Многомерные массивы

После знакомства с одномерными массивами пришло время познакомиться с многомерными. В математических вычислениях часто используются матрицы — двумерные массивы. Также можно представить кубические (трехмерные) массивы. Но на самом деле это представление не совсем верное с точки зрения организации массивов в Java. Для начала познакомимся с формой записи двумерного массива а потом уже более глубоко покопаемся в том, как на самом деле устроен двумерный массив. Итак, вариант записи достаточно простой:

- Расширенное описание классов
- Исключения
- Решения на основе классов
- Список контактов — начало
- Коллекции — базовые принципы
- Коллекции — продолжение
- Список контактов — GUI приложение
- Что такое JAR-файлы
- Многопоточность — первые шаги
- Многопоточность и синхронизация
- Работаем с XML
- Reflection — основы
- Установка СУБД PostgreSQL
- Базы данных на Java — первые шаги
- Возможности JDBC — второй этап
- JDBC — групповые операции
- Список контактов — работаем с БД
- Переезжаем на Maven
- Потоки ввода-вывода
- Сетевое взаимодействие
- С чего начинается Web

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

Мы объявили двумерный массив символов и создали объект, который можно использовать. Как видите он не сильно отличается от того, что мы видели раньше при знакомстве с одномерными массивами. Прежде чем мы углубимся (как я обещал) в устройство двумерного массива, напишем несложный пример, который заполнит массив символов определенным рисунком, а потом выведет его на экран.
Для начала сделаем простое заполнение массива одним символом — например ‘#’.

```
1 package edu.javacourse.array;
2
3 public class MultiArray
4 {
5     public static void main(String[] args) {
6         // Объявим коснтанту для размера массива
7         int SIZE = 10;
8         // Создаем двумерный массив
```

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

        // Цикл по первой размерности (первые квадратные скобки)
        for (int i = 0; i < SIZE; i++) {
            // Цикл по второй размерности (вторые квадратные скобки)
            for (int j = 0; j < SIZE; j++) {
                graph[i][j] = '#';
            }
        }

        // Теперь выводим массив на экран
        // Цикл по первой размерности выводит строки
        for (int i = 0; i < SIZE; i++) {
            // Цикл по второй размерности выводит колонки - вывод одной строки
            for (int j = 0; j < SIZE; j++) {
                // Используем оператор print - без перехода на следующую строку
                System.out.print(graph[i][j]);
            }
            // Переход на следующую строку
            System.out.println();
        }
    }
}
```

Рассмотрим подробнее два двойных цикла (мы кстати сними уже встречались при рисовании фигур в разделе Управление порядком операций.
Первый двойной цикл позволяет нам с помощью переменных i и j пройти по всем индексам массива graph и каждому его элементы присвоить значение ‘#’. Вот и вся его задача — пройтись по каждому элементу путем обращения к нужному индексу. Никаких облегчающих конструкций в Java пока нет (есть такое понятие как «замыкание», но они ожидаются в Java 8 в середине 2013 года. Вполне допускаю, что я не успею закончить курс до ее выхода и Вы сможете сказать о моих статьях что-то вроде «старье» :)).
Второй цикл предназначен для вывода массива на экран. Эту конструкцию мы тоже уже видели — в нем первый цикл отвечает просто за переход на следующую строку. А внутренний печатает одну строку без каких-либо переходов. Думаю. что Вы сможете разобраться.
Следующий пример заполняет массив символов более сложной фигурой — квадратом с пустым содержанием. Символ ‘#’ встречается здесь только по самому краю. Весь алгоритм заполнения содержится в первом цикле. условие достаточно простое — если индекс является либо первым равным 0) либо последним (равен SIZE-1), то используем символ ‘#’. Иначе — символ пробела ‘ ‘.

```
package edu.javacourse.array;

public class MultiArray
{
    public static void main(String[] args) {
        // Объявим коснтанту для размера массива
        int SIZE = 10;
```

```
        // Создаем двумерный массив
        char[][] graph = new char[SIZE][SIZE];

        // Цикл по первой размерности (первые квадратные скобки)
        for (int i = 0; i < SIZE; i++) {
            // Цикл по второй размерности (вторые квадратные скобки)
            for (int j = 0; j < SIZE; j++) {
                if (i == 0 || i == SIZE - 1 || j == 0 || j == SIZE - 1) {
                    graph[i][j] = '#';
                } else {
                    graph[i][j] = ' ';
                }
            }
        }

        // Теперь выводим массив на экран
        // Цикл по первой размерности выводит строки
        for (int i = 0; i < SIZE; i++) {
            // Цикл по второй размерности выводит колонки - вывод одной строки
            for (int j = 0; j < SIZE; j++) {
                // Используем оператор print - без перехода на следующую строку
                System.out.print(graph[i][j]);
            }
            // Переход на следующую строку
            System.out.println();
        }

    }
}
```

Думаю, что Вы сможете разобраться в этом примере самостоятельно. А после этого примера предлагаю Вам попробовать «нарисовать» фигуры, которые Вы возможно уже пробовали создавать — треугольники, ромбы и прочая. Я же хотел предложить Вам посмотреть на многомерные массивы более глубоко.

Двумерность как массивы массивов

Надеясь на то, что Вы попробовали порисовать фигуры и получили четкое понимание, что сами массивы вещь достаточно несложная — сложны именно алгоритмы, которые используют массивы для своей реализации, предлагаю посмотреть как устроены многомерные массивы.На самом деле двумерный массив не является матрицей — это массив массивов. Т.е. Вы создаете массив, внутри которого находятся указатели на одномерные массивы — массив массивов. Я могу конечно еще раз повторить эти слова, но думаю, что это вряд ли поможет, если Вы еще не поняли эту идею. Попробуйте рассмотреть это на примере кода, где написаны комментарии.

```java
package edu.javacourse.array;

public class MultiArray
{
    public static void main(String[] args) {
        // Объявим константу для размера массива
        int SIZE = 5;
        // Создаем массив, в котором есть другие массивы
        // Причем массивы не создаются - они равны NULL
        char[][] graph = new char[SIZE][];

        // Цикл по элементам массива - все они пока равны NULL
        for (int i = 0; i < graph.length; i++) {
            // Проверяем равенство NULL - это правда
            System.out.println(graph[i] == null);
        }

        for (int i = 0; i < graph.length; i++) {
            // Создаем случайное число от 25 до 75 для указания размера массива
            int size = (int)(Math.round(Math.random()*50) + 25);
            // Теперь создаем массив нужного размера
            graph[i] = new char[size];
        }

        // Цикл по элементам массива - все они теперь проинициализированы
        for (int i = 0; i < graph.length; i++) {
            // Выводим размеры массивов, которые мы создали
            System.out.println(graph[i].length);
        }

    }
}
```

Как видите в начале програмы мы создаем массив массивов — он имеет размер SIZE, но внутри его элементы указывают на null. Первый цикл нам это может продемонстрировать — каждый элемент graph[i]равен null.
Второй цикл создает массивы случайной размерности от 25 до 75 элементов. И вот уже эти элементы и есть символы. Для создания случайного числа используется специализированный класс Math. Метод random() создает случайное число от 0 до 1, которые мы умножаем на 50 (получаем случайное число от 0 до 50) и округляем его с помощью второй функции round. К полученному целому числу прибавляем 25, тем самым сдвигая диапазон на 25 — от 25 до 75. Последним шагом является приведение полученного числа из типа long к типу int — размер массива использует тип int. Вторая строка уже создает массив случайного размера. Третий цикл печатает размеры массивов. Таким образом на самом деле мы создаем не матрицу, а массив массивов.
После недолгих размышлений Вы поймете, что трехмерный массив на самом деле это массив массивов массивов — т.е. массив, который содержит ссылки на массивы, которые уже в свою очередь содержат настоящие элементы (только учтите, что если элементами массивов являются классы, то конечные элементы без инициализации будут тоже ссылки на null).

Упрощенная конструкция инициализации массива

В заключении я хочу показать Вам конструкцию, которая позволяет присвоить элементам массива значения без сложных циклов. Не буду придумывать слова для объяснения — просто приведу пример.

```
package edu.javacourse.array;

public class ArrayInit
{
    public static void main(String[] args) {
        // Создание одномерного массива чисел
        int[] a = {1, 2, 56, 78, 34, 12, 89};

        // Создание двумерного массива символов
        // Обратите внимание на фигурные скобки для выделения
        // массивов внутри массива. Второй массив - пустой
        char[][] graph = {{'1', 'R', 'H', '&', '5', '@'}, {}, {'L', '0', 'I'}};

        // Печать массива символов через конструкцию foraech
        // Проходим по элементам массива массивов - т.е. получаем
        // одномерный массив при каждом цикле
        for(char[] g1 : graph) {
            for(char g2 : g1) {
                // Печатаем строку из массивов
                System.out.print(g2);
            }
            // Переходим на следующую строку
            System.out.println();
        }
    }
}
```

Как видите, все достаточно просто — вы перечисляете внутри фигурных скобок нужные Вам значения. В случае, если нужен многомерный массив, для каждого массива внутри массива тоже надо открыть (и потом закрыть) фигурные скобки. В случае, если нужны объекты, то Вы можете вызывать конструкторы прямо внутри скобок. Например для создания массива из трех роботов можно сделать так:

```
    Robot[] robots = {new Robot(0,0), new Robot(10, 10), new Robot(5,24)};
```

На этом мы можем закончить знакомство с массивами. Я показал Вам вполне достаточно для того, чтобы Вы могли использовать этот инструмент в своих программах. Теперь дело за малым — надо начать писать программы и принимать решение надо ли Вам использовать массив в данной программе или нет. И если Вы сочтете это нужным сделать — небольшой справочник у Вас уже есть. Примеры из данной статьи доступны для скачивания: MultiArray1, MultiArray2, MultiArray3, ArrayInit

Самостоятельно

В качестве самостоятельной работы можете попробовать сделать несколько заданий, которые когда-то я сам делал и мне они понравились. Для их выполнения я хочу предложить вам познакомится с конструкцией, которая позволяет выводить на экран числа в определенном формате. Если вас заинтересует более подробная информация по способам форматирования, можете посмотреть их сами (правда на английском языке) прямо в документации по Java — Formatter

Например, я хочу вывести число 12 и чтобы оно занимало не 2 позиции, а 4.
Т.е. не так

```
12
```

А вот так

```
  12
```

Как видите перед 12 есть еще 2 пробела. Для такого вывода используется конструкция

```
System.out.format("%4d\r\n", 12);
```

Эта конструкция содержит внутри скобок два аргумента, Второй — это наше число 12. А вот первый гораздо интереснее — эта строка говорит о том, как именно я хочу вывести число 12 (а точнее, аргумент, который идет после строки форматирования).
Сначала идет «%4d» — это значит, что я вывожу целое число и это число (даже если оно занимает 2 символа) будет как минимум занимать 4 символа с ведущими пробелами в случае, если число цифр в нем меньше четырех.
Набор «\r\n» означает, что курсор перейдет на следующую строку. Хотя часто достаточно либо \r либо \n — это уже зависит от операционной системы, под которой вы работаете.
Что еще более интересно, вы можете использовать несколько аргументов (Java с 1.5 позволяет это). Мы пока не рассматривали эту возможность. но обязательно упомянем о ней. Для печати нескольких чисел можно написать так

```
System.out.format("%4d и %4d\r\n", 99, 88);
```

Думаю, что вы догадались, что вывод будет вот такой:

```
99 и 88
```

Думаю, что этой информации будет достаточно.

1. В качестве тренировки создайте массив 5 на 5, заполните его числами от 1 до 25 и выведите его на экрана чтобы каждое число занимало 3 позиции. Если разобрались с выводом, то тогда задания посложнее.

2. Заполните массив числами, которые увеличиваются на 1 по спирали. Для примера массив 4 на 4 должен выглядеть вот так

```
  1  2  3  4
 12 13 14  5
 11 16 15  6
 10  9  8  7
```

3. Заполните массив «ходом коня» — надеюсь вы знаете как ходит конь в шахматах (буквой Г). Так вот существует простой алгоритм, который позволяет гарантированно заполнить доску ходом коня размерами от 5 до 70 (квадратную конечно). Т.е. сначала заполните массив числом 0, а потом на первой клетке (элементе массива) ставится число 1, на следующей, на которую прыгает конь — 2 и так до тех пор, пока не останется клеток, на которые конь не ступал. Если остались нулевые значения — значит что-то не так.
Алгоритм достаточно простой — надо прыгать на ту клетку, с которой будет меньше всего ходов.

В следующих статьях мы продолжим изучение конструкций для языка Java.

Один из ответов (на вопрос 2) — Спасибо пользователю **Grif**.

```
/*
 Заполнение по спирали двумерного массива со сторонами A X B любого размера
 при условии, что A = B
 */
public class SpiralArray {

    public static void main(String[] args) {
        // Выбираем размер стороны квадрата и рассчитываем общее количество элементов массива
        int Size = 8;
```

```
        int Max = Size * Size;
        /*
         координаты и рамки заполнения массива
         x[0],y[0] - нижние динамические границы
         x[1],y[1] - верхние динамические границы
         x[2],y[2] - текущие координаты
         x[3],y[3] - вектор смещения
         */
        int RC = 4;
        int[] x = new int[RC];
        int[] y = new int[RC];
        // Объявляем массив для заполнения
        int[][] Sp = new int[Size][Size];
        // Задаём начальные значения координат и рамок
        for (int i = 0; i < RC; i++) {
            x[i] = y[i] = 0;
        }
        x[1] = y[1] = Size - 1;
        x[3] = 1;
        // Заполняем массив
        for (int i = 0; i < Max; i++) {
            //Используем массив как координатное поле
            Sp[x[2]][y[2]] = i + 1;
            // Рассчитываем текущие координаты
            x[2] = x[2] + x[3];
            y[2] = y[2] + y[3];
            // Рассчёт вектора смещения
            if (y[3] == 0) {
                if ((x[2] > x[0]) && (x[3] > 0)) {
                    x[3] = 1;
                }
                if ((x[2] < x[1]) && (x[3] < 0)) {
                    x[3] = -1;
                }
                if ((x[2] == x[1]) && (x[3] > 0)) {
                    x[3] = 0;
                    x[1]--;
                    y[3] = 1;
                }
                if ((x[2] == x[0]) && (x[3] < 0)) {
                    x[3] = 0;
                    x[0]++;
                    y[3] = -1;
                }
            }
            if (x[3] == 0) {
                if ((y[2] > y[0]) && (y[3] > 0)) {
```

```
                    y[3] = 1;
                }
                if ((y[2] < y[1]) && (y[3] < 0)) {
                    y[3] = -1;
                }
                if ((y[2] == y[1]) && (y[3] > 0)) {
                    y[3] = 0;
                    y[1]--;
                    x[3] = -1;
                }
                if ((y[2] == x[0]) && (y[3] < 0)) {
                    y[3] = 0;
                    y[0]++;
                    x[3] = 1;
                }
            }
        }
        // Печать массива заполненного данными по спирали
        System.out.println();
        for (int i = 0; i < Size; i++) {
            for (int j = 0; j < Size; j++) {
                System.out.format("%4d", Sp[j][i]);
            }
            System.out.println();
        }
    }
}
```

И теперь нас ждет следующая статья: Абстрактные классы.

39 comments to *Многомерные массивы*

- 
  Июль 20, 2015 at 14:57
  Grif says:

  Не знаю причин … но часть программы при копировании исчезает …

  Reply

Июль 22, 2015 at 10:29
Grif says:

Отправил в теле письма и на всякий случай в виде файла RTF.

Reply

- 

  Июль 23, 2015 at 06:05
  admin says:

  Опубликовал в статье

  Reply

- 

  Июль 31, 2015 at 13:50
  Grif says:

  Здравствуйте!
  Сейчас в свободное работаю над решением задачи №3 «Ход конём», стараюсь применить принципы объектного программирования и вот столкнулся с затруднением.
  Подскажите пожалуйста почему конструкция типа:

  ```
  public class Horse {
  public static void main(String[] args) {
  int Size = 10;
  int [] Array = new int [Size];
  for (int i = 0; i<Size; i++){Array[i]=0;}}}
  ```

  работает

  а конструкция типа:

```
public class Horse {
private int Size;
private int [] Array = new int [Size];
public Horse (int Size){
this.Size = Size;
for (int i = 0; i<Size; i++){this.Array[i]=0;}}}
```

выдаёт ошибку в "{this.Array[i]=0;}"?

Reply

- Август 3, 2015 at 07:14
  admin says:

  Честно говоря я здесь про ООП не думал — тут на мой взгляд проще делать обычный процедурный вариант. Сделать фнукцию, которая может посчитать количество свободных шагов с любого поля и уже ее использовать для реализации алгоритма. Объекты тут как-то странно выглядят. Разве что для отображения процесса заполнения можно что-то придумать.

  Reply

- Август 4, 2015 at 18:17
  Grif says:

  Я решил задачу №3 “Ход конём” при помощи ООП, у меня получилось четыре класса:
  1-Класс Конь (принимает текущие координаты коня и рассчитывает возможные ходы без ограничений);
  2-Класс Поле (создает поле по заданной ширине и длине на Поле остаются ходы Которые совершил Конь);
  3-Класс Жокей (Управляет конем на поле сопоставляет размеры участка поля (загона) выделенного для коня, стиль езды коня и рассчитывает путь в зависимости от задачи.)
  4-Класс Менеджер (тут и так все ясно).
  Мой конь умеет заполнять не только квадратные поля но и прямоугольные, если поле заполнено не полностью Жокей дает отчет о последней остановке коня.
  Хотелось бы поделиться решением с читателями, но весь код с комментариями занимает около 350 строчек, вряд ли мне удастся выложить его сюда. Если администратор согласится, то я вышлю файлы решения а он прикрепит их к сайту для желающих.

  Reply

Август 5, 2015 at 06:14
admin says:

Администратор согласится 🙂 Высылайте

Reply

Август 5, 2015 at 09:51
Grif says:

Спасибо, выслал на почту 🙂

Reply

Август 5, 2015 at 11:37
Grif says:

Интересно, что немного доработав код, я дал задачу менеджеру проанализировать все массивы размером от 0X0 до 100X100 (квадратных и прямоугольных) и за 1,5 минуты (у меня моноблок на базе нетбучного двухъядерника т.е. слабенький ПК) получил ответ : Jokey — kon vypolnit zadachu dlya 8784 massivov razmerom ot 0X0 do 100X100.

Reply

Август 5, 2015 at 12:09
admin says:

О, уже правильно думаете — «дал задачу менеджеру». Каждый класс занимается своим делом.
Скорость мне сложно оценить. Но при вычислениях Java достаточно неплохо работает.

Reply

Август 5, 2015 at 13:17
Grif says:

О … там было очень много вычислений (обработано 10000 массивов различной величины при этом для каждой ячейки проводился анализ на 8 возможных ходов и для каждого возможного ещё столько же и т.д.) т.е. учитывая, что мой ПК имеет по части вычислений уровень ПК 2004 года, то представленный результат весьма не плох.

Reply

Апрель 5, 2016 at 13:18
Екатерина says:

```
public class MultiArray
{
public static void main(String[] args) {
// Объявим константу для размера массива
int SIZE = 8;
// Создаем двумерный массив
int[][] graph = new int[SIZE][SIZE];
// Объявляем начальное значение элемента в массиве
int n=1;
// Объявляем шаг уменьшения квадрата
int k=1;
for (int i=0; i<SIZE/2;i++)
{
for (int j=i; j<=SIZE-k;j++)
{
graph[i][j]=n;
n++;
if (j==SIZE-k){
for (int m=k; m=k;j—) {
graph[SIZE-k][j]=n;
n++;
```

```
if (j==k)
for(int l=SIZE-k;l>=k;l—) {
graph[l][j-1]=n;
n++;
}
}

k++;
}
// Теперь выводим массив на экран
for (int i = 0; i < SIZE; i++) {
for (int j = 0; j < SIZE; j++) {
System.out.format("%3d",graph[i][j]);
}
System.out.println();
}

}
}
```

Reply

Апрель 5, 2016 at 14:23
admin says:

У меня не получилось это запустить. Лучше код оборачивать тэгами
<pre>
код здесь
</pre>

Reply

Июнь 5, 2016 at 14:35
Nibbler says:

Решил задачу «ход конем» — здесь не стал выкладывать, чтобы не загромождать комментарии. Отправил решение на почту. Буду очень благодарен за критическую оценку. У меня получилось 3 класса: 1) Основной: GeHorse.java; 2) Конь: Horse.java; 3) Шахматная доска: ChessDesk.java
Доска определяет естественные границы возможных перемещений коня и запоминает «карту сделанных ходов» коня.
Конь обладает собственной «конской логикой» и принимает решение, куда ему дальше идти, исходя из своих функциональных возможностей (только буквой «Г»), а также, руководствуясь принципом поиска клетки, откуда было бы минимальное количество ходов и ограничениями и состоянием доски.
Сначала поставил себе цель решить задачу с помощью обычного процедурного подхода, но сами-собой напрашивались методы и я, также, пошел по стопам Grif. Конструктор доски позволяет создавать доску любой размерности, однако, форматирование вывода сейчас настроено под доски размерности не больше 30 (900 клеток). Коня можно устанавливать (при его создании) на любую из клеток доски — задача будет решена в любом случае. Метод «минимума ходов», как выяснилось работает только для досок четной размерности (8, 10, 12 и т.п.). Для досок нечетной размерности задача решения не имеет — остаются незанятые клетки.

Reply

- 
  Июнь 6, 2016 at 15:13
  admin says:

  Я получил письмо — постараюсь посмотреть. Удачи.

  Reply

- Июнь 30, 2016 at 00:17
  Firefly says:

  Скажите пожалуйста, для форматированного вывода ведь еще подойдет System.out.printf()?
  А в чем существенное отличие этих команд?

  Reply

  - 
    Июнь 30, 2016 at 10:40
    admin says:

Для форматированного вывода он и предназначен.
Что касается отличия — о каких командах идет речь ?

Reply

- 
  Июнь 30, 2016 at 23:04
  Firefly says:

  System.out.printf() и System.out.format()

  Reply

  - 
    Июль 3, 2016 at 18:37
    admin says:

    Возможность передавать неопределенное число параметров появилась только в Java 1.5. До этого времени использование printf было по сути невозможно. Вот и не было его — было println — не самая удачная поделка.

    Reply

- 
  Июль 17, 2016 at 05:28
  Oleh says:

```
1 public class TestSpiral
2 {
3     public static void main(String[] args) {
4     //Spiral s=new Spiral(5);
5     //s.paintsp();
6     Spiral.paintsp(5);
7     }
8 }
9
```

```
10
11 class Spiral {
12 //private int size;
13 public Spiral (){}
14 //{
15 //this.size=s;
16 //}
17 public static void paintsp(int size)
18 {
19 int[][]sp=new int[size][size];
20 int znach=1;
21 int kolhodov = size*2-1;
22 int ri=0;
23 int rj=0;
24 int si=size;
25 int sj=size;
26 int mi=0;
27 int mj=0;
28 int x=0;
29 int vpravo=1;
30 int vlevo=0;
31 int vniz=0;
32 int vverh=0;
33 for (;kolhodov>0;kolhodov--)
34 {
35   if (vpravo==1){
36    for(int j=rj;j<sj;j++)
37    {
38     sp[ri][j]=znach;
39     znach++;
40     x=j;
41    }
42   vniz=1;
43   vpravo=0;
44   ri=ri+1;
45   rj=x;
46    sj--;
47   }
48   if (vniz==1){
49      for(int i=ri;i=mj;j--){
50            sp[ri][j]=znach;
51       znach++;
52 x=j;
53      }
54
55       vverh=1;
56       vlevo=0;
```

```
       ri=ri-1;
       rj=x;
       mj++;

       }

      if (vverh==1){

      for(int i=ri;i>mi;i--){
      sp[i][rj]=znach;
      znach++;
x=i;
      }
       vpravo=1;
       vverh=0;
       rj=rj+1;
       ri=x;
       mi++;
      }
      }
for(int i=0;i<5;i++){
    for (int j=0;j<5;j++){
     System.out.printf ("%4d",sp[i][j]);}
   System.out.printf ("\n\n");
}

}

}
```

Reply

- Сентябрь 28, 2016 at 15:29
Creed says:

```
package my.array;

public class Spiral {
private int size;
private int arr[][];
```

```
public Spiral(int size) {
this.size = size;
arr = new int[size][size];
}

public void CreateSpiral (){
int num=1;
int left=0;
int right = size-1;
int top=0;
int bottom = size-1;

while ( num <= size*size){
for (int i = left; i<=right;i++)
arr[top][i] = num++;
for (int j = top+1; j=left;i--)
arr[bottom][i] = num++;
for (int j =bottom-1;j>top;j--)
arr[j][left] = num++;
left++;
right--;
top++;
bottom--;
}
}

public void PrintSpiral (){
for (int i = 0; i < size; i++) {
for (int j = 0; j < size; j++)
System.out.format("%4d", arr[i][j]);
System.out.println();
}
}
}
```

Reply

- Октябрь 5, 2016 at 09:16
Денис says:

Заполняет по спирали массивы любой размеренности (кроме одномерных)

```java
package simpleStructures.spiralMatrix;

public class SpiralApplication {

    protected static final int SIZE_X = 8;
    protected static final int SIZE_Y = 11;

    public static int[][] matrix;
    private static Configuration config;

    public static void main(String[] args) {

        matrix = new int[SIZE_X][SIZE_Y];

        int ost = Math.min(SIZE_X, SIZE_Y) % 2;

        int iterations = (int)Math.min(SIZE_X, SIZE_Y) / 2 + (ost==0 ? 0 : 1); // количество итераций. Т.е. сколь

        config = new Configuration();

        config.setMax(SIZE_X-1, SIZE_Y-1);

        int hops = 4;   // кол-во веток в каждой спирали. Обычно 4, а для массива в котором 2 строки - 3 ветки. Н

        int counter = 0; // счетчик для заполнения

        // начинаем заполнять
        for (int k = 0; k < iterations; k++) {

            // точка старта каждого витка спирали всегда в углу (чем дальше - тем глубже)
            config.setStart(k, k);

            config.setDirection(0, 1); // первый хоп витка всегда заполняем по часовой стрелке. Если против часов

            int startI = config.getStartX();
            int startJ = config.getStartY();

            int maxI = config.getMaxX();
            int maxJ = config.getMaxY();

            // если матрица не квадратная а прямоугольная, может остаться для заполнения одномерный массив
            if (startI == maxI || startJ == maxJ)
                hops = 1;

            // для первой ветки витка текущие координаты совпадают с начальными
            int curI = startI;
            int curJ = startJ;
```

```
48
49              // делаем витки
50              int h=1;
51              counter++;
52
53              do {
54                  // устанавливаем
55                  matrix[curI][curJ] = counter++;
56
57                  // сдвигаем
58                  curI = curI + config.getDirectionX();
59                  curJ = curJ + config.getDirectionY();
60
61                  //проверяем
62                  // если координаты выходят за границы текущего прямоугольника - заканчиваем виток
63                  if (curI  maxI || curJ  maxJ || (curI==startI && curJ==startJ)) {
64                      h++;
65                      counter--; // счетчик надо уменьшать потому, что мы возвращаемся на предыдущую угловую уже за
66                      // вышли за границы, сдвинем координаты обратно
67                      curI = curI - config.getDirectionX();
68                      curJ = curJ - config.getDirectionY();
69                      //continue;
70
71                      // определяем новое направление движения ветки
72                      switch (h) {
73                      case 1:
74                          config.setDirection(0, 1); // двигаемся слева направо
75                          break;
76                      case 2:
77                          config.setDirection(1, 0); // двигаемся сверху вниз
78                          break;
79                      case 3:
80                          config.setDirection(0, -1); // двигаемся справа налево
81                          break;
82                      case 4:
83                          config.setDirection(-1, 0); // двигаемся снизу вверх
84                          break;
85                      default:
86                          break;
87                      }
88                  }
89
90              } while (h<=hops); // конец хопов
91
92              // и передвинем их в новом направлении
93              curI = curI + config.getDirectionX();
94              curJ = curJ + config.getDirectionY();
```

```
95
96              config.setMax(--maxI, --maxJ);
97
98          } // конец итераций
99
100         printArray();
101     }
102
103     private static void printArray() {
104         for (int i = 0; i<SIZE_X; i++) {
105             for (int j=0; j<SIZE_Y; j++) {
106                 System.out.printf("%4d", matrix[i][j]);
107             }
108             System.out.println();
109         }
110
111     }
112 }
```

Reply

Октябрь 5, 2016 at 09:17
Денис says:

код класса Configuration

```
1  package simpleStructures.spiralMatrix;
2
3  public class Configuration {
4
5      // Определяем массив в котором будет храниться
6      protected final static int DIRECTION = 0;    // Определяет направление движения курсора по размеренностям
7                                                   // Три значения -1: движение справа на лево; 0 - не изменяетс
8      protected final static int START = 1;        // Начальная координата движения (cX - по строкам, cY - по ст
9      protected final static int MAX = 2;          // Конечная координата движения (cX - по строкам, cY - по сто
10
11     protected final static int cX = 0;
12     protected final static int cY = 1;
13
14     protected int[][] config = new int[3][2];
```

```
15
16      protected void configuration() {
17
18          config[DIRECTION][cX] = 0;  config[DIRECTION][cY] = 1;
19          config[START][cX] = 0;      config[START][cY] = 0;
20          config[MAX][cX] = 1;        config[MAX][cY] = 1;
21      }
22
23      public void setDirection(int directionX, int directionY) {
24          config[DIRECTION][cX] = directionX;
25          config[DIRECTION][cY] = directionY;
26      }
27
28      public void setStart(int startX, int startY) {
29          config[START][cX] = startX;
30          config[START][cY] = startY;
31      }
32
33      public void setMax(int maxX, int maxY) {
34          config[MAX][cX] = maxX;
35          config[MAX][cY] = maxY;
36      }
37
38
39      public int getDirectionX() {
40          return config[DIRECTION][cX];
41      }
42
43      public int getDirectionY() {
44          return config[DIRECTION][cY];
45      }
46
47      public int getStartX() {
48          return config[START][cX];
49      }
50
51      public int getStartY() {
52          return config[START][cY];
53      }
54
55      public int getMaxX() {
56          return config[MAX][cX];
57      }
58
59      public int getMaxY() {
60          return config[MAX][cY];
61      }
```

```
62 }
```

Reply

- Ноябрь 4, 2016 at 17:11
  Oleg says:

  Что-то вы слишком много всего намудрили. Про массивы в Java на http://proglang.su/java/142 проще написано.

  Reply

  

  - Ноябрь 7, 2016 at 11:17
    admin says:

    Я не мудрил — мне показалось интересно и важно рассказать так, как я это сделал. Но на вкус и цвет….

    Reply

- Ноябрь 8, 2016 at 13:55
  v says:

  Задание1 разными вариантами:

```
1 public class ArrayFormat {
2     public static void main(String[] args) {
3         int[] arr = new int[25];
4         int k = 0;
5         int l = 5;
6         for (int i = 0; i < 25; i++) {
7             arr[i] = i + 1; //начало с единицы
8         }
```

```
        for (int i = 0; i < l; i++) {
            for (int j = 0; j < l; j++) {
                System.out.format("%3d", arr[j+k]);
            }
            System.out.println();
            k += l;
        }
    }
}

public class ArrayFormat {
    public static void main(String[] args) {
        int[] arr = new int[25];
        int k = 0;
        int l = 5;
        for (int i = 0; i < 25; i++) {
            arr[i] = i + 1; //начало с единицы
        }
        for (int i = 0; i < l; i++) {
            for (int j = 0; j < l; j++) {
                if (arr[j + k] == l + k) {
                    System.out.format("%3d\r\n", arr[j + k]);
                } else {
                    System.out.format("%3d", arr[j + k]);
                }
            }
            k += l;
        }
    }
}
```

Reply

- Ноябрь 8, 2016 at 14:16
v says:

Вариант с двумерным массивом

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

```
        int size = 5;
        int step = 1;
        int[][] arr = new int[size][size];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
                arr[i][j] = step++;
            }
        }
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr.length; j++) {
                System.out.format("%3d", arr[i][j]);
            }
            System.out.println();
        }
    }
}
```

Reply

- 

  Март 5, 2017 at 14:59
  Юрий says:

  Тоже сразу пришло в голову такое решение. Но потом засомневался, на самом ли деле на печать выводится двухмерный массив?
  Есть подозрения, что получился массив с числами от 1 до 25, а вторая размерность равна нулю и на печать мы выводим только первый массив, просто разбив его на несколько строчек.

  если очистить вторые скобки ничего не меняется при выдаче
  да и можно в выдачу тупо step++ поставить, тоже ничего не меняется. Выходит в выдаче только одномерный массив присутствует.

```
public class Array6 {
    public static void main(String[] args) {
        int size = 5;
        int step = 1;
        int[][] arr = new int[size][];
        for (int i = 0; i < size; i++) {
            for (int j = 0; j < size; j++) {
      //          arr[i][j] = step++;
            }
        }
```

```
        for (int i = 0; i < arr.length; i++) {
            for (int j = 0; j < arr.length; j++) {
                System.out.format("%3d", step++);
            }
            System.out.println();
        }
    }
}
```

Большая просьба к Админу прокомментировать.

Reply

Март 6, 2017 at 10:58
admin says:

В принципе в статье указывается, что двумерный массив — это одномерный массив одномерных массивов. Что касается приведенного примера — я не очень понимаю, что здесь надо комментировать.
В первом двойном цикле (если убрать комментарий) — это будет падать по NullPointerException — что вполне разумно. Массив массивов создан, а вот подмассивы — нет. Код и будет падать.
Второй двойной цикл просто выводит 25 раз счетчик — что здесь связано с массивом кроме ограничения цикла за счет его размера ?

Reply

Ноябрь 18, 2016 at 15:27
K7gt says:

Вариант решения 2ой задачи.
Вроде работает…

```
public class practice {
    public static void  main(String[]  args){
        int SIZE1=7;
        int SIZE2=7;
        int[][] spiral = new int[SIZE1][SIZE2];
```

```
        int p=1;
        int x=0;
        int y=0;
        while (p<=SIZE1*SIZE2) {
            //vpravo
            for (int i = x,j=y; j < SIZE2; j++) {
                if((i==x&&j==y)&&p!=1){
                    continue;
                }
                else if(spiral[i][j]==0){
                    spiral[i][j] = p;
                    p++;
                    x=i;
                    y=j;
                }else{
                    break;
                }
            }
            //vniz
            for (int i = x,j=y; i = 0; j--) {
                if(i==x&&j==y){
                    continue;
                }
                else if(spiral[i][j]==0){
                    spiral[i][j] = p;
                    p++;
                    x=i;
                    y=j;
                }else{
                    break;
                }
            }
            //vverh
            for (int i = x,j=y; i >= 0; i--) {
                if(i==x&&j==y){
                    continue;
                }
               else if(spiral[i][j]==0){
                    spiral[i][j] = p;
                    p++;
                    x=i;
                    y=j;
                }
                else {
                    break;
                }
            }
```

```
53
54          }
55          for (int[] g1:spiral){
56              for(int g2:g1){
57                  System.out.format("%4d",g2);
58              }
59              System.out.print("\n");
60          }
61      }
62  }
```

Reply

Февраль 10, 2017 at 23:44
Евгения says:

Еще вариант 2 задачи

```
1  class SpiralTwoArray{
2    public static void fill(int[][] a){
3     int x=0; //первый индекс
4     int y=0; //второй индекс
5     int k=0; //отступ
6     int n=a.length;
7     for (int i=0; i<n*n; i++){
8       a[x][y]=i+1;
9       if(x==k && y<n-1-k)
10        y++; //направо
11      else if(y==n-1-k && xk)
12        y--; //налево
13      else if(y==k && x>k+1)
14        x--; //вверх
15      else{
16        k++; // увеличиваем отступ
17        y++;} // меняем направление
18       }
19      }
20   public static void print(int[][] a){
21     for(int i=0; i<a.length; i++){
22       for(int j=0; j<a[i].length; j++)
```

```
        System.out.printf("%4d", a[i][j]);
        System.out.println();
       }
      }
 public static void main(String[] args){
   int[][] a1=new int[4][4];
   fill(a1);
   print(a1);
   int[][] a2=new int[6][6];
   fill(a2);
   print(a2);
  }
 }
```

Reply

- Март 16, 2017 at 23:15
Grif says:

Понравилось людям решения выкладывать 🙂
Ресурс тоже благодаря стараниям Админа вырос … жаль только порог вхождения в разработчики тоже.

Reply

- Июнь 24, 2017 at 01:48
Алексей says:

public class SimpleFrame {

public static void main(String[] args) {
int size1=6;
int size2=5;
int k=1;
int i=0;
int j=0;

```
int mas[][]=new int[size1][size2];

while(k<size1*size2){

while(j<size2 && mas[i][j]==0){
mas[i][j]=k;
k++;
j++;
}
j—;
i++;
while(i=0 && mas[i][j]==0){
mas[i][j]=k;
k++;
j—;
}
j++;
i—;
while (i>=0 && mas[i][j]==0){
mas[i][j]=k;
k++;
i—;
}
j++;
i++;
}

for (int i1=0; i1<mas.length;i1++){
for(int j1=0; j1<mas[i1].length; j1++){
System.out.format("%4d",mas[i1][j1]);
}
System.out.println();
}

}

}
```

Reply

Июнь 24, 2017 at 02:03
Алексей says:

```
public class SimpleFrame {

    public static void main(String[] args) {
        int size1=5;
        int size2=9;
        int k=1;
        int i=0;
        int j=0;

        int mas[][]=new int[size1][size2];


        while(k<=size1*size2){

            while(j<size2 && mas[i][j]==0){
                mas[i][j]=k;
                k++;
                j++;
            }
            j--;
            i++;
            while(i=0 && mas[i][j]==0){
                mas[i][j]=k;
                k++;
                j--;
            }
            j++;
            i--;
            while (mas[i][j]==0){
                mas[i][j]=k;
                k++;
                i--;
            }
            j++;
            i++;
        }

        for (int i1=0; i1<mas.length;i1++){
```

```
            for(int j1=0; j1<mas[i1].length; j1++){
                System.out.format("%4d",mas[i1][j1]);
            }
            System.out.println();
        }

    }

}
```

Reply

- Июнь 24, 2017 at 02:08
Алексей says:

Странно, куда то один while потерялся при вставке

Reply

- Июнь 24, 2017 at 12:33
Алексей says:

Задача с конем

```
public class SimpleFrame {

    public static void main(String[] args) {

        int size=10;
        int ar[][]=new int[size+4][size+4];
        for (int i1=0; i1<ar.length;i1++){
            for(int j1=0; j1<ar[i1].length; j1++){
                if (i1ar.length-3 || j1ar[i1].length-3)
                    ar[i1][j1]=1;
                else
                    ar[i1][j1]=0;
```

```
                System.out.format("%4d",ar[i1][j1]);
            }
            System.out.println();
        }

        int i3=0;
        int j3=0;
        int min;
        int m;
        int k=1;
        int i=2;
        int j=2;
        while(k<=size*size){
            ar[i][j]=k;
            min=9;
            m=horse(ar,i+2,j+1);
            if (m<min){
                i3=i+2;
                j3=j+1;
                min=m;
            }
            m=horse(ar,i+1,j+2);
            if (m<min){
                i3=i+1;
                j3=j+2;
                min=m;
            }
            m=horse(ar,i-2,j+1);
            if (m<min){
                i3=i-2;
                j3=j+1;
                min=m;
            }
            m=horse(ar,i-1,j+2);
            if (m<min){
                i3=i-1;
                j3=j+2;
                min=m;
            }
            m=horse(ar,i+2,j-1);
            if (m<min){
                i3=i+2;
                j3=j-1;
                min=m;
            }
            m=horse(ar,i+1,j-2);
            if (m<min){
```

```
60                  i3=i+1;
61                  j3=j-2;
62                  min=m;
63              }
64              m=horse(ar,i-2,j-1);
65              if (m<min){
66                  i3=i-2;
67                  j3=j-1;
68                  min=m;
69              }
70              m=horse(ar,i-1,j-2);
71              if (m<min){
72                  i3=i-1;
73                  j3=j-2;
74                  min=m;
75              }
76              i=i3;
77              j=j3;
78              k++;
79          }
80          System.out.println();
81          for (int i1=0; i1<ar.length;i1++){
82              for(int j1=0; j1<ar[i1].length; j1++){
83
84                  System.out.format("%4d",ar[i1][j1]);
85              }
86              System.out.println();
87          }
88      }
89
90      public static int horse (int ar[][],int i, int j){
91          int k=0;
92          if (ar[i][j]!=0)
93              return 9;
94          if (ar[i+2][j+1]==0)
95              k++;
96          if (ar[i+1][j+2]==0)
97              k++;
98          if (ar[i-2][j+1]==0)
99              k++;
100         if (ar[i-1][j+2]==0)
101             k++;
102         if (ar[i+2][j-1]==0)
103             k++;
104         if (ar[i+1][j-2]==0)
105             k++;
106         if (ar[i-2][j-1]==0)
```

```
            k++;
        if (ar[i-1][j-2]==0)
            k++;
        return k;
    }
}
```

Reply

- Август 26, 2017 at 23:22
olejoman says:

Как-то сложно у вас всё

public static int[][] getSpiral() {
int n = 5;
int[][] array = new int[n][n];

int row = 0;
int col = 0;
int dx = 1;
int dy = 0;
int dirChanges = 0;
int visits = n;

for (int i = 0; i < n * n; i++) {
array[row][col] = i + 1;
if (—visits == 0) {
visits = n * (dirChanges % 2) + n * ((dirChanges + 1) % 2) — (dirChanges / 2 — 1) — 2;
int temp = dx;
dx = -dy;
dy = temp;
dirChanges++;
}
col += dx;
row += dy;
}

```
for (int i = 0; i < n; i++) {
for (int j = 0; j < n; j++)
System.out.print(array[i][j] + "\t");
System.out.println();
}

return array;
}
```

Reply

- Февраль 20, 2018 at 20:07
Артем says:

```
package studyjava;
public class Spiral
{
public static void main(String[] args)
{
int size=10;
int [][] a= new int [size][size];
int count=1;
int top=0;
int right=size-1;
int niz=size-2;
int left=0;
while (count<=size*size)
{
for (int i=top; i<right+1; i++)
{
a[top][i]=count;
count++;
}

for (int i=top+1; i=left; i—)
{
a[right][i]=count;
```

```
count++;
}
for (int i=niz;i>top; i—)
{
a[i][top]=count;
count++;
}
top++;
right—;
niz—;
left++;

}

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

Reply

Июль 9, 2018 at 10:30
rav says:

Хороший материал, по многомерным массивам прочел еще

Reply

Leave a reply

Comment

You may use these HTML tags and attributes: <a href="" title=""> <abbr title=""> <acronym title=""> <b> <blockquote cite=""> <cite> <code class="" title="" data-url=""> <del datetime=""> <i> <q cite=""> <s> <strike> <pre class="" title="" data-url="">

Имя *

E-mail *

Сайт

шесть + = девять

Add comment