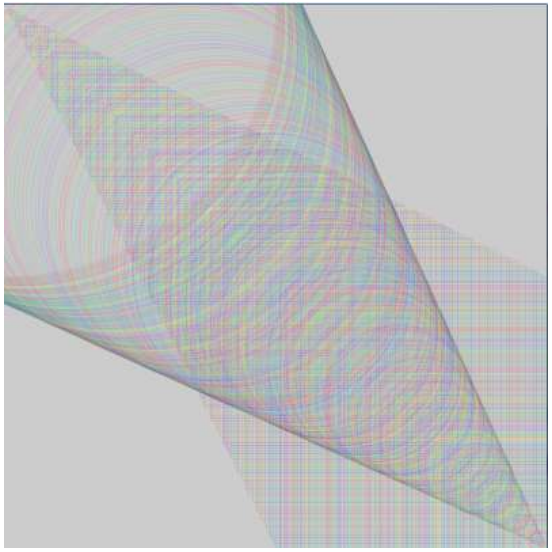
応用例：斜めに並べて配置する、図形のサイズを大きくする（小さくする）

```
size(400,400);
noFill();
int h = 0;
for (int x = 0; x <= width; x+=5) {
  stroke(h, 100, 100, 50);
  rect(x, x, 5, 5);
  h+=5;
}
```

y座標もx座標と同じだけずれていく

問題：繰り返しで図形を配置

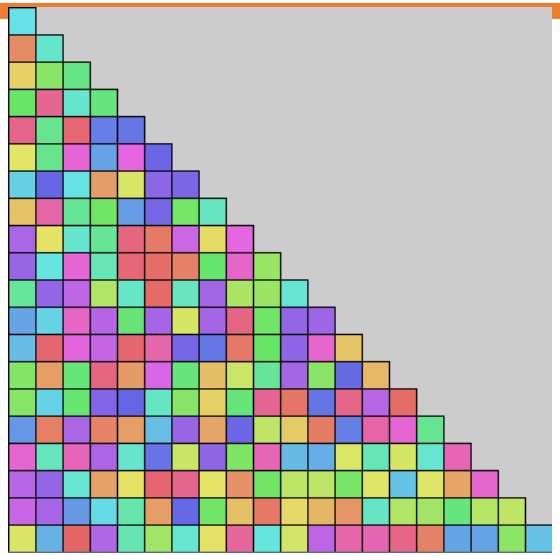




- 問題1：大きくなる円
  - 中心座標は固定
  - 幅、高さをループ変数に

- 問題2：斜め配置の組み合わせ
  - 繰り返しを2回書く
    - 左上から右下への四角
    - 左上から右下への丸

問題3：階段状に配置

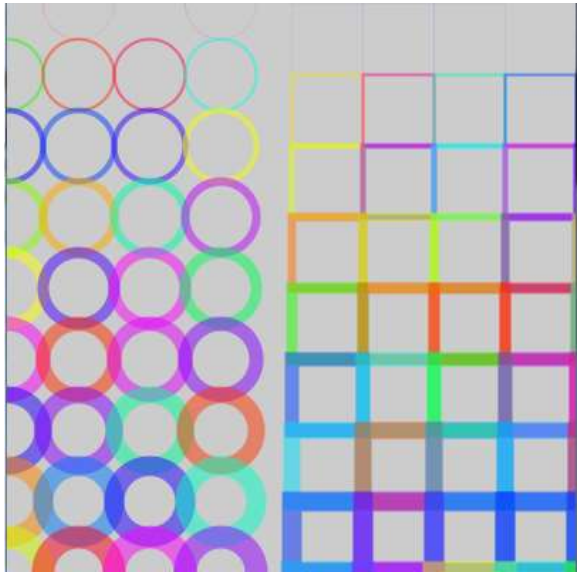


- 原型は2重ループのプログラム
- xの繰り返し条件がポイント
- 1段目は1個分の幅まで
- 2段目は2個分の幅まで
- 3段目は3個分の幅まで
- y段目は??個分の幅になる？

```
for(int y = 0; y <= height; y+=20) {
      for(int x = 0; x <= ???; x+=20) {
        rect(x , y , 20 , 20);
      }
}
```

??? が0,20,40,60,,,と増えていくと階段状になる

問題４：左右で違う図形を配置



- 2重ループを2回書く
  - 左側の丸の繰り返し
  - 右側の四角の繰り返し

図形を描き始めるx座標を変えれば・・・

3分割・4分割などにすれば制作課題の工夫点にもなる

配列：沢山の変数を作る、使う

複数の図形を動かす→変数も複数必要

- 8個の図形を動かそうとすると・・・

```
int x1, y1, vx1, vy1;
int x2, y2, vx2, vy2;
int x3, y3, vx3, vy3;
int x4, y4, vx4, vy4;
int x5, y5, vx5, vy5;
int x6, y6, vx6, vy6;
int x7, y7, vx7, vy7;
int x8, y8, vx8, vy8;
```

- 座標用や速度用などの変数が8個ずつ必要
  - for文で同じ動きをさせる場合は除く

複数の図形を動かす＝処理も複数必要

```
void draw(){
  background(255);
  ellipse(x1,y1,10,10);
  if(x1 <= 0 ) {
    vx1 *= -1;
  }
  if(x1 >= width) {
    vx1 *= -1;
  }
  x1 += vx1;
  ellipse(x2,y2,10,10);
 if(x2 <= 0 ) {
    vx2 *= -1;
  }
  if(x2 >= width) {
    vx2 *= -1;
  }
  x2 += vx2;

  ･･･これを後6個分書かないといけない
}
```

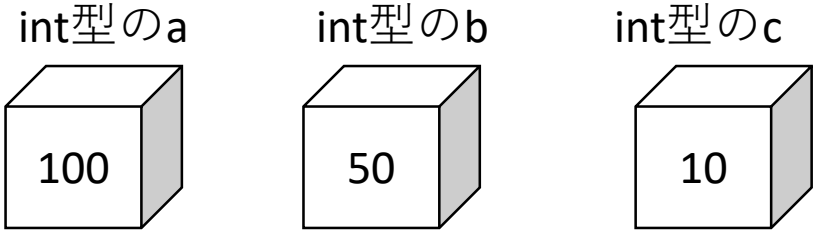
8個ではなく100個なら･･･？

同じようなことを100回書かなければいけない

同じようなことなら、繰り返しで！

配列：沢山の箱を一まとめに

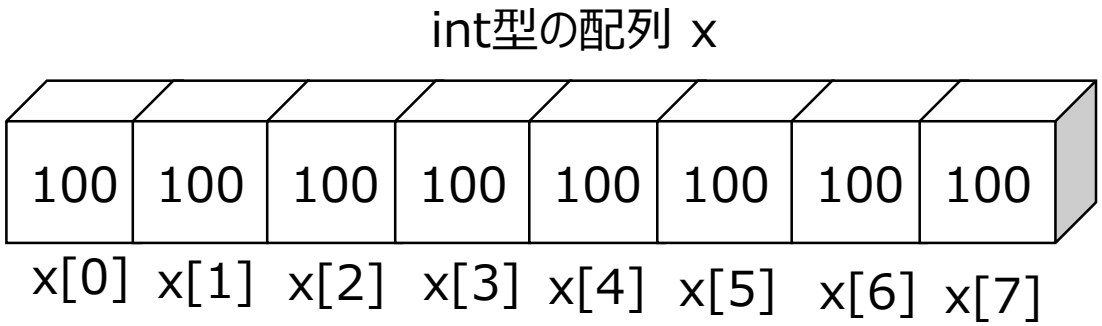
- 変数：データを入れることが出来る箱



箱は一つずつ

10個のデータを扱いたければ、
変数も10個作らないといけない

- 配列：一つの大きな箱に複数の箱を入れたもの



複数の箱を一まとめに

名前[番号] で管理する

配列の宣言（配列を作る）

- 変数の宣言：変数の型 変数名; （ int x; ）

- 配列の宣言

  変数の型[] 配列名 = {データ1, データ2, …};

  int [] x = {5, 8, 20, 100, 49, 62, 1, 40};

- 配列を使う：配列名[番号]

  ellipse(x[0], 10, 10, 10);

  ellipse(x[1], 10, 10, 10);

  ellipse(x[2], 10, 10, 10);

配列の番号は0から！

宣言が楽になっただけ？

for文と配列の組み合わせ

- for文を使えば、配列を効率的に扱える

```
ellipse(x[0],  0, 10, 10);
ellipse(x[1], 10, 10, 10);
ellipse(x[2], 20, 10, 10);
ellipse(x[3], 30, 10, 10);
ellipse(x[4], 40, 10, 10);
ellipse(x[5], 50, 10, 10);
ellipse(x[6], 60, 10, 10);
ellipse(x[7], 70, 10, 10);
```

x.length は配列xの長さ（今は8)

```
for ( int i = 0; i < x.length; i++ ) {
  ellipse(x[i], i * 10, 10, 10);
}
```

たった3行で書ける！

宣言と初期化でもfor文を使うと・・・

- 最初に配列の箱を100個作る

```
int [] x = new int[100];
```

箱の数

- for文で初期値を入れる

```
for ( int i = 0; i < x.length; i++ ) {
  x[i] = i * 2;  //箱の中身を2ずつ大きくする
}
```

```
for ( int i = 0; i < x.length; i++ ) {
  x[i] = random(100); //箱の中身をランダムにする
}
```

この書き方なら100でも1000でも一気に作れる

100個のボール

- 100個のボールを描く

```
float[] x = new float[100];  // float型の配列x を100個宣言
float[] y = new float[100];  // float型の配列y を100個宣言
void setup() {
  size(400, 400);
  for(int i = 0; i < x.length; i++) {
    x[i] = random(width);
    y[i] = random(height);
  }
}
void draw() {
  background(0,0,0);
  for(int i = 0; i < x.length; i++) {
    ellipse(x[i], y[i], 10, 10);
  }
}
```

変数初期化時にランダム指定
（描く場所をバラバラにする）

それぞれを描画

100個のボールを動かす

- 100個のボールを描く

```
float[] x = new float[100];  // float型の配列x を100個宣言
float[] y = new float[100];  // float型の配列y を100個宣言
void setup() {
  …
}
Void draw() {
  background(0,0,0);
  for(int i = 0; i < x.length; i++)
    ellipse(x[i], y[i], 10, 10);
    x[i]+=1;
    y[i]+=2;
}
}
```

変数初期化時にランダム指定
（描く場所をバラバラにする）

それぞれを描画

100個のボールを跳ね返す(setup)

```
float[] x = new float[100];  // float型の配列x を100個宣言
float[] y = new float[100];  // float型の配列y を100個宣言
float[] vx = new float[100];  // float型の配列x を100個宣言
float[] vy = new float[100];  // float型の配列y を100個宣言
void setup() {
  size(400, 400);
  for(int i = 0; i < x.length; i++) {
    x[i] = random(width);
    y[i] = random(height);
    vx[i] = random(3) – 1.5;  //-1.5 ~ 1.5 までの乱数
    vy[i] = random(3) – 1.5;  //-1.5 ~ 1.5 までの乱数
  }
}
```

vx[]とvy[]を作って初期化する

100個のボールを跳ね返す(draw)

```
void draw() {
  background(0,0,0);
  for(int i = 0; i < x.length; i++) {
    ellipse(x[i], y[i], 10, 10);
    x[i] += vx[i];
    if(x[i] >= width || x[i] < 0){
      vx[i]*=-1;
    }
    //y[i] についても同じように書いて、上下左右に跳ね返そう

  }
}
```

枠外に出たらvx[i]の符号を反転（逆に動くようにする)

今回のレポート：跳ね返るボールのアレンジ

- 沢山の図形を動かすプログラムを作成
- 配列の初期化、インスタンス型配列の使い方、これまで作ったプログラムを配列に応用

- 次回からの課題の引き出しづくりの意味合いもある
  - たくさんの図形を動かすパターンを覚えるつもりで取り組もう

アレンジのヒント

- 色を変えてみる
- 図形を変えてみる
- 組み合わせ図形を動かす
- 動きを変えてみる

- 跳ね返ったタイミングで
  - 色を変えてみる
  - 図形を変えてみる
  - 動きを変えてみる

レポートチェックリスト（第10回）

- ミニテストを受験した(レポートの前と後)
- 雛型のプログラムを完成させた(上下にも跳ね返るようにした)
- 雛型プログラムをアレンジした
  - 色を変えた
  - 図形を変更した
  - 第11回に何を作るのかを決めた
- PowerPointでレポートを作成した
  - タイトル、作品紹介(工夫点)、プログラムの3枚
- Moodleでレポートを提出した

各図形の種類を変える

- iの数によって、書く図形を変える

```
void draw() {
  background(0,0,0);
  for(int i = 0; i < x.length; i++) {
    if( i < 50){  //もしも,iが50未満なら(0番～49番までなら)
      ellipse(x[i], y[i], 10, 10);  //丸を描く
    } else {  //そうでないなら(50番以降なら)
      rect(x[i], y[i], 10, 20);  //四角を描く
    }
    x[i] += vx[i];
    if(x[i] >= width || x[i] < 0){
      vx[i]*=-1;
    }
    //y[i] についても同じように書いて、上下左右に跳ね返そう
  }
}
```

応用：複数図形との当たり判定


```
void draw() {
  background(0,0,0);
  for(int i = 0; i < x.length; i++) {
    if( i < 50){  //もしも,iが50未満なら(0番～49番までなら)
      ellipse(x[i], y[i], 10, 10);  //丸を描く
    } else {  //そうでないなら(50番以降なら)
      rect(x[i], y[i], 10, 20);  //四角を描く
    }
    x[i] += vx[i];
    if(x[i] >= width || x[i] < 0){
      vx[i]*=-1;
    }
    //y[i] についても同じように書いて、上下左右に跳ね返そう
    if( checkCollision(x[i], y[i], 10, t1x, t1y, 20) ) {
      text(“gameOver”, 100, 100);
    }
  }
}
```

配列の応用例：尾を引く軌跡を残す

```
int [] xArray = new int [50];
int [] yArray = new int [50];
int fillColor = 200;
void setup() {
  size(600, 600);
  smooth ();
  background(0);
}
//前に描いた部分を徐々に消していくメソッド
void fadeOut() {
  noStroke();
  fill(0, 30);  //背景色と同じ色で半透明
  rectMode(CORNER);
  //全体を塗り潰す
  rect(0, 0, width, height);
}
```

```
void draw() {
  //配列のデータをスライドする
  for ( int i = 0 ; i < xArray.length-1 ; i++){
    xArray[i] = xArray[i+1];
    yArray[i] = yArray[i+1];
  }
  //配列の最後の要素をマウス座標に
  xArray[xArray.length-1] = mouseX;
  yArray[yArray.length-1] = mouseY;
  //丸を小さくしながら軌跡を描く
  for ( int i = 0 ; i < xArray.length-1 ; i++){
    fill(50, 50, fillColor--, 50); //少しずつ色を変える
    ellipse(xArray[i], yArray[i], i, i+5);
  }
  fillColor = 200; //色を元に戻す
  fadeOut();
}
```

加点項目

- 複数の円形模様メソッドを作成した
  - 1つの画面内に2つ以上の円形模様を繰り返しで表示
- 円形模様の並びを、雛型以外の方法で並べた
  - 斜めに配置、円状に配置、交互に配置、階段状など

- 繰り返し分を複数書き、1画面の中で複数の並べ方をした
  - forが沢山ある
- 前回のプログラムの敵キャラに円形模様を使った
  - これは別のプログラムとして提出