計算機プログラミング I (5)

沢山のデータを使った計算

2000 年 11 月 6 日 増原 英彦

前回までに、繰り返しなどを使って大量の計算を行うことを学んだ。一方、複数のデータを扱おうと思ったら、その数だけ変数を用意する必要があった。(例: 預金期間に応じて異なる金利を適用するためには、それぞれの期間ごとに金利を定義する変数を宣言する必要があった。また、「n 年間預金した後の残高を求める」例では、「現在の残高」を表わす一つの変数を繰り返し使っていたので、10 年目を計算したときに 2 年目の残高は忘れてしまっていた。)

プログラミング言語では、沢山のデータを一様に扱う方法として、配列(array) というものがある。配列を用いると、「n 番目のデータ」「$(3m+k)$ 番目のデータ」のように、沢山のデータを番号によって扱うことができるようになる。

1 データの列

名簿や数列などのように、1 列に並べられたデータを配列によって扱う方法を見てみる。配列は、「整数」「実数」「文字列」等のデータ型の 1 つである。

```
1 // 各月の日数: daysOfMonth[x-1]=x 月の日数 (閏年でないとする)
2 int[] daysOfMonth = new int[] { 31, 28, 31, 30, 31, 30,   // 1-6 月
3                                 31, 31, 30, 31, 30, 31 }; // 7-12 月
4 // 1 月から (month-1) 月までの日数を表示する
5 for (int month = 1; month <= 12; month = month+1)
6   System.out.println(month+"月は"+daysOfMonth[month-1]+"日あります。");
```

型: 配列は、「整数を入れる配列」や「文字列を入れる配列」のように中身 (要素) の型によって区別される。2 行目の "`int[]`" のように、中身の型 (`int`) の後ろに "`[ ]`" を続ける。

例: `int[]` (整数を入れる配列)、`String[]` (文字列を入れる配列)、`String[][]` (「文字列を入れる配列」を入れる配列)。

作る: 新しく配列を作る式は、2–3 行目の `new int[] { 31, ... }`のように、キーワード`new`、配列の型`int[]`、その中身`{ 31, 28, ... }`の順に書く。

中身を使う: 配列中の値を使って計算する式は、6 行目の`daysOfMonth[month-1]` のように、配列を表わす式 (`daysOfMonth`) に続けて、何番目の中身を使うかを示す添字(`month-1`) を "`[ ]`" で囲んで書く。

注意: 最初の中身は **0** 番目である。 例中で`month` 月の日数を`daysOfMonth[month-1]` という式によって求めているのはそのためである。

また、10 個の中身を持つ (つまり、0 ~ 9 番目まで中身がある) 配列の 10 番目の中身をとり出そうとすると、プログラムの実行中にエラーが報告され、プログラムが停止する。(対処方法は図 1 を参考にせよ。)

配列などを扱うプログラムに誤りがある場合、コンパイル (javac を実行) したときではなく、プログラム (java) を実行したときにエラーが報告されることがある。そのような場合、java コマンドに-debug オプションをつけて実行すると、エラーの起きた場所が表示されるため、誤りのある部分を見つけやすい。

```
masuhara@as301> cat Array.java                    (プログラムの中身を表示)
class Array { // 配列を使ったプログラム (誤りあり)
  public static void main(String[] args) {
    int[] scores = { 10, 9, 3, 8, 9 }; // 大きさ 5 の配列の
    System.out.println(scores[5]);      // 5 番目を表示
  }
}
masuhara@as301> javac Array.java                  (コンパイル)
masuhara@as301> java -debug Array                 (実行)
Agent password=4cieqp
java.lang.ArrayIndexOutOfBoundsException: 5
        at Array.main(Array.java:4)               (***)
```

上の例では、(***) の行が、「Array クラスのmain メソッドの中、(Array.java というファイルの 4 行目) でArrayIndexOutOfBoundsException(配列の添字が範囲外になった) という例外が起きた」ことを示している。

図 1: プログラムの誤りを見つける

練習 5-1: (配列の宣言)

1. 曜日の名前を順に格納した配列dayOfWeek を宣言せよ。ただし、dayOfWeek[0] の中身が「日」、dayOfWeek[1] の中身が「月」、のようになるとする。(ヒント: 配列の中身は String 型である。)
2. 同様に、月の名前を順に格納した配列nameOfMonth を宣言せよ。ただし、nameOfMonth[0] の中身が "January"、nameOfMonth[1] の中身が "Feburary" のようになるとする。

練習 5-2: (配列を使う 1) 例で定義したdaysOfMonth を使って、(1) 月の日数の平均値および (2)1 年の総日数を求めるプログラムを作れ。

練習 5-3: (配列を使う 2) m 月 d 日が元旦から数えて何日目かを計算するプログラムを作り、11 月 6 日が何日目かを計算させよ。
(ヒント: m 月 d 日までの日数は、((1 月から $(m-1)$ 月までの「月の日数」の合計) $+ d$) になる。)

2 計算によって作られる配列

配列の中身を計算によって作ることもできる。例えば、10 進数を 2 進数表示に直す場合、各桁の値を配列によって覚えさせることで、下の桁から計算して上の桁から表示するといったことが可能になる。

```
 1  int decimal = 987654;   // もとの数
 2  int maxDigits = 32;     // 最大の桁数
 3  int[] place = new int[maxPlaces];// place[x]=2 進数表示したときの x 桁目
 4  // 最下位桁から計算
 5  for (int index = 0; index < place.length; index = index + 1) {
 6    // decimal=(もとの数を 2 進数にしたときの index 桁目以上の部分)
 7    place[index] = decimal % 2; // もとの数の index 桁目は現在の decimal の最下位桁
 8    decimal = decimal / 2;      // decimal を 1 桁ずらす
 9  }
10  // 最下位桁から表示
11  for (int index = place.length-1; index >= 0; index = index -1)
12    System.out.print(place[index]);
```

配列を作る: 3 行目の `new int[maxPlaces]` は、整数型の値が`maxPlaces` 個入れられる配列を作る式である。このようにして作られた配列には、全て 0(あるいは空を示す値 (後述)) が入っている。

配列の大きさ: 5, 11 行目の `places.length` は、配列 `places` の大きさを得る式である。

配列への代入: 7 行目の `places[index] = decimal % 2;` は配列`places` の`index` 番目の中身へ計算結果を代入する式である。

練習 5-4: (コマンドの引数) Java プログラムを実行する際、下の例のように、ターミナルウィンドウ (kterm) に "`java` クラス名" に続けて、スペースで区切った適当な文字列を書いてゆくと、`String` 型の配列としてプログラムに渡される。例えば、

```
class ExampleProgram {
  public static void main(String[] args) {
    ...
```

というプログラムをコンパイルして、ターミナルウィンドウ (kterm) から "`java Example Animal Bird Cat Dog`" のように実行した場合、`args[0]` は`"Animal"`、`args[1]` は`"Bird"`のようになっている。

ターミナルウィンドウ (kterm) から

```
java Example 44 16 7 19 22
```

のように実行した場合、与えられた数の合計を表示するプログラムを作れ。なお、文字列を整数にするには、ライブラリを使って`Integer.parseInt("33")` のようにするとよい。

練習 5-5: (フィボナッチ数) フィボナッチ数とは、

$$f_n = \begin{cases} 1 & n = 0, 1 \\ f_{n-2} + f_{n-1} & \text{それ以外} \end{cases}$$

という関係によって定義される数列である。100 番目までのフィボナッチ数を求め、順に表示するプログラムを作れ。

練習 5-6: (エラトステネスのふるい) 素数の列を作る方法として、「エラトステネスのふるい」(sieve of Eratosthenes) と呼ばれるやり方がある:

- まず、2 から始まる自然数の列を用意する
- 列の先頭から順に値を調べてゆき、その値 x が
  - 消されていなければ、x は素数である。列の残りから x の倍数を消す。
  - 消されていれば x は素数でない。

このようにすると、消されずに残った数は素数になっている。

「数 x が消されていない」場合に `isPrime[x]` が `true` であるような をboolean 型の配列を定義し、1,000,000 未満の素数を大きい順に 10 個表示させるプログラムを作れ。

3 高次の配列

ここまでで見た配列はデータが 1 列に並んでおり、1 つの添字を使って中身を扱っていた。つまり、座標系としては**1** 次元の座標になる。

世の中には、縦横にデータが並んでいるものもある。例えば、Washington D.C. では、東西方向に走る道路に南から A, B, C, ... と名前を付け、南北方向に走る道路には東から 1, 2, 3, ... のと名前を付けて、「ホワイトハウスは 16 の F」のような番地の表わし方をしている。つまり、**2** 次元の座標系を使って場所を表わしている。同様に、3 次元、4 次元... ... の座標系を使って表わされる情報もあるだろう。

3.1 碁石でお絵描き

Java 言語では、高次元のデータは配列の配列として扱う。例として碁盤の上に碁石を並べてるように、2 次元の格子状に文字を並べることで図形を描くプログラムを見てみる。この場合、碁盤の格子点 (オセロだったらマス目)1 つを、1 つの文字とした配列を使う。

```
1  int width=60, height=25;                    // 「画面」の幅と高さ
2  char[][] canvas = new char[height][width]; // 「画面」は 2 次元の文字配列
3  // 「画面」の全ての場所を空白にする
4  for (int row = 0; row < height; row=row+1)
5    for (int column = 0; column < width; column=column+1)
6      canvas[row][column] = ' ';
7  // 「画面」に正弦曲線を描く
8  for (int column = 0; column < width; column=column+1) {
9    double x = 2*Math.PI*column/width;          // column:[0,width]..x:[0,2π]
10   double y = Math.sin(x);                     // y=sin(x) のグラフ
11   int row = (int)(-height*y/2.2 + height/2); // row:[0~height]..y:[1.1,-1.1]
12   canvas[row][column] = '*';                  // 計算した場所を「*」にする
13 }
14 // 「画面」の中身を端末に書く
15 for (int row = 0; row < height; row=row+1) {
16   for (int column = 0; column < width; column=column+1)
17     System.out.print(canvas[row][column]); // 中身を表示
18   System.out.println(); // 1 行書いたら改行する
19 }
```

図 2: 正弦曲線を描くプログラム

- 2 行目では、大きさ (height × width) の文字からなる配列を定義している。このプログラムでは、canvas[r][c] に、「画面」の上からr 行目の左から c 桁目に書かれる文字を入れることにする。

  ここで、char[][] のように、「[]」を続けて書くことで高次の配列の型を表わすことができる。

  また、new char[height][width] のように、各次元の大きさを並べて書くことで高次の配列を作ることができる。

- 3–6 行は、「面画」を空白で埋めつくすための作業。' ' は、空白を表わす文字型である。

  ここで、char[row][column] = ' '; のように、各次元の添字を順に「[ ]」で囲んで書くことで、高次の配列の中身を使ったり、代入したりできる。

- 7–13 行は、「画面」上に $y = \sin x$ $(0 \leq x < 2\pi)$ を描く部分である。

  – まず、「画面」上の各桁に 1 つずつ点を打てるように、桁 column を 0 から (width − 1) までの間繰り返しをする。

- row 行 coulmn 桁を $0 \leq x < 2\pi, -1.1 \leq y < 1.1$ という平面に対応させるために、column から $x = 2\pi\texttt{column}/\texttt{width}$ を求め、
- $y = \sin x$ を計算し、
- y 座標から $\texttt{row} = -\texttt{height} \cdot y/2.2 + \texttt{height}/2$ を求める。($\Rightarrow$ (int) の意味についてはテキスト 4.8 節を見よ。)
- 最後に、row 行 column 桁の文字を「*」にする。

- 14–19 行は、「画面」の中身を端末に表示する部分である。単純に、0 行目から (height − 1) 行目までの行を順に、0 桁目から (width − 1) 桁目までの文字を順に表示することで表示している。

練習 **5-7:** (円を描く) 図 2 を参考に、円を描くプログラムを作れ。

3.2 でこぼこの 2 次元配列

Java 言語では、高次の配列は「配列の配列」として扱うため、必ずしも大きさがそろっている必要はない。例えば 2 次元配列を使ってcalendar[3-1][23-1] が 3 月 23 日に対応する[1]ようなカレンダを扱うのであれば、11 月は 30 個、12 月は 31 個の中身があるように作るのが自然なやり方だろう。

```
1  // 各月の日数: days[x-1]=x 月の日数（閏年）
2  int[] days = new int[] { 31, 29, 31, 30, 31, 30,   // 1-6 月
3                           31, 31, 30, 31, 30, 31 }; // 7-12 月
4  // 曜日を 0 ~ 6 の整数で表わす
5  int Sun=0, Mon=1, Tue=2, Wed=3, Thu=4, Fri=5, Sat=6;
6
7  int[][] calendar = new int[12][]; //calendar[m-1][d-1]=m 月 d 日の曜日
8  int dayOfWeek = Sat; //2000 年元旦は土曜日
9  // カレンダ (曜日表) を作る
10 for (int month = 1; month <= 12; month = month+1) {
11   // 各月の曜日表のための配列を作る
12   calendar[month-1] = new int[days[month-1]];
13   // 1 日から月末までの曜日を埋めてゆく
14   for (int day = 1; day <= days[month-1]; day=day+1) {
15     calendar[month-1][day-1] = dayOfWeek;
16     dayOfWeek=(dayOfWeek+1)%7; // 曜日を 1 つ進める
17   }
18 }
19
20 // 全ての月について調べる
21 for (int month = 1; month <= 12; month=month+1)
22   if (calendar[month-1][13-1] == Fri) // 13 日が金曜か?
23     System.out.println(month+"月 13 日は金曜日。");
```

図 3: 13 日の金曜日プログラム

図 3 のプログラムでは、そのようなことをやっている。ここでは、

- 7 行目のint[][] calendar = ...; で、整数型の配列の配列 (**2** 次元配列) を入れる変数を宣言し、
- ... = new int[12][]; で、整数型の配列 (**1** 次元配列を)**12** 個入れる配列を作り、calendar の値としている。この時点で、それぞれの月は空であることに注意。

[1] Java 言語では配列の添字は 0 から始まるので、1 月 1 日はcalendar[0][0] に対応する。

- 12 行目のcalendar[month-1] = new int[days[month-1]]; で、整数をその月の日数個入れる配列を作り、配列 calendar の (month-1) 番目に代入している。ここで、作る配列の大きさは月によって違うことに注意。

結局、このようにすると m 月 y 日に対応してcalendar[m-1][d-1] が使えるようになる。

練習 5-8: (組み合わせ数)　組み合わせ数 ${}_nC_m$ は、

$$ {}_nC_m = \begin{cases} 1 & m = 0, n \\ {}_{n-1}C_{m-1} + {}_{n-1}C_m & \text{それ以外} \end{cases} $$

という関係式で定義されている。${}_nC_m$ を C[n][m] という配列で表わすことにして、$n = 30$ までの組み合わせ数を全て計算して表示させよ。(ヒント: C[n] は大きさ $(n+1)$ の整数配列になる。)