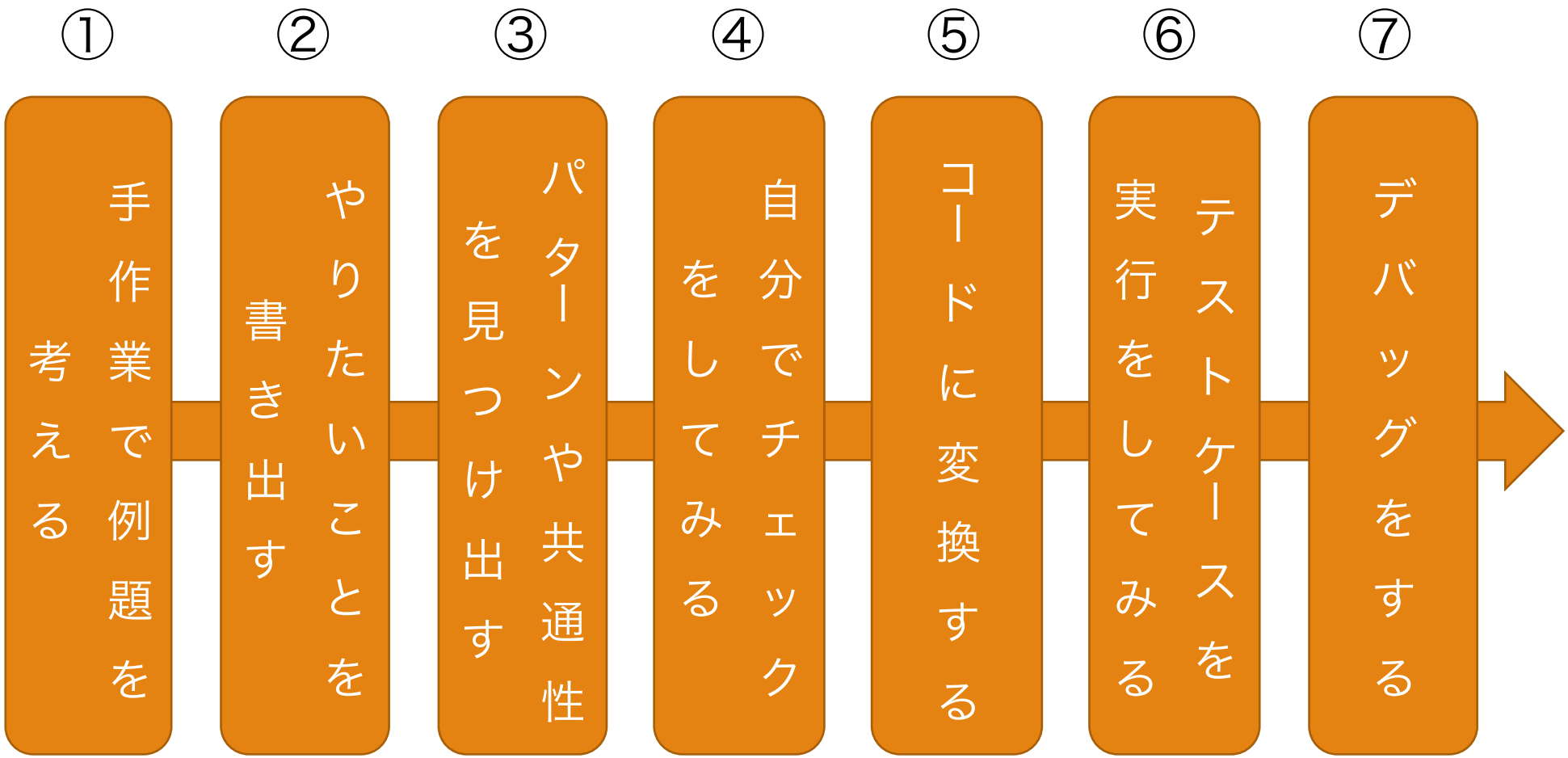
情報メディア基盤ユニット

5月10日

プログラムの作り方と条件分岐処理

情報メディア学科佐藤尚

プログラムを作るための7つのステップ



①手作業で例題を考える

手作業で例題を考える

小さな（簡単な）例題を
手で解いてみる

状況を表す図（絵）を
描いてみる

問題や状況は
明確になったか？

関連知識が必要かも？

②やりたいことを書き出す

やりたいことを書き出す

やりたいことの手順を
きちんと書き出す

③パターンや共通性を見つけ出す

パターンや共通性を見つけ出す

それぞれの場合のアルゴリズムを考える

パターンや共通性を見つけ出す

条件分岐処理、繰り返し処理、変数などが利用出来ないかを考える

このステップが難しければ、①や②に戻って考える

別な入力や状況ではどうようになるか？

④自分でチェックをしてみる

自分でチェックをしてみる

考えもらしたパターンや状況がないかを考える

他の入力や状況でのチェックをする

⑤コードに変換する

コードに変換する

考えたアルゴリズムを
コードに変換する

この授業では
Processingを使う

①〜④までの処理は他の
言語を利用する場合でも
同じ

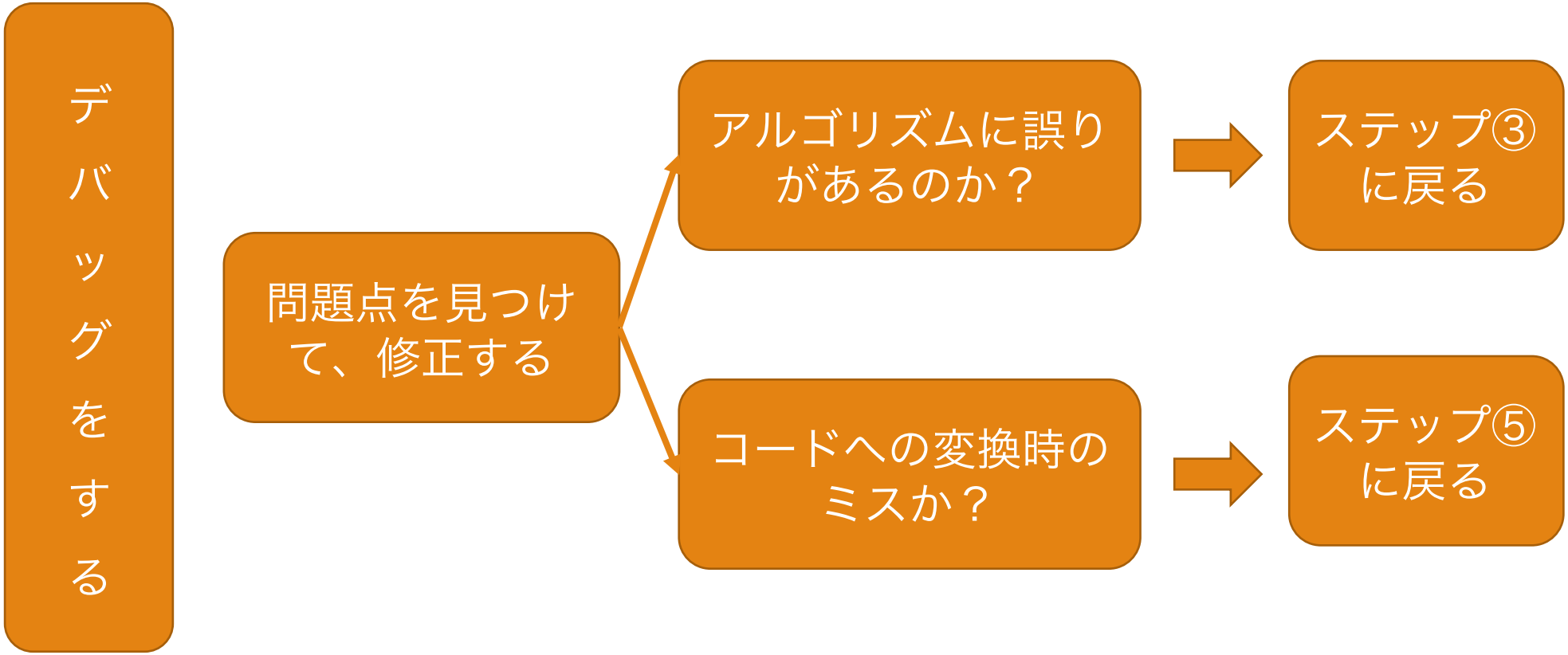
⑥テストケースを実行をしてみる

テストケースを実行をしてみる

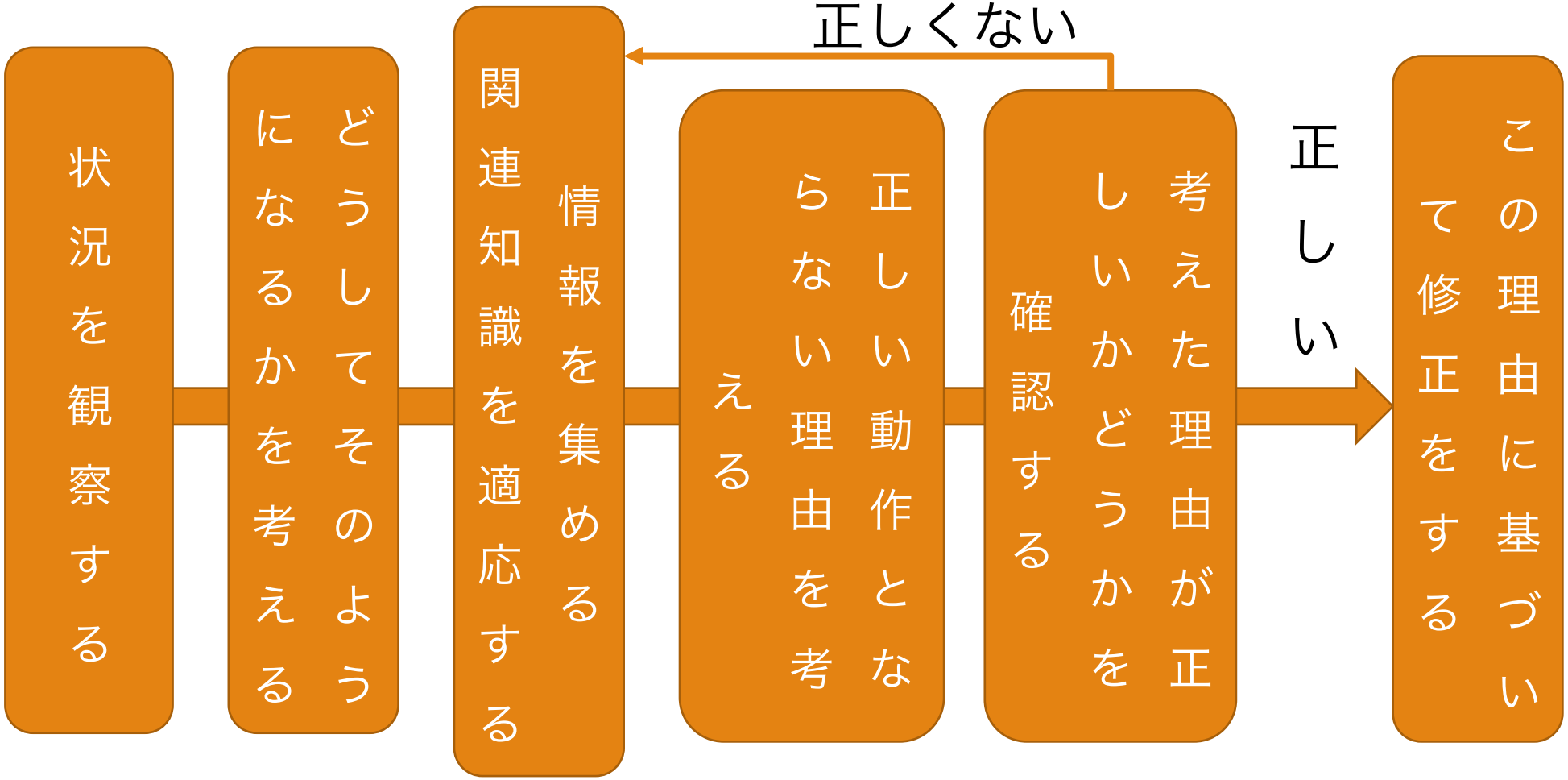
プログラムを
実行してみる

実行結果が正しいかを
確認する

デバッグをする


デバッグをする
問題点を見つけて、修正する
アルゴリズムに誤りがあるのか？
ステップ③に戻る
コードへの変換時のミスか？
ステップ⑤に戻る

デバッグの手順


状況を観察する
どうしてそのようになるかを考える
関連知識を適応する
情報を集める
正しい動作とならない理由を考える
考えた理由が正しいかどうかを確認する
正しくない
正しい
この理由に基づいて修正をする

情報を集める

関連知識を適応する

情報を集める

println関数

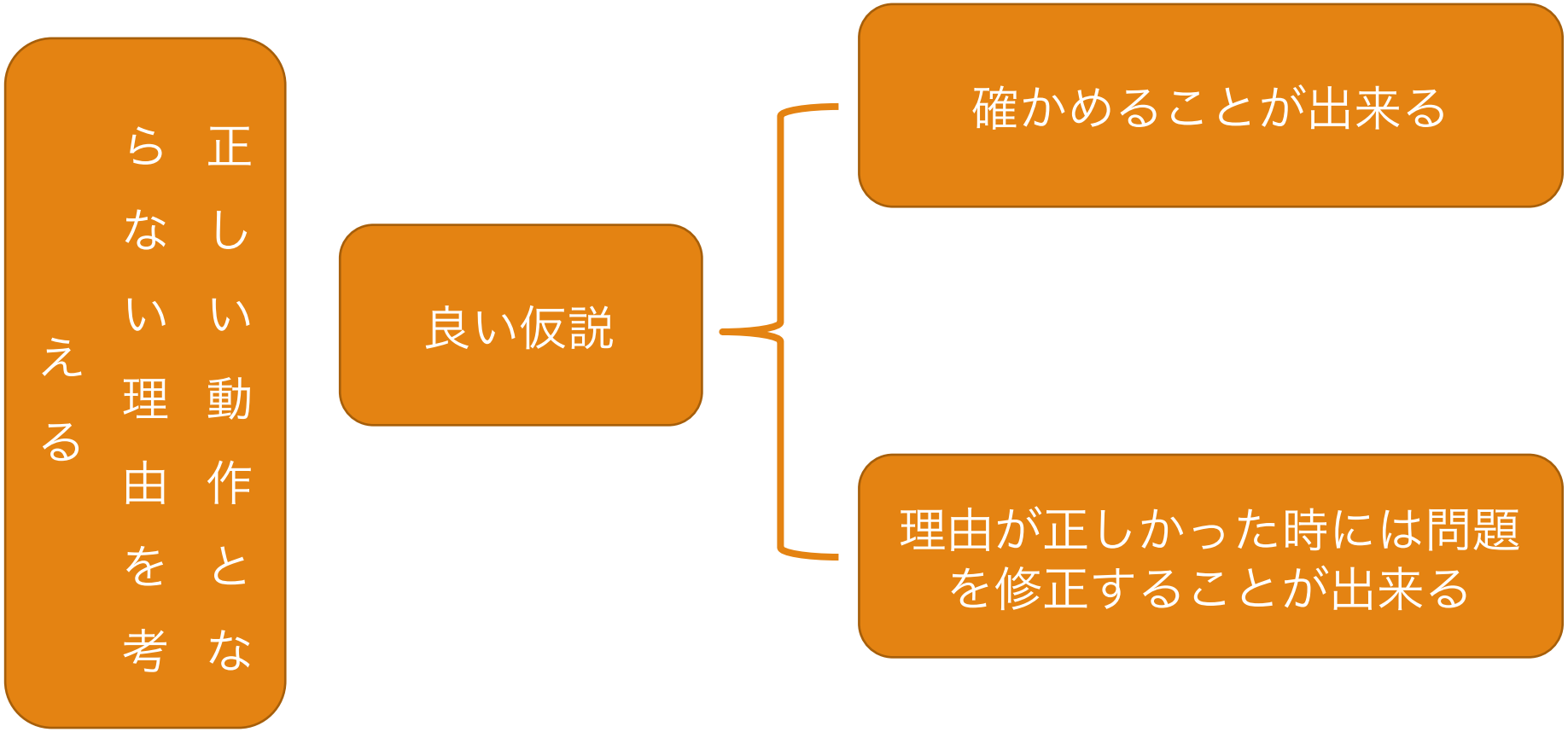
デバッグツールを使う

別の機会に紹介する予定

コードの命令を自分で実行してみる

前回のクイズのようにやってみる

正しい動作とならない理由（仮説）を考える



良い仮説、悪い仮説

悪い仮説

プログラムは間違っている

良い仮説

5行目に問題がある

5行目の条件判定に間違いがある

Ever better

良い仮説、悪い仮説

良い仮説

5行目に問題がある

5行目の条件判定に間違いがある

Ever better

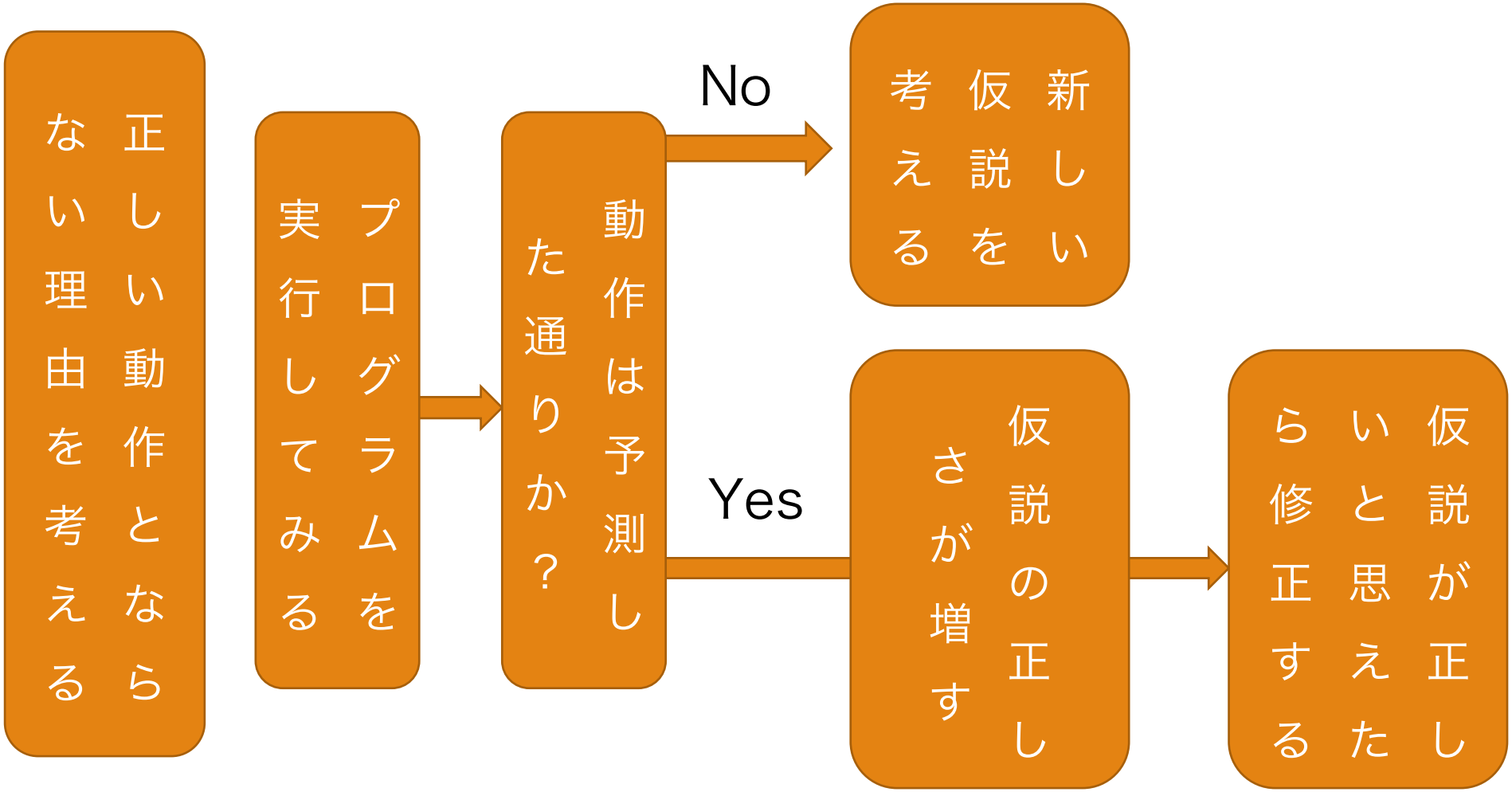
5行目の条件判定の不等号の向きが反対

Very Good

テスト可能

修正可能

正しい動作とならない理由（仮説）を考える



問題

ウインドウの端に円が衝突したときに跳ね返るような処理

考えるべきこと

跳ね返る処理はどのようにする？

円を動かすためにはどのようにするか？

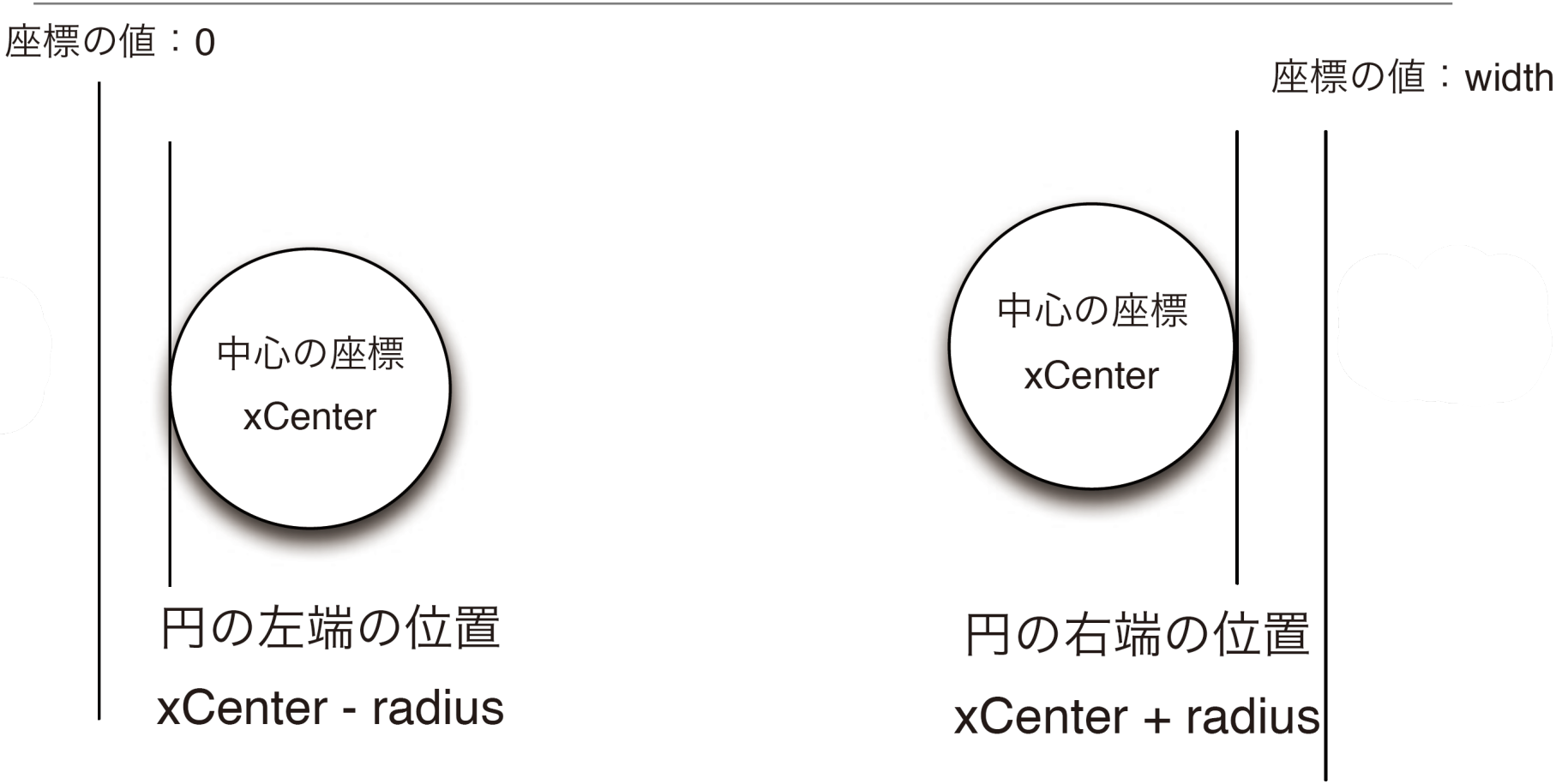
- 円の移動方向は水平方向だとする。
- 円の中心の座標を変化させればよい。
- xCenter　←変化する
- yCenter　←変化しない
- 移動速度
- speed
- 円の大きさ
- 半径：radius

考えるべきこと

跳ね返る処理はどのようにする？

- 移動方向は水平方向にすると決めた
- ぶつかるのは左端か右端

円を動かすためにはどのようにするか？


座標の値：0
座標の値：width
中心の座標
xCenter
中心の座標
xCenter
円の左端の位置
xCenter - radius
円の右端の位置
xCenter + radius

背景（関連）知識

円は中心から距離が
同じ点の集まり

ウインドウの左端
$x = 0$

Y軸に平行な直線 $x = x_0$ と
点の (x_1, y_1) 距離は $|x_1 - x_0|$

ウインドウの右端
$x = width$

絶対値

絶対値を計算する仕組み

abs：関数

やりたいこと

ウインドウの左端に球がぶつかったか？

ウインドウの左端に球がぶつかったら、移動方向を左から右にする

ウインドウの右端に球がぶつかったか？

ウインドウの右端に球がぶつかったら、移動方向を右から左にする

ちょっと冷静に考えると

xCenterの値は
0以上

$$|xCenter - 0| = xCenter$$

xCenterの値は
width以下

$$|xCenter - width| = width - xCenter$$

abs(xCenter - 0)< radius

は

xCenter < radius

abs(xCenter - width) < radius

は

width-xCenter < radius

width - radius < xCenter

xCenterを主語のように考えると

xCenter > width - radius

やりたいこと

ウインドウの左端に球がぶつかったら、移動方向を左から右にする

円の中心のX座標の値を増やすようにする

ウインドウの右端に球がぶつかったら、移動方向を右から左にする

円の中心のX座標の値を減らすようにする

やりたいことの別の言い方

ウインドウの端に球がぶつかったら、移動方向を反対向きにする

speedの値を-1倍する

- ウインドウの左端に球がぶつかる
- ウインドウの右端に球がぶつかる

ウインドウの端に球がぶつかったら、移動方向を反対向きにする

- ウインドウの左端に球がぶつかる
- ウインドウの右端に球がぶつかる

プログラムの構成要素

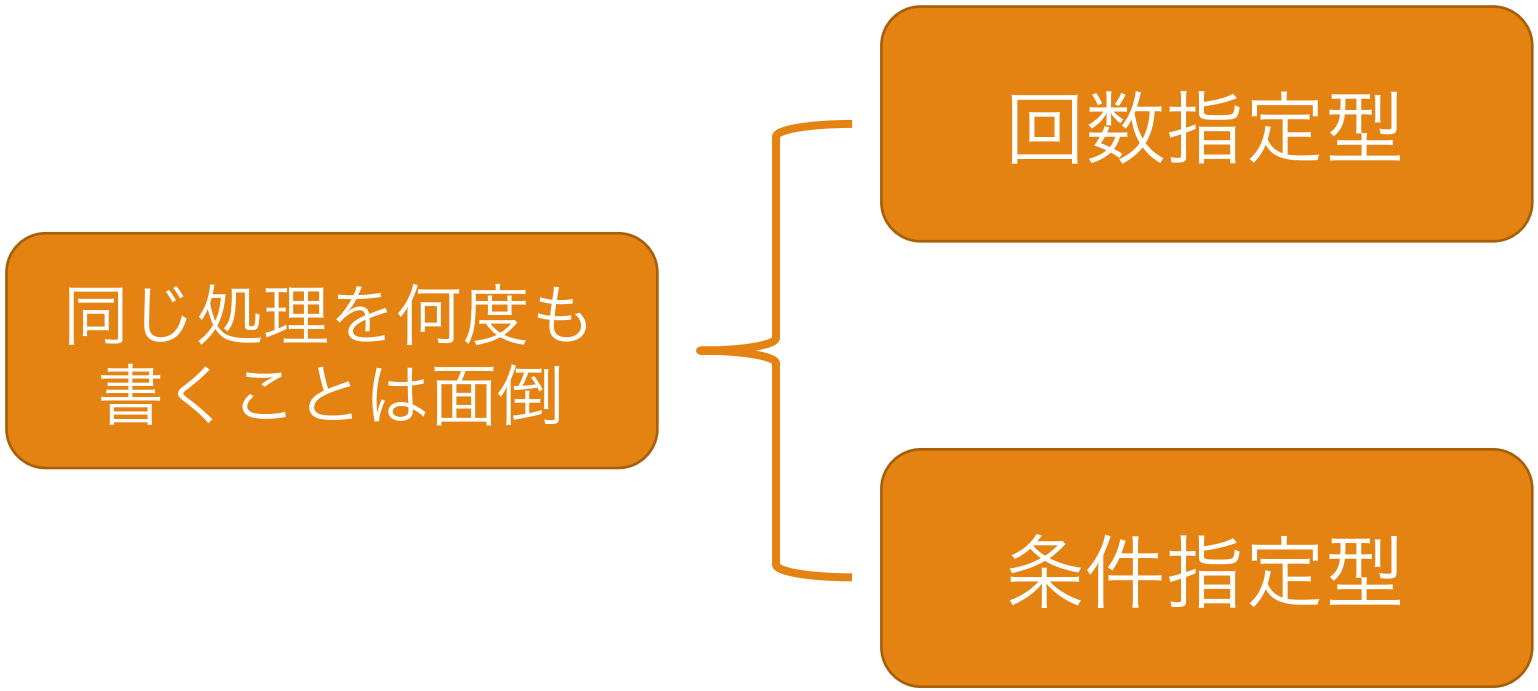
逐次処理	前々回までやっていた、 上から順番に命令を実行
条件分岐処理	前回にやった条件に応じて 実行する命令を選ぶ
繰り返し処理	同じ命令（の塊）を 繰り返し実行
関数	処理の塊に名前をつけて、 その名前で処理を行う

乱数

乱数
でたらめな数を作り出す仕組み

random：関数

繰り返し処理



繰り返しパターンを見つ出す

```
float x,y;
size(400,200);
smooth();
background(150);
fill(255);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
 :
 :
```

```
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
```

繰り返しパターンを見つ出す

```
float x,y;
size(400,200);
smooth();
background(150);
fill(255);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
 :
 :
```

```
x = random(width);
y = random(height);
ellipse(x,y,20,20);
x = random(width);
y = random(height);
ellipse(x,y,20,20);
```

繰り返し回数は何回か？

回数指定型

6ページ目から

```
for(int i=0;i<10;i++){
  x = random(width);
  y = random(height);
  ellipse(x,y,20,20);
}
```

カウンタ変数
繰り返し回数を
おぼえている変数

少し複雑な回数指定処理

カウンタ変数の利用を
利用して処理を行う

7ページ目から

繰り返しパターンを見つけ出す

```
size(300,200);
background(255);
stroke(0);

line(25,20,25,180);
line(50,20,50,180);
line(75,20,75,180);
line(100,20,100,180);
line(125,20,125,180);
line(150,20,150,180);
line(175,20,175,180);
line(200,20,200,180);
line(225,20,225,180);
line(250,20,250,180);
```

```
size(300,200);
background(255);
stroke(0);

line( 0*25+25,20, 0*25+25,180);
line( 1*25+25,20, 1*25+25,180);
line( 2*25+25,20, 2*25+25,180);
line( 3*25+25,20, 3*25+25,180);
line( 4*25+25,20, 4*25+25,180);
line( 5*25+25,20, 5*25+25,180);
line( 6*25+25,20, 6*25+25,180);
line( 7*25+25,20, 7*25+25,180);
line( 8*25+25,20, 8*25+25,180);
line( 9*25+25,20, 9*25+25,180);
```

繰り返しパターンを for命令に置き換える

```
size(300,200);
background(255);
stroke(0);

line( 0*25+25,20, 0*25+25,180);
line( 1*25+25,20, 1*25+25,180);
line( 2*25+25,20, 2*25+25,180);
line( 3*25+25,20, 3*25+25,180);
line( 4*25+25,20, 4*25+25,180);
line( 5*25+25,20, 5*25+25,180);
line( 6*25+25,20, 6*25+25,180);
line( 7*25+25,20, 7*25+25,180);
line( 8*25+25,20, 8*25+25,180);
line( 9*25+25,20, 9*25+25,180);
```

```
size(300,200);
background(255);
stroke(0);

for(int i=0;i<10;i++){
  line(i*25+25,20,
       i*25+25,180);
}
```

授業時に配布した資料

http://www.sato-lab.jp/imfu/index.html

においてあります。