Массивы

Одномерные массивы

Массивы позволят вам создавать не множество отдельных переменных, в каждой из которых хранится одно значение, а одну переменную, в которой можно хранить множество значений.

Массив – тип данных, позволяющий хранить в одной переменной сразу несколько значений. Чтобы отличать эти значения их нумеруют. Номер переменной называют индексом, поэтому иногда массивы называются индексированными переменными. Каждое значение в массиве называется элементом массива.

Для создания переменной массива указывается тип элемента, имя массива и пустые квадратные скобки. Скобки могут стоять как после типа, так и после имени.

String args[]

Данная строка имеется в объявлении main и означает, что входной переменной метода является массив args типа String.

При создании массива следует помнить, что массивы являются особым типом объектов, поэтому приведенная выше запись создает только ссылку на массив, а не сам массив. Сам массив создается с помощью оператора new.

```
int marks[] = new int[20];
```

Мы создали массив **marks** для хранения оценок двадцати учеников, то есть массив, состоящий из 20 элементов. Как уже было сказано, каждый элемент массива имеет собственный номер. **Нумерация начинается с нуля**, и идет по порядку, то есть в данном случае в массиве будут элементы с номерами от 0 до 19.

Если вам заранее известно, какие значения должны находиться в элементах вашего массива, их можно перечислить в фигурных скобках через запятую:

```
int daysInMonth[] = {31,28,31,30,31,30,31,31,30,31,30,31};
```

В данном случае будет создан массив с двенадцатью элементами. Номера элементам будут заданы автоматически начиная с 0 и далее по порядку. То есть с номером 0 будет иметь значение 31 и соответствовать январю, элемент с номером 1 будет иметь значение 28 и соответствовать февралю, элемент с номером 2 будет иметь значение 31 и так далее.

Как работать с его значениями? Чтобы обратиться к значению, которое хранится в массиве, следует использовать не только имя массива, но и индекс нужного вам значения в квадратных скобках:

marks[2] = 4;

В результате такой операции в элемент с индексом 2 (то есть ученику, идущему по списку третьим) заносится оценка 4.

При записи индекса элемента можно писать не только число, но и практически любое выражение, результатом которого будет целое число.

```
int i = 6;
```

```
marks[i + 2] = 10;
```

В результате этих операций в элемент массива индекс 8 попадет значение 10, то есть эту оценку получил девятый по счету ученик.

С элементами массивов можно выполнять любые операции, которые допустимы с обычными переменными.

Например, средняя оценка первого и последнего учеников в списке:

```
int averageMark = (marks[0] + marks[19])/2;
```

Пример: заполнение массива из двадцати элементов, случайными числами:

```
int marks[] = new int[20];
for (int i=0; i < 20; i++) {
        marks[i] = (int) (Math.random()*10 + 1);
}
```

Здесь для формирования самой оценки используется стандартный метод Math.random(), возвращающий случайное число от 0 до 1

Если следует ввести массив другого размера, нужно поменять числа в объявлении массива и в заголовке цикла.

Важной является также операция вывода массива. Когда нам необходимо посмотреть оценки учеников класса, вывод может выглядеть так:

```
for (int i=0; i < 20; i++) {
        System.out.println("Ученик №"+ i + " = " + marks[i]);
}
```

В результате на экране будут выведены ученики с их номерами и оценками, в виде:

УченикN№0 = 6

и т. д.

Часто мы не знаем заранее, какого размера будет наш массив, что может создавать проблемы. Предыдущий пример написан исходя из того, что в классе строго 20 учеников, если их больше, то часть из них не будет распечатана, если меньше – будет выведено сообщение об ошибке. Можно воспользоваться значением поля length массива:

```
System.out.println(marks.length);
```

При выполнении данной строки будет выведено количество элементов в массиве (то есть учеников в нашем классе) в виде числа. С помощью этой переменной мы можем распечатать массив, сколько бы элементов в нем не было:

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

```
        System.out.println("Ученик №"+ i + " = " + marks[i]);
}
```

В Java нумерация элементов массива начинается с 0, поэтому номер последнего элемента массива равен marks.length – 1. Поэтому в заголовке условие содержит знак «меньше», а не «меньше либо равно».

Пример: для отчета необходимо указать наивысшую оценку, полученную учениками, то есть перед нами стоит задача поиска максимального элемента:

```
int maxMark = marks[0];
for (int i=0; i < marks.length; i++) {
        if (maxMark < marks[i]) {
                maxMark = marks[i];
        }
}
System.out.println("максимальная оценка =" + maxMark);
```

Переменная **maxMark** хранит максимальную оценку, обнаруженную в массиве. В самом начале надо присвоить этой переменной начальное значение, которое должно быть больше либо равно максимальному значению массива. Поэтому наиболее надежным действием будет присвоить m значение любого элемента массива. Логично для этого использовать нулевой элемент.

В каждом проходе цикла очередное значение массива, то есть оценка очередного ученика, сравнивается с найденной ранее, и если она оказалась больше, то она становится наибольшей. Так как в цикле перебираются оценки всех учеников, после завершения цикла в переменной maxMark будет наибольшее значение, то есть самая высокая оценка в классе.

Пример: Провести воспитательную работу с учениками, получившими неудовлетворительные оценки (то есть 3 и ниже), для этого необходимо получить список номеров этих учеников:

```
for (int i = 0; i < marks.length; i++) {
        if (marks[i] <= 3){
                System.out.println("Ученик №="+ i + " = " + marks[i]);
        }
}
```

То есть здесь мы решаем задачу распечатки массива, только не всех его элементов, а тех, которые соответствуют определенным условиям.

Цикл foreach

Пример: необходимо вывести на экран массив, но при этом нам не важен порядок вывода и текущая позиция элемента. Для этого используется улучшенный цикл for:

```
for (int element : marks) {
        System.out.println(element);
}
```

Справа от **:** пишем массив, а слева – переменную, в которую при каждой итерации цикла for будет записываться очередной элемент массива.

Сортировка массива методом прямого выбора

Метод заключается в том, что ищется максимальный элемент массива и меняется с первым местами, а затем ищется максимальный элемент среди оставшихся и меняется местами со вторым, и так далее.

```
for (int i = 0; i < marks.length; i++) {
        maxIndex = i;
        for (int j = i; j < marks.length; j++) {
                if (marks[maxIndex] < a[j]) {
                        maxIndex = j;
                }
        }
        int temp = marks[maxIndex];
        marks[maxIndex] = marks[i];
        marks[i] = temp;
}
```

Внешний цикл предназначен для перебора всех элементов массива. Внутренний служит для поиска наибольшего элемента. Поиск должен начинаться каждый раз с нового элемента, поэтому во втором цикле начальным значением является не 0, а i. В отличие от предыдущего примера, где искалось значение наибольшего элемента, здесь ищется номер, поэтому условие в if и присваиваемое значение отличаются.

После того как внутренний цикл заканчивается, выполняется обмен найденного наибольшего элемента с текущим. Для этого применяется метод "трех стаканов".

Многомерные массивы

Язык Java допускает создание многомерных массивов. Для этого следует при объявлении указать несколько пар квадратных скобок.

```
int m[][] = new int[5][5];
```

Данная операция создает квадратный двухмерный массив. Пример заполнения массива единицами:

```
for (int i = 0; i < m.length; i++) {
        for (int j = 0; j < m[i].length; j++) {
                m[i][j] = 1;
        }
}
```

Следует обратить внимание на то, что свойство length в одном случае вызывается от самого массива, а во втором – от его строки. В первом случае оно возвращает количество строк в массиве, а во втором – количество элементов в строке.

Еще одной особенностью массивов в Java является возможность создания многомерных массивов с переменным количеством элементов в строках (для двухмерного случая). Создание такого массива будет иметь вид:

```
int m[][] = new int[5][];
```

После этого каждую строку следует создать отдельно, например, если нам нужен массив треугольной формы, можно использовать цикл:

```
for (int i = 0; i < m.length; i++) {
        m[i] = new int[i+1];
}
```

Результат распечатки массива будет следующим:

```
1
1 1
1 1 1
1 1 1 1
1 1 1 1 1
```