
配列その2
基数ユニット
PROCESSING FRIENDS

Web授業アンケート回答手順

授業改善のためのアンケートに協力いただくようお願いします。

※担当教員は、選択式回答はその集計結果のみ、記述式回答は記述内容のみを知ることができ、回答学生個人（学籍番号・氏名）が特定されることはありません。

授業アンケートの提出方法

① KAIT Walkerホームページの「教育・学修支援」にある「授業アンケート」をクリックする。

② SecureMatrixのログイン画面でユーザーIDとパスワードを入力する。

③ マイページの「大学からの課題・アンケート」にある「授業アンケート20nn年度前期（または後期）」をクリックする。

④ 一覧から、回答する授業を選択する。

⑤ アンケートの授業名を確認し、「スタート」をクリックする。

⑥ 順次各設問に回答する。

・この授業について　：　12問
・教員オリジナル設問　：　4問
（先生が指示した場合）
・自由記述　：　2問

⑦ 全問回答したら、「提出確認」をクリックする。
注意！「提出確認」をクリックした段階では
まだ提出はされていません。

⑧ 回答内容に問題がなければ、「提出」をクリックする。

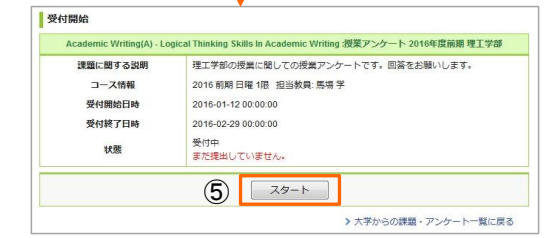
以上で提出が完了です。

＜KAIT Walker＞

＜画面例＞

マイページ

大学からの課題・アンケート一覧

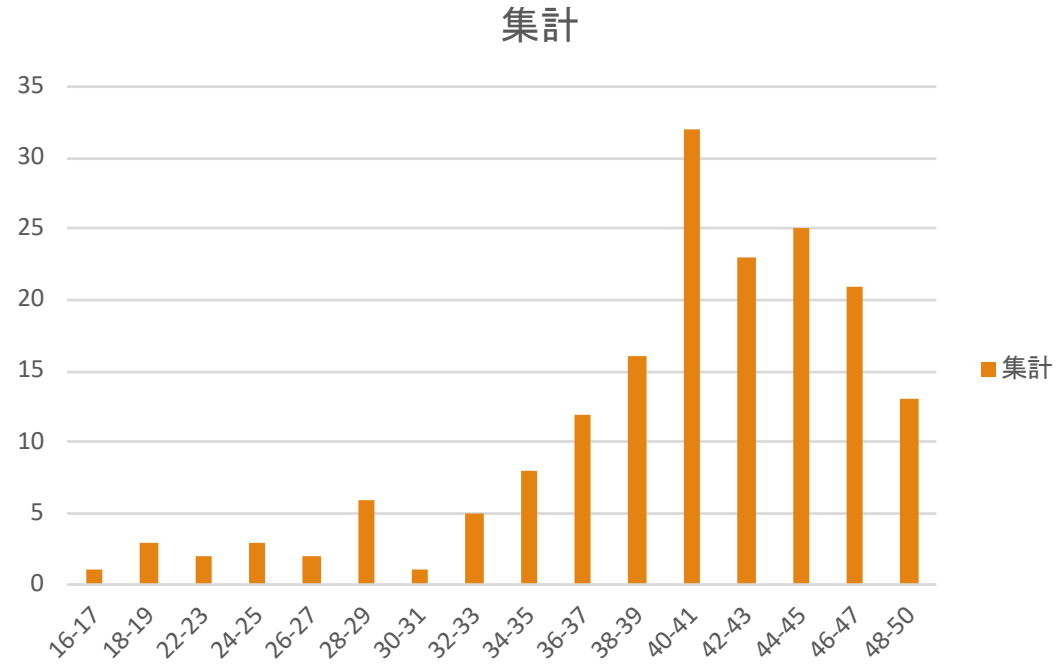


受付開始

中間試験の結果

記述問題は採点中なので、マーク問題のみの結果

正解数平均：40.1(37.3)
中央値：41(40)



前回の内容(1/2)

配列（一次元配列）

変数名と添え字のペアで一つの変数を指定する

変数名[番号]

配列変数の宣言

データ型名[] 変数名

配列変数名 = new データ型[個数]

前回の内容(2/2)

配列は関数の引数にもすることができる

```
void foo(int a,int b){
  a = 1;
  b = 2;
}
void bar(int[] a){
  a[0] = 1;
  a[1] = 2;
}
```

```
int a = 0;
 int b = 1;
 int[] c = new int[2];
 c[0] = 0;
 c[1] = 1;
 println(a,b);
 foo(a,b);
 println(a,b);
 println(c[0],c[1]);
 bar(c);
 println(c[0],c[1]);
```

出力結果

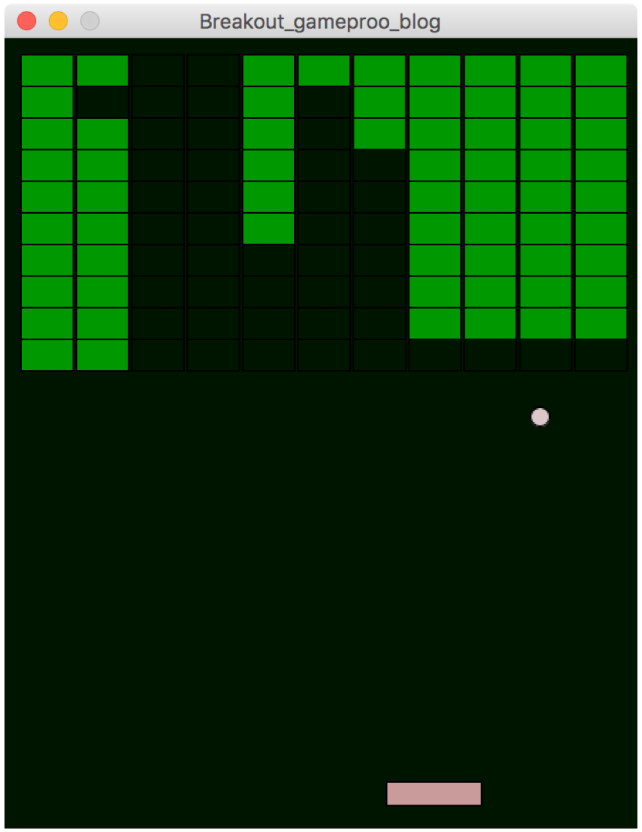
0 1
0 1
0 1
1 2

関数内で配列となっている引数の値を変更すると、関数の外側にも反映される。

今日の内容

配列を使ったプログラム

多次元配列



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）

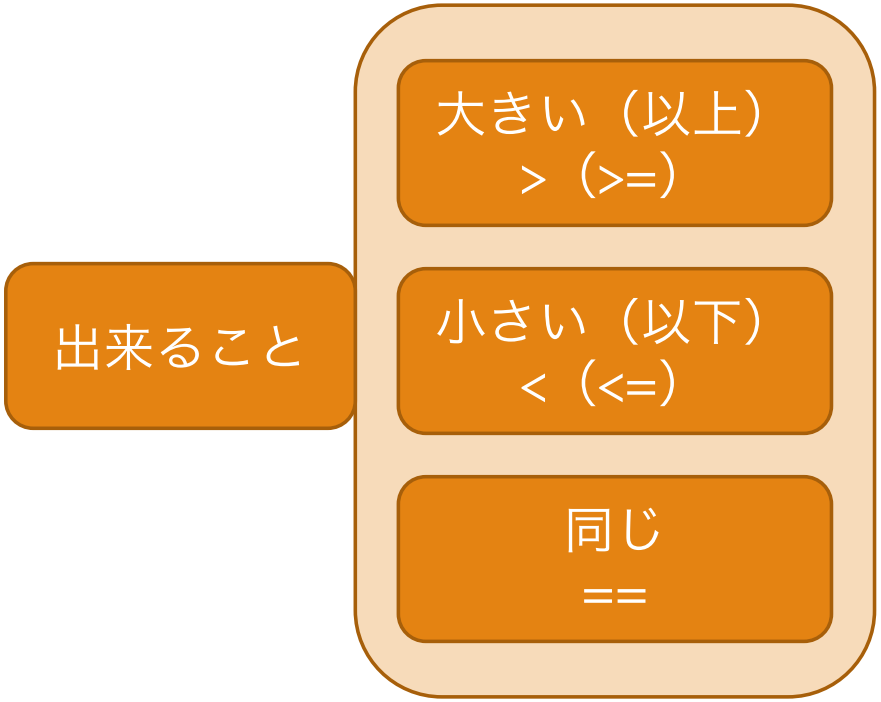
接触判定

物体の移動

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 ≧ B ≧ C	B ≧ A ≧ C	C ≧ A ≧ B
A ≧ C ≧ B	B ≧ C≧ A	C≧ B≧ 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
```

擬似コード

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>maxSoFar){
  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
```

なくても
OK

```
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

配列と繰り返し処理は相性が良い

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

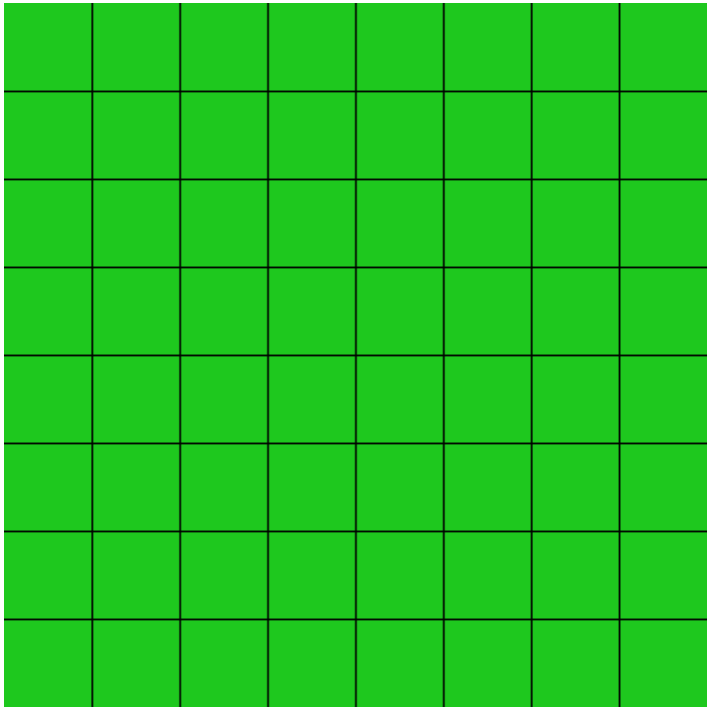
配列（1次元配列）

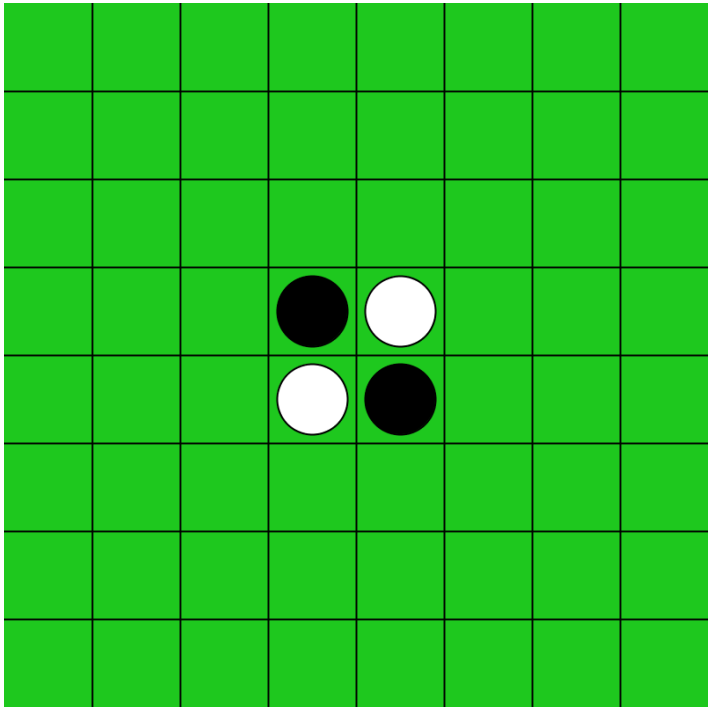
変数名と数字のペアで変数を表す

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

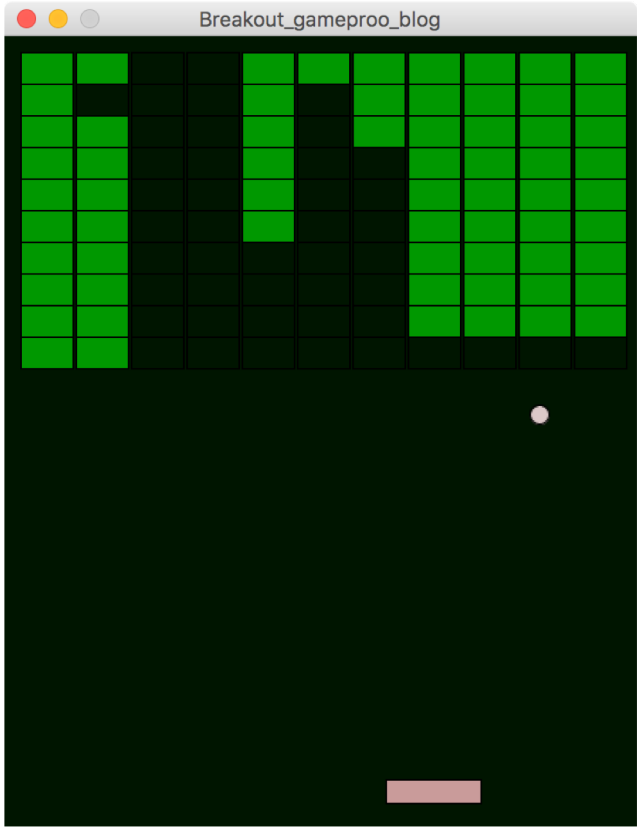
こんな盤面の情報を表すには？





多次元配列

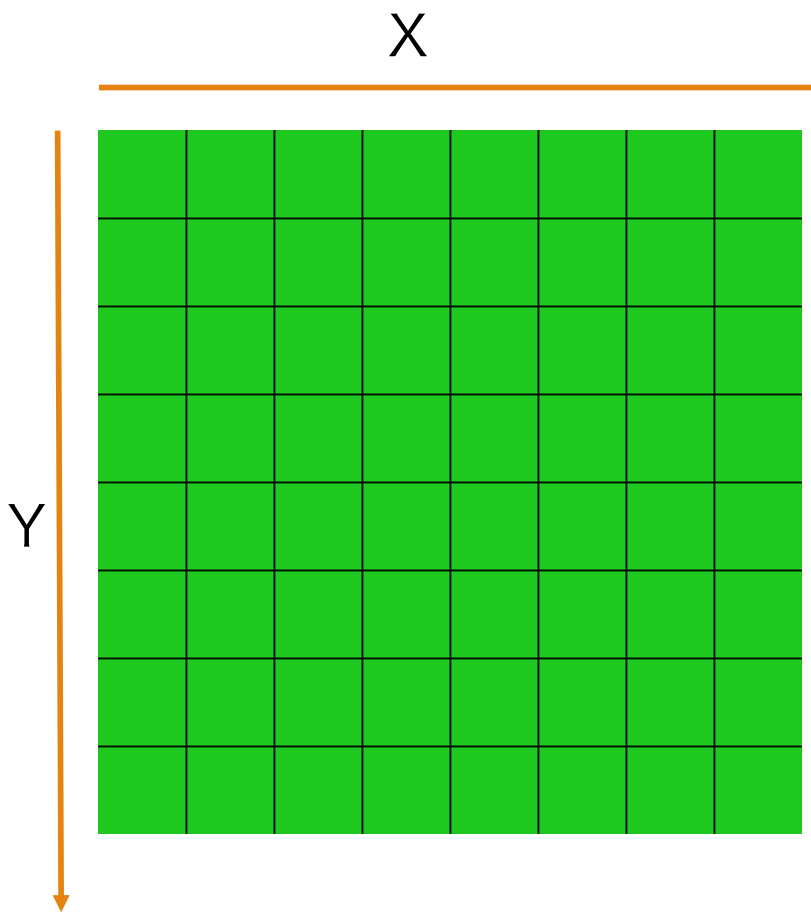
こんなゲームのブロックの情報を表すには？

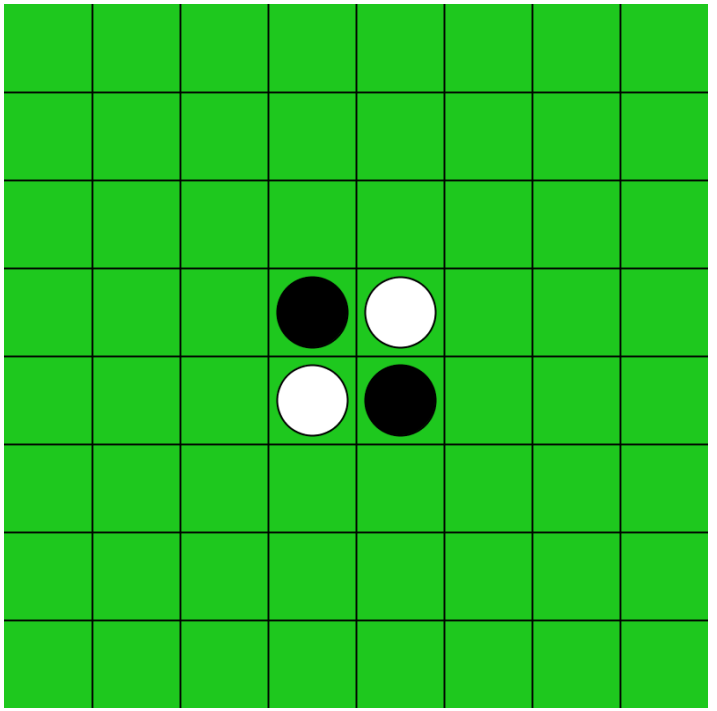


多次元配列

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

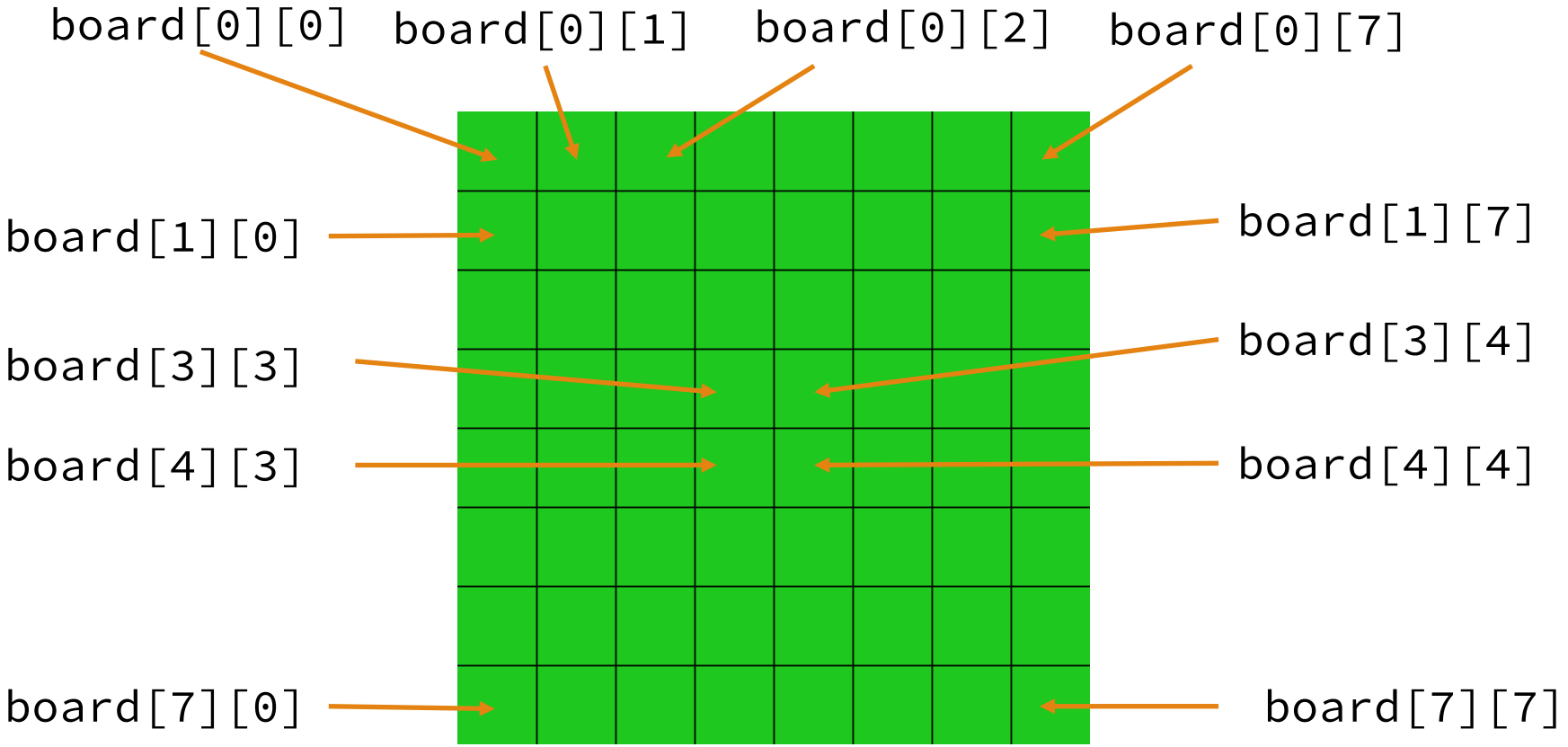
変数名をboardとする、何もない0、白は1、黒は2

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）



board[3][3]=1

int[][] board;

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
　board[3][3],board[3][4]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

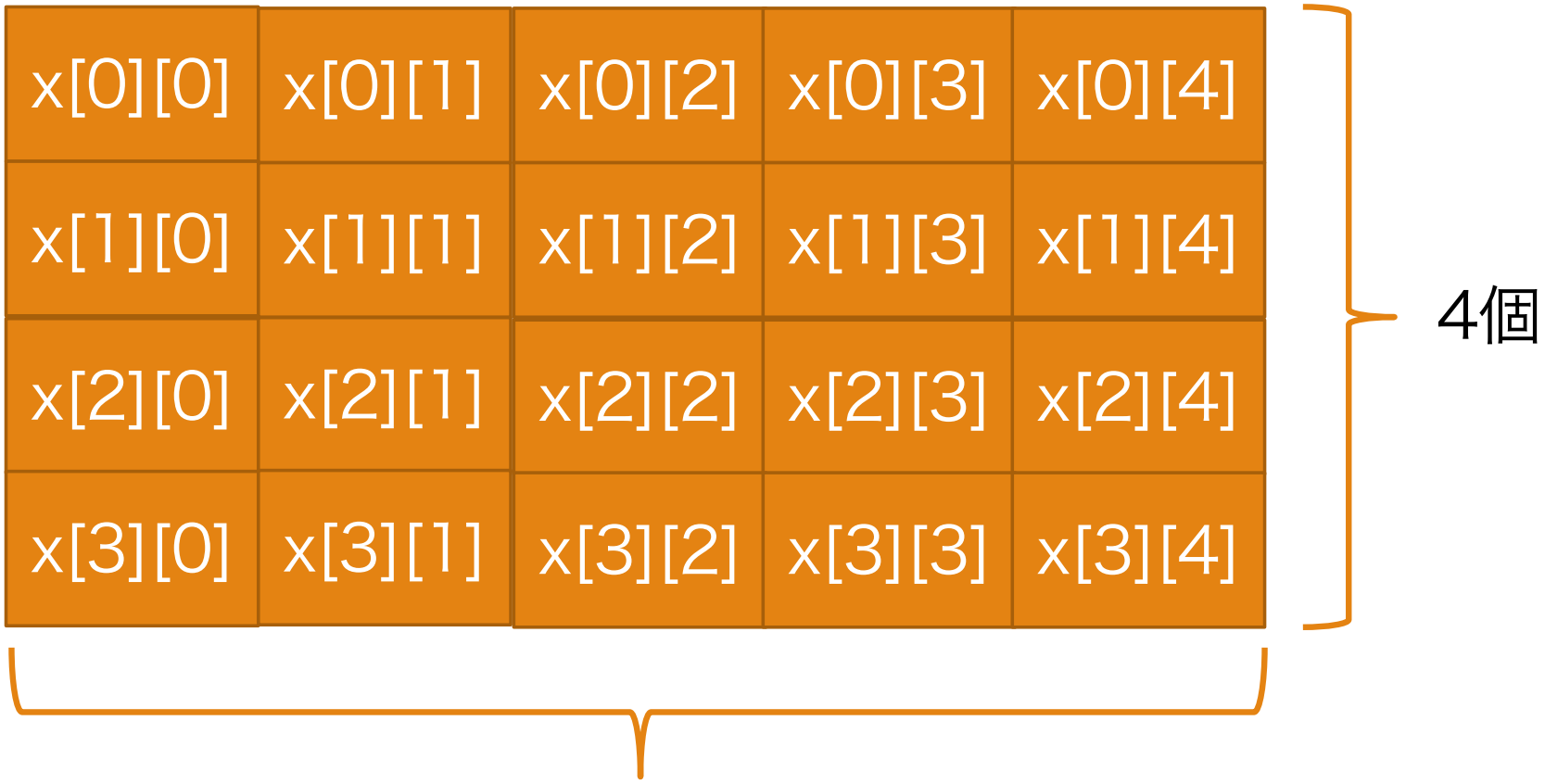
何個のデータを使うかのかを指定する

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

StringとChar

String

色々な長さの文字列を扱う
“abc”,”Kait 2018”,”!”,””

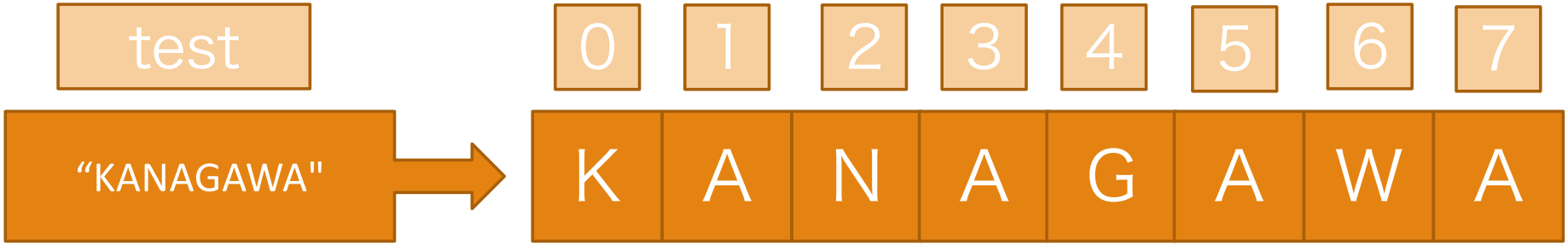
参照型

char

一文字だけの情報を表す
’a’,’B’,’!’,’1’

character

文字列(String)



String test=”KANAGAWA”;

test.length()

文字列の含まれる文字数
（文字列の長さ）

test.length()->8

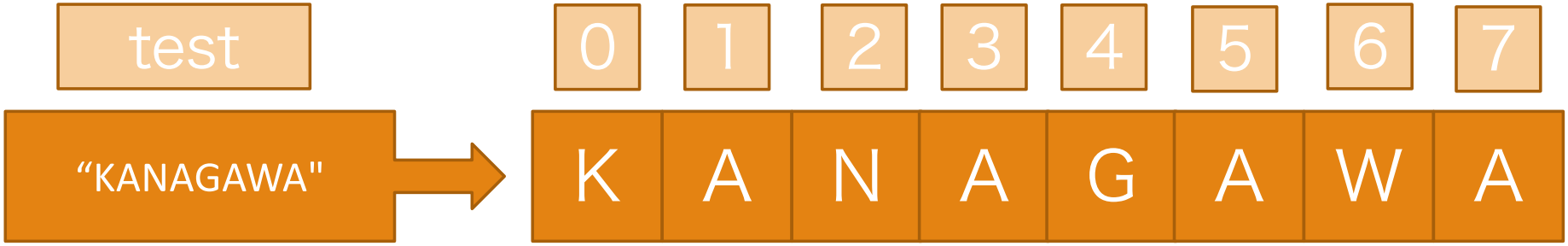
test.charAt()

i番目の位置の文字

test.charAt(0)->‘K’
test.charAt(7)->‘A’

Immutable：変わらない

文字列(String)



String test=”KANAGAWA”;

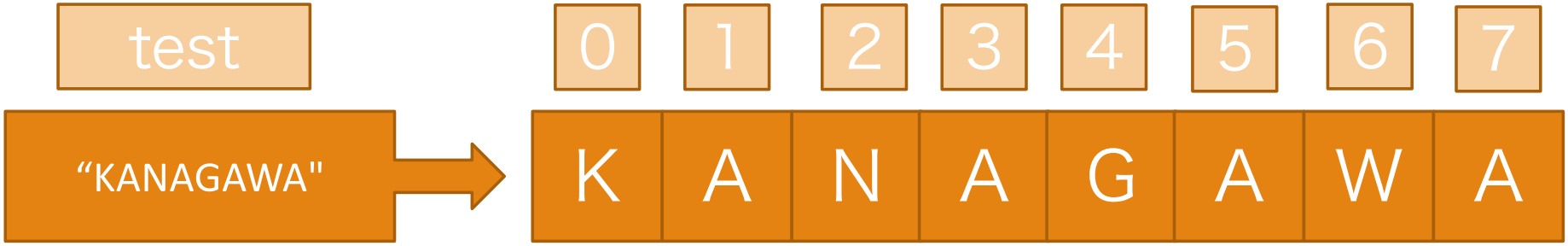
test.length()

文字列の含まれる文字数
（文字列の長さ）

test.length()->8

“abcde”.length()　　.は「の」

文字列(String)



String test=”KANAGAWA”;

test.indexOf()

引数で指定した文字列が
最初に現れる位置

test.indexOf(“A”)->1
test.indexOf(“N”)->2
test.indexOf(“NA”)->2
test.indexOf(“IT”)->-1

indexOf(探す文字列、探し始める位置)
test.indexOf(“A”,2) ->3

指定した文字列が見つからなければ、戻り値は-1となる

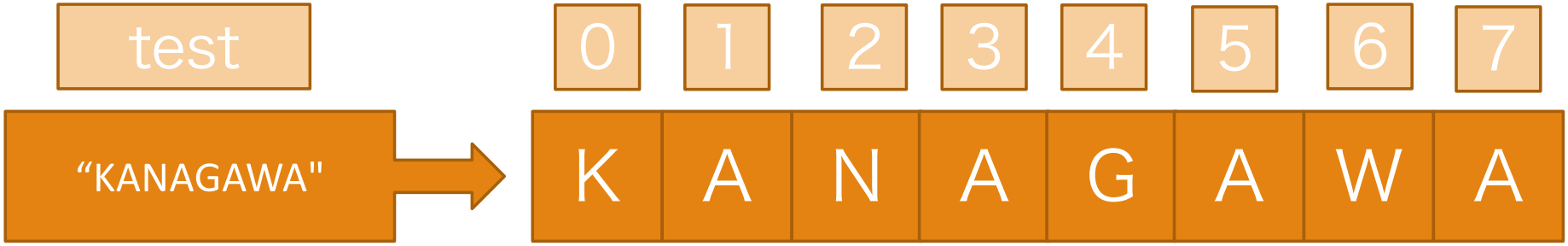
例えば、

String alphabet = “abcdefghijklmnopqrstuvwxyz”;

alphabet.charAt(i) → i番目のアルファベットを取り出す

alphabet.indexOf(alpha) → 文字alphaが何番目の
アルファベットかを調べる

文字列(String)



String test=”KANAGAWA”;

test.substring()

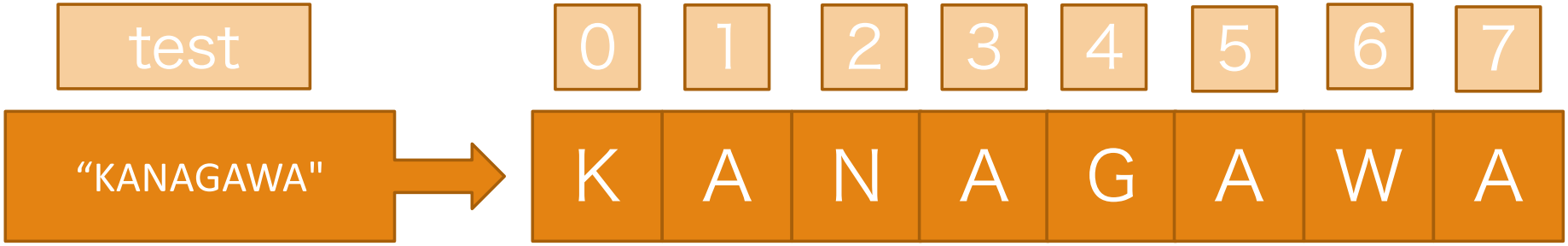
指定した場所からの
文字列を取り出す

部分文字列

test.substring(4)->”GAWA”
test.substring(6)->”WA”
test.substring(0)->”KANAGAWA”

test.substring(4,6)->”GA”
test.substring(1,6)->”ANAGA”
test.substring(2,5)->”NAG”

文字列(String)



String test=”KANAGAWA”;

文字列の連結（つなげる）

test + ” Institute” -> “KANAGAWA Institute ”
test + “2016” -> “KANAGAWA2016”
test + test -> “KANAGAWAKANAGAWA”
test + test+ ” Institute” -> “KANAGAWA KANAGAWA Institute ”

文字列（String）

split → 指定した文字で文字列を分解する

String[] moji = “Akagi, aircraft carrier,1925”.split(“,”)

↓ “,”の位置で文字列を分割

moji[0] = “Akagi”,moji[1]=“aircraft carrier”,moji[2]=“1925”

文字列（String）

trim

文字列中の前後の
ホワイトスペース（空白など）を取り除く

“ K A ”.trim() -> ”K A”

isEmpty()

空の文字列かを判定する

test.isEmpty() -> false
“Kanagawa”.isEmpty() -> false
“”.isEmpty() -> true

文字列の比較

2つの文字列が同じかどうかは==では判定できない

”test” == “kait”

equalsを使う

2つが同じ：true
異なっている：false

“test”.equals(“test”)

文字列と数値の変換

関数名	機能	使用例
int(String s)	引数が表している数値をint型の数値に変換	int(“123”)
float(String s)	引数が表している数値をfloat型の数値に変換	float(“3.14”)
str(int x)	int型引数xを文字列に変換	str(123)
str(float x)	float型引数xを文字列に変換	str(3.14)

最終課題制作のお知らせ

提出物

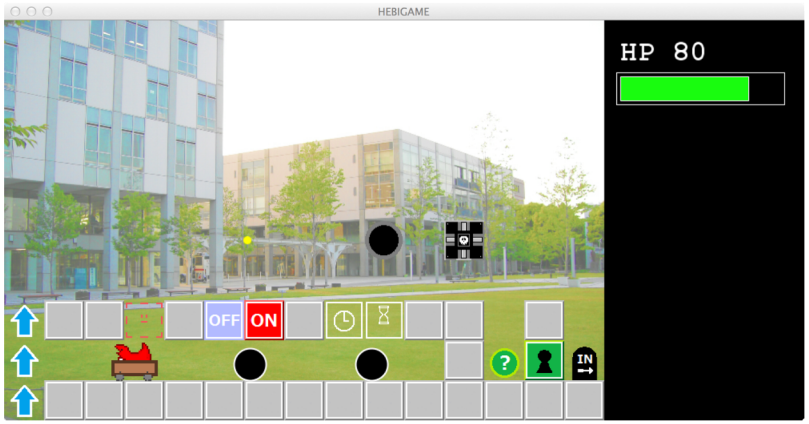
- Processingで作ったプログラム
  - （「スケッチをアーカイブ」の機能を利用して、zipファイルとして提出）
- 自分の制作物の内容を説明するWordで作ったレポート
- 5分間のプレゼンテーションを想定した、自分の作品紹介用のPowerPoint

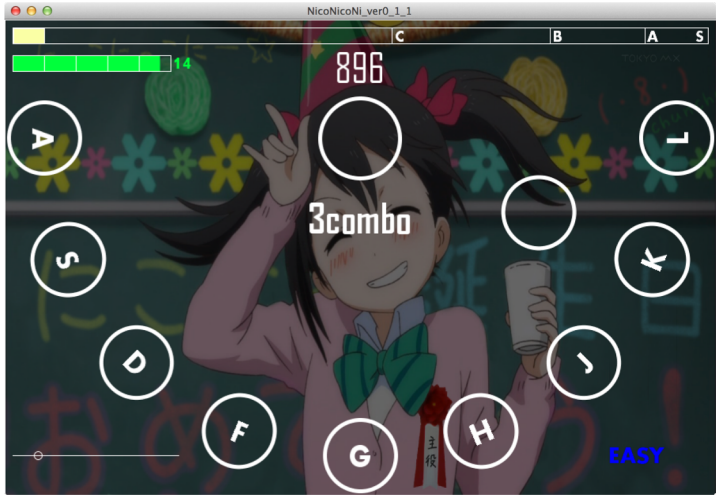
提出締め切り：7月24日（水）の15時まで

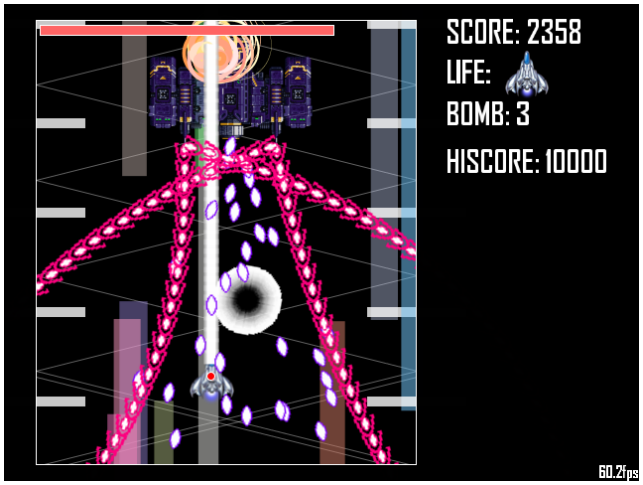
発表会

- 日時：7月26日（金）1,2限の演習の時間帯
- 会場：未定（おそらくK3-3506?）

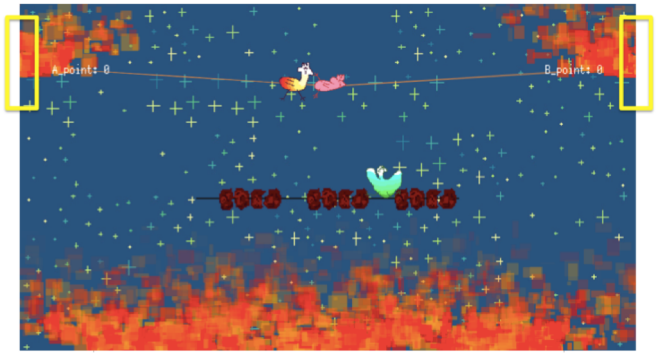
2014年度1年前期の最終課題

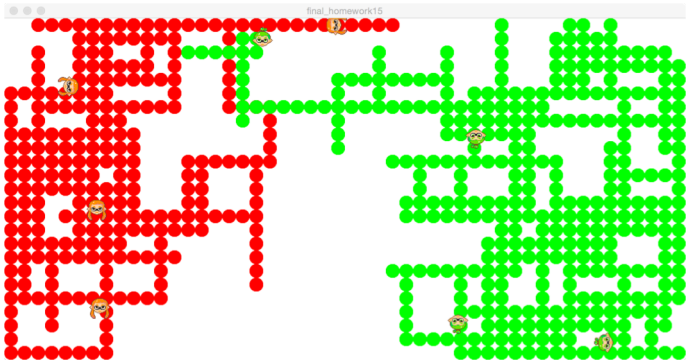




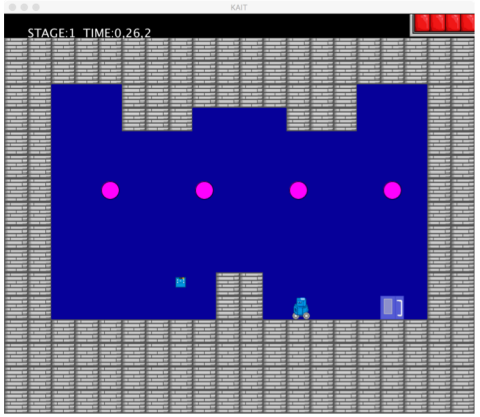


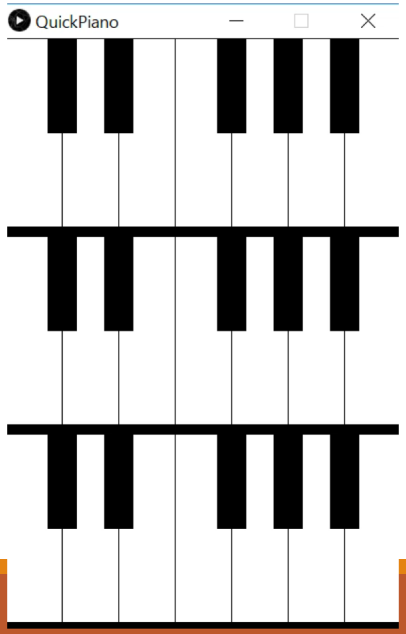
2015年度1年生前期の最終課題





2016年度1年生前期の最終課題





今後の予定

- 簡単な文字列処理
- ファイルへの入出力
- 簡単なサウンド処理

今日やった内容

- 配列を使ったプログラム
- 多次元配列
- 文字列

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。