



# ji-6. 配列

(Java プログラミング入門)

URL: <https://www.kkaneko.jp/pro/ji/index.html>



金子邦彦



# 内容



- 例題 1. 月の日数

配列とは. 配列の宣言. 配列の添字.

- 例題 2. ベクトルの内積

- 例題 3. 棒グラフを描く

- 例題 4. Horner 法による多項式の計算

- 例題 5. エラトステネスのふるい

配列と繰り返し計算の関係

# 目標

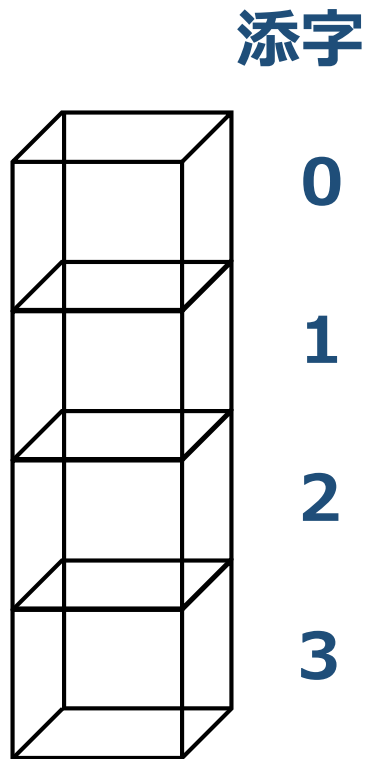


- **配列**とは何かを理解し, **integer, real の配列を使ったプログラム**を書けるようになる
- **配列と繰り返し文**を組み合わせて, **多量のデータ**を扱えるようになる

# 配列



データの並びで，番号（添字）が付いている





- プログラミングを行えるオンラインのサービス

<https://www.onlinegdb.com>

- ウェブブラウザを使う

- たくさんの言語を扱うことができる

Python3, Java, C/C++, C#, JavaScript,  
R, アセンブリ言語, SQL など

- オンラインなので、「秘密にしたいプログラム」  
を扱うには十分な注意が必要

# Online GDB で Java を動かす手順



① ウェブブラウザを起動する

② 次の URL を開く

<https://www.onlinegdb.com>

A screenshot of a search bar with a magnifying glass icon on the left and the text 'https://www.onlinegdb.com' entered inside. The search bar is set against a light gray background.



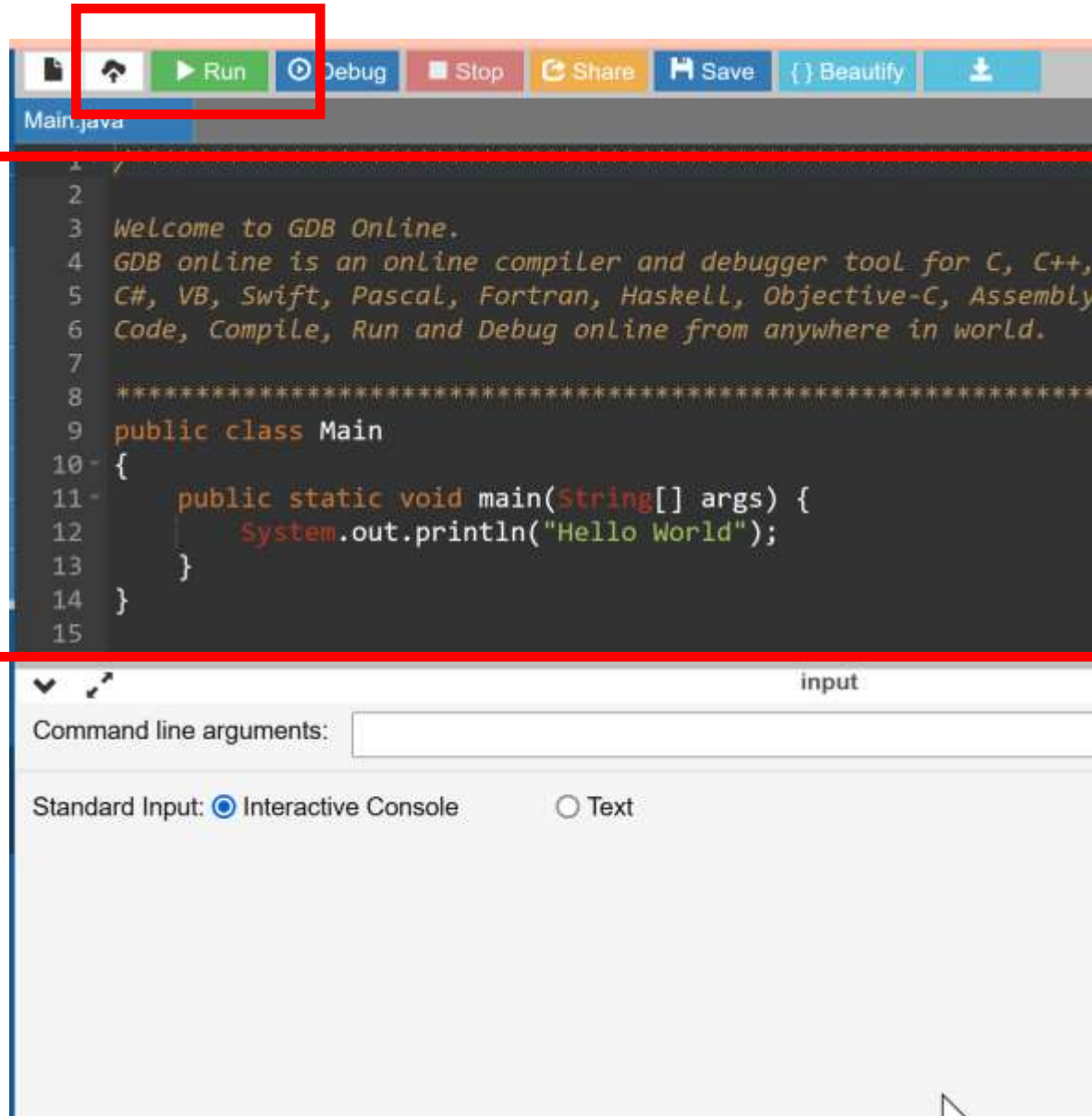
### ③ 「Language」 のところで、「Java」 を選ぶ

The screenshot shows the GDB Online web interface. At the top, there is a navigation bar with buttons for Run, Debug, Stop, Share, Save, Beautify, and a Language dropdown menu. The Language dropdown menu is open, showing a list of programming languages. The 'Java' option is highlighted in blue. The main area of the interface displays a C program source code with a 'Hello World' message.

```
1 - /*****  
2  
3 Welcome to GDB OnLine.  
4 GDB online is an online compiler and debugger tool for C, C++, Python  
5 C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6 Code, Compile, Run and Debug online from anywhere in world.  
7  
8 *****/  
9 #include <stdio.h>  
10  
11 int main()  
12 {  
13     printf("Hello World");  
14  
15     return 0;  
16 }  
17
```



# 実行ボタン



エディタ画面

プログラムを  
書き換えること  
ができる



# 例題 1 . 月の日数



- 年と月を読み込んで、日数を求めるプログラムを作る
  - うるう年の2月ならば 29
  - 日数を求めるために、サイズ12（1から12まで）の integer の 配列 を使う

例) 2021年11月 → 30



```
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        int y, m;
        int num[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
        Scanner s = new Scanner(System.in);
        System.out.println("Please Enter y(year) =");
        y = s.nextInt();
        System.out.println("Please Enter m(month) =");
        m = s.nextInt();
        if ((m == 2) && (((y % 400) == 0) || (((y % 100) != 0) && ((y % 4) == 0)))) {
            System.out.printf("number of days %d/%d is 29¥n", y, m);
        } else {
            System.out.printf("number of days %d/%d is %d¥n", y, m, num[m]);
        }
    }
}
```

配列への  
書き込み

うるう年の判定

配列からの  
読み出し

## 実行結果の例

```
Please Enter y(year) =  
2022  
Please Enter m(month) =  
6  
number of days 2022/6 is 30
```

```
Please Enter y(year) =  
2022  
Please Enter m(month) =  
2  
number of days 2022/2 is 28
```

## メモリ

num[m];

配列からの値の  
読み出し

num[0]	0
num[1]	31
num[2]	28
num[3]	31
num[4]	30
num[5]	31
num[6]	30
num[7]	31
num[8]	31
num[9]	30
num[10]	31
num[11]	30
num[12]	31

# 配列の宣言



- **配列**には、**名前**と**型**（データの種類のこと）と**サイズ**がある
  - 整数データ `int`
  - 浮動小数データ `float` や `double`
- **配列**を使うには、**配列の使用をコンピュータに伝えること**（宣言）が必要

```
int num[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

整数 名前は  
num

配列の中身を読み書きするときには、配列の名前と添字を書く

例) num[m]

添字

# 配列の読み書き



- 値の初期化

```
int num[] = {0, 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};
```

- 名前と添字で読み書き

```
System.out.printf("number of days %d/%d is %d\n", y, m, num[m]);
```

## 例題 2. ベクトルの内積



- ベクトル (1.9, 2.8, 3.7) と, ベクトル (4.6, 5.5, 6.4) の内積を表示するプログラムを作る
  - 2つのベクトルの内積の計算のために, サイズ 3 の配列を2つ使う



# ベクトルの内積

ベクトルの成分から内積を求める

$\vec{a} = (a_0, a_1, a_2)$   $\vec{b} = (b_0, b_1, b_2)$  のとき

$$\vec{a} \cdot \vec{b} = a_0 b_0 + a_1 b_1 + a_2 b_2$$





```
public class Main
{
    public static void main(String[] args) {
        double p;
        int i;
        double u[] = {1.9, 2.8, 3.7};
        double v[] = {4.6, 5.5, 6.4};
        p = 0;
        for(i = 0; i <= 2; i++) {
            p = p + (u[i] * v[i]);
        }
        System.out.printf("p(product) = %f¥n", p);
    }
}
```

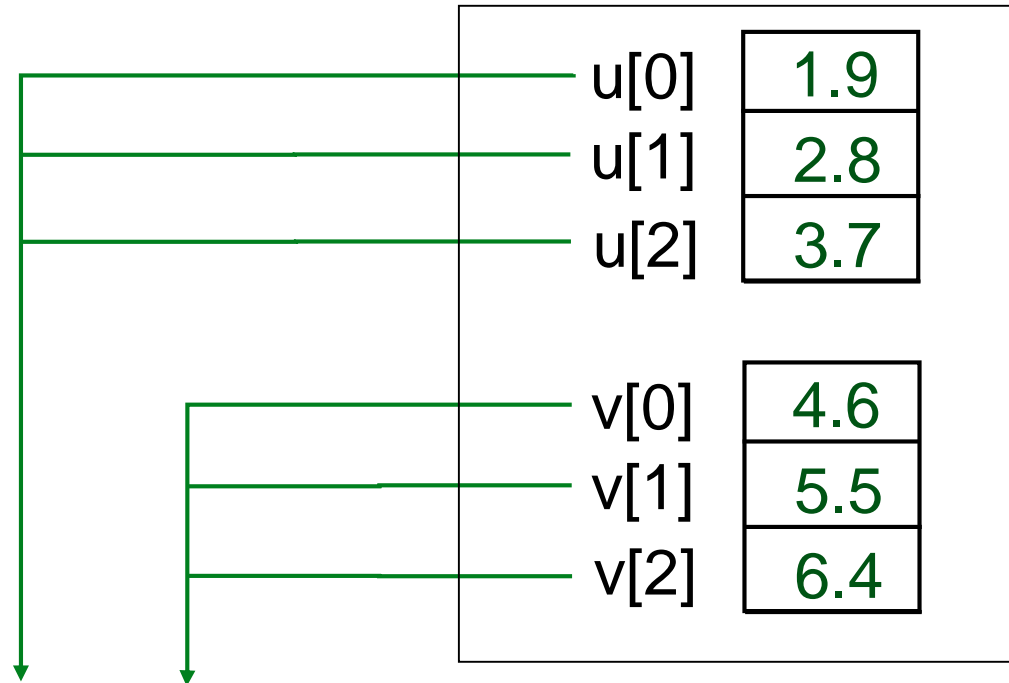
配列への  
書き込み

配列からの  
読み出し

## 実行結果の例

```
p(product) = 47.820000
```

## メモリ



```
p = p + (u[i] * v[i]);
```

配列からの  
読み出し

# ベクトルの内積



	i の値	繰り返し条件式 が成り立つか	ip の値
繰り返し 1 回目	$i = 0$	$i < 3$ が成り立つ	$p = p + u[0] * v[0];$ つまり ip の値は $u[0]*v[0]$
繰り返し 2 回目	$i = 1$	$i < 3$ が成り立つ	$p = p + u[1] * v[1];$ つまり ip の値は $u[0]*v[0] + u[1]*v[1]$
繰り返し 3 回目	$i = 2$	$i < 3$ が成り立つ	$p = p + u[2] * v[2];$ つまり ip の値は $u[0]*v[0] + u[1]*v[1]$ $+u[2]*v[2]$
繰り返し 4 回目	$i = 3$	$i < 3$ が成り立たない	



## 例題 3 . 棒グラフを描く

- 整数の配列から, その棒グラフを表示するプログラムを作る.
  - ループの入れ子で, 棒グラフの表示を行う



```
public class Main
{
    public static void main(String[] args) {
        int i, j;
        int a[] = {6, 4, 7, 1, 5, 3, 2};
        for(i = 0; i < 7; i++) {
            for(j = 1; j <= a[i]; j++) {
                System.out.printf("*");
            }
            System.out.printf("¥n");
        }
    }
}
```

配列への  
書き込み

配列からの  
読み出し

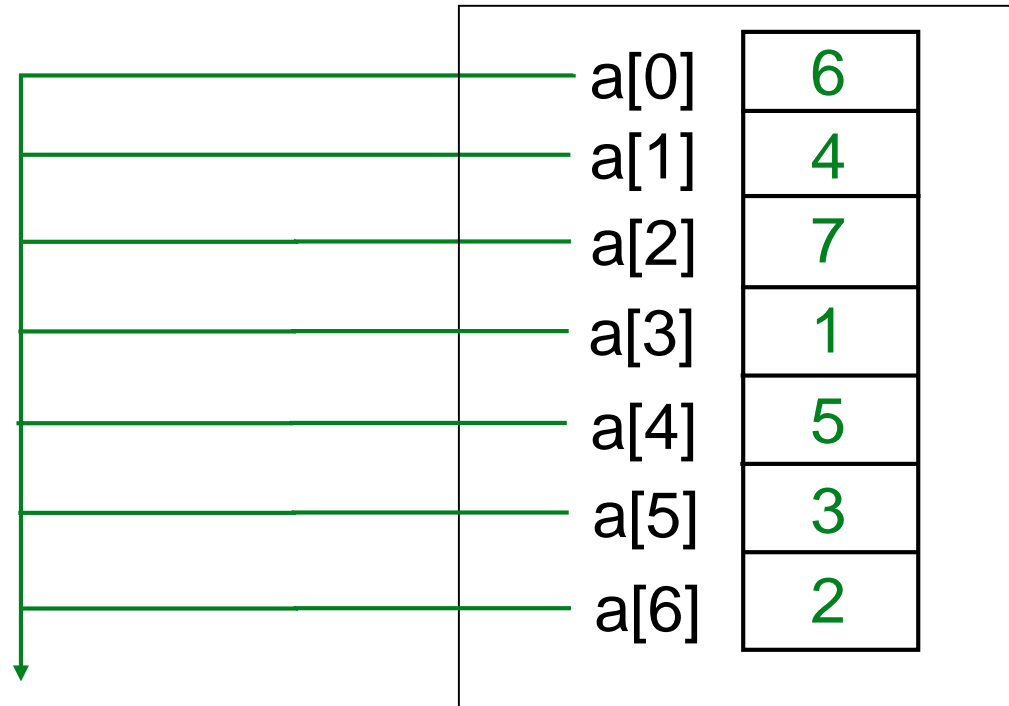
# 棒グラフを描く



## 実行結果の例

```
*****  
****  
*****  
*  
*****  
***  
**
```

## メモリ



```
for(j = 1; j <= a[i]; j++) {
```

配列からの  
読み出し



# 例題 4 . Horner 法による多項式の計算



## • n次の多項式

$$f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdot \cdot \cdot + a_n \cdot x^n$$

について, **次数 n** と, **係数 a<sub>0</sub> から a<sub>n</sub>** を読み込んで, **f(x)** を計算するプログラムを作る

- まず n を読み込んで, その後に a<sub>0</sub> から a<sub>n</sub> を読み込む. 最後に x を読み込む.
- 次ページで説明する Horner法を使う
- 読み込んだ係数は, いったん配列に格納する. n は高々 20 とする.

# Horner法



$$\begin{aligned}f(x) &= a_0 + a_1 \cdot x + a_2 \cdot x^2 + \dots + a_n \cdot x^n \\ &= a_0 + (a_1 + (a_2 + \dots + (a_{n-1} + a_n \cdot x) x \cdot \\ &\quad \cdot \cdot) x) x\end{aligned}$$

$$\begin{aligned}\text{例えば, } & 5 + 6x + 3x^2 \\ &= 5 + (6 + 3x) x\end{aligned}$$

計算手順

- ①  $a_n$
- ②  $a_{n-1} + a_n \cdot x$
- ③  $a_{n-2} + (a_{n-1} + a_n \cdot x) x$
- ⋯ (  $a_0$  まで続ける )

```
import java.util.Scanner;
public class Main
{
```



```
    public static void main(String[] args) {
        int i, n;
        double x, y;
        double a[] = new double[20];
        Scanner s = new Scanner(System.in);
        System.out.println("Please Enter n =");
        n = s.nextInt();
        for(i = 0; i <= n; i++) {
            System.out.printf("Please Enter a[%d] =¥n", i);
            a[i] = s.nextFloat();    配列への書き込み
        }
        System.out.println("Please Enter x =");
        x = s.nextFloat();
        y = a[n];    配列からの読み出し
        i = n - 1;
        while(i >= 0) {
            y = y * x + a[i];    配列からの読み出し
            i = i - 1;
        }
        System.out.printf("y = %8.3f", y);
    }
}
```

# 実行結果の例



```
Please Enter n =  
2  
Please Enter a[0] =  
5  
Please Enter a[1] =  
6  
Please Enter a[2] =  
3  
Please Enter x =  
3  
y = 50.000
```

## 例題 5. エラトステネスのふるい



- 「**エラトステネスのふるい**」の原理に基づいて**素数**を求め、結果を表示するプログラムを作成する
  - **配列**を使う
  - 配列には、**添字が素数なら 1**，そうでなければ **0** をセットする

# エラトステネスのふるい (1/3)



2   3   4   5   6   7   8   9   10   11   . . .

          ○          ○          ○          ○

$2 \times 2$            $2 \times 3$            $2 \times 4$            $2 \times 5$

まず、2の倍数を消す

# エラトステネスのふるい (2/3)



2 3 4 5 6 7 8 9 10 11 . . .

$3 \times 2$   $3 \times 3$

次に, 3の倍数を消す

# エラトステネスのふるい (3/3)



2 3 4 5 6 7 8 9 10 11 . . .  
5 × 2

次に, 5 の倍数を消す  
(「4 の倍数」は考えない.  
それは, 「4」がすでに消えているから)





```
import java.lang.Math;
import java.util.Scanner;
public class Main
{
    public static void main(String[] args) {
        int i, n, max;
        int p[] = new int[100000];
        Scanner s = new Scanner(System.in);
        System.out.println("Please Enter max =");
        max = s.nextInt();
```

```
for(i = 1; i <= max; i++) {
    p[i] = 1;
}
```

p[1] から a[100] までを  
「1」にセット

```
n = 2;
```

まず「2」の倍数は、素数ではない

```
while(n <= Math.sqrt(max)) {
```

```
    i = 2;
    while((n * i) <= max) {
        p[n * i] = 0;
        i = i + 1;
    }
```

n の倍数は素数ではない

```
    n = n + 1;
    while((p[n] == 0) && (n <= Math.sqrt(max))) {
        n = n + 1;
    }
```

n の「次」の  
素数を探す

```
    }
    for(i = 2; i <= max; i++) {
        if(p[i] == 1) {
            System.out.printf("%d, ", i);
        }
    }
}
```

求めた素数の表示

# 実行結果の例



```
Please Enter max =  
100  
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97,
```