Циклы

Оператор циклов while

Общий вид оператора while:

```
while (условие) {
        операторы
}
```

Цикл while работает следующим образом: проверяется условие, и если оно верно, выполняется оператор. Затем снова проверяется условие и выполняется оператор, и так будет продолжаться до тех пор, пока условие верно. Если условие стало не верным, цикл завершается.

Пример: следует вычислить сумму чисел от 1 до 200.

```
int sum = 0;
int i = 1;
while (i <= 200) {
        sum += i;
        i++;
}
```

Сначала создаются две переменные: sum – в которой хранится сумма чисел, i – переменная счетчик, позволяющая перебирать числа от 1 до 200. Изначально сумма = 0, на каждом шаге i добавляется к сумме, и после этого увеличивается на 1. Когда i станет больше 200, цикл прекратится, и выполнение программы продолжится.

В качестве условия может использоваться любое логическое выражение, включая условные и логические операторы.

Пример: получить ряд случайных чисел и посчитать среднее арифметическое от них. Ввод чисел прекращается, если очередным числом будет ноль.

```
double averageNum = 0;
double sum = 0;
int n = 0;
int x = (int) (Math.random()*20);
while(x != 0) {
        sum += x;
        n++;
        x = (int) (Math.random()*20);
}
```

```
if (n != 0){
        averageNum = sum / n;
} else {
        averageNum = 0;
}
System.out.println("среднее значение: " + averageNum);
```

Переменная averageNum служит для хранения среднего значения, переменная sum служит для хранения суммы чисел, переменная n служит для хранения количества введенных чисел. Сначала получается первое число в переменную x. Если оно не равно нулю, оно прибавляется к сумме, и увеличивается счетчик n на единицу. Затем получается новое число и цикл повторяется. Если оно равно нулю, цикл прекращается. В конце стоит проверка на случай, если первое же число будет равным нулю, чтобы избежать деления на ноль.

Оператор do while

Существует еще один вариант оператора while это do while. Общий вид:

```
do {
        оператор
} while (условие);
```

Условие здесь проверяется не перед началом прохода цикла, а после его завершения, отсюда следует особенность, что в отличие от while, даже если условие не выполняется изначально, хотя бы один раз тело цикла будет выполнено.

Пример:

```
int s = 50;
int i=0;
do {
        s = s – 4; i++;
} while (i < 4 && i > 0);
System.out.println("s=" + s + " i=" + i);
```

В данном случае мы имеем дело с ситуацией, когда условие в while на момент запуска цикла не выполняется, так как i равно 0. Но поскольку цикл проверяет условие в конце тела, он запускается, в теле значение i меняется и цикл продолжает свою работу.

Оператор цикла for

Предназначен для случаев, когда код выполняется определенное, заданное заранее, число раз. Для отсчета количества повторений используют специальную переменную, называемую переменной-счетчиком или параметром.

Общий вид оператора for:

```
for (инициализация; условие; инкремент) {
        операторы
}
```

Инициализация – присвоение начального значения счетчику;

условие - пока истинно цикл продолжается;

инкремент – изменение счетчика.

В качестве условия в операторе for может использоваться любое логическое выражение, так же как и в операторе while.

Пример:

```
int sum = 0;
for (int i = 1; i <= 200; i++) {
        sum += i;
}
```

for позволяет записать цикл с параметром более компактно, чем while.

Пример: вывести на экран первые десять целых чисел, которые делятся на 3 без остатка.

```
int n = 10;
int k = 1;
for (int i = 1; k <= n; i++) {
        if (i % 3 == 0) {
                System.out.println(k + ":" + i);
                k++;
        }
}
```

В переменной n содержится количество чисел, которые следует вывести. Переменная k служит для подсчета количества выведенных чисел. Переменная i служит для перебора всех целых чисел по порядку. В теле цикла находится условный оператор, проверяющий остаток от деления

очередного i на 3. Если остаток равен нулю, число выводится, и переменная k увеличивается на единицу. Когда переменная k превышает 10, цикл завершается.

Вложенные циклы

В теле цикла может находиться практически любой оператор, в том числе и другой оператор цикла.

Пример: вывести таблицу умножения.

```
for (int i = 1; i < 10; i++) {
        for (int j = 1; j < 10; j++) {
                System.out.print(i * j + " ");
        }
        System.out.println("");
}
```

Сначала начинается цикл, перебирающий строки таблицы. Тело данного цикла содержит вывод строки. В нем запускается второй цикл, создающий элементы таблицы. Каждый раз выводится произведение номера строки и столбца плюс пробел, для отделения чисел друг от друга. Когда этот цикл завершается, выводится перевод строки с помощью println. После этого происходит увеличение номера строки и процесс повторяется. Второй цикл называется вложенным, и полностью выполняется столько раз, сколько проходов содержит первый цикл.

Операторы break и continue

Иногда выполняющийся цикл следует остановить. Для этого существует специальный оператор break. Оператор break прекращает выполнение цикла и передает управление операторам, которые находятся после тела цикла.

Пример: Вывести первое число в диапазоне 10 - 100 делящееся без остатка на 7;

```
int result = 0;
for (int i = 10; i <= 100; i++) {
        if (i % 7 == 0) {
                result = i;
                break;
        }
}
System.out.println(result);
```

Оператор continue пропускает одну текущую итерацию цикла, т.е. весь код после continue выполнен не будет, цикл перейдет к следующей итерации.

Пример: вывести значение выражения y = 1/x, при значениях x от -10 до 10, с шагом 1. Обычный цикл будет выглядеть следующим образом:

```
int y;
for (int x = -10; x <= 10; x++) {
        y = 1 / x;
        System.out.println("x = " + x + " y= " + y + "\n" );
}
```

С данным фрагментом программы имеется только одна проблема, в один из моментов x может быть равен 0. Деление на 0 вызовет ошибку. Пропустить нежелательный проход цикла, можно воспользоваться оператором continue. Данный оператор заставляет пропустить все операторы до конца тела цикла, и начать новый проход.

```
int y;
for (int x = -10; x <= 10; x++) {
        if (x == 0){
                continue;
        }
        y = 1 / x;
        System.out.println("x = " + x + " y = " + y + "\n" );
}
```

В данном случае будут выданы все значения функции, кроме одного нежелательного.