情報メディア基盤ユニット

配列その2

情報メディア学科佐藤尚

中間試験のお知らせ

試験日：7月1日(金)1限の講義時間
試験範囲：6月17日（金）までの授業範囲
主な内容：図形・文字等の描画、変数、分岐処理、
繰り返し処理、座標変換, 関数、1次元配列

形式：マークシート
持ち込み：プリント、ノート、ノートPC、本は可。人は不可。
注意事項：持ち込んだノートPCをネットに接続することは認めません。
カンニング、相談などもダメです。

試験期間中に期末試験がある
最終課題制作

Processingでプログラムをつくる
発表会を実施（最後の金曜日）

プログラムを作るさいの共通処理

選び出す（選択）

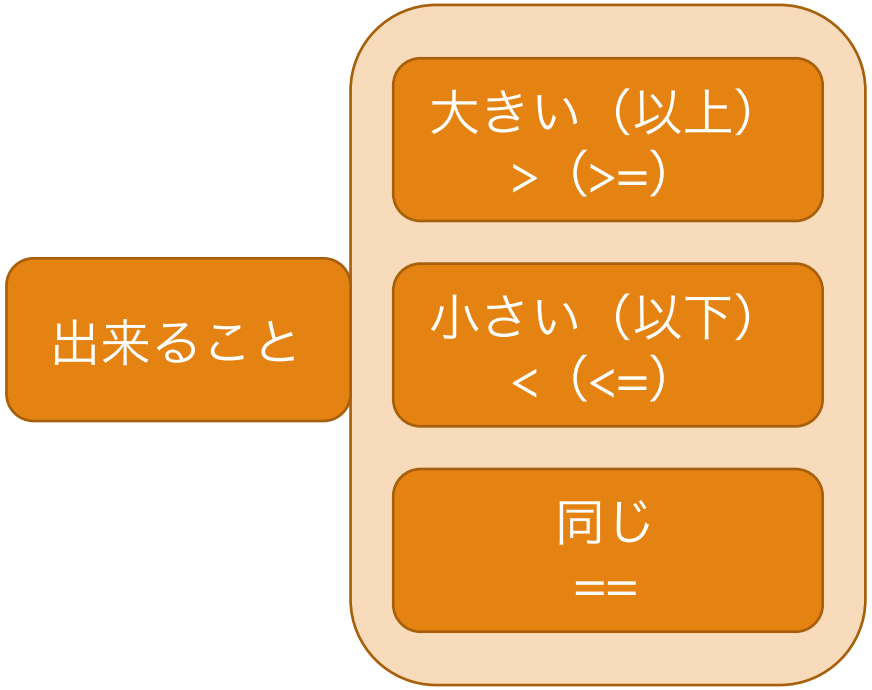
並び替える（ソート）

接触判定

跳ね返る（物理的なものの動き）

最大値を求める

２つの数A,Bのうち大きい方（最大値）を求めるためには？



2つの数A,Bの値を比較して、
大きい方が最大値

```
if(A>=B){
  Aが最大値
}else{
  Bが最大値
}
```

最大値を求める

3つの数A,B,Cの中の最大値を求めるためには？

3つの数A,B,Cの大小関係には6通りの可能性がある

Aが最大値	Bが最大値	Cが最大値
$A \geqq B \geqq C$	$B \geqq A \geqq C$	$C \geqq A \geqq B$
$A \geqq C \geqq B$	$B \geqq C \geqq A$	$C \geqq B \geqq A$

3つの数A,B,Cの中の最大値を求める

```
if(A>=B && B >= C){
  Aが最大値
}else if(A>=C && C >= B){
  Aが最大値
} else if(B>=A && A >= C){
  Bが最大値
} else if(B>=C && C >= A){
  Bが最大値
} else if(C>=A && A >= B){
  Cが最大値
} else if(C>=B && B >= A){
  Cが最大値
}
```

この方法は比べる数が
増えるとちょっと面倒

4つの場合は24通りの可能性がある
5つの場合は120通りの可能性がある
6つの場合は720通りの可能性がある

120個もif命令を書くのはちょっと…

3つの数A,B,Cの中の最大値を求める

別な方法を考える → 勝ち抜き戦方式

AとBの大きさを比べ、大きい方を最大値の候補(maxSoFar)とする。
Cと最大値の候補(maxSoFar)を比べ、大きい方が最大値となる。

```
if(A>B){
  Aの値をmaxSoFarに代入する
}else{
  Bの値をmaxSoFarに代入する
}
if(C>maxSoFar){
  Cの値をmaxSoFarに代入する
}
maxSoFarの値が最大値となっている
```

これならできる

so far = 今までのところ

4つの数A,B,C,Dの中の最大値を求める

勝ち抜き戦方式ならできる

AとBの大きさを比べ、大きい方をmaxSoFarとする。
Cとを比べ、大きい方をmaxSoFarとする。
Dとを比べ、大きい方をmaxSoFarとする。
maxSoFarが最大値。

```
if(A>B){
  Aの値をmaxSoFarに代入する
}else{
  Bの値をmaxSoFarに代入する
}
if(C>maxSoFar){
  Cの値をmaxSoFarに代入する
}
if(D>maxSoFar){
  Dの値をmaxSoFarに代入する
}
maxSoFarが最大値
```

これならできる

4つの数A,B,C,Dの中の最大値を求める

Aの値をmaxSoFarとする。
Bとmaxsoを比べ、大きい方をmaxSoFarとする。
Cとmaxsoを比べ、大きい方をmaxSoFarとする。
Dとmaxsoを比べ、大きい方をmaxSoFarとする
maxSoFarが最大値。

```
maxSoFar = A;
if(B>maxSoFar){
  maxSoFar = B;
}
if(C>maxSoFar){
  maxSoFar = C;
}
if(D>maxSoFar){
  maxSoFar = D;
}
maxSoFarが最大値
```

これを関数を使って書くと

A,B,C,Dはfloat型、関数名はmax4とする

```
float max4(float A,float B,float C,float D){
  float maxSoFar = A;
  if(B > maxSoFar){
    maxSoFar = B;
  }
 if(C > maxSoFar){
    maxSoFar = C;
  }
 if(D > maxSoFar){
    maxSoFar = D;
  }
  return maxSoFar;
}
```

これを関数を使って書くと

こんな風に書いても同じ

```
float max4(float a0,float a1,float a2,float a3){
  float maxSoFar = a0;
  if(a1 > maxSoFar){
    maxSoFar = a1;
  }
 if(a2 > maxSoFar){
    maxSoFar = a2;
  }
 if(a3 > maxSoFar){
    maxSoFar = a3;
  }
  return maxSoFar;
}
```

5つの数A,B,C,D,Eの中の最大値を求める

勝ち抜き戦方式ならできる

```
maxSoFar = A;
if(B>maxSoFa){
  maxSoFar = B;
}
if(C>maxSoFar){
  maxSoFar = C;
}
if(D>maxSoFar){
  maxSoFar = D;
}
if(E>maxSoFar){
  maxSoFar = E;
}
maxSoFarが最大値
```

これならできる

5つの数a0,a1,a2,a3,a4の中の最大値を求める

勝ち抜き戦方式ならできる

```
maxSoFar = a0;
if(a1>maxSoFar){
  maxSoFar = a1;
}
if(a2>maxSoFar){
  maxSoFar = a2;
}
if(a3>maxSoFar){
  maxSoFar = a3;
}
if(a4>maxSoFar){
  maxSoFar = a4;
}
```

maxSoFarが最大値

これならできる

5つの数a0,a1,a2,a3,a4の中の最大値を求める

```
maxSoFar = a0;
if(a0>maxSoFar){
  maxSoFar = a0;
}
if(a1>maxSoFar){
  maxSoFar = a1;
}
if(a2>maxSoFar){
  maxSoFar = a2;
}
if(a3>maxSoFar){
  maxSoFar = a3;
}
if(a4>maxSoFar){
  maxSoFar = a4;
}
```

maxSoFarが最大値

これならできる

5つの数の最大値を求める

データが配列a[0]～a[4]に入っているのなら

```
maxSoFar = a[0];
if(a[0]>maxSoFar){
  maxSoFar = a[0];
}
if(a[1]>maxSoFar){
  maxSoFar = a[1];
}
if(a[2]>maxSoFar){
  maxSoFar = a[2];
}
if(a[3]>maxSoFar){
  maxSoFar = a[3];
}
if(a[4]>maxSoFar){
  maxSoFar = a[4];
}
maxSoFarが最大値
```

```
maxSoFar = a[0];
for(int i=0;i<5;i++){
  if(a[i] > maxSoFar){
    maxSoFar = a[i];
  }
}
maxSoFarが最大値
```

これならできる

5つの数の最大値を求める

データが配列a[0]～a[4]に入っているのなら
関数の形で書けば

これならできる


```
float max5(float[]a ){
  float maxSoFar = a[0];
  for(int i=0;i<5;i++){ // 5は配列の要素の個数=a.length
    if(a[i] > maxSoFar){
       maxSoFar = a[i];
    }
  }
  return maxSoFar;
}
```

配列の中に入ってる数の最大値を求める

配列の中の要素数はlengthでわかるから

```
float myMax(float[]a ){
  float maxSoFar = a[0];
  for(int i=0;i<a.length;i++){
    if(a[i] > maxSoFar){
       maxSoFar = a[i];
    }
  }
  return maxSoFar;
}
```

実は…

Processingでは、組み込み関数として配列の最大値や最小値を求める関数が用意されている

max

min

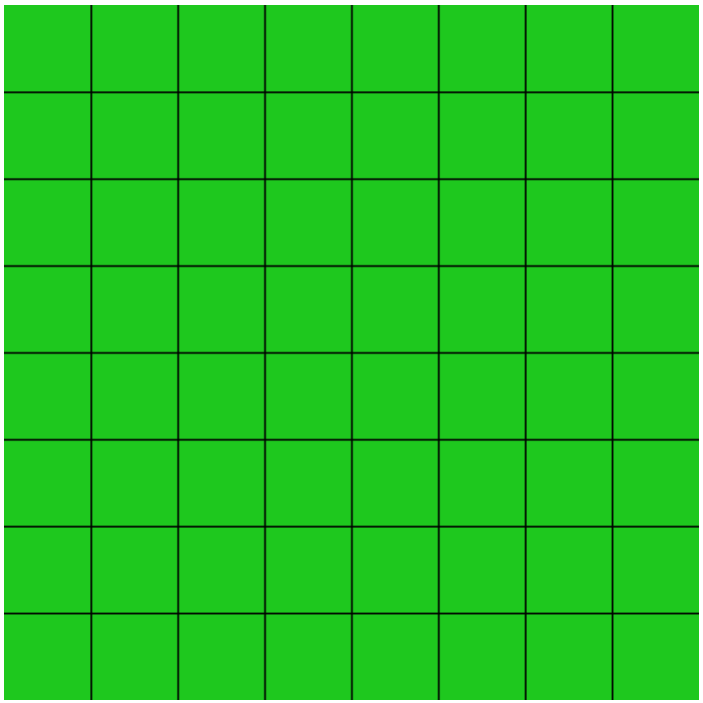
配列（1次元配列）

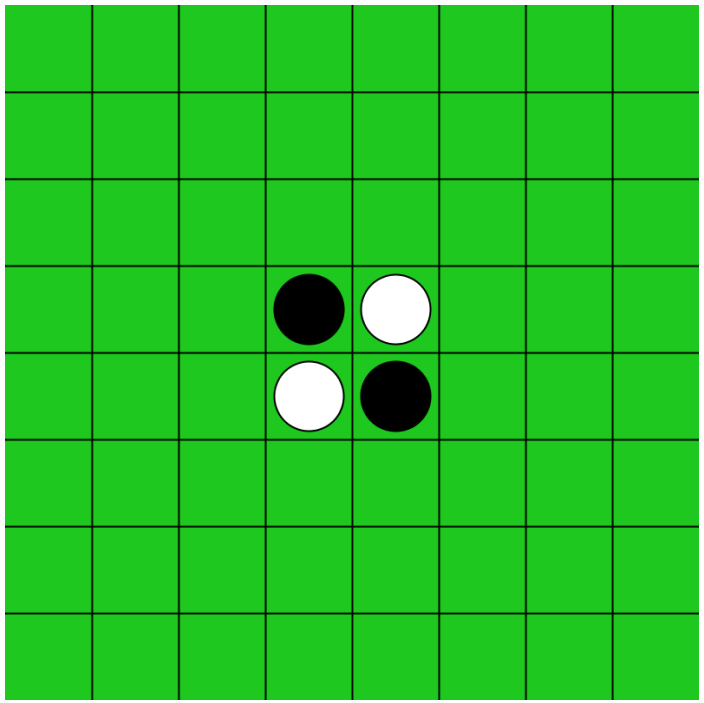
変数名と数字のペアで変数を表す

数字のことを添え字
変数名[数字]で変数を表す
　　数字は0から数え始める、
　　先頭からどれだけ移動したか
例えば、
　　x[0],x[1],x[2]など

多次元配列

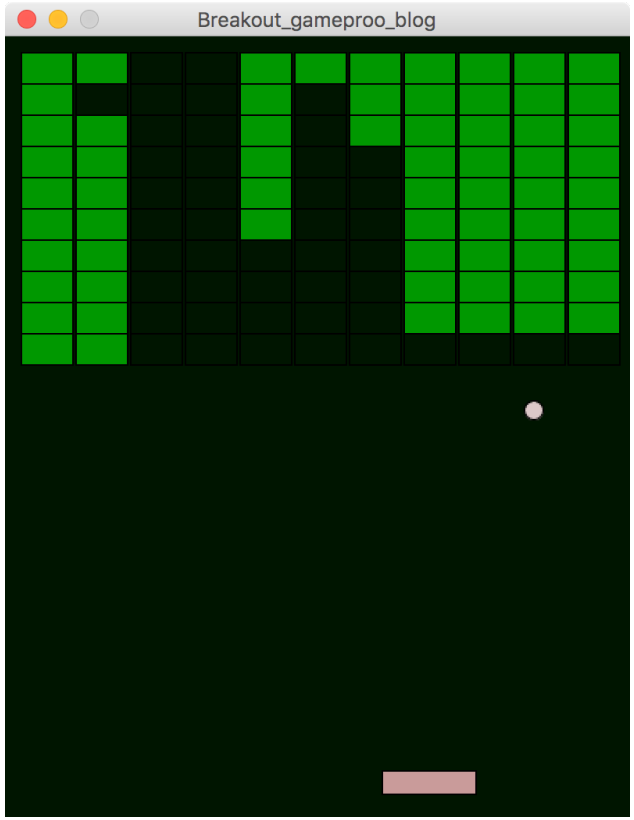
こんな盤面の情報を表すには？





多次元配列

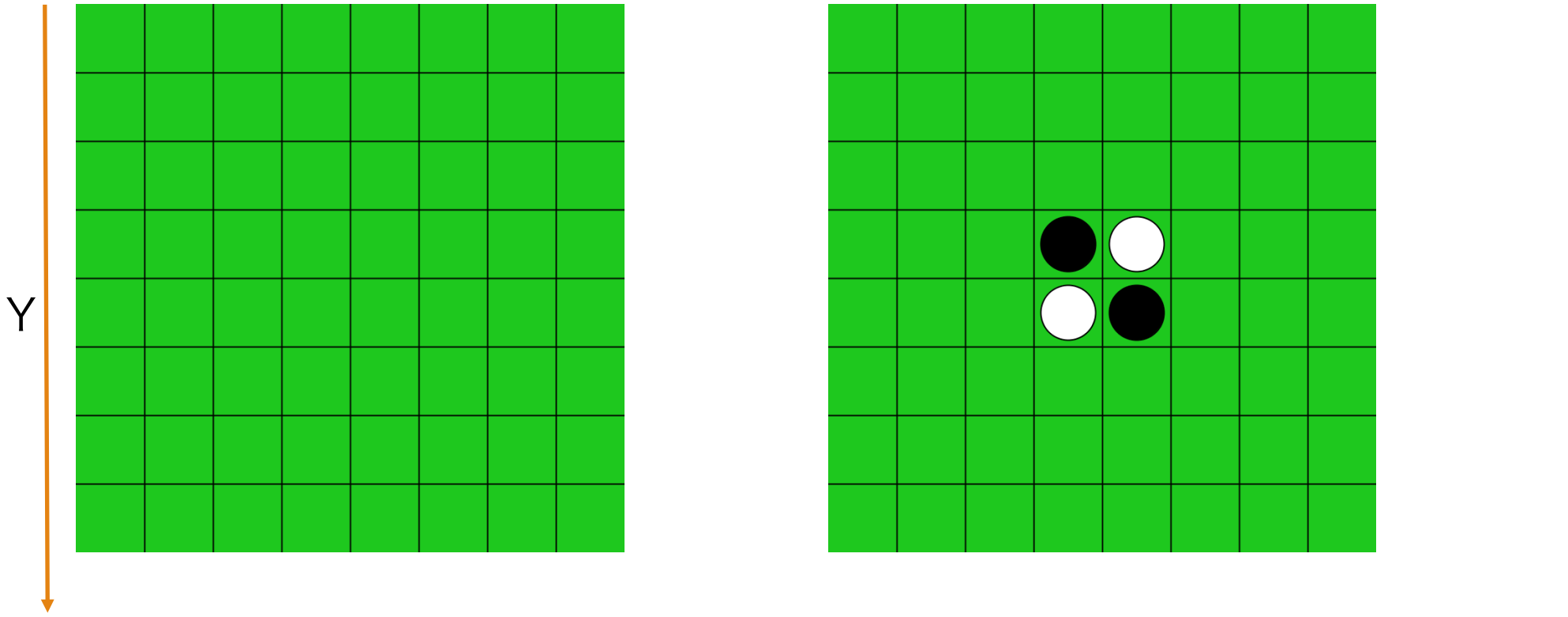
こんなゲームのブロックの情報を表すには？



多次元配列

変数名をboardとする、何もない0、黒は1、白は2

「board(3,4)=2」みたに書けないか？



多次元配列

変数名をboardとする、何もない0、白は1、黒は2

board(3,4)だと関数と区別で出来ない

そこで、board[3][4]と書くことにする

二つの数字で指定するので、2次元配列と呼ばれている

多次元配列（2次元配列）

変数名と2つ数字のペアで変数を表す

数字のことを添え字
変数名[数字] [数字]で変数を表す
　数字は0から数え始める例えば、
　x[0][0],x[1][2],x[2][0]など

配列（2次元配列）を使うためには

変数宣言が必要

データ型[] [] 変数;
例えば、
float[] [] y;

PImage[] [] faces;
int[] [] x;
String [][] names;

配列を表すマーク

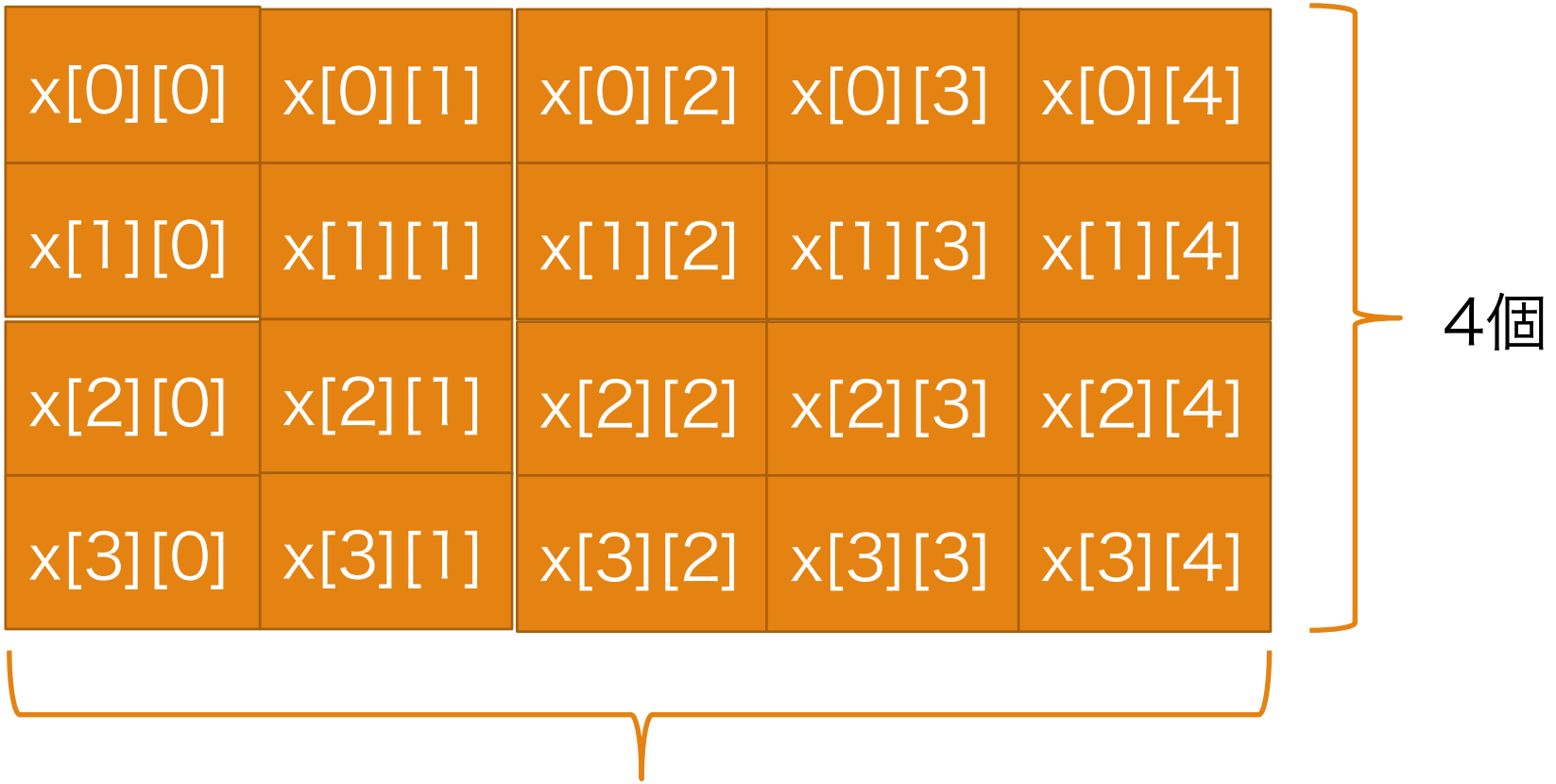
何個のデータを使うかのかを指定する

データを入れる場所を確保

データ型[] [] 変数 = new データ型[必要な個数] [必要な個数];
例えば、
float[] y = new float[100][10];
100*10=1000個のデータが確保される

y[0][0],y[0][1],…,y[0][9],
y[99][0],y[99][1],…,y[99][9]

float[][] x = new float[4][5];



x.length --> 4

配列型変数の宣言と初期化

```
データ型 [][]配列変数名 =
{{[0][0]に入れるデータ,[0][1]に入れるデータ,…},
 {[1][0]に入れるデータ,[1][1]に入れるデータ,…},
 {[2][0]に入れるデータ,[2][1]に入れるデータ,…},
  …
};
int[][] radius = new int[10][3];
String [] msg = new String[20][2];
```

多次元配列

変数名と2個以上数字のペアで変数を表す

2,3くらいが多いかも

多次元配列の宣言

データ型[][] 変数名； // 2次元配列
データ型[][][] 変数名;// 3次元配列

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。