§3. УСЛОВИЯ. ЛОГИЧЕСКИЕ ОПЕРАТОРЫ. ЦИКЛЫ (УПРАВЛЯЮЩИЕ ИНСТРУКЦИИ)

Цель занятия: научиться создавать алгоритмы, основанные на ветвлении и зацикливании возможных вариантов исполнения программы.

Использованная литература:

- Шилдт, Герберт, Java: руководство для начинающих, 7-е изд.: Пер. с англ. — СПб.: ООО “Диалектика”, 2019. — 816 с.
- https://ru.wikipedia.org/wiki/Таблица_истинности

Условные конструкции

Ранее мы рассмотрели простые программы, исполнение которых производилось строго сверху-вниз — **линейно**, и разработали ряд программ для разрешения простых задач.

```
int i = 146_171_015; // Население РФ
long l = 7_902_614_691L; // Население мира
double pi = 3.1415926535897; // число Пи
boolean toNorth = true; // направление движения поезда метро
String str = "Это строка. Здесь можно записывать мысли и не только...";
```

Код 1. Основные типы данных, рассмотренные ранее

Но на практике нам необходимо изменять поведение программы в зависимости от входных данных, поскольку универсальных сценариев для решения задач не существует.

Например, необходимо проверить вводимые пользователем данные на корректность, т. е. обеспечить «**защиту от дураков**». Предположим, надо проверить, что значение переменной a не обращается в нуль:

```
Scanner in = new Scanner(System.in);
int a = in.nextInt();

if (a == 0) {
    System.out.println("Такое значение недопустимо!");
}
```

Код 2. Проверка входного значения на корректность

Представленная конструкция называется **условной конструкцией** и работает следующим образом: если указанное условие истинно, то исполняется код внутри нее. Так, если входное целое число a равно нулю, то будет выведена фраза «Такое значение недопустимо!».

Чтобы указать, что нужно делать в случае неудовлетворения условия, добавляют конструкцию **else**:

```
if (a == 0) {
    System.out.println("Такое значение недопустимо!");
} else {
    System.out.println(8.0 / a);
}
```

Код 3. Условная конструкция if-else

Условные инструкции называют также командами ветвления, поскольку с их помощью выбирается ветвь кода, подлежащая выполнению. Ниже приведена полная форма условной инструкции if.

```
if (у с л о в и е ) инструкция;

else инструкция;
```

Здесь *условие* — это некоторое условное выражение, а после ключевого слова `if` или `else` стоит одиночная инструкция. Предложение `else` не является обязательным. После ключевых слов `if` и `else` могут также стоять блоки инструкций. Ниже приведена общая форма условной инструкции `if`, в которой используются блоки кода.

```
if (у с л о в и е ) {

     последовательность инструкций

} else {

     последовательность инструкций

}
```

Если условное выражение оказывается истинным, то выполняется ветвь **`if`**. В противном случае выполняется ветвь **`else`**, если таковая существует. Выполнение сразу двух ветвей невозможно. Условное выражение, управляющее инструкцией **`if`**, должно давать результат типа **`boolean`**.

Запись условий с помощью логических операторов

Как могут быть записаны условия для условных конструкций? В программирование приходят классические логические операции: НЕ, И, ИЛИ.

В математической логике за истинностное значение (true) принимают 1, а за ложное — 0. Рассмотрим таблицы истинности следующих классических логических функций.

Эквиваленция

a	b	$a \leftrightarrow b$
0	0	1
0	1	0
1	0	0
1	1	1

Отрицание (NOT)

a	$\neg a$
0	1
1	0

Конъюнкция (AND)

a	b	$a \wedge b$
0	0	0
0	1	0
1	0	0
1	1	1

Дизъюнкция (OR)

a	b	$a \vee b$
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 1 Таблицы истинности элементарных логических функций

Союз	Математическое название	Математический символ				Оператор в Java	Смысл истинности высказывания
	Равенство	$a \leftrightarrow b$				==	Равенство двух переменных
	Сравнение	$<$	$\leq$	$>$	$\geq$	< <= > >=	Сравнение двух целочисленных типов
НЕ	Отрицание	$\neg a$		$\bar{a}$		!	Инверсия условия
И	Коньюнкция	$a \wedge b$		ab		&	Выполнение нескольких условий **одновременно**
ИЛИ	Дизъюнкция	$a \vee b$		$a + b$		|	Выполнение **хотя бы одного** условия

Рис. 2 Обобщающая таблица операторов и логических функций

Пример 1. Запишите условие на языке Java для проверки числа на принадлежность интервалу $a \in [5, 14]$.

Ответ: `(a >= 5) & (a <= 14).`

Пример 2. Дан номер линии метрополитена Санкт-Петербурга $a \in [1, 5]$. Напишите логическую функцию, принимающую истинностное значение тогда и только тогда, когда “линия a пересекается с зеленой линией №3”.

Решение. Единственная линия метрополитена, не имеющая прямой пересадки с третьей Невско-Василеостровской линией, — пятая Фрунзенско-Приморская линия, а все остальные линии обладают необходимой пересадкой.

Ответ: `(a != 5).`

Цикл for

В программировании часто необходимо повторить один и тот же инструкций несколько раз. Для этого используют **циклы**.

```
// Вывести 3 числа: 0, 1, 2
for (int i = 0; i < 3; i++) {
    System.out.println(i);
}
```

Код 4 (Demo.ForLoop). Цикл for

Цикл **for** исполняется, неформально, следующим образом:

1. Объявляется некоторая числовая переменная-итератор, например, **int i = 0**
2. Если выполнено **условие цикла (i < 3)**:
    a. Исполняется содержимое тела цикла (**System.out.println()**)
    b. Выполняется действие над переменной (**i++**, т. е. увеличение на единицу)
    c. Возврат в п. 2
3. Если условие не выполнено (**i < 3**), то цикл завершается.

Ниже приведен общий синтаксис цикла **for** для повторного исполнения блока кода:

```
for (инициализация; условие; итерация) {
    последовательность инструкций
}
```

Часто возникает задача обработать не фиксированный набор переменных, а последовательность произвольного размера. Покажем, как прочитать из стандартного входа набор чисел.

```
5
1 2 3 4 5
```

```
// Чтение множества чисел из стандартного потока.
Scanner in = new Scanner(System.in);
int n = in.nextInt();

for (int i = 0; i < n; i++) {
    int t = in.nextInt();
    // ... обработка очередного числа t ...
}
```

Код 5 (Demo.ForLoop). Чтение чисел из стандартного ввода

Сначала читают число n — количество чисел в потоке, а затем n раз считывают следующее число.

Задачи

Тема: Условия и логические операторы [раздел на informatics.msk.ru]

Тема: Циклы [раздел на informatics.msk.ru]

Вопросы

Тема: Логические операторы

Угринович (учебник, 10 класс, профильный уровень, с. 177):

3.9. Доказать справедливость первого $\overline{A \vee B} = \overline{A}\ \&\ \overline{B}$ и второго $\overline{A\ \&\ B} = \overline{A} \vee \overline{B}$ законов де Моргана, используя таблицы истинности.

3.10. Упростить логическое выражение: $(A \vee B)\ \&\ (A \vee \overline{B})$.

3.11. Решить логическое уравнение: $\overline{X\ \&\ B}\ \&\ \overline{X\&\ \overline{B}} = A$.

Семакин (задачник-практикум, том 1):

№ 10

Используя логические операции, запишите высказывания, которые являются истинными при выполнении следующих условий:

1) хотя бы одно из чисел X, Y, Z положительно;
2) хотя бы одно из чисел X, Y, Z отрицательно;
3) хотя бы одно из чисел X, Y, Z не является положительным;
4) только одно из чисел X, Y, Z является отрицательным.

№ 11

Используя логические операции, запишите высказывания, которые являются истинными при выполнении следующих условий:

1) только одно из чисел X, Y, Z больше 10;
2) только одно из чисел X, Y, Z не больше 10;
3) ни одно из чисел X, Y, Z не равно 104;
4) каждое из чисел X, Y, Z равно 0.

1. Запишите выражения на языке Java для логических функций, принимающих истинностное значение тогда и только тогда, когда:
    a. Число a входит в полуинтервал $[0, 4)$
    b. Число b нацело делится на 7, но не делится нацело на 11
    c. Логические переменные x, y, z либо все истинны, либо все ложны
2. Постройте таблицы истинности для следующих логических функций, предварительно упростив, если требуется:
    a. $a \vee b \vee c$
    b. $\overline{a \wedge b} \vee a$