Массивы

Массив – это структура, содержащая в себе несколько переменных. Например, на рисунке 1 представлен массив, длинной 10. Обратите внимание, что нумерация ячеек делается от 0 до 9.

Рисунок 1. Массив

Если переменную можно представить, как доску, на которую можно написать значение, можно стереть и написать новое и всегда посмотреть, что написано на ней в любой момент времени, то массив можно представить как ряд таких досок. У каждой доски есть свой порядковый номер.

Массив объявляется также как, обычные переменные, но необходимо добавить квадратные скобки. Тип данных в начале объявления переменной массива определяет тип данных всех элементов массива.

```
int [] arr;
int arr2[];
```

А какое чему в таком случае будет равна переменная arr или arr2?

Чтобы разобраться с этим вопросом, нужно понять, что такое массив. По факту массив – это первый объект класса, который Вы создадите. Т.к. в Java все переменные объектов хранятся как ссылки на них (т.е. адрес памяти, где лежат данные, принадлежащие объекту), то без инициализации переменные классов будут хранить адрес null. Он создан специально для того, чтобы показывать, что указатель ни на что не указывает.

Инициализировать массив можно явным перечислением значений:

```
int [] arr = {1,2,3,4,5,6,7,8,9,10};
```

обратите внимание, что тип данных массива должен совпадать с типом значений, перечисляемых в фигурных скобках через запятую.

Второй способ инициализации массива более сложный для восприятия, но он приближает нас к миру ООП.

Так как любой массив является объектом класса, пока не будем выяснять, какого, то для явного создания нам понадобится ключевое слово new.

```
int arr[] = new int[8];
```

Эта команда создаст массив arr, состоящий из восьми целочисленных элементов.

В квадратных скобках необходимо указать кол-во элементов. Обратите внимание, что при такой инициализации тип данных массива указывается 2 раза.

При такой записи создаётся массив заданной длины, его элементы заполняются значениями по умолчанию. Для численных типов данных – это 0.

Обращаться к элементам массива можно по его номеру, указав его в квадратных скобках.

```
import java.util.Scanner;

public class Main {

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

        int [] arr = {6,7,8,9,10};
        System.out.println(arr[0]+" "+arr[3]);
        arr[3]=0;
        System.out.println(arr[0]+" "+arr[3]);

    }
}
```

Обратите внимание, что нумерация элементов массива начинается с нуля!

В этом случае программа выведет в консоль следующее:

>>6 9

>>6 0

Т.к. массив – объект, то у него есть поля (переменные). Одно из них – это length. Бывают ситуации, что нам заранее неизвестна длина массива, тогда нам это поле очень пригодится.

```
import java.util.Scanner;

public class Main {

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

        int [] arr = {6,7,8,9,10};
        int ln = arr.length;
        System.out.println(ln);

    }
}
```

На консоль будет выведено:
>>5

Попробуем решить следующую задачу: с клавиатуры вводится натуральное число n требуется создать массив длины n и заполнить его элементами от n до 1.

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int n = sc.nextInt();
        int arr[] = new int[n];
        for(int i=0; i<n; i++){
            arr[i] = n-i+1;
        }
    }
}
```

Для чтения всех элементов массива создана специальная конструкция for-each. Дословно можно переместить как «для каждого».
Работает эта конструкция следующим образом: по очереди во временную переменную копируются один за другим элементы массива и обрабатываются. При этом, что самое важное, нам не надо знать длину массива и обращаться к каждому элементу по индексу.
Попробуем при помощи такой конструкции решить задачу: пусть массив уже задан, требуется вывести сумму его элементов.

```
for (int a : arr) {
    sum += a;
}
System.out.println(sum);
```

Чтобы использовать for-each конструкцию, надо в скобках указать тип массива, потом имя временной переменной, потом «:», потом имя массива. Каждое выполнение тела цикла производится с новым значением переменной a, равным соответствующему элементу массива.

Теперь возникает вопрос: а что если уже после инициализации массиву присвоить значение второго массива?

```
int [] arr = {1,2,3,4,5};
int [] arr2 = {6,7,8,9,10};
arr = arr2;
```

После присвоения arr = arr2 никакого копирования элементов, конечно, не произойдёт, копироваться будет только адрес оперативной памяти, где лежат переменные массива. Поэтому после такого присвоения arr и arr2 будут указывать на одну и ту же структуру данных, а та структура, на которую указывал arr автоматически удалится.

```
import java.util.Scanner;

public class Main {

    public static void main(String[] args) {
        int [] arr = {1,2,3,4,5};
        int [] arr2 = {6,7,8,9,10};
        arr = arr2;
        for(int a: arr){
            System.out.print(a+" ");
        }
        arr2[0] = 0;
        System.out.println();
        for(int a: arr){
            System.out.print(a+" ");
        }
    }
}
```

Вывод в консоль будет следующий:

>>6 7 8 9 10

>>0 7 8 9 10