Модуль 1. Основы программирования

Тема 1.11. Многомерные массивы. *Неровные массивы

2 часа

Оглавление

Тема 1.11. Многомерные массивы. *Неровные массивы

После знакомства с одномерными/линейными массивами пришло время познакомиться с многомерными. В результате изучения этой темы мы усвоим понятие многомерного массива, научимся обращаться к каждой ячейке с данными отдельно.

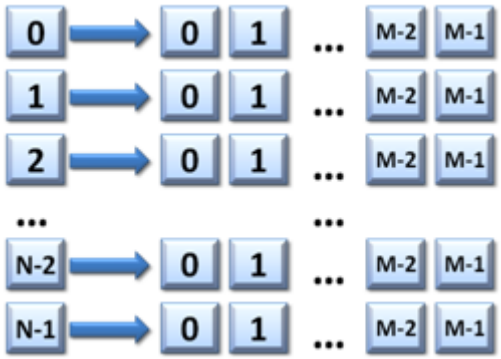
При работе с многомерными массивами надо научиться решать три задачи:

- выделять память нужного размера под массив
- записывать данные в нужную ячейку
- читать данные из ячейки

В математических вычислениях часто используются двумерные массивы. Также можно представить параллелепипедные и как частный случай кубические (трехмерные) массивы. Но на самом деле это представление не совсем верное с точки зрения организации массивов в Java.

Массив может состоять не только из элементов какого-то встроенного типа (int, double, char и пр.), но и, в том числе, из объектов какого-то существующего класса и даже из других массивов.

Массив, который в качестве своих элементов содержит другие массивы называется **многомерным массивом**.



Чаще всего используются двумерные массивы. Такие массивы можно легко представить в виде матрицы. Каждая строка которой является обычным одномерным массивом, а объединение всех строк — двумерным массивом, в каждом элементе которого хранится ссылка на какую-то ячейку матрицы.

Например, на рисунке приведен примере двумерного массива, у которого N строк и каждая строка ссылается на одномерный массив из M элементов. Таким образом, получается задан массив, например **matrix[N][M]**.

Нумерация элементов массива начинается с 0. Поэтому при обращении к последнему элементу массива мы пишем не N, а N-1.

*Например, последний элемент с картинки будет иметь следующие индексы **matrix[N-1][M-1].***

*Если N=7, а M=10, то последний элемент будет **matrix[6][9]**.*

Трёхмерный массив можно представить себе как набор матриц, каждую из которых можно записать, например, на библиотечной карточке. Тогда, чтобы добраться до конкретного числа, сначала нужно указать номер карточки (первый индекс трёхмерного массива), потом указать номер строки (второй индекс массива) и только затем номер элемента в строке (третий индекс).

Соответственно, для того, чтобы обратиться к элементу n - мерного массива, нужно указать n индексов.

1.11.1 Объявление и создание массивов

Для того, чтобы использовать массив, нам необходимо объявить и инициализировать его. Объявление массива можно совместить с его созданием, используя следующий синтаксис:

```
тип_массива[][] имя_массива = new тип_массива [размер_массива]
[размер_массива];
```

Например:

```
char[][] graph = new char[10][10];
```

Объявить массив без создания можно так:

```
тип_массива [][] имя_массива;
```

Например:

```
  char[][] graph = new char[10][10];
  int[] A1; // Обычный, одномерный массив целых
  int[][] A2; // Двумерный массив целых
  double[][][] A3; // Трёхмерный массив вещественных
  int[][][][][] A5; // Пятимерный массив целых
```

И конечно, потом его нужно не забыть инициализировать.

При создании массива можно указать явно размер каждого его уровня:

```
int[][] a2 = new int[5][7]; // Матрица из 5 строк и 7 столбцов
```

1.11.2. Доступ к элементам массива

При создании массива происходит его инициализация, т.е. присваивание начальных значений элементам массива. По умолчанию компилятор инициализирует числовые массивы нулевыми значениями, символьные – нулевым символом (символом с кодом 0), булевские – значением false. Имеется возможность инициализировать элементы массива другими значениями, добавив в оператор создания массива список выражений, заключенный в фигурные скобки:

```
тип_массива[][] имя_массива = new тип_массива[][] {список_выражений};
```

Мы можем создать массив, явно указав его элементы. Например, так:

```
int[][] array = { { 3, 6, 5, 7 }, { 3, 2, 1, 6 }, { 7, 8, 9, 0 } };
```

И результатом будет заполненный массив размером 3 x 4, или точнее **`array[3][4]`.**

После создания массива его размер сохраняется в свойстве length, которое рекомендуется использовать в различных алгоритмах обработки массивов. Обращение к этому свойству имеет синтаксис:

```
имя_массива.length
```

Вывод на экран размера массива:

```
out.println(array.length);
```

А для определения размера строки используется следующее обращение array[i].length.

Элементы массива нумеруются, начиная с нулевого значения, и номер элемента называется его индексом. Таким образом, первому элементу массива соответствует значение индекса 0, второму – значение индекса 1, элементу с порядковым номером k – значение индекса k-1. Для доступа к отдельным элементам массива используется следующий синтаксис:

```
имя_массива[выражение_целого_типа][выражение_целого_типа];
```

Значение выражения в квадратных скобках рассматривается как индекс элемента массива и поэтому должно иметь значение в диапазоне от 0 до length – 1, включительно. Выход за границы этого диапазона приводит к ошибке времени выполнения (ArrayIndexOutOfBoundsException).

Строки матрицы можно легко переставлять:

```
int[][] array = { { 3, 6, 5, 7 }, { 3, 2, 1, 6 }, { 7, 8, 9, 0 } };
int[] temp = array[0];
array[0] = array[1];
array[1] = temp;
```

В этом фрагменте для перестановки используется буферный массив *temp* типа *int[]*.

Перестановка строк в рассмотренном примере происходит очень быстро. Потому что меняются местами только ссылки на области памяти, а сами элементы строк при этом не перемещаются.

1.11.3. Двумерный массив

Чаще всего используются двумерные массивы с равным количеством элементов в каждой строке. Для обработки двумерных массивов используются два вложенных друг в друга цикла с разными счётчиками.

Матрица – это прямоугольная таблица однотипных элементов или другими словами это массив одномерных массивов.

Говоря о квадратной матрице, мы сталкиваемся с понятиями:

- элементы главной диагонали (i=j);
- элементы побочной диагонали (i+j=n-1, n - количество строк или столбцов квадратной матрицы);
- элементы ниже главной диагонали (i>j);
- элементы выше главной диагонали (i<j);
- элементы ниже побочной диагонали (i+j>n-1);
- элементы выше побочной диагонали (i+j<n-1).

15	2	3	7	-9
6	16	2	-8	4
-5	0	3	15	23
9	-7	2	-1	17
20	-6	-4	0	5

Элементы главной диагонали (15, 16, 3, -1, 5).

Элементы побочной диагонали (-9, -8, 3, -7, 20).

Всю работу с массивами можно разделить на несколько блоков:

Упражнение 1.11.1

В качестве примеров обращения к отдельным элементам двумерного массива рассмотрим циклы заполнения массива случайными числами с последующим выводом полученных значений на экран. Для этого нам потребуется функция, генерирующая последовательность случайных чисел. Такой функцией является метод random() класса Math.

Метод random() позволяет получить последовательность случайных значений типа double из диапазона [0, 1), которые затем можно масштабировать, умножая на соответствующий коэффициент, и преобразовать к целому типу.

Заполним двумерный массив случайными числами от 0 до 9 и выведем его на экран в виде матрицы.

Создание и заполнение двумерного массива случайными числами.

```
int[][] array = new int[6][4];/*объявление массива */

 /* Блок ввода массива  */
for(int i = 0; i < array.length; i++) {
   for(int j = 0; j < array[i].length; j++) {
     array[i][j] = (int)(Math.random() * 10);
   }
 }
```

Обработка массива, замена всех четных элементов на “0”, а нечетных на “1”

```
/* Блок обработки массива */
for (int i = 0; i < array.length; i++) {
  for (int j = 0; j < array[i].length; j++) {
     if (array[i][j] % 2 == 0)
           array[i][j] = 0;
     else
           array[i][j] = 1;
   }
}
```

Вывод массива на экран в виде таблицы

```
  /* Блок вывода массива */
  for (int i = 0; i < array.length; i++) {
     for (int j = 0; j < array[i].length; j++) {
        System.out.print(array[i][j] + "\t");
     }
     System.out.println(); // Переходим на следующую строку
  }
```

Соберем блоки в единую программу и посмотрим что получится.

```
/*Основная программа */
import java.io.PrintStream;
import java.util.Scanner;

public class Array {
   public static Scanner in = new Scanner(System.in);
   public static PrintStream out = System.out;

   public static void main(String[] args) {
       out.println("Массив, заполненный случайными числами");
       // Объявление массива
       int[][] array = new int[6][4];
       // Заполнение элементов массива случайными числами
       /* Блок ввода */

       for (int i = 0; i < array.length; i++) {
       for (int j = 0; j < array[i].length; j++) {
              array[i][j] = (int) (Math.random() * 10);
       }
       }
       /* Блок вывода заданного массива */
       for (int i = 0; i < array.length; i++) {
       for (int j = 0; j < array[i].length; j++) {
              out.print(array[i][j] + "\t");
       }
       out.println(); // Переходим на следующую строку
       }

       /* Блок обработки */
       for (int i = 0; i < array.length; i++) {
       for (int j = 0; j < array[i].length; j++) {
              if (array[i][j] % 2 == 0)
              array[i][j] = 0;
              else
              array[i][j] = 1;
       }
       }
       out.println("Измененный массив ");
       /* Блок вывода изменённого массива */
       for (int i = 0; i < array.length; i++) {
       for (int j = 0; j < array[i].length; j++) {
              out.print(array[i][j] + "\t");
       }
       out.println(); // Переходим на следующую строку
       }
```

```
    }
}
```

Пример вывода программы:

```
Массив, заполненный случайными числами
2     8     2     4
0     4     7     1
5     3     0     3
0     7     0     3
2     3     1     4
1     1     4     6
Измененный массив
0     0     0     0
0     0     1     1
1     1     0     1
0     1     0     1
0     1     1     0
1     1     0     0
```

Упражнение 1.11.2

Программа, демонстрирующая алгоритм нахождения минимального и максимального значения в двумерном массиве.

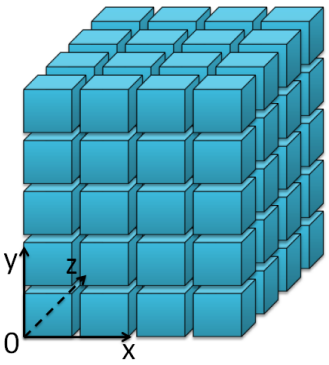
```
import java.io.PrintStream;
import java.util.Scanner;

public class Array {
  public static Scanner in = new Scanner(System.in);
  public static PrintStream out = System.out;

  public static void main(String[] args) {
       out.println("Массив, заполненный случайными числами");
    // Объявление массива
    int[][] matrix = new int[6][6];
    // блок ввода
       for (int i = 0; i < matrix.length; i++) {
       for (int j = 0; j < matrix[i].length; j++) {
       matrix[i][j] = (int) (Math.random() * 50);
       out.print(matrix[i][j] + "\t");
       }
      out.println(); // Переходим на следующую строку
    }
    // Поиск минимального и максимального значений
    int imin = 0, jmin = 0, imax = 0, jmax = 0;
    for (int i = 0; i < matrix.length; i++) {
       for (int j = 0; j < matrix[i].length; j++) {
       if (matrix[i][j] < matrix[imin][jmin]) {
       imin = i;// запоминаем индекс минимального элемента
       jmin = j;
       } else if (matrix[i][j] > matrix[imax][jmax]) {
       imax = i;// запоминаем индекс максимального элемента
       jmax = j;
         }
```

```
      }
    }
    out.println("Минимальное значение " + matrix[imin][jmin]
        + " у элемента с индексами array[" + (imin + 1) + "]["
        + (jmin + 1) + "]");
    out.println("Максимальное значение " + matrix[imax][jmax]
        + " у элемента с индексами array[" + (imax + 1) + "]["
        + (jmax + 1) + "]");
  }
}
```

Результат работы программы:

```
Массив, заполненный случайными числами
4     47    9     2     39    19
28    23    23    36    12    43
16    37    32    6     10    32
31    16    6     41    47    6
44    24    9     31    28    44
5     37    33    45    27    12
Минимальное значение 2 у элемента с индексами matrix[1][4]
Максимальное значение 47 у элемента с индексами matrix[1][2]
```

1.11.4. Трехмерный массив



Четырехмерные и пятимерные массивы мы рассматривать не будем, так как они сложны для восприятия. Обработка данных массивов сводится все к тем же вложенным циклам. Для каждого уровня массива свой цикл обработки. Сложность обработки увеличивается с размером массива, например, для одномерного массива - N операций, где N количество элементов в массиве. Для двумерного массива, N*M, для трехмерного соответственно N*M*K, для четырехмерного соответственно N*M*K*L и так далее для каждого нового уровня массива.

Остановимся на трехмерном массиве. Трехмерный массив - это массив двумерных массивов.

Работа с трехмерным массивом, похожа на работу с двумерным массивом, только используется не два, а три цикла для работы с индексами элемента массива .

```
import java.io.PrintStream;
import java.util.Scanner;

public class Array {
	public static Scanner in = new Scanner(System.in);
	public static PrintStream out = System.out;

	public static void main(String args[]) {
		int[][][] n = { { { 1, 2 }, { 5, 6 }, { 2, 8, 9, 4, 5 } }, { { 3 } },
				{ {} } };
		// Вывод элементов трехмерного массива

		out.println("n.length=" + n.length);
```

```
            for (int x = 0; x < n.length; x++) {
                  for (int y = 0; y < n[x].length; y++) {
                        for (int z = 0; z < n[x][y].length; z++) {
                              out.println("n[" + x + "][" + y + "][" + z + "]="
                                          + n[x][y][z]);
                        }
                  }
            }
      }
}
```

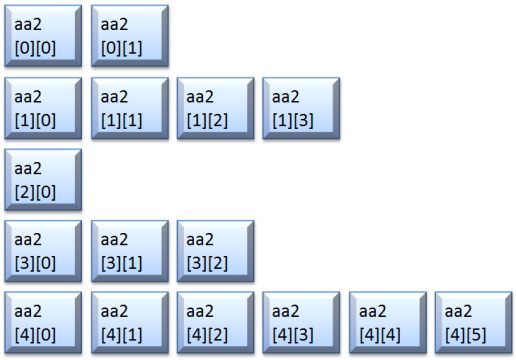
Вывод программы:

```
n.length=3
n[0][0][0]=1
n[0][0][1]=2
n[0][1][0]=5
n[0][1][1]=6
n[0][2][0]=2
n[0][2][1]=8
n[0][2][2]=9
n[0][2][3]=4
n[0][2][4]=5
n[1][0][0]=3
```

*1.11.5. "Неровные" массивы

При создании массива можно указать явно размер каждого его уровня, но можно указать только размер первого уровня:

```
/*
 * Матрица из 5 строк. Сколько элементов будет в каждой строке пока не
 * известно.
*/
int[][] aa2 = new int[5][];
```



В этом случае, можно создать двумерный массив, который не будет являться матрицей из-за того, что в каждой его строке будет разное количество элементов.

Например, случайное задание количества элементов в строке, выглядит следующим образом :

```
  for (int i = 0; i < 5; i++) {
       int k = (int) (Math.random() * 6) + 1;// генерация количества
                                             // столбцов(элементов) в строке
       aa2[i] = new int[k];
  }
```

После заполнения такого массива случайными числами:

```
 for (int i = 0; i < aa2.length; i++) {
       for (int j = 0; j < aa2[i].length; j++) {
           aa2[i][j] = (int) (Math.random() * 10);
       }
  }
```

Можно получить следующую реализацию неровного массива:

```
0     4
0     8     7     4
1
5     8     4
4     8     7     1     2     3
```

Мы могли создать массив, явно указав его элементы. Например, так:

```
int[][] matrix2 = { { 6, 5 }, { 3, 2, 1, 6, 7 }, { 7, 8, 9 } };
```

При этом можно обратиться к элементу с индексом 4 во второй строке **matrix2[1][4]**, но если мы обратимся к элементу **matrix2[0][4]** или **matrix2[2][4]** — произойдёт ошибка, поскольку таких элементов просто нет. При этом ошибка будет происходить уже во время исполнения программы (т.е. компилятор её не увидит).

Получается, что в Java можно создавать "неровные" массивы, имеющие строки разной длины.

Упражнение 1.11.3

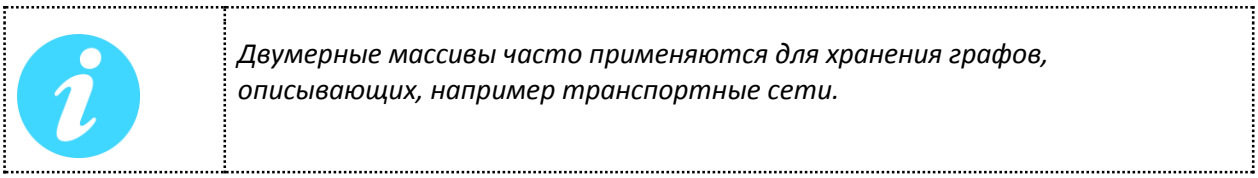
Следующая программа создает массив, каждый элемент которого равен количеству возможностей "выбрать" j чисел из i:

```
import java.io.PrintStream;
import java.util.Scanner;

public class Array {
       public static Scanner in = new Scanner(System.in);
       public static PrintStream out = System.out;

       public static void main(String[] args) {
              final int NMAX = 10;
              // Выделение памяти для треугольной матрицы
              int[][] odds = new int[NMAX + 1][];

              for (int n = 0; n <= NMAX; n++) {
                     odds[n] = new int[n + 1];
              }
```

```
            // Заполнение треугольной матрицы
            for (int n = 0; n < odds.length; n++) {
                  for (int k = 0; k < odds[n].length; k++) {
                        /*
                         * Вычисление выражения
                         *    n*(n-1)*(n-2)* ... *(n-k+1)
                         *  ---------------------------------
                         *    1 * 2 * 3 * ... * k
                         */
                        int digit = 1;
                        for (int i = 1; i <= k; i++) {
                              digit = digit * (n - i + 1) / i;
                        }
                        odds[n][k] = digit;
                  }
            }
            // Вывод треугольной матрицы
            for (int[] row : odds) {
                  for (int odd : row) {
                        out.printf("%4d", odd);
                  }
                  out.println();
            }
      }
}
```

Результат:

```
      0   1   2   3   4   5   6   7   8   9   10
  |-----------------------------------------------
0 | 1
1 | 1   1
2 | 1   2   1
3 | 1   3   3   1
4 | 1   4   6   4   1
5 | 1   5  10  10   5   1
6 | 1   6  15  20  15   6   1
7 | 1   7  21  35  35  21   7   1
8 | 1   8  28  56  70  56  28   8   1
9 | 1   9  36  84 126 126  84  36   9   1
10| 1  10  45 120 210 252 210 120  45  10   1
```



Двумерные массивы часто применяются для хранения графов, описывающих, например транспортные сети.

Задания, которые чаще всего встречаются при работе с двумерными массивами.

- перестановка элементов строк/столбцов;
- вычисление суммы элементов строк/столбцов;
- нахождение самого часто повторяющегося значения среди элементов массива;
- нахождение среднего арифметического и среднего геометрического значений элементов массива строк/столбцов;

- заполнить массив значениями членов арифметической прогрессии, полученными по заданной учителем формуле;
- подсчитать количество положительных и отрицательных значений в массиве;
- подсчитать количество четных и количество нечетных элементов в массиве;
- переместить в начало массива все отрицательные значения;
- сортировка элементов массива (в порядке возрастания или убывания, по последней цифре);
- поиск заданного элемента в массиве.

Задание 1.11.1

Выполните задания на сайте http://informatics.mccme.ru/

Задачи: №354, №355, №361, №363(Задача из вступительного теста), №365, №1458.

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателю IT ШКОЛЫ SAMSUNG Муль Павлу Фридриховичу.