配列

ここまでにデータを保存するための領域として int 型や double 型といった変数を利用してプログラムを作成してきた。これらのデータ型は単純に一つの変数に一つのデータを格納するために利用し、変数を説明する上では以下のように「データを出し入れする箱」として例えられる。

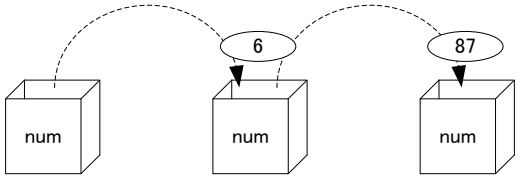

図１　変数のイメージ

それではプログラムを作成する際に変数が多数必要となった場合はどうなるだろうか。例えば同じようなデータを扱うために int 型の変数が 100 個必要となった場合、変数の宣言は 100 個の変数名を記述しなくてはならない。これでは非常に使い勝手が悪いので、多数の同じ型のデータを扱うために配列というデータ構造が用意されている。配列は for 文などの繰り返し処理と組み合わせることですっきりとした記述が可能となる。本日はこの配列の宣言と操作方法について説明する。

配列変数の宣言

ここでは同じ型のデータをまとめて管理する配列の役割と宣言方法について述べる。

例えば 5 人の英語の点数を管理することを考える。今までに習った単純な int 型変数を用いて宣言すると、以下のようになり、

```
int a0, a1, a2, a3, a4;
```

この場合では次の図のようなイメージとなる。

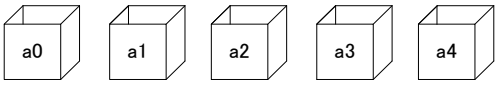

図２　多くの変数が定義された場合のイメージ

この方法でプログラムを作成すると、変数の宣言や値の代入の数が多くなるだけでなく、データ数が後から変更になった場合には、プログラム全体の見直しが必要になるため、かなりの手間がかかってしまう。

それでは配列を使った場合はどうなるだろうか。通常の変数の場合と等しく、まずは宣言を行う必要があり、以下のように 2 種類の方法が用意されている。

書式
①データ型[ ] 配列変数名; ②データ型 配列変数名[ ];

以下に使用例として、int 型のデータで、「配列変数名」を a として宣言したものを示す。

使用例
①int[ ] a; ②int a[ ];

ここでの変数 a は配列変数と呼ばれる特殊な変数であり、データを格納する配列本体ではない。

配列本体の生成

配列本体は以下の書式に従って、別に生成することになる。

書式
配列変数名 = new データ型[要素数]

int 型のデータで「配列変数名」を a として宣言した後に、配列の要素数を 5 として配列本体を生成した使用例を示す。

使用例
a = new int[5];

また、配列の生成は配列変数の宣言と同時に行うこと（配列変数の初期化）も可能である。

使用例
int[ ] a = new int[5];

この使用例の場合には、以下のように a[0]～a[4]といった「データを出し入れする連続した箱」が用意される。

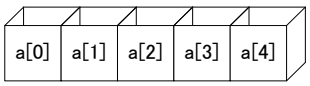

図３　配列のイメージ

この連続した箱の一つ一つは「配列の要素」と呼ばれ、以下の図のようにして、配列変数名と添字（インデックス）を用いることで、通常の変数と同様に値を自由に出し入れすることができる。

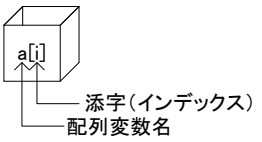

図４　配列の要素

注意点は配列の要素数を 5 として生成した場合（a = new int[5];）、添字は 0～4 までが指定できる。つまり、配列の要素は 0 番目から 4 番目の要素（a[0]～a[4]）として用意されるのである。

また、int 型変数に i を用いると、配列の i 番目の要素は a[i]として書くことができる。あるいは a[i/2]のように式を書いて要素を指定することもできる。ただし、それは整数である必要がある。

※Java のコンパイラでは、配列の上限を超えた値、または 0 より小さい値を添字として利用してしまった場合にはコンパイルエラーは出ない。しかし、たいがいの場合にはプログラムは正常に動作しなくなり、実行時にエラーを出して、プログラムは止まる。

配列の構成要素

まずは簡単なプログラム例で配列の使用方法を確認していく。以下は int 型の配列を作り、配列の要素に値を代入して、表示するプログラムである。

プログラム例 1

```
public class Prog06_01
{
        public static void main(String[ ] args)
        {
                int[] a = new int[5];
```

```
                a[1] = 37;                      // a[1]に 37 を代入
                a[2] = 51;                      // a[2]に 51 を代入
                a[4] = a[1] * 2;                // a[4]に 74 を代入

                System.out.println("a[" + 0 + "] = " + a[0]);
                System.out.println("a[" + 1 + "] = " + a[1]);
                System.out.println("a[" + 2 + "] = " + a[2]);
                System.out.println("a[" + 3 + "] = " + a[3]);
                System.out.println("a[" + 4 + "] = " + a[4]);
        }
}
```

実行結果

```
a[0] = 0
a[1] = 37
a[2] = 51
a[3] = 0
a[4] = 74
```

実行結果から分かるように、値を代入していない配列の要素は 0 が出力される。通常の変数の場合には値を代入する処理を行っていないとコンパイル時にエラーが発生したが、「**配列の要素は明示的に初期化しなくても 0 で初期化される**」という特徴がある。構成要素が初期化される値のことを「**既定値**」という。既定値について以下の表にまとめておく。

表１．各型の既定値

型	既定値
byte	ゼロ　すなわち　(byte)0
short	ゼロ　すなわち　(short)0
int	ゼロ　すなわち　0
long	ゼロ　すなわち　0L
float	ゼロ　すなわち　0.0f
double	ゼロ　すなわち　0.0d
char	空文字　すなわち　‘¥u0000’
boolean	偽　すなわち　false
参照型	空参照　すなわち　null

注：¥u が頭についた数値は 16 進数を表す。配列の要素だけでなく、これ以降で学ぶインスタンス変数やクラス変数といったものも、この既定値で初期化される。

配列の要素数の取得

次に要素数の取得方法を示す。これは以下の書式を用いる。

書式

```
配列変数名.length
```

配列の要素数は別名では配列の長さとも呼ばれる。この取得方法を用いたプログラム例を以下に示す。配列変数名は a であるので、a.length と書き、for 文の条件式に利用した。

プログラム例２

```
public class Prog06_02
{
        public static void main(String[ ] args)
```

```
        {
                int[] a = new int[5];   // 配列の宣言

                for(int i = 0; i < a.length; i++)
                        a[i] = i + 1;

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

実行結果

```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
```

図３で配列のイメージを示したが、より正しくは以下のようになる。

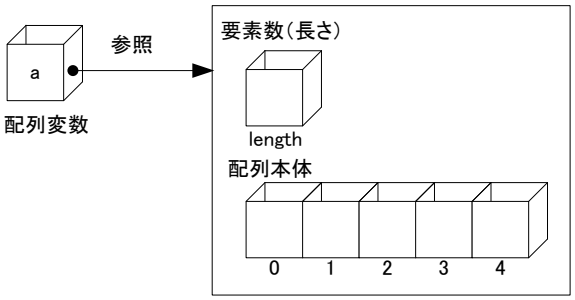

図５　配列

配列の要素への値の読み込み

次にキーボードから配列の要素数を入力して、配列本体を生成し、配列の要素に値を代入するプログラムの例を示す。

プログラム例３

```
import java.util.Scanner;

public class Prog06_03
{
        public static void main(String[ ] args)
        {
                Scanner stdIn = new Scanner(System.in);

                System.out.print("要素数:");
                int n = stdIn.nextInt();        // 要素数を読み込む
                int[] a = new int[n];           // 配列を生成

                for(int i = 0; i < n; i++)
                {
                        System.out.print("a[" + i +"] = ");
                        a[i] = stdIn.nextInt(); // 要素に値を代入
                }
```

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

実行結果（斜体は入力値）

```
要素数：5
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
```

ここでは変数 n に要素数を代入してあるので、for 文の条件式では n を用いたが、a.length として書いてもかまわない。

配列の初期化と代入

次に配列の初期化と代入方法について述べる。配列の全要素は配列が生成されると既定値である 0 で初期化が行われることを説明したが、あらかじめ、要素に入れる値が決まっているのであれば、明示的に初期化を行うことができる。以下のプログラム例を確認しよう。

プログラム例４

```
public class Prog06_04
{
        public static void main(String[ ] args)
        {
                int[] a = {1, 2, 3, 4, 5};

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

```
a[0] = 1
a[1] = 2
a[2] = 3
a[3] = 4
a[4] = 5
```

「int[] a = {1, 2, 3, 4, 5};」の部分が配列本体の初期化を行っている。new を使って配列本体を生成する処理が無いことに注意すること。配列変数の宣言と本体の生成を別に行った上で、初期化するには以下のように書く。

```
int[] a;
a = new int[ ]{1, 2, 3, 4, 5};
```

ただし、以下のように書くと間違いである。

```
int[] a;
a = {1, 2, 3, 4, 5};
```

a は配列本体への参照用の変数であり、そこに本体を代入することはできない。図５と照らし合わせて確認すること。

拡張 for 文

ここまでに示したように、配列は繰り返し文、特に for 文と組み合わせて利用する場合がほとんどである。ここで紹介する拡張 for 文は配列と組み合わせた特定の処理を簡潔に記述する方法である。また、ここまでに利用してきた for 文は拡張 for 文に対して明確に区別する場合には「基本 for 文」とも呼ばれることがある。拡張 for 文を用いると、配列の全要素へのアクセス（これを配列の走査という）を簡潔に表現できる。以下に示すプログラム例は配列の全要素の合計を求めて表示するプログラムである。

プログラム例５

```
public class Prog06_05
{
        public static void main(String[ ] args)
        {
                double[] a = { 1.0, 2.0, 3.0, 4.0, 5.0 };

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

                double sum = 0; // 合計
                for(double i : a)
                        sum += i;

                System.out.println("全要素の和は" + sum + "です。");
        }
}
```

```
a[0] = 1.0
a[1] = 2.0
a[2] = 3.0
a[3] = 4.0
a[4] = 5.0
全要素の和は 15.0 です。
```

拡張 for 文の記述は以下の部分である。

```
for(double i : a)
        sum += i;
```

for 文の () 内のコロン：は、“～の中の”という意味であり、そのため、この例での拡張 for 文は“ﾌｫｰ ﾀﾞﾌﾞﾙ ｱｲ ｲﾝ ｴｲ”と読まれることもある。この処理は以下の処理と等しい。

```
for(int i = 0; i < a.length; i++)
        sum += a[i];
```

この拡張 for 文は「配列の先頭から末尾までの全要素に１つずつアクセスし、ループの本体では、

現在着目している要素を変数 i で表す」。拡張 for 文内の「double」は走査の対象となっている配列 a のデータ型に合わせて変更する。拡張 for 文を用いた場合は、簡潔な形で配列の走査を行うことができる。ただし、「ループ本体でインデックスの値を使わないプログラム」である必要がある。以下のプログラムでは a[i]は拡張 for 文では i に置き換えることができるが、下線を付けた i はインデックスの値であるため、拡張 for 文に対応できない。

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

演習

プログラム例１～５を作成し、実行しなさい。

課題４

kadai4_1　キーボードから複数の整数値を入力した後、最大値、最小値、平均値を求めて、出力するプログラムを作成しなさい。ただし、入力するデータ数もキーボードからあらかじめ入力すること。また、平均値は実数で正しく表示されなくてはならない。

kadai4_2　サイコロを振ったときにそれぞれのサイコロの目に対して次の表のような得点が与えられるゲームを考える。

サイコロの目	1	2	3	4	5	6
得点	30	10	15	2	8	28

このゲームを２回繰り返したときの合計得点の表を、配列と繰り返し処理を利用して求め、表示するプログラムを作成しなさい。出力例を以下に示す。

```
サイコロの得点表
[1 回目][2 回目]  [得点]
[  1  ][  1  ]   [ 60 ]
[  1  ][  2  ]   [ 40 ]
       :
```

kadai4_3　0より大きい10個の異なる数字を入力したときに大きい数から順番に並び替えて表示するプログラムを作成しなさい。この処理はソートと呼ばれ、いろいろな処理方法があるが、プログラムの練習のため、作成方法を以下のように指定する。

方針

・キーボードから入力した10個のデータを配列に格納する。
・入力したデータを格納する配列とは別に結果を格納する配列を用意する。
・並び替えは 2 重ループを利用して、大きい数を順番に見つけることで実現する。
・そのときそのときの最大値を見つけ、見つけた配列の要素に 0 を代入する。このようにすれば、同じものを二度最大値として見つけることはなくなる。

具体的には

・外側ループは10回の繰り返し処理を行い、内側のループでは最大値を見つける処理を行うことになる。内側のループの処理では最大値となる配列の要素の添字を更新する。
・内側のループを抜けた状態で、そのときそのときの最大値が格納されている配列の添字が確定する。この添字を利用して、結果を格納する配列の要素にその値を代入する。その後、入力した値を記憶している配列において、最大値があった配列の要素に 0 を代入する。
・最後に結果を表示する。

配列に関する課題は続きがあるので、提出期日は後で指定する。