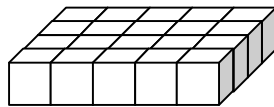


● 2次元配列

2次元配列

配列要素が直線上に並ぶ一次元配列に対して、
平面上に並ぶ配列要素をもつ配列

直観的には、



● 2次元配列の準備

配列変数の宣言は型と識別子を指定して次のように行う

```
型 識別子 [][];
```

または

```
型 [][] 識別子;
```

配列要素の確保は型と配列要素の個数を指定して次のように行う

```
識別子 = new 型 [配列要素の個数] [配列要素の個数];
```

配列要素の初期化は識別子と添え字を用いて次のように行う

```
識別子 [添え字] [添え字] = 値;
```

ソースコード例

ソースファイル名 : Sample11_1.java

```
// 配列を用いて 3 人の学生の 2 科目の点数を管理する
class Sample11_1
{
    public static void main(String[] args)
    {
        int i;

        // 配列変数の宣言
        int test[][];
        // int[][] test; と同記述可能

        // 配列要素の確保
        test = new int[2][3];

        // 配列変数の宣言と配列要素の確保は同時に記述可能
        // int test[][] = new int[2][3];
        // int[][] test = new int[2][3];

        // 各配列要素の初期化（兼、値の代入）
        // 添え字は 0 から（要素数） - 1 まで！！
        test[0][0]=80; test[0][1]=60; test[0][2]=22;
        test[1][0]=50; test[1][1]=75; test[1][2]=90;

        // 各配列要素を順番に出力
        for(i=0; i<3; i++)
        {
            System.out.println((i+1)+"番目の学生の国語の点数は"+test[0][i]+"です。");
            System.out.println((i+1)+"番目の学生の算数の点数は"+test[1][i]+"です。");
        }
    }
}
```

実行画面

```
>java Sample11_1
1 番目の学生の国語の点数は 80 です。
1 番目の学生の算数の点数は 50 です。
2 番目の学生の国語の点数は 60 です。
2 番目の学生の算数の点数は 75 です。
3 番目の学生の国語の点数は 22 です。
3 番目の学生の算数の点数は 90 です。
-- Press any key to exit (Input "c" to continue) --
```

● 2次元配列の初期化

2次元配列の初期化は識別子の宣言時に{}をさらに入れ子（ネスト）にして次のように行う

```
型 識別子 [][] = {{ 値1, 値2, . . . }, {値A, 値B, . . . }, . . . { . . . } };
```

または

```
型 [][] 識別子 = {{ 値1, 値2, . . . }, {値A, 値B, . . . }, . . . { . . . } };
```

ソースコード例

ソースファイル名 : Sample11_2.java

```
// 2次元配列の初期化
class Sample11_2
{
    public static void main(String[] args)
    {
        int i;
        // 2次元配列の初期化
        int test[][]={{80, 60, 22}, {50, 75, 90}};
        // int[][] test={{80, 60, 22}, {50, 75, 90}}; と同記述可能

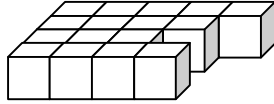
        // 各配列要素を順番に出力
        for(i=0; i<3; i++)
        {
            System.out.println((i+1)+"番目の学生の国語の点数は"+test[0][i]+"です。");
            System.out.println((i+1)+"番目の学生の算数の点数は"+test[1][i]+"です。");
        }
    }
}
```

実行画面

```
>java Sample11_2
1 番目の学生の国語の点数は 80 です。
1 番目の学生の算数の点数は 50 です。
2 番目の学生の国語の点数は 60 です。
2 番目の学生の算数の点数は 75 です。
3 番目の学生の国語の点数は 22 です。
3 番目の学生の算数の点数は 90 です。
-- Press any key to exit (Input "c" to continue) --
```

● 不規則な 2 次元配列

Java では、各行の配列要素の数がそれぞれ異なる 2 次元配列を容易に作成できる



不規則な 2 次元配列は次の 2 通りの方法で作成できる

1. 配列の初期化を利用して作成する方法
2. 配列変数の宣言と配列要素の確保の手順にしたがい作成する方法

1. 配列の初期化を利用して作成する方法

ソースコード例

ソースファイル名 : Sample11_3.java

```
// 不規則な 2 次元配列で初期化する
class Sample11_3
{
    public static void main(String[] args)
    {
        int i, j;

        // 2 次元配列の初期化
        int test[][]={{80, 60, 22}, {50, 75}, {72, 33, 75, 63}};

        // 各行の配列要素数 (.length の詳細は次節) と要素を順番に出力
        for(i=0; i<3; i++)
        {
            System.out.println((i+1)+"行目の要素数は"+test[i].length+"です。");
            for(j=0; j<test[i].length; j++)
                System.out.print(test[i][j]+" ");
            System.out.println();
        }
    }
}
```

実行画面

```
>java Sample11_3
1 行目の要素数は 3 です。
80 60 22
2 行目の要素数は 2 です。
50 75
3 行目の要素数は 4 です。
72 33 75 63
-- Press any key to exit (Input "c" to continue) --
```

○ .length 修飾子とその利用

.length 修飾子 配列要素の数を得るための特別な文法

1 次元配列の場合 :

<code>配列変数.length</code>	配列要素の数
--------------------------	--------

2 次元配列の場合 :

<code>配列変数.length</code>	行数
<code>配列変数[i].length</code>	第 i 行の配列要素の数

3 次元配列の場合 :

<code>配列変数.length</code>	行数
<code>配列変数[i].length</code>	第 i 行の列数
<code>配列変数[i][j].length</code>	第 i 行 j 列の配列要素の数

(4 次元以降、同様)

ソースコード例

ソースファイル名 : Sample11_4.java

```
// .length 修飾子
class Sample11_4
{
    public static void main(String[] args)
    {
        // 配列の初期化
        int test[]={72, 33, 75, 63};
        System.out.println("配列要素の数は"+test.length+"です。");
    }
}
```

実行画面

```
>java Sample11_4
配列要素の数は 4 です。
-- Press any key to exit (Input "c" to continue) --
```

2. 配列変数の宣言と配列要素の確保の手順にしたがい作成する方法

ソースコード例

ソースファイル名 : Sample11_5.java

```
// 不規則な 2 次元配列を宣言する
class Sample11_5
{
    public static void main(String[] args)
    {
        int i;

        // 2 次元配列 = 1 次元配列の配列 と考える
        int[] test[]; // int の配列型からなる配列

        // 下記と同じ意味であるが、上のように考えたほうが分かりやすい
        // int test[][];
        // int[][] test;

        test = new int[3][]; // 配列は 3 行からなる

        test[0] = new int[3]; // 1 行目の配列要素は 3 つ
        test[1] = new int[2]; // 2 行目の配列要素は 2 つ
        test[2] = new int[4]; // 3 行目の配列要素は 4 つ

        // 各行の配列要素数 (.length の詳細は次節) と要素を順番に出力
        System.out.println("配列は"+test.length+"行からなる");
        for(i=0; i<test.length; i++)
        {
            System.out.println((i+1)+"行目には"+test[i].length+"個の配列要素がある");
        }
    }
}
```

実行画面

```
D:\home\makio\Fit\Lectures\JavaProgramming1\11\Sources>java Sample11_5
配列は 3 行からなる
1 行目には 3 個の配列要素がある
2 行目には 2 個の配列要素がある
3 行目には 4 個の配列要素がある
-- Press any key to exit (Input "c" to continue) --
```

参考：例題 Sample11_5 の変数の振る舞いは図的に次のように理解できる

```
int[] test[]; // int 型の配列型の配列型の変数
```

