Лекция 5

Операторы организации цикла в JAVA

Циклом в ЯП Java называется некоторый фрагмент программного кода, который повторяется многократно. При этом цикл может состоять даже из одного выражения или вообще быть пустым.

Выделяют 2 типа циклов: Цикл типа ≪пока≫ и цикл типа ≪n раз≫.

Цикл типа «n раз»

Данный вид циклов используется в том случае, когда заранее известно, какое количество повторений потребуется. Обычно цикл типа ≪n раз≫ связан с каким-то счётчиком, который и отмеряет количество повторений.

Общая схема цикла:

```
for (инициализация; условие повторения; итерация) {
//Тело цикла  }
```

Инициализация – создание некоторого счётчика и указание его начального значения. Для одного цикла при необходимости может быть создано сразу несколько счётчиков.

Инициализация выполняется один раз до самого первого шага цикла.

Итерация – некоторое выражение, описывающее, каким образом будет изменятьсясчётчик/счётчики цикла после каждого его шага.

Условия повторения – некоторое логическое выражение, переменная или константа,значение которой будет проверяться перед началом каждого шага цикла (в том числе исамого первого). Если условие истинно, то очередной шаг цикла будет выполняться, иначецикл остановится.

Тело цикла – набор каких-то операций ЯП Java, который и будет повторяться на каждомшаге цикла. Все значения переменных, полученных на текущем шаге, будут переданы в следующий шаг.

Перед первым шагом цикла счётчику присваивается начальное значение (выполняется инициализация). Это происходит лишь однажды.

Перед каждым шагом цикла (но после инициализации) проверяется условие повторения, если оно истинно, то в очередной раз выполняется тело цикла. При этом, тело цикла может не выполниться ни разу, если условие будет ложным в момент первой же проверки.

После завершения каждого шага цикла и перед началом следующего (и, значит, перед проверкой условия повторения) выполняется итерация.

Представленная программа выводит на экран числа от 1 до 100:

```
for (int i = 1; i <= 100; i++) {
System.out.print(i + " ");
}
```

Представленная программа выводит на экран числа от 10 до −10:

```
for (int s = 10; s > -11; s--) {
System.out.print(s + " ");
}
```

Представленная программа выводит на экран нечётные числа от 1 до 33:

```
for (int i = 1; i <= 33; i = i + 2) {
System.out.print(i + " ");
}
```

Представленная программа вычислит сумму элементов фрагмента последовательности 2, 4, 6, 8,… 98, 100. Итак:

```
int sum = 0; // Сюда будем накапливать результат
for (int j = 2; j <= 100; j=j+2) {
sum = sum + j; }
System.out.println(sum);
```

Представленная программа будет возводить число из переменной **a** в натуральную степень из переменной **n**:

```
double a = 2;
int n = 10;
double res = 1; // Сюда будем накапливать результат
for (int i = 1; i <= n; i++) {
res = res * a;
}
System.out.println(res);
```

Представленная программа выведет на экран 10 первых элементов последовательности $2n+2$, где $n=1, 2, 3\ldots$:

```
for (int i = 1; i < 11; i++) {
System.out.print(2*i + 2 + " ");
}
```

Представленная программа выведет на экран 10 первых элементов последовательности $2a_{n-1}+3$, где $a_1=3$:

```
int a = 3;
for (i=1; i<=10;i++) {
System.out.print(a + " ");
a = 2*a + 3;
}
```

В одном цикле можно задавать сразу несколько счётчиков. При этом несколько выражений в итерации и в инициализации разделяются запятыми. Условие повторения можно задавать только одно, но оно может быть выражением, содержащим сразу несколько счётчиков.

Представленная программа выведет на экран 10 первых элементов последовательности $2a_{n-1}-2$, где $a_1=3$:

```
for (int a=3, i=1; i<=10; a=2*a-2, i++) {
System.out.print(a + " ");
}
```

Представленная программа выведет на экран такую последовательность «0 -1 -4 -9 -16 -25»:

```
for (int a=0, b=0; a-b<=10; a++, b--) {
System.out.print(a*b + " "); }
```

Задания для самостоятельной работы: Что выведется на экран в результате работы представленной выше программы.

Пример 1.

```
for (inti=1; i<=10; i++) {
System.out.println («!»);
}
```

Пример 2.

```
for (int j=99; j > 0; j =j-2) {
System.out.println (j + « »);
}
```

Пример 3.

```
for (int a = 5; b = 5; a-b>=0; a--, b++) {
System.out.println (a*b);
}
```

Пример.

```
int n = 6, f = 1;
for (inti = 2; i<= n, i++) {
f = f*i;
}
System.out.println (f);
```

Пример.

```
int n = 24;
for (inti = 1; i<=n; i++) {
if (n%i == 0) {
System.out.println (i+» «);
}
```

Цикл типа «пока».

Данный вид циклов используется в том случае, когда набор некоторых действий нужно повторять до тех пор, пока выполняется определённое условие, при этом заранее можно не знать, сколько раз выполнится цикл.

Общая схема:

```
while (условие) {
// тело цикла
}
```

Условие – некоторое логическая переменная, выражение или константа, истинность которой проверяется перед каждым шагом цикла, включая первый).

Если условие истинно, то выполняется очередной шаг цикла, иначе происходит выход из цикла и выполняется та часть программы, которая расположена после него.

Тело цикла – набор операций, повторяемых на каждом шаге цикла.

Таким образом, цикл типа ≪пока≫ может не выполняться ни разу. Как правило, условие повторения цикла составляется таким образом, чтобы после очередного шага цикла оно всё-таки стало ложным. Если этого не сделать, то получится цикл, повторяющийся бесконечное число раз.

Условие, определяющее будет ли цикл повторятся снова, проверяется перед каждым шагом цикла, в том числе перед самым первым. Говорят, что происходит *предпроверка* условия.

Оператор `while` повторяет указанные действия до тех пор, пока его параметр имеет истинное значение.

Например, такой цикл выполнится 4 раза, а на экран будет выведено «1 2 3 4 »:

```
int i = 1;
while (i < 5) {
    System.out.print(i + " ");
    i++;
}
```

Такой цикл не выполнится ни разу и на экран ничего не выведется:

```
int i = 1;
while (i < 0) {
    System.out.print(i + " ");
    i++;
}
```

Такой цикл будет выполняться бесконечно, а на экран выведется «1 2 3 4 5 6 7 …»:

```
int i = 1;
while (true) {
    System.out.print(i + " ");
    i++;
}
```

Существует разновидность цикла типа ≪пока≫, в которой условие выполнения следующего шага цикла проверяется не перед ним, а после него (с *постпроверкой* условия).

Схема:

```
do {
// тело цикла
} while (условие);
```

Цикл do while, в отличие от цикла while, выполняется по крайней мере один раз.

Такой цикл выполнится 4 раза, а на экран будет выведено «2 3 4 5 »:

```
int i = 1;
do {
    i++;
    System.out.print(i + " ");
} while (i < 5);
```

Такой цикл выполнится 1 раз, а на экран будет выведено «2 »:

```
int i = 1;
do {
    i++;
    System.out.print(i + " ");
```

```
} while (i < 0);
```

Задания для самостоятельной работы: Что выведется на экран в результате работы представленной выше программы.

Пример 1

```
int s = 1;
while (s<11) {
System.out.print(«!»);
s++;
}
```

Пример 2

```
int s = 1;
do {
System.out.print («!»);
s++;
} while (s<10);
```

Пример 3

```
int s = 0;
do {
s++;
System.out.print («!»);
} while (s<11);
```

Пример 4

```
Scanner inp = new Scanner(System.in);
double u = inp.nextDouble();
while (u<=0 || u != Math.Floor(u)) {
u = input.nextDouble();
}
```

Пример 5

```
Scanner inp = new Scanner(System.in);
double u;
do {
u = inp.nextDouble();
} while (!(u>0 &&Math.floor(u) == u));
```

Пример 6

```
Int kol=0;
int n = 1875;
while (n>0) {
n = n/10 ;
kol++ ;
}
System.out.print(kol);
```