Массивы

Обсуждаемые вопросы

- Определение и характеристики массива
- Принципы работы с массивами
- Объявление (декларация)
- Создание (выделение памяти)
- Инициализация массивов
- Типовые алгоритмы обработки одномерных массивов
- Двухмерные массивы
- Непрямоугольные двухмерные массивы

Определение и характеристики

- **Массив** - группа элементов одного типа, имеющих одно имя и различающихся по номеру элемента внутри массива – индексу
- Массивы в Java являются объектами встроенного класса, => они имеют ряд атрибутов и методов, облегчающих работу с ними и предоставляющих дополнительные возможности
- Характеристики массива:
  - Мерность (количество измерений)
    - Одномерные массивы (векторы)
    - Двухмерные массивы (матрицы)
    - Многомерные массивы
  - Размер (кол-во элементов) каждого измерения

Принципы работы

- Объявление переменной-ссылки на массив (декларация)
- Создание объекта-массива (выделение памяти)
- Инициализация (присвоение начальных значений)
- Обработка (обращение к элементам)

Объявление

- Синтаксис объявления массива

```
Тип[] имя;
```
или
```
Тип имя[];
```

- Например, одномерный массив целых чисел:

```
int[] iData;
```
или
```
int iData[];
```

- Если переменная объявлена, но ещё не инициализирована, выделение памяти под массив не производится
- => Указывать размер массива на этом этапе нельзя
- Создаётся переменная, кот. в будущем будет содержать ссылку на массив создаваемый динамически

Создание (выделение памяти)

- Синтаксис:

```
Имя = new Тип [размер];
```

- Пример – создание массива целых чисел из 10 элем.:

```
iData= new int [10];
```

- В отличие от локальных переменных, элементы массивов примитивных типов инициализируются значениями по умолчанию
  - Числовые элементы – нулями
  - Символьные – значением ‘\0’ (нулевой символ)
  - Логические – значением false
  - Массивы объектов – значениями **null**
- Можно создать массив сразу при его определении:

```
Int n = getSize();
Int []iData= new int [n];
```

Обработка

- Массивы обрабатываются не целиком, а поэлементно
- Доступ к элементу массива осуществляется по его индексу (номеру)
- Как правило, доступ к элементам массива осуществляется в цикле
- Начальный элемент массива в Java имеет номер 0
- Конечный элемент массива из N элементов имеет номер N-1
- Например:

```
boolean[] barr = new boolean[3];
boolean flag = barr[0];
```

flag будет иметь значение **false**

```
int[] arr = new int[3];
Int a = arr[0];
```

a будет иметь значение **0**

Напоминание: локальные переменные, в отличие от элементов массива, не инициализируются по умолчанию. Во избежание ошибок при компиляции они должны быть инициализированы явно.

Инициализация

- При создании переменной-ссылки на массив можно явно произвести его инициализацию, что приведёт к созданию массива, выделению необходимого объёма памяти и размещению в ней заданных значений:

```
int a[] = new int[] {5, 7, 9};
int temper[] = {25,28,31,26,33,30,32,24,30,32};
```

- Массив temper будет состоять из 10-ти элементов и занимать в памяти 40 байт

Примечание: в Java любая инициализация переменных выполняется на этапе выполнения, а не компиляции. Поэтому для инициализации можно использовать не только литеральные константы, но и переменные и значения выражений.

Типовые алгоритмы обработки

- Присвоение начальных значений или генерация значений элементов случайным образом
- Поиск элемента массива и его номера
  - Максимальный
  - Минимальный
  - Заданный
- Обработка значений
  - Вычисление суммы, разности, произведения, среднего арифметического и т.п.
    - безусловное и условное вычисление
- Сортировка элементов массива (упорядочение)
- Перестановка элементов массива

Пример обработки

Подсчёт среднего арифметического температуры

```
int temper[] = {25,28,31,26,33,30,32,24,30,32};

double avg;
int sum = 0;
int n = temper.length;
for (int i = 0; i < n; i++) {
    sum += temper[i];
}

avg = (double)sum / n;
```

Проход по всем элементам

- В Java есть специальная форма цикла for, которая упрощает полный перебор всех элементов массива или коллекции

```
for (Тип Имя_Переменной : Имя_массива) {
    тело цикла;
}
```

- Например:

```
for (int t : temper) {
    sum += t;
}
```

- В некоторых других языках (Perl, PHP, VB и др.) подобный цикл записывается как «for each» («для каждого элемента»)
- Отсутствие счётчика делает применение этого вида цикла ограниченным

Свойство length

- Для прохода по всем элементам массива можно использовать цикл со счётчиком **for**, используя в качестве верхней границы свойство объекта-массива **length**:

```
for (int i = 0; i < temper.length; i++) {
    sum += temper[i];
}
```

- Использование свойства **length** делает программу более универсальной и не зависящей от конкретного значения размера массива
- Использование этого свойства предпочтительно

Двухмерные массивы

- Создание и инициализация двухмерного массива:

```
public class Matr {
public static void main(String[] args) {
 double a[][] = { { 1.0, 9.0, 3.1 },
                  { 0.2, 1.0, 5.8 },
                { 3.7, 0.4, 1.0 } };
 for( int i=0; i < 3; i++ ) {
     String s = "";
     for( int j=0; j < 3; j++ ) {
                 s += ("  "+a[i][j]);
      }
     System.out.println( s );
   }
 }
}
```

Непрямоугольные массивы

- В Java разные измерения одного и того же массива могут иметь разные размеры
- Пример: создание треугольного массива:

```
int[][] arr;
    arr = new int[3][];      // это ошибка: new int[][3]
    arr[0] = new int[1];
    arr[1] = new int[2];
    arr[2] = new int[3];
 for( i = 0; i < 3; i++ )
   for( int j=0; j <= i; j++ ) arr[i][j] = j;

 String s = "";
 for( i = 0; i < 3; i++ ) {
   s = "";
   for(int j = 0; j <= i; j++ ) s += (arr[i][j]+"   ");
   System.out.println( s );
  }
```

Выход за границы массива

- Во время выполнения приложения виртуальная машина Java отслеживает выход за границы массива.
- Если приложение пытается выйти за границу массива, генерируется исключение `java.lang.ArrayIndexOutOfBoundsException`

Копирование массивов

- Если присвоить одной переменной-ссылке на массив другую переменную-ссылку на массив, то будет скопирован только адрес массива:
- int[] a = new int [3]; int[] b = a;
- Если изменить элемент массива b, то это скажеться и на массиве a, т.к. эти переменные-ссылки указывают на один и тот же массив.
- Скопировать значения элементов массива можно в цикле
- Есть системный метод копирования массивов:
- System.arraycopy(a, index1a, b, index1b, count);
- Из a в b копируются count элементов начиная с индекса index1a в массиве a. Они размещаются в массиве b начиная с индекса index1b.

Методы обработки массивов

- Используется класс Arrays из пакета java.util (т.е. нужно импортировать этот пакет: import java.util.*)
- Arrays.fill(mas, znach) – заполняет массив одинаковыми значениями znach
- Arrays.equals(a, b) – сравнивает два массива по элементам. (Сравнивать a == b нельзя, т.к. будут сравниваться адреса массивов, а не значения)
- Arrays.sort(a) – сортирует массив
- И др.

Рассмотрены вопросы:

- Определение и характеристики массива
- Принципы работы с массивами
- Объявление (декларация)
- Создание (выделение памяти)
- Инициализация массивов
- Типовые алгоритмы обработки одномерных массивов
- Двухмерные массивы
- Непрямоугольные двухмерные массивы

Комментарии к заданиям

- Каждое задание состоит из нескольких частей в порядке возрастания их сложности
- Вы можете выполнить несколько заданий по одному пункту, либо все пункты одного задания и т.д. в зависимости от вашего уровня
- Не обязательно делать все задания!

Задания

- В одномерном массиве, состоящем из n вещественных элементов, вычислить:
- 1) сумму отрицательных элементов массива;
- 2) произведение элементов массива, расположенных между максимальным и ми
- нимальным элементами.
- Упорядочить элементы массива по возрастанию.

Задания

- в одномерном массиве, состоящем из п целых элементов, вычислить:
- 1) произведение элементов массива с четными номерами;
- 2) сумму элементов массива, расположенных между первым и последним нуле
- выми элементами.
- Преобразовать массив таким образом, чтобы сначала располагались все положи
- тельные элементы, а потом — все отрицательные (элементы, равные О, считать
- положительными).

Задания

- в одномерном массиве, состоящем из п вещественных элементов, вычислить:
- 1) максимальный элемент массива; Упражнения к части I 1 3 7
- 2) сумму элементов массива, расположенных до последнего положительного эле
- мента.
- Сжать массив, удалив из него все элементы, модуль которых находится в интер
- вале [a,Ь]. Освободившиеся в конце массива элементы заполнить нулями.

Задания

- в одномерном массиве, состоящем из п целых элементов, вычислить:
- 1) минимальный по модулю элемент массива;
- 2) сумму модулей элементов массива, расположенных после первого элемента,
- равного нулю. 1 3 8 Часть I. Структурное программирование
- Преобразовать массив таким образом, чтобы в первой его половине располага
- лись элементы, стоявшие в четных позициях, а во второй половине — элементы,
- стоявшие в нечетных позициях.

Задания

- Дана целочисленная прямоугольная матрица. Определить:
- 1) количество строк, не содержащих ни одного нулевого элемента;
- 2) максимальное из чисел, встречающихся в заданной матрице более одного раза.

Задания

- Дана целочисленная прямоугольная матрица. Определить:
- 1) количество столбцов, содержащих хотя бы один нулевой элемент;
- 2) номер строки, в которой находится самая длинная серия одинаковых элементов.