プログラミング基礎

配列 (array)

今回の内容

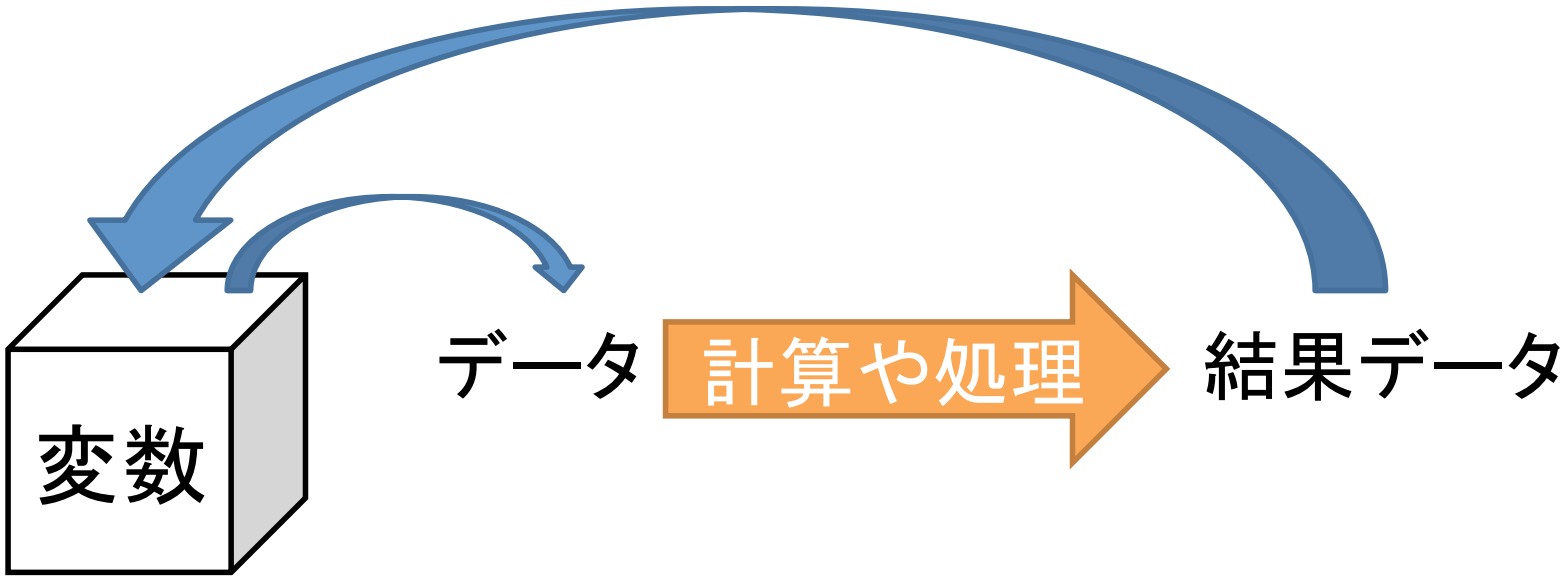
- 配列
- 乱数 (random 関数)
- 変数の有効範囲 (scope)

- 課題 : 棒グラフを描くプログラム

- 配列と画像を使ったアニメーション

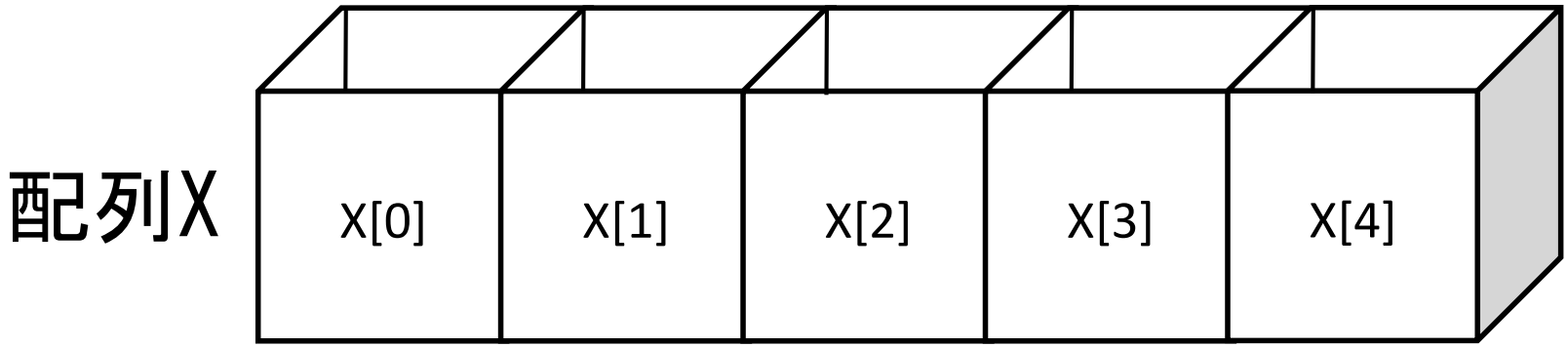
復習：変数（variable）

- データを記憶しておくための領域
  - 領域ごとに異なる名前を付けて区別する
- 変数を使うプログラムの基本的な流れ：



配列 (Array)

- たくさんの変数をひとまとめにしたもの
- 番号が振られている
- 配列にも型がある



配列の作り方・使い方

```
int[] scores = new int[3];

scores[0] = 88;
scores[1] = 80;
scores[2] = 76;

for(int i = 0; i < scores.length; i = i + 1){
  println(scores[i]);
}
```

【書式】 型[] 配列名 = new 型[個数];

【書式】 配列名[番号]

最初の番号は 0
最後は 個数-1

繰り返しと一緒に使うことが多い！

【書式】 配列名.length

```
int[] scores = {88, 80, 76};
```

と書いても良い

練習

- 練習１: int型の配列の最大値を求める関数
  - 引数の型は int[]　結果の型は？
  - 繰り返しを使います
  - 途中段階の最大値を覚えておく変数を使います
- 練習２: int型の配列の平均値を求める関数
  - 結果の型は float

random 関数（乱数）

- サイコロを振ったときのように適当な結果をランダムに返してくれる機能

```
int[] scores = new int[5];

for(int i = 0; i < scores.length; i++){
  float f= random(6); // 0 以上 6 未満の乱数
  scores[i] = ceil(f); // f の小数点以下を切り上げ
}

for(int i = 0; i < scores.length; i++){
  println(scores[i]);
}
```

変数の有効範囲

関数の「外」で配列を作る

```
int[] scores;

void setup(){
  scores = new int[10];

  for(int i = 0; i < scores.length; i++){
    float f= random(50, 100);
    scores[i] = ceil(f);
  }
}

void draw(){
  for(int i = 0; i < scores.length; i++){
    line(i * 10, 100, i * 10, 100 - scores[i]);
  }
}
```

プログラム全体でscoresが使える

関数の「中」で配列を作る

```
void setup(){
  int[] scores = new int[10];

  for(int i = 0; i < scores.length; i++){
    float f= random(50, 100);
    scores[i] = ceil(f);
  }
}

void draw(){
  for(int i = 0; i < scores.length; i++){
    line(i * 10, 100, i * 10, 100 - scores[i]);
  }
}
```

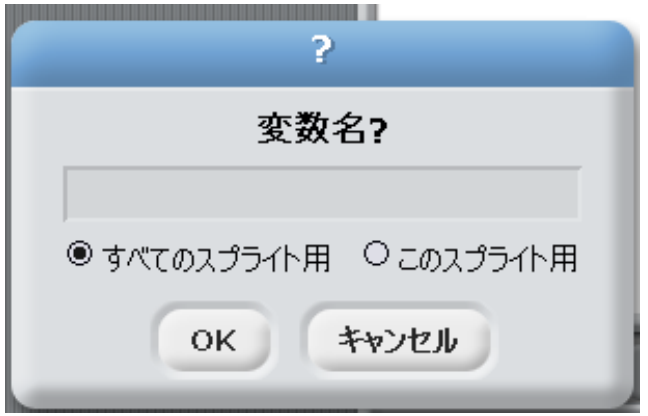
Cannot find anything named “scores”

draw関数ではscoresが使えない！

変数の有効範囲（補足）

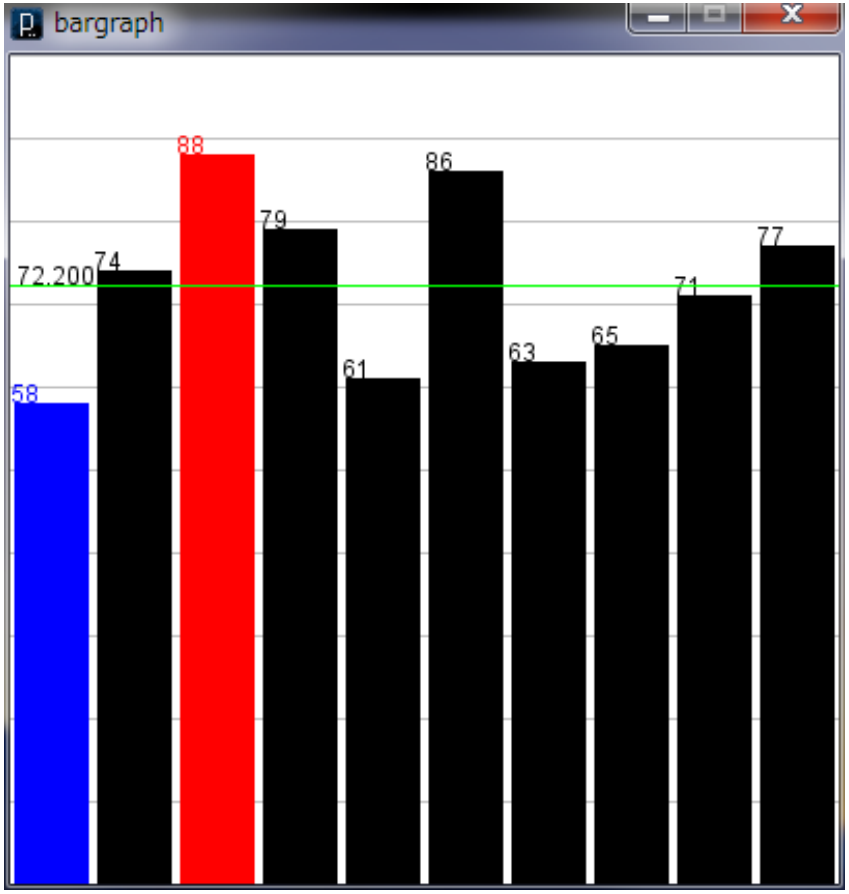
- 有効範囲は { … } で区切られる
  - 関数以外にも、for, while, if 等でも区切られる
- 変数・配列どちらも有効範囲の考え方は同じ
- 英語では scope

- [関連] Scratch の変数



課題3：点数の棒グラフ

- 要件
  - 点数の配列は乱数で初期化する
  - 最高点：赤、最低点：青、その他：好きな色
  - 平均点の横線：緑



課題のヒント

- 最高･最低･平均点を計算する関数を作る
- setup関数
  - 点数の配列を乱数で初期化する
  - 最高･最低･平均点を計算して変数に入れておく
- draw関数
  - 繰り返しと条件分岐を使って描く

パラパラ漫画風アニメーション
（次回の課題で使います）

Processingで画像を扱う

1. プログラムで使いたい画像ファイルを用意 (gif, jpg, png)
2. プログラムに名前を付けて保存する
3. メニューから Sketch > Add File… から用意した画像を追加する
4. loadImage 関数でプログラムに読み込む
   - Pimage 型の変数に入れる
5. image 関数で描画する

Processingで画像を扱う

（先に前のスライドの１～３をしておくこと）

```
PImage img;

void setup(){
  size(400, 400);
  img = loadImage("cat1-a.gif");
}

void draw(){
  background(255);
  image(img, mouseX, mouseY);
  image(img, mouseX, mouseY, img.width / 2, img.height / 2);
}
```

配列と画像を使ったアニメーション（パラパラ漫画風）

```
String[] filenames = {"cat1-a.gif", "cat1-b.gif"};
PImage[] images;

void setup(){
  size(95, 111); // ウィンドウのサイズを画像に合わせている
  frameRate(2); // 1秒間にdrawを実行する回数を設定

  images = new PImage[filenames.length];
  for(int i = 0; i < filenames.length; i = i + 1){
    images[i] = loadImage(filenames[i]);
  }
}

void draw(){
  background(255);
  int frame = frameCount % images.length;
  image(images[frame], 0, 0);
}
```

※画像の扱いについては
次のスライドを参照

frameCount
= draw 関数が実行された回数

image関数

- image(img, x, y)
- image(img, x, y, width, height)
- img
  - 描画したい PImage
- x, y
  - 描画する場所（左上の点）
- width, height
  - 描画する大きさ
  - 省略した場合は元画像の大きさになる