Массивы в Java

Массивом называется именованное множество переменных одного типа. Каждая переменная в данном массиве называется ***элементом*** массива. Чтобы сослаться на определенный элемент в массиве нужно знать ***имя*** массива в соединении с целым значением, называемым ***индексом***. Индекс указывает на позицию конкретного элемента относительно начала массива. Индексы массивов в Java начинаются с нуля.

Массивы в Java иметь любой тип и иметь несколько измерений.

**Общая форма объявления массива:**

type variableName[]; int monthDays[];

**Определение массива:**

variableName = new type[size]; monthDays = new int[10];

**Объявление можно совместить с определением**:

int monthDays[] = new int[10];

**Объявление можно совместить с инициализацией:**

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

Синтаксис массивов полностью аналогичен С++, за исключением того, что имя массива в С++ - это указатель а не ссылка на объект.

Управляющие операторы в Java

**Операторы выбора**

* **if-else**   
  В обобщенной форме этот оператор записывается следующим образом:   
  if (логическое выражение) оператор1; [ else оператор2;]   
    
  Раздел else необязателен. На месте любого из операторов может сто­ять составной оператор, заключенный в фигурные скобки. Логическое выражение — это любое выражение, возвращающее значение типа boolean.
* **switch**Оператор switch обеспечивает ясный способ переключения между раз­личными частями программного кода в зависимости от значения одной переменной или выражения. Общая форма этого оператора такова:  
  switch ( выражение ) {  
  case значение1:   
  break;  
  case значение2:  
  break;  
  …  
  case значение n:  
  break;  
  default:  
  }

Результатом вычисления выражения может быть значение любого простого типа, при этом каждое из значений, указанных в операторах case, должно быть совместимо по типу с выражением в операторе switch. Все эти значения должны быть уникальными литералами. Если указать в двух операторах case одинаковые значения, транслятор выдаст сообщение об ошибке.

Если же значению выражения не соответствует ни один из операторов case, управление передается коду, распо­ложенному после ключевого слова default. Оператор default необязателен. В случае, когда ни один из операторов case не соответствует значению выражения и в switch отсутствует оператор default, выполнение программы продолжается с оператора, следующего за оператором switch.

Внутри оператора switch break без метки при­водит к передаче управления на код, стоящий после оператора switch. Если break отсутствует, после текущего раздела case будет выполняться следующий. Иногда бывает удобно иметь в операторе switch несколько смежных разделов case, не разделенных оператором break.

Начиная с JDK 7 в качестве выражения можно использовать строку.

**Операторы перехода**

* **break**
  + Прекращает выполнение именованного блока и пере­дает управление оператору, следующему за данным блоком. Для имено­вания блоков в языке Java используются метки:   
    a: {   
    …  
    break a;  
    }  
    Этот вариант использования представляет собой замену goto. Чаще всего можно обойтись без именованных блоков кода;
  + Немедленное завершение цикла. Когда программа встречает оператор break внутри цикла, она прекращает выполнение цикла и управление передается оператору, следующему за циклом;
  + Внутри оператора switch break приводит к передаче управления на код, стоящий после оператора switch.
* **Continue**
  + В некоторых ситуациях возникает потребность досрочно перейти к выполнению следующей итерации, проигнорировав часть операторов тела цикла, еще не выполненных в текущей итерации. Для этой цели предус­мотрен оператор continue.
* **Return**
  + В любом месте программного кода метода можно поста­вить оператор return, который приведет к немедленному завершению ра­боты и передаче управления коду, вызвавшему этот метод.

Циклы

Любой цикл можно разделить на 4 части — *инициализацию, тело, итерацию и условие завершения.* В Java есть такие циклические конструкции: while (с пред-условием), do-while (с пост-условием) и for (с параметрами).

* **while**Этот цикл многократно выполняется до тех пор, пока значение логического выражения равно true. Ниже приведена общая форма оператора while:   
  [ инициализация; ]  
  while ( условие завершения ) {  
  тело;  
  [итерация;] }
* **do-while**  
  Иногда возникает потребность выполнить тело цикла по крайней мере один раз - даже в том случае, когда логическое выражение с самого на­чала принимает значение false. Для таких случаев в Java используется цикли­ческая конструкция do-while. Ее общая форма записи такова:   
  [ инициализация; ] do { тело; [итерация;] } while (условие завершения );
* **for**В этом операторе предусмотрены места для всех четырех частей цикла. Ниже приведена общая форма оператора записи for:  
  for ( инициализация; завершение; итерация ) тело;  
  Любой цикл, записанный с помо­щью оператора for, можно записать в виде цикла while, и наоборот.
* **for-each**Данный тип цикла является усовершенствованным вариантом цикла for. Особенность такого цикла в том, что он применяется к коллекциям объектов, таким как массив. Преимущество данного цикла в том для перебора элементов коллекции не требуется дополнительная инициализация. Java не предлагает специального ключевого слова для этого типа циклов в отличии от C# и C++. Общая форма оператора: for (тип итер\_перем: коллекция) тело;