Лабораторная работа №6
Управление выполнением программы. Операторы цикла

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

Оператор while

Этот цикл многократно выполняется до тех пор, пока значение логического выражения равно true. Ниже приведена общая форма оператора while:

- простая
while (условие) оператор;
или
while (условие)
{
<группа операторов>
}

- расширенная
[инициализация;]
while (завершение)
{
тело;
[итерация;] }

Инициализация и итерация необязательны. Ниже приведен пример цикла while для вывода десяти чисел.

```
class WhileDemo {
public static void main(String args[]) {
int n = 10;
while (n > 0) {
System.out.println("Число n = " + n);
n--;
}
} }
```

Оператор 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 (завершение);

В следующем примере тело цикла выполняется до первой проверки условия завершения. Это позволяет совместить код итерации с условием завершения:

```
class DoWhile {
public static void main(String args[]) {
int n = 10;
do {
System.out.println("Число n = " + n);
} while (--n > 0);
} }
```

Оператор 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. Если начальные условия таковы, что при входе в цикл условие завершения не выполнено, то операторы тела и итерации не выполняются ни одного раза. В каноническая форме цикла **for** происходит увеличение целого значения счетчика с минимального значения до определенного предела.

```
class ForDemo {
public static void main(String args[]) {
for (int i = 1; i <= 10; i++)
System.out.println("i = " + i);
} }
```

Следующий пример — вариант программы, ведущей обратный отсчет.

```
class ForTick {
public static void main(String args[]) {
for (int n = 10; n > 0; n--)
System.out.println("Число n = " + n);
} }
```

Обратите внимание — переменные можно объявлять внутри раздела инициализации оператора for. Переменная, объявленная внутри оператора for, действует в пределах этого оператора.
А вот — новая версия примера с временами года, в которой используется оператор for.

```
class Months {
static String months[] = {
"January", "February", “March", "April", "May", "June", "July", "August", "September", "October", "November", "December" };
static int monthdays[] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
static String spring = "spring";
static String summer = "summer";
static String autumn = "autumn";
static String winter = "winter";
```

```
static String seasons[] = { winter, winter, spring, spring, spring, summer, summer, summer, autumn, autumn, autumn, winter };
public static void main(String args[]) {
for (int month = 0; month < 12; month++) {
System.out.println(months[month]  + " is a " +
seasons[month] + " month with " + monthdays[month] + " days.");
} } }
```

При выполнении эта программа выводит следующие строки:

```
January is a winter month with 31 days.
February is a winter month with 28 days.
March is a spring month with 31 days.
April is a spring month with 30 days.
May is a spring month with 31 days.
June  is a summer  month with 30 days.
July is a summer  month with 31 days.
August is a summer  month with 31 days.
September is a autumn  month with 30 days.
October is a autumn  month with 31 days.
November is a autumn month with 30 days.
December a winter month with 31 days.
```

Следующий пример — нахождение суммы, произведения и количества элементов массива.

```
  int i;                 //індекс масиву
 float av=0;           //середнє значення ел-тів масиву
 int  s=0;             //сума ел-тів масиву
double  p=1;        //добуток ел-тів масиву
int  nn=0;            //кількість ел-тів масиву

/* заповнення масиву числами */
 int Ar[]={2,-4,1,-8,11,-3};

for(i=0;i<6;i++)
            {
            s=s+Ar[i];
             p=p*Ar[i];
            nn++;
            }
av=(float)s/nn;

System.out.println("Середне арифметичне = "+av);
System.out.println("Добуток = "+p);
```

Иногда возникают ситуации, когда разделы инициализации или итерации цикла for требуют нескольких операторов. Поскольку составной оператор в фигурных скобках в заголовок цикла for вставлять нельзя, Java предоставляет альтернативный путь. Применение запятой (,) для разделения нескольких операторов допускается только внутри круглых скобок оператора for. Ниже приведен тривиальный пример цикла for, в котором в разделах инициализации и итерации стоит несколько операторов.

```
class Comma {
public static void main(String args[]) {
int a, b;
for (a = 1, b = 4; a < b; a++, b--) {
System.out.println("a = " + a);
System.out.println("b = " + b);
}
} }
```

Вывод этой программы показывает, что цикл выполняется всего два раза.

```
a = 1
b = 4
a = 2
b = 3
```

Операторы break и **continue**

В *языке* Java отсутствует оператор goto. Для того, чтобы в некоторых случаях заменять goto, в Java предусмотрен оператор break. Этот оператор сообщает исполняющей среде, что следует прекратить выполнение именованного блока и передать управление оператору, следующему за данным блоком. Для именования блоков в языке Java используются метки. Оператор break при работе с циклами и в операторах switch может использоваться без метки. В таком случае подразумевается выход из текущего блока.

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 в Java предусмотрен оператор **continue**. Ниже приведен пример, в котором оператор continue используется для того, чтобы в каждой строке печатались два числа.

```
class ContinueDemo {
public static void main(String args[]) {
for (int i=0; i < 10; i++) {
System.out.print(i + " ");
if (i % 2 == 0) continue;
System.out.println("");
}
}}
```

Если индекс четный, цикл продолжается без вывода символа новой строки. Результат выполнения этой программы таков:

```
0 1
2 3
4 5
5 7
8 9
```

Для операторов break и continue можно задавать метку, указывающую, в каком из вложенных циклов вы хотите досрочно прекратить выполнение текущей итерации. Для иллюстрации служит программа, использующая оператор continue с меткой для вывода треугольной таблицы умножения для чисел от 0 до 9:

```
class ContinueLabel {
public static void main(String args[]) {
outer: for (int i=0; i < 10; i++) {
for (int j = 0; j < 10; j++) {
if (j > i) {
System.out.println("");
continue outer;
}
System.out.print("  " + (i * j));
}
}
}}
```

Оператор continue в этой программе приводит к завершению внутреннего цикла со счетчиком j и переходу к очередной итерации внешнего цикла со счетчиком i. В процессе работы эта программа выводит сле-дующие строки:

```
0
0 1
0 2 4
0 3 6 9
0 4 8 12 16
0 5 10 15 20 25
0 6 12 18 24 30 36
0 7 14 21 28 35 42 49
0 8 16 24 32 40 48 56 64
0 9 18 27 36 45 54 63 72 81
```

Задание 1.
Вычислить произведение и сумму, используя три оператора цикла

$$X=\prod_{i=1}^{4}(2i+xi^2),$$

$$Y=\sum_{n=3(2)}^{10}\frac{n^3}{n^2+1}$$

Задание 2.
В цикле for целые значения переменных x и y увеличиваются на единицу от нуля до тех значений, пока их сумма не станет больше 100. Вывести значения этих сумм в одну строку через пробел. Использовать два счетчика цикла и проверку выполнять в круглых скобках после слова for.
Задание 3.
В цикле for вводить различные целые числа и выводить их на экран до тех пор, пока не будет введено число 123.
Задание 3.
Найти первые десять членов ряда Фибоначчи (каждое следующее число равно сумме двух предыдущих – 0, 1, 1, 2, 3, 5, 8, 13, …) и вычислить их сумму.