

11回目 多次元配列

■ 今日の講義で学ぶ内容 ■

- ・ 2次元配列とその使い方
- ・ 不規則な2次元配列
- ・ .length 修飾子

2次元配列

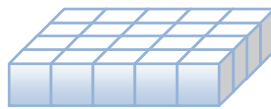
1次元配列

配列要素が直線的に並ぶ配列です
次のように考えると分かりやすいでしょう



2次元配列

配列要素が平面的に並ぶ配列です
次のように考えると分かりやすいでしょう



⚠ 2次元以上の配列のことを多次元配列といいます

2次元配列の準備

- 2次元配列の準備手順
1. 配列変数の宣言
 2. 配列要素の確保
 3. 配列要素の初期化

⚠ 配列の次元が2次元以上の場合も1次元配列の場合と同様です

1. 配列変数の宣言

型と識別子を指定して次のように行います

```
型 識別子[ ][ ];
```

または

```
型[ ][ ] 識別子; // Java での標準のスタイル
```

⚠ 括弧[]の数が配列の次元を表します

2. 配列要素の確保

型と配列要素の個数を指定して次のように行います

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

3. 配列要素の初期化

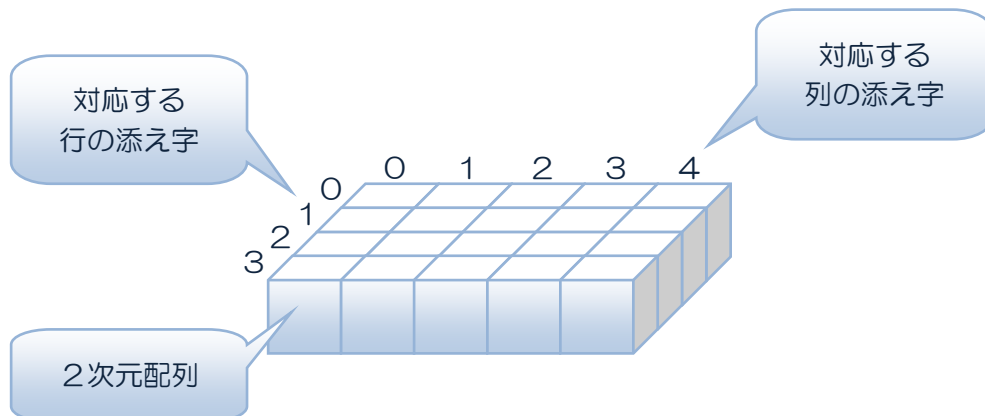
添え字を用いて各配列要素を参照して次のように行います

```
識別子 [ 行の添え字 ][ 列の添え字 ] = 値;
```

添え字には配列要素の位置を表す行方向または列方向の値を指定します

- ・ 行の添え字には 0 ～ (行の配列要素の個数 - 1) までの値を指定します
- ・ 列の添え字には 0 ～ (列の配列要素の個数 - 1) までの値を指定します

例えば、4行5列の配列要素をもつ2次元配列の場合は次のようにして添え字を指定します



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（3行）の2科目の点数（2列）を管理する
class Sample11_1
{
    public static void main(String[] args)
    {
        // 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[3][2];

        // 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[3][2];
        // int[][] test = new int[3][2];

        // 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]=50;
        test[1][0]=60; test[1][1]=75;
        test[2][0]=22; test[2][1]=90;

        // 各配列要素を順番に出力
        for(int i=0; i<3; i++)
        {
            System.out.println((i+1)+"番目の学生：");
            System.out.println("国語"+test[i][0]+"点 算数"+test[i][1]+"点");
        }
    }
}
```

実行画面

```
>java Sample11_1
1 番目の学生：
国語 80 点 算数 50 点
2 番目の学生：
国語 60 点 算数 75 点
3 番目の学生：
国語 22 点 算数 90 点
-- 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、配列変数の宣言時に次のように行います
指定する値が指定する行と列に代入された配列要素をもつ配列変数が宣言されます

```
型 識別子[][] = {{ 1行1列, 1行2列, ... }, { 2行1列, 2行2列, ... }, ... { ... } };
```

または

```
型[][] 識別子 = {{ 1行1列, 1行2列, ... }, { 2行1列, 2行2列, ... }, ... { ... } };
```

⚠ 括弧{ }が入れ子になっていることに注意しましょう

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)
    {
        // 2次元配列の初期化
        int[][] test={{80, 50}, {60, 75}, {22, 90}};
        // int test[][]={{80, 50}, {60, 75}, {22, 90}}; と同記述可能

        // 各配列要素を順番に出力
        for(int i=0; i<3; i++)
        {
            System.out.println((i+1)+"番目の学生：");
            System.out.println("国語"+test[i][0]+"点 算数"+test[i][1]+"点");
        }
    }
}
```

実行画面

```
>java Sample11_2
1 番目の学生：
国語 80 点 算数 50 点
2 番目の学生：
国語 60 点 算数 75 点
3 番目の学生：
国語 22 点 算数 90 点
-- Press any key to exit (Input "c" to continue) --
```

2次元配列の配列要素の初期化は、配列要素の確保時に次のように行います
指定する値が指定する行と列に代入された配列要素が確保されます

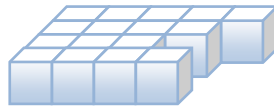
```
型[][] 識別子 = new 型[][] {{ 1行1列, ... }, { 2行1列, ... }, ... { ... } };
```



配列の初期化では配列要素の数を指定しません
配列要素の数は{}内の値より自動的に計算されます

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ます



不規則な2次元配列は次の2通りの方法で作成することができます

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}};

        // 各行の列数と要素を順番に出力します
        // .length は指定行の列数を値としてもちます (.length の詳細は次節)
        for(i=0; i<3; i++)
        {
            System.out.print((i+1)+"行目の要素数は"+test[i].length+"で、");
            for(j=0; j<test[i].length; j++)
                System.out.print(test[i][j]+" ");
            System.out.println("です");
        }
    }
}
```

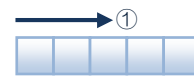
実行画面

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

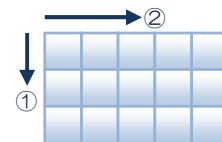
? .length 修飾子とその利用

.length 修飾子 配列要素の数を得るための修飾子

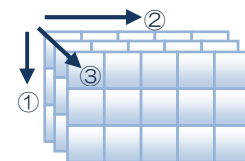
1 次元配列の場合： ① `配列変数.length` 配列要素の数



2 次元配列の場合： ① `配列変数.length` 行数
② `配列変数[i].length` 第 i 行の列数



3 次元配列の場合： ① `配列変数.length` 行数
② `配列変数[i].length` 第 i 行の列数
③ `配列変数[i][j].length` 第 i 行 j 列の高さ



(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)
    {
        // (ポイント)
        // 2次元配列は、1次元配列の配列 である

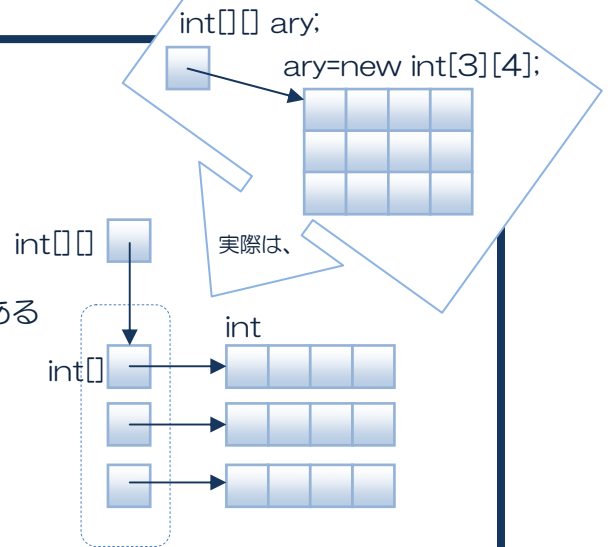
        // int 型の配列型の配列型の変数
        int[] test[];

        // 下記と同じ意味である
        // int test[][];
        // int[][] test;

        // 行数のみで部分的に 2 次元配列を作成
        test = new int[3][]; // 配列は 3 行からなる

        // 各行に列を作成して 2 次元配列は完成
        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

        // 各行の列数 (.length の詳細は前節) と要素を順番に出力
        System.out.println("配列は "+test.length+" 行からなる");
        for(int i=0; i<test.length; i++)
            System.out.println((i+1)+"行目には "+test[i].length+" 個の列がある");
    }
}
```



実行画面

```
>java Sample11_5
配列は 3 行からなる
1 行目には 3 個の列がある
2 行目には 2 個の列がある
3 行目には 4 個の列がある
-- 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ます

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

