Масиви

Масив - це впорядкований набір однотипних елементів, на які посилаються по спільному імені. Це доволі зручний засіб групування інформації. Масиви можна створювати з елементів будь-якого типу. До конкретного елементу в масиві звертаються по індексу (номеру). Вони можуть бути як одновимірні так і багатовимірні.

Одновимірні масиви

Одновимірні масиви - це список однотипних елементів. Загальний формат оголошення такого масиву:

тип-елементів назва-масиву[];

Наприклад:

int month_days[]; *// масив цілих чисел*

Існує також інша форма оголошення масиву:

```
int[] month_days;
```

Проте для того, щоб масив почав існувати необхідно виділити під нього пам'ять, за допомогою операції *new*.

назва-масиву = new тип-елементів [розмір];

де *розмір* - планована кількість елементів у масиві.

```
month_days = new int[12];
```

або зразу ж:

```
int month_days[] = new int[12];
```

Таким чином відбувається виділення пам'яті під масив і ініціалізації елементів масиву нулями.

В подальшому можна напряму звертатися до елементів масиву вказуючи індекс у квадратних дужках.

Нумерація елементів в масиві в java відбувається з нуля. Тобто в наведеному прикладі звернення до першого (нульового) елемента - month_days[0], а до останнього - month_days[11].

Java не дозволить програмі звернутися поза межі масиву, щоправда помилка буде вказана лише на етапі виконання програми через викидання винятку(виключення).

```
month_days[5] = 30;
System.out.println(month_days[5]);
```

Масиви також можна ініціалізувати зразу ж при їхньому оголошенні, не використовуючи операції new, аналогічно як це відбувається при роботі з простими типами даних.

Наступний приклад зразу ж при оголошенні ініціалізує масив month_days[] кількістю днів в місяцях.

```
public class DaysOfMonth { public static void
    main(String[] args) {
     int month_days[] = {31, 28, 31, 30, 31, 30, 31, 31,
    30, 31, 30, 31}; //оголошуємо та ініціалізуємо
    масив
          System.out.println("Травень має " +
    month_days[4] + " день"); // вивід на консоль
}
}
```

Наступний приклад демонструє знаходження максимального числа в одновимірному масиві.

Як бачимо спочатку змінній max присвоюється значення нульового елементу масиву, після чого в циклі іде послідовне порівняння з кожним наступним числом до останнього. Якщо при порівнянні чергове значення в масиві більше за максимальне в змінній max, то змінній max присвоюється дане значення. Як Ви уже зрозуміли, по закінченню циклу у змінній max міститиметься максимальне значення, яке і буде виведене на консоль:

```
public class ArrayMax {
    public static void main(String[] args) {
        double array[] = {1.1, 2.2, 1.1, 3.2, 1.2, 2.1};
        double max = array[0];
        for (int i = 0; i < 6; i++) {
            if (max < array[i])
                max = array[i];
        }
        System.out.println("Максимальне число в масиві: " + max);
    }
}
```

Можна дізнатися довжину масиву таким чином array.length. Для вищенаведеного прикладу можна замінити рядок з циклом таким чином:

```
for (int i =0; i < array.length; i++)
```

Багатовимірні масиви

Багатовимірні масиви по суті – це масив масивів. Робота з багатовимірними масивами подібна до роботи з одновимірними. Відмінність лише в тому, що використовуються додаткові квадратні дужки. Переважно використовуються двовимірні масиви, які служать для роботи з табличними даними та трьохвимірні масиви. Двовимірний масив та трьохвимірний, можна оголосити наступним чином:

```
int twoD[][] = new int [4][5]; //створення масиву 4x5
int threeD[][][] = new int[5][5][5]; //створення масиву 5x5x5
```

Для двовимірного лівий індекс означає номер рядка, а правий номер стовпця. Це можна уявити наступним чином:

[0,0][0,1][0,2][0,3][0,4]
[1,0][1,1][1,2][1,3][1,4]
[2,0][2,1][2,2][2,3][2,4]
[3,0][3,1][3,2][3,3][3,4]

Трьохвимірний масив можна уявити у вигляді куба. Крім номера рядка і номера стовпця, додається ще індекс елемента вглибину.

Наступна програма створює масив 5 на 4, заповнює його випадковими числами і виводить на екран.

```
import java.util.Random;                    // імпортуємо клас Random

public class RandomArray {
    public static void main(String[] args) {
        int m = 5, n = 4;                          //оголошуємо і ініцілізуємо змінні з розмірами масиву
        int Array[][] = new int[m][n];     //оголошуємо і ініціалізуємо масив
        Random generator = new Random();     // створюємо генератор випадкових чисел
        int gn;                  //змінна в яку буде записуватися згенероване генератором число

        /* заповнюємо масив випадковими числами */
        for (int i = 0; i < m; i++)              //проходимось по стовпцях
            for (int j = 0; j < n; j++) {         //проходимось по рядках
                gn = generator.nextInt(100);  //генерація випадкового числа від 0 до 100;
                Array[i][j] = gn;             //записуємо згенероване випадкове число
            }

        /* Виводимо результат */
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < n; j++)          // зверніть увагу на відсутність фігурної дужки
                System.out.print(Array[i][j] + "    ");//даний рядок відноситься до масиву по j
            System.out.println();           //виводимо символи переводу каретки і нового рядка
            //після кожного проходження стовпцевих елементів рядка
        }
    }
}
```

В наведеному прикладі в кожному рядку однакова кількість елементів(стовбців). В Java можна створити двовимірні масиви з різною кількістю елементів в рядках.

```
int twoD[][] = new int[5][]; //створюємо двовимірний масив з 5-ма стовбцями
twoD[0] = new int[5]; // виділяємо пам'ять для 5-ти елементів нульового рядка
twoD[1] = new int[4]; // перший рядок матиме 4-ри елементи
twoD[2] = new int[3]; // другий - 3
twoD[3] = new int[2]; // третій - 2
twoD[4] = new int[1]; // четвертий - 1
```

Використання таких нерівних (нерегулярних) масивів не рекомендується, оскільки з ними важче працювати і можна припуститися ряд помилок, але в деяких ситуаціях можуть бути доволі корисними.

Як і з одновимірними масивами. Ми можемо зразу ж ініціалізувати масив необхідними значеннями при його оголошенні.

Приклад - Array2.java

```
public class Array2 {
    public static void main(String[] args) {

        int[][] Array= {
            {5, 6, 1, 3},
            {3, 4, 2, 1},
            {1, 2, 2, 2}
         };
         for (int i = 0; i < 3; i++){
            for (int j = 0; j < 4; j++)
                 System.out.print (Array[i][j]+"    ");
            System.out.println();
        }
    }
}
```

Результат виконання:

```
5    6    1    3
3    4    2    1
1    2    2    2
```

завдання

1. Створіть масив з усіх парних чисел від 2 до 20 і виведіть елементи масиву на екран спочатку в рядок, відокремлюючи один елемент від іншого пробілом, а потім в стовпчик (відокремлюючи один елемент від іншого початком нового рядка). Перед створенням масиву подумайте, якого він буде розміру.

   2 4 6 ... 18 20
   2
   4
   6
   ...
   20

2. Створіть масив з усіх непарних чисел від 1 до 99, виведіть його на екран у рядок, а потім цей же масив виведіть на екран теж в рядок, але в зворотному порядку (99 97 95 93 ... 7 5 3 1).

3. Створіть масив з 15 випадкових цілих чисел з відрізка [0; 9]. Виведіть масив на екран. Підрахуйте скільки в масиві парних елементів і виведете це кількість на екран на окремому рядку.

4. Створіть масив з 8 випадкових цілих чисел з відрізка [1; 10]. Виведіть масив на екран у рядок. Замініть кожен елемент з непарним індексом на нуль. Знову виведете масив на екран на окремому рядку.

5. Створіть 2 масиву з 5 випадкових цілих чисел з відрізка [0; 5] кожен, виведіть масиви на екран в двох окремих рядках. Порахуйте середнє арифметичне елементів кожного масиву і повідомте, для якого з масивів це значення виявилося більше (або повідомте, що їх середнє арифметичне рівні).

6. Створіть масив з 4 випадкових цілих чисел з відрізка [10; 99], виведіть його на екран у рядок. Визначити і вивести на екран повідомлення про те, чи є масив строго зростаючої послідовністю.

7. Створіть масив з 20-ти перших чисел Фібоначчі і виведіть його на екран. Нагадуємо, що перший і другий члени послідовності рівні одиницям, а кожен наступний - сумою двох попередніх.

8. Створіть масив з 12 випадкових цілих чисел з відрізка [-15; 15]. Визначте який елемент є в цьому масиві максимальним і повідомте індекс його останнього входження в масив.

9. Створіть два масиви з 10 цілих випадкових чисел з відрізка [1; 9] і третій масив з 10 дійсних чисел. Кожен елемент з i-им індексом третього масиву повинен дорівнювати відношенню елемента з першого масиву з i-им індексом до елемента з другого масиву з i-им індексом. Вивести всі три масиву на екран (кожен на окремому рядку), потім вивести кількість цілих елементів в третьому масиві.

10. Створіть масив з 11 випадкових цілих чисел з відрізка [-1; 1], виведіть масив на екран у рядок. Визначте який елемент зустрічається в масиві найчастіше і виведіть про це повідомлення на екран. Якщо два якихось елемента зустрічаються однакову кількість разів, то не виводьте нічого.

11. Користувач повинен вказати з клавіатури парне позитивне число, а програма повинна створити масив зазначеного розміру з випадкових цілих чисел з [-5; 5] і вивести його на екран у рядок. Після цього програма повинна визначити і повідомити користувачеві про те, сума модулів який половини масиву більше: лівої чи правої, або повідомити, що ці суми модулів рівні. Якщо користувач введе невідповідний число, то програма повинна вимагати повторного введення до тих пір, поки не буде вказано коректне значення.

12. Програма повинна створити масив з 12 випадкових цілих чисел з відрізка [-10; 10] таким чином, щоб негативних і позитивних елементів там було порівну і не було нулів. При цьому порядок проходження елементів повинен бути випадковий (т. Е. Не підходить варіант, коли в масиві постійно випадає спочатку 6 позитивних, а потім 6 негативних чисел або ж коли елементи постійно чергуються через один і ін.). Вивести отриманий масив на екран.

13. Користувач вводить з клавіатури натуральне число більше 3, яке зберігається в змінну n. Якщо користувач ввів не підходящий число, то програма повинна просити користувача про необхідність повторного введення. Створити масив з n випадкових цілих чисел з відрізка [0; n] і вивести його на екран. Створити другий масив тільки з парних елементів першого масиву, якщо вони там є, і вивести його на екран.

Сортування масиву

Сортуванням називається такий процес перестановки елементів масиву, коли всі його елементи шикуються по зростанням або за спаданням.

Сортувати можна не тільки числові масиви, а й, наприклад, масиви рядків (за тим же принципом, як розставляють книги на бібліотечних полицях). Взагалі сортувати можна елементи будь-якого безлічі, де задано відношення порядку.

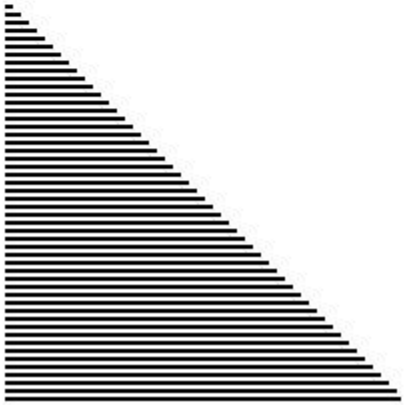
Існують універсальні алгоритми, які виконують сортування незалежно від того, яким було початковий стан масиву. Але крім них існують спеціальні алгоритми, які, наприклад, дуже швидко можуть впорядкувати майже впорядкований масив, але погано справляються з сильно перемішаним масивом (або взагалі не справляються). Спеціальні алгоритми потрібні там, де важлива швидкість і вирішується конкретне завдання, їх докладне вивчення виходить за рамки нашого курсу.

Сортування вибором

Розглянемо приклад сортування по зростанню. Тобто на початковій позиції в масиві повинен стояти мінімальний елемент, на наступній - більший або рівний і т. Д., На останньому місці повинен стояти найбільший елемент.

Суть алгоритму така. У всьому відшукуємо мінімальний елемент, міняємо його місцями з початковим. Потім в решти масиву (т. Е. Серед всіх елементів крім початкового) знову відшукуємо мінімальний елемент, міняємо його місцями вже з другим елементом в масиві. І так далі.

Ілюстрація:



Код:

```
for (int i = 0; i < a.length; i++) {
    /* Предполагаем, что начальный элемент рассматриваемого
     * фрагмента и будет минимальным.
     */
    int min = a[i]; // Предполагаемый минимальный элемент
    int imin = i; // Индекс минимального элемента
    /* Просматриваем оставшийся фрагмент массива и ищем там
     * элемент, меньший предположенного минимума.
     */
    for (int j = i+1; j < a.length; j++) {
        /* Если находим новый минимум, то запоминаем его индекс.
         * И обновляем значение минимума.
         */
        if (a[j] < min) {
            min = a[j];
            imin = j;
        }
    }
    /* Проверяем, нашёлся ли элемент меньше, чем стоит на
     * текущей позиции. Если нашёлся, то меняем элементы местами.
     */
    if (i != imin) {
        int temp = a[i];
        a[i] = a[imin];
        a[imin] = temp;
    }
}
```

Сортування методом бульбашки

Суть алгоритму така. Якщо пройдемося по будь-якому масиву встановивши правильний порядок в кожній парі сусідніх елементів, то після того проходу на останньому місці масиву гарантовано буде стояти потрібний елемент (найбільший для сортування по зростанню або найменший для сортування по спадаючій). Якщо ще раз пройтися по масиву з такими ж перетвореннями, то і на передостанньому місці гарантовано виявиться потрібний елемент. І так далі.

приклад:

2 9 1 4 3 5 2 → порядок правильний, що не буде перестановки

2 9 1 4 3 5 2 → 2 1 9 4 3 5 2

2 1 9 4 3 5 2 → 2 1 4 9 3 5 2

2 1 4 9 3 5 2 → 2 1 4 3 9 5 2

2 1 4 3 9 5 2 → 2 1 4 3 5 9 2

2 1 4 3 5 9 2 → 2 1 4 3 5 2 **9**

Код:

```
/* Внешний цикл постоянно сужает фрагмент массива,
 * который будет рассматриваться, ведь после каждого прохода
 * внутреннего цикла на последнем месте фрагмента будет
 * оказываться нужный элемент (его не надо рассматривать снова).
 */
for (int i = a.length - 1; i >= 2; i--) {
    /* В переменной sorted мы будем хранить признак того,
     * отсортирован ли массив. Перед каждым проходом внутреннего
     * цкла будем предполагать, что отсортирован, но если совершим
     * хоть одну перестановку, то значит ещё не конца отсортирован.
     * Этот приём, упрощающий сортировку, называется критерием Айверсона.
     */
    boolean sorted = true;
    /* Во внутреннем цикле мы проходимся по фрагменту массива, который
     * определяется внешним циклом. В этом фрагменте мы устанавливаем
     * правильный порядок между соседними элементами, так попарно
     * обрабатывая весь фрагмент.
     */
    for (int j = 0; j < i; j++) {
        /* Если порядок соседних элементов не правильный, то их
         * надо обменять местами. И запомнить, что была перестановка.
         */
        if (a[j] > a[j+1]) {
            int temp = a[j];
            a[j] = a[j+1];
            a[j+1] = temp;
            sorted = false;
        }
    }
    /* Если массив отсортирован (т.е. не было ни одной перестановки
     * во внутреннем цикле, значит можно прекращать работу внешнего
     * цикла.
     */
    if(sorted) {
        break;
    }
}
```

Для обробки двовимірних масивів використовуються два вкладених один в одного циклу з різними лічильниками.

Приклад (заповнюємо двовимірний масив випадковими числами від 0 до 9 і виводимо його на жкран у вигляді матриці):

```
int[][] da = new int[6][4];
for(int i=0; i<da.length; i++) {
  for(int j=0; j<da[i].length; j++) {
    da[i][j] = (int)(Math.random()*10);
  }
}
for(int i=0; i<da.length; i++) {
  for(int j=0; j<da[i].length; j++) {
    System.out.print(da[i][j] + "\t");
  }
  System.out.println(); // Переходим на следующую строку
}
```

завдання

1. Створити двовимірний масив з 8 рядків по 5 стовпців в кожній з випадкових цілих чисел з відрізка [10; 99]. Вивести масив на екран.

2. Створити двовимірний масив з 5 рядків по 8 стовпців в кожній з випадкових цілих чисел з відрізка [-99; 99]. Вивести масив на екран. Після на окремому рядку вивести на екран значення максимального елемента цього масиву (його індекс не має значення).

3. Створена двовимірний масив з 7 рядків по 4 шпальти в кожної з випадкових цілих чисел з відрізка [-5; 5]. Вивести масив на екран. Визначити і вивести на екран індекс рядка з найбільшим по модулю твором елементів. Якщо таких рядків декілька, то вивести індекс першої зустрілася з них.

4. Створити двовимірний масив з 6 рядків по 7 стовпців в кожній з випадкових цілих чисел з відрізка [0; 9]. Вивести масив на екран. Перетворити масив таким чином, щоб на першому місці в кожному рядку стояв її найбільший елемент. При цьому змінювати склад масиву не можна, а можна тільки переставляти елементи в рамках одного рядка. Порядок інших елементів рядка не важливий (тобто можна збрешеш тільки одну перестановку, а можна впорядкувати за спаданням кожен рядок). Вивести перетворений масив на екран.

5. Для перевірки залишкових знань учнів після літніх канікул, вчитель молодших класів вирішила починати кожен урок з того, щоб задавати кожному учневі приклад з таблиці множення, але в класі 15 чоловік, а приклади серед них не повинні повторюватися. На допомогу вчителю напишіть програму, яка буде виводити на екран 15 випадкових прикладів з таблиці множення (від 2 * 2 до 9 * 9, тому що завдання по множенню на 1 і на 10 - занадто прості). При цьому серед 15 прикладів не повинно бути повторюваних (приклади 2 * 3 і 3 * 2 і їм подібні пари вважати повторюваними).