

## 錬成問題

- 配列本体は、`new` 演算子によって動的に生成しなければならない (1) である。

```
int[] a = new int[5];
```

ここで宣言されている `a` は、配列本体を参照するための変数であり、(2) 変数と呼ばれる。

なお、どこからも参照されなくなった (1) は、(3) の機能によって自動的に回収される。

- 配列内の任意の構成要素を参照するための演算子 `[]` の名称は (4) 演算子である。この演算子を適用した式 `a[i]` は、配列 `a` における先頭から (5) 個目の構成要素にアクセスする。構成要素の位置を指定するための `[]` 内の整数値のことを (6) と呼ぶ。

- 以下の宣言を考える。

```
int[][] b = new int[4][3];
```

`b` の型は (7) で、`b[0]` の型は (8) で、`b[0][0]` の型は (9) であり、`b.length` の値は (10) で、`b[0].length` の値は (11) である。

- 何も参照しない (参照先をもたない) 参照を (12) と呼ぶ。その型は (13) 型であり、それを表すリテラルが (14) である。

- 配列本体の構成要素は、既定値と呼ばれる値で初期化される。各型の既定値は右のようになっている。

型	既定値
<code>int</code>	(15)
<code>long</code>	(16)
<code>float</code>	(17)
<code>double</code>	(18)
<code>boolean</code>	(19)
参照型	(20)

- 配列内の要素を一つずつ順になぞっていく手続きのことを (21) と呼ぶ。

- 以下に示すのは、要素の値が先頭から順に 1, 2, 3 である配列の宣言である。

```
int[] a = (22);
```

```
int[] a = new int (23) (22);
```

- 以下に示すプログラムの実行結果を示せ。

```
int[] a = new int[3];
for (int i = 0; i < 3; i++)
    System.out.println(a[i] + " ");
```

(24)

- 以下に示すのは、要素数5の **int** 型配列 *a* を生成し、その要素に、先頭から順に 10, 20, 30, 40, 50 を代入するプログラムである。

```
int[] a = (25);
for (int i = 0; i < (26); i++)
    a[i] = (27);
```

- 以下に示すのは、要素数が *n* である **int** 型配列 *a* の要素の最大値と最小値の差を求めて表示するプログラムである。

```
int min = (28);
int max = (29);
for (int i = (30); i < n; i++) {
    if (min > (31)) min = (31);
    if (max < (32)) max = (32);
}
System.out.println("最大値と最小値の差は" + (33) + "です。");
```

- 以下に示すのは、**double** 型配列 *a* の全要素の並びを逆転させるプログラムである（要素の値が先頭から順に 1.0, 2.0, 3.0 であれば 3.0, 2.0, 1.0 にする）。

```
for (int i = 0; i < (34); i++) {
    double t = a[i];
    a[i] = a[(35)];
    a[(36)] = (37);
}
```

- 以下に示すのは、**float** 型配列 *a* の全要素の合計を変数 *sum* に格納するプログラムである。なお、2 番目のプログラムの **for** 文は、(38) 文と呼ばれる。

```
float sum = (39);
for (int i = 0; i < (40); i++)
    sum += (41);
```

```
float sum = (39);
for ((42) i : (43))
    sum += (44);
```

- 以下に示すのは、**int** 型配列 *a* の全要素の値を先頭から順にコンマで区切って表示するプログラムである（要素の値が先頭から順に 1, 2, 3 であれば『1, 2, 3』と表示）。

```
if ((45) >= 2)
    for (int i = 0; i < (46); i++)
        System.out.print(a[i] + ", ");
if ((47) >= 1)
    System.out.print(a[(48)]);
```

- 配列変数は、型に分類され、**int** 型や **double** 型などの基本型とは異なる。
- 以下に示すプログラムの実行結果を示せ。

```
int[] a = {1, 2, 3, };
for (int i = 0; i < 3; i++)
    System.out.print(a[i] + " ");
```

- 以下に示すのは、**int** 型配列 *a* の全要素の値を記号文字 \* を横に並べたグラフで表示するプログラムである (実行例に示すのは要素が {3, 5, 2, 7} である場合)。

```
for (int i = 0; i < ; i++) {
    System.out.print("a[" +  + "] : ");
    for (int j = 0; j < ; j++)
        System.out.print('*');
    System.out.println();
}
```

```
a[0] : ***
a[1] : *****
a[2] : **
a[3] : *****
```

- 以下に示すのは、**int** 型配列 *a* の全要素の値 (正であるとする) を記号文字 \* を縦に並べた下向きグラフで表示するプログラムである (実行例に示すのは要素が {3, 5, 2, 7, 8, 4, 1, 9, 1, 0, 3, 4, 5} である場合)。最初の行に出力するのは、インデックスの最下位桁である。

```
int max = ;
for (int i = 0; i < ; i++)
    if (a[i] > )  = a[i];

for (int i = 0; i < ; i++)
    System.out.print();
System.out.println();

for (int i = 1; i < ; i++) {
    for (int j = 0; j < ; j++)
        if (a[j] >= )
            System.out.print("*");
        else
            System.out.print(" ");
    System.out.println();
}
```

```
0123456789012
***** * **
***** * ***
** *** * ***
* *** * **
* ** * *
** *
** *
* *
*
```

- データの集合から、ある値をもつ要素の存在を調べることを  と呼び、調べる値のことを  と呼ぶ。配列からの探索は、配列の全要素を順に走査することによって実現できる (探すべき  と同じ値の要素に出会ったら、探索に成功である)。このアルゴリズムは、 と呼ばれる。以下に示すのは、配列 *a* の要素のうち、値が *key* である要素を探し、見つけた位置のインデックスの値を *idx* に格納するプログラムである。なお、見つからなかった場合には *idx* に -1 を格納する。

```
int i;
for (i = 0; i < ; i++)
    if ( == key) ;
int idx = ;
```

- 以下に示すのは、`int` 型配列 `a` の複製（要素数が同一で、すべての要素の値が同じ）を `b` として作るプログラムである。

```
int[] b = (69);
for (int i = 0; i < (70); i++)
    b[i] = (71);
```

- 以下に示すのは、`int` 型配列 `a` の要素の中で値が正であるものを順に並べた要素をもつ配列 `b` を作るプログラムである（配列 `a` が {5, -1, 3, 4, -2, 7} であれば、配列 `b` は {5, 3, 4, 7} となる）。

```
int count = 0;
for (int i = 0; i < (72); i++)
    if (a[i] > 0) (73);
int[] b = (74);
int j = 0;
for (int i = 0; i < (75); i++)
    if (a[i] > 0) (76);
```

- 以下に示すのは、すべての要素の値が 0 である 2 行 3 列の `int` 型の 2 次元配列 `a` を作るプログラムである。

```
int[][] a = new (77);
```

```
int[][] a = {(78)};
```

```
int[][] a = {new (79), new (80)};
```

```
int[][] a;
a = new (81);
a[0] = new (82);
a[1] = new (83);
```

- 以下に示すのは、`int` 型 2 次元配列 `a` の複製（要素数が同一で、すべての要素の値が同じ）を `b` として作るプログラムである。

```
int[][] b = (84);
for (int i = 0; i < (85); i++) {
    b[i] = (86);
    for (int j = 0; j < (87); j++) {
        b[i][j] = (88);
    }
}
```

- 以下に示すプログラムについて、コンパイルエラーとなる行には×を、エラーとならない行には○を記入せよ。

```

int[] a = new int[5];
final int[] b = new int[5];
(89) a.length = 10;
(90) b.length = 10;
(91) a = b;
(92) b = a;
(93) a[0] = 10;
(94) b[0] = 10;

```

- 以下に示すのは、月を表す英単語の学習プログラムである。
  - 1月～12月の英単語 "January", "February", … を入力させる。
  - たとえば、『8月:』と表示した場合は、"August"を入力しなければならない。
  - 入力された英単語の綴りがあったら『正解です。』と表示し、そうでなければ『違います。』と表示する（間違えた場合再入力はできない）。
  - 出題は12回であり、すべての月を出題する。
  - 出題の順序はランダムである。
  - 12回のうちの何回正解したのかを表示する。

※変数 `rand` は、本文 p.34 で解説したとおりに宣言されているものとする（次章以降の錬成問題でも同様である）。

```

(95) monthString = (96)
    "January", "February", "March", "April", "May", "June", "July",
    "August", "September", "October", "November", "December"
(97);

System.out.println("英語の月名を入力してください。");
System.out.println("先頭は大文字で、2文字目以降は小文字とします。");

(98) order = new int[12];
for (int i = 0; i < (99); i++)
    order[i] = i;
for (int i = 0; i < 24; i++) {
    int idx1 = rand.nextInt((100));
    int idx2 = rand.nextInt((101));
    int t = order[idx1];
    order[idx1] = (102);
    order[idx2] = (103);
}

int correct = 0; // 正解した回数
for (int i = 0; i < 12; i++) {
    int month = order[i];
    System.out.print((104) + "月:");
    String s = stdin.next();

    if (s.equals(monthString[(105)])) {
        System.out.println("正解です。");
        (106);
    } else {
        System.out.println("違います。");
    }
}

System.out.println("12回中" + correct + "回正解でした。");

```

- 以下に示すのは、2次元配列 a の全要素の値を表示するプログラムである（行によって列数が異なる可能性がある）。

```

System.out.println("{}");
for (int i = 0; i < (107); i++) {
    System.out.print("  {}");
    if ((108) >= 2)
        for (int j = 0; j < (109); j++)
            System.out.print(" (110) + ", "");
    if ((111) >= 1)
        System.out.println(" (112) + ", "");
}
System.out.println("{}");

```

```

{
  {1, 2, 3, 4, 5},
  {1, 3, 5},
  {2, 4, 6, 8},
}

```

## 6

## 配列

- 以下に示すのは、行によって列数が異なる可能性がある2次元配列 a の全要素の値をキーボードから読み込んで表示するプログラムである。

- 要素の値は0～99である。この範囲外の値が入力された場合は再入力させる。
- 最大列数に満たない行に関しては、存在しない要素の箇所に『-』を表示する。

```

System.out.print("行数: ");
int height = stdIn.nextInt();

int[][] a = new int (113);
int maxWidth = 0;

for (int i = 0; i < (114); i++) {
    System.out.print(i + "行目の列数: ");
    int width = stdIn.nextInt();
    a[i] = new int (115);
    if (width > maxWidth) maxWidth = (116);
}

System.out.println("各要素の値[0~99]を入力せよ。");
for (int i = 0; i < (117); i++) {
    for (int j = 0; j < a[i].length; j++) {
        do {
            System.out.printf("a[(118)][(119)]: ", i, j);
            a[i][j] = stdIn.nextInt();
        } while (a[i][j] < (120) || a[i][j] > (121));
    }
}

for (int i = 0; i < (122); i++) {
    int j = 0;
    for (; j < (123); j++)
        System.out.printf("%3d", (124));
    for (; j < (125); j++)
        System.out.print(" -");
    System.out.println();
}

```

```

行数: 3
0行目の列数: 5
1行目の列数: 3
2行目の列数: 4
各要素の値[0~99]を入力せよ。
a[0][0]: 1
a[0][1]: 2
a[0][2]: 3
a[0][3]: 4
a[0][4]: 5
a[1][0]: 6
a[1][1]: 7
a[1][2]: 8
a[2][0]: 9
a[2][1]: 10
a[2][2]: 11
a[2][3]: 12
  1  2  3  4  5
  6  7  8  -  -
  9 10 11 12  -

```