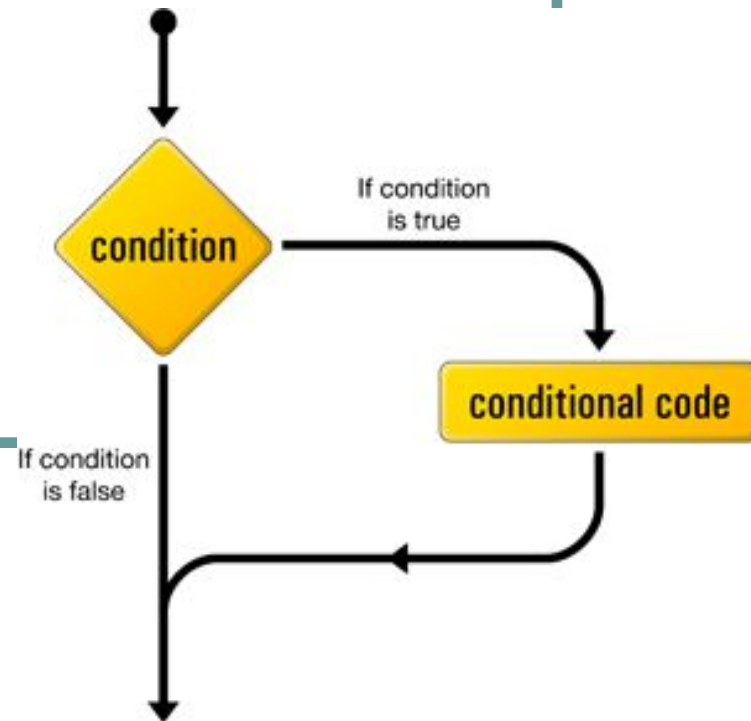


Conditions



Контрольные вопросы

- Сокращённые арифметические формы
- Виды преобразований типов
- Что такое переполнение?
- Как работает оператор %?
- Что такое сноплет?

Операторы сравнения

== (равно, эквиваленция)

!= (не равно)

> (больше)

>= (больше или равно)

< (меньше)

<= (меньше или равно)

Результат их работы – значение типа **boolean!**

Пример операций сравнения

boolean isEqual, isNonEqual, isGreater,
isGreaterOrEqual, isLess, isLessOrEqual;

int a = 5, b = 5, c = 3;

isEqual = a == b; // isEqual = true

isNonEqual = a != b; // isNonEqual = false

isGreater = a > c; // isGreater = true

isGreaterOrEqual = b >= c; // isGreaterOrEqual = true

isLess = c < a; // isLess = true

isLessOrEqual = a <= c; // isLessOrEqual = false

Логические операторы

Булевские операции выполняются над переменными типа **boolean** и их результатом также является значение типа **boolean**.

Логические операторы

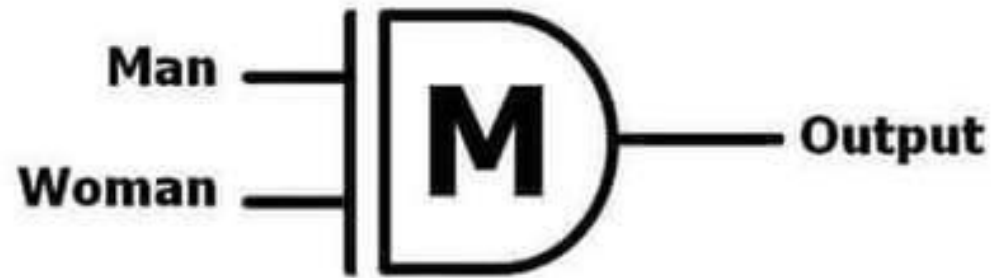
- **отрицание "!"** – замена **false** на **true**, или наоборот
- **операция И "&&"** (конъюнкция) – результат равен **true**, только, если оба операнда равны **true**, иначе результат – **false**
- **операция ИЛИ "||"** (дизъюнкция) – результат равен **true**, только, если хотя бы один из операндов равен **true**, иначе результат – **false**
- **операция исключающее ИЛИ "^"** – результат равен **true**, только, если операнды не равны друг другу, иначе результат – **false**

Таблица результатов применения логических операций

A	B	OR	AND	XOR
<i>false</i>	<i>false</i>	false	false	false
<i>true</i>	<i>false</i>	true	false	true
<i>false</i>	<i>true</i>	true	false	true
<i>true</i>	<i>true</i>	true	true	false



Marriage Logic Gate



Man	Woman	Result
Wrong	Right	Woman is Right
Right	Right	Woman is Right
Right	Wrong	Woman is Right
Wrong	Wrong	Man is Wrong

Пример

```
boolean isInRange, isValid, isValid, isEqual,  
isNotEqual;
```

```
int x = 8;
```

```
isInRange = x > 0 && x < 5; // isInRange = false
```

```
isValid = x > 0 || x > 5; // isValid = true
```

```
isValid = !isValid; // isValid = false
```

```
isEqual = isInRange == isValid; // isEqual = false
```

```
isNotEqual = isInRange != isValid // isNotEqual = true
```

Понятие инструкции

Инструкция (statement) – это минимальная единица программы, представляющая собой один шаг программы.

Одна инструкция может занимать несколько строк. В одной строке может быть несколько инструкций. Любое выражение, которое заканчивается точкой с запятой является инструкцией.

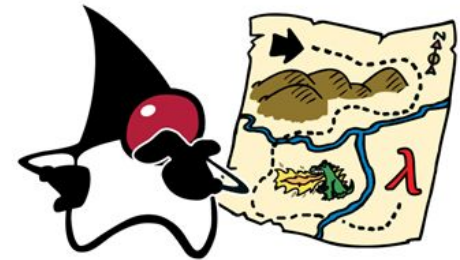
Составная инструкция (блок)

Несколько инструкций, которые заключены в фигурные скобки, образуют **составную инструкцию**. Составная инструкция синтаксически эквивалентна одной инструкции. Составная инструкция может употребляться везде, где допускается обычная инструкция. Точка с запятой не ставится после закрывающей фигурной скобки. Составная инструкция является блоком и может содержать объявления локальных переменных. Эти переменные будут уничтожены после выполнения составной инструкции, т.е. после выхода из блока.

Условная инструкция

Условная инструкция **if** (if statement) – это конструкция языка программирования, которая позволяет выполнять инструкцию в зависимости от истинности некоторого выражения. Общий синтаксис:

```
if (условие)  
    инструкция
```



В данном случае условие – это выражение, которое даёт в результате значение логического типа. Если условие будет истинно, то инструкция выполнится. Если ложно – инструкция будет пропущена.

Условный оператор if

Оператор **if** является основным оператором выбора в Java и позволяет выборочно изменять ход выполнения программы – и, пожалуй, это одно из основных отличий между программированием и простыми вычислениями.

if (условие) оператор // если условие истинно, то оператор отработает

Конструкция if - else

Когда необходимо обеспечить реакцию на ложность условия, применяется конструкция **if-else**. Общий синтаксис:

if (условие)

инструкция1

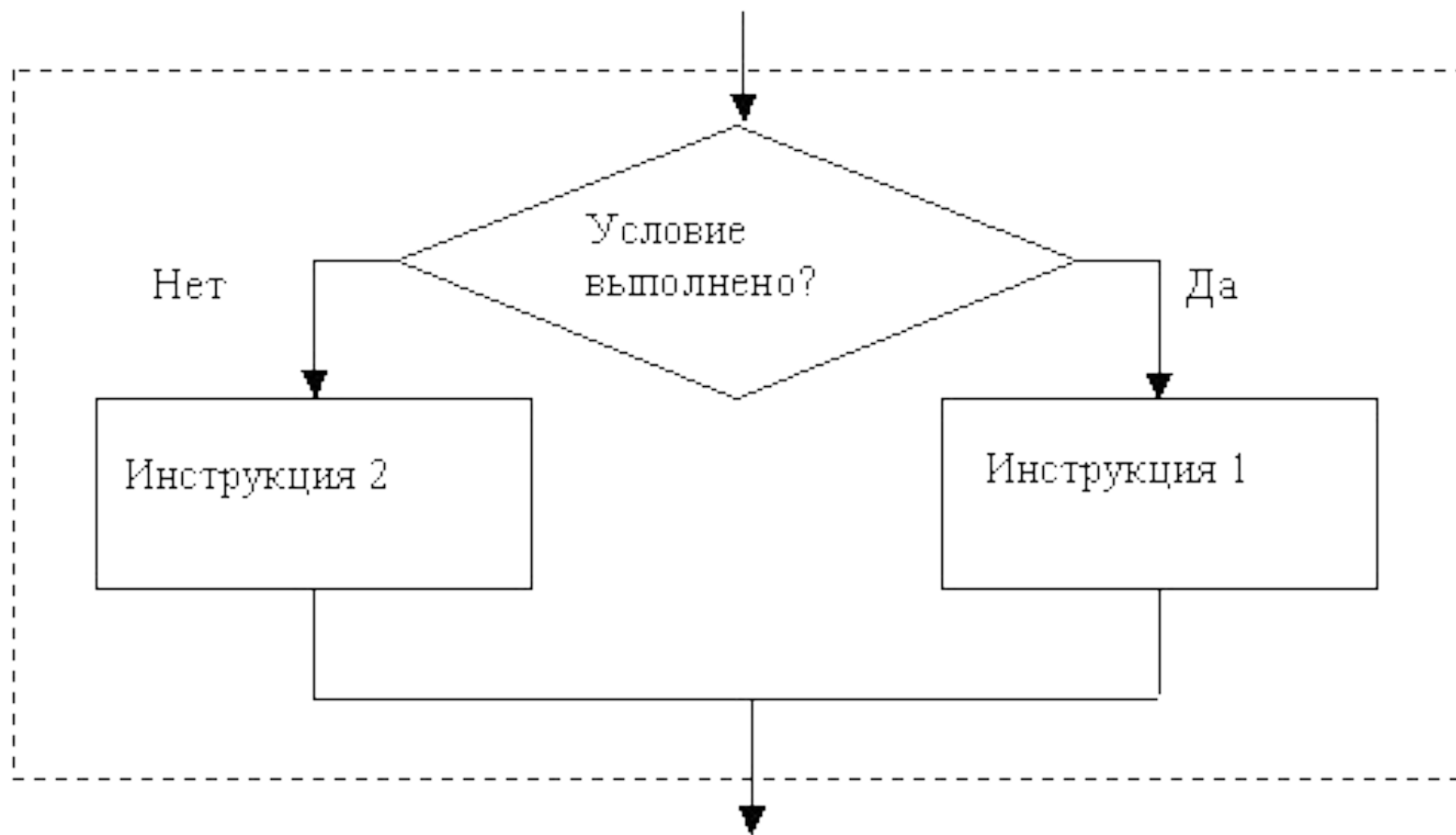
else

инструкция2

В данном случае, если условие будет истинным – будет выполнена только инструкция1, а затем выполнение кода продолжится с инструкций, которые написаны после **if-else**. Если условие будет ложным, будет выполнена только инструкция2.

Если необходимо выполнить несколько инструкций в теле **if** или **else** применяется составная инструкция: **if** (условие) {
инструкция1; инструкция2; }

Блок-схема if - else



Вложенность условий

Условная инструкция является единой инструкцией. То есть, например:

```
if (n < 0)
```

```
    sout(“Отрицательное”);
```

```
else
```

```
    sout(“Положительное или ноль”);
```

воспринимается компилятором как одна цельная инструкция. Следовательно, возможна вложенность.

Неопределённость вложенности

Однако, иногда может возникнуть неопределённость со вложенностью. Например:

```
if (n > 0)
    if (a > b)
        z = a;
else
    z = b;
```

Хоть код и отформатирован, компилятор решает эту проблему таким правилом: `else` относится к самому внутреннему `if`.

```
if (n > 0)
    if (a > b)
        z = a;
else
    z = b;
```

Лесенка ифов

Когда необходимо выбрать один вариант из **более чем двух**, применяется конструкция if-else if. Общий синтаксис:

```
if (условие1)
    инструкция1
else if (условие2)
    инструкция2
...
else if (условиеN)
    инструкцияN
else
    инструкцияПоУмолчанию
```

Принцип работы лесенки

Условия будут проверяться по очереди до тех пор, пока не будет найдено истинное, либо пока выполнение не дойдет до **else**. Если истинное условие найдено – проверки прекращаются, выполняется указанная инструкция, а выполнение кода продолжается с кода, который написан далее за **if-else if-else**.

Цепочка операторов if-else-if

```
if (condition1)
    statement1
else if (condition2)
    statement2
else if (condition3)
    statement3
...
else statementN
```

Условные выражения оцениваются сверху вниз. Как только найдено условие, принимающее значение **true**, выполняется ассоциированный с этим условием оператор, а остальная часть цепочки пропускается. Если ни одно из условий не принимает значение **true**, то выполняется последний оператор **else**, который можно рассматривать как оператор по умолчанию.

Составной оператор

```
if (age > 18) buyCount++;  
    System.out.print("Продано");
```

```
if (age > 18) {  
    buyCount++;  
    System.out.print("Продано");  
}
```

Операторы, заключённые в фигурные скобки, считаются за одну операцию, и называются составным (блочным) оператором.

Тернарный оператор

```
int e = (b == 0) ? 0 : (a / b);
```

В качестве первого операнда — «выражение 1» — может быть использовано любое выражение, результатом которого является значение типа **boolean**. Если результат равен **true**, то выполняется оператор, заданный вторым операндом, то есть «выражение2». Если же первый операнд равен **false**, то выполняется третий операнд — «выражение3». Второй и третий операнды, то есть «выражение2» и «выражение3», должны возвращать значения одного типа, и не должны иметь тип **void**!

Тернарный оператор

условие ? действие1 : действие2;

Пример условной операции:

```
int x = n > 1 ? 0 : 1;
```

Переменной **x** будет присвоено значение **0**, если **n > 1** (выражение **n > 1** имеет значение **true**) или **1**, если **n ≤ 1** (выражение **n > 1** имеет значение **false**).

Оператор множественного выбора switch

В отличие от операторов **if-else**, оператор **switch** применим к известному числу возможных ситуаций. Можно использовать простые типы **byte**, **short**, **char**, **int**. Также можно использовать **Enum** и **String** (начиная с JDK7), а также специальные классы, которые являются обёрткой для примитивных типов: **Character**, **Byte**, **Short**, **Integer**.

Оператор switch

```
switch (выражение для сравнения) {  
    case совпадение1:  
        команда; break;  
    case совпадение2:  
        команда; break;  
    case совпадение3:  
        команда; break;  
    default: оператор; break;  
}
```

Оператор switch

Каждая секция **case** обычно заканчивается командой **break**, которая передаёт управление к концу команды **switch**. Если не использовать **break**, выполнение кода продолжится.

Дублирование значений **case** не допускается. Тип каждого значения должен быть совместим с типом выражения.

Пример использования switch

```
int month = 3;
String monthString;
switch (month) {
    case 1:
        monthString = "Январь";
        break;
    case 2:
        monthString = "Февраль";
        break;
    default:
        monthString = "Что-то другое";
        break;
}
System.out. print(monthString);
```

Случайные числа




Генератор псевдослучайных чисел - это алгоритм, порождающий последовательность чисел, элементы которой почти независимы друг от друга и обычно подчиняются равномерному распределению. Сфера применения СЧ достаточно широка — от имитационного моделирования до криптографии.

Источники настоящих СЧ

Источники настоящих случайных чисел найти очень трудно. Физические шумы, такие, как детекторы событий ионизирующей радиации или космическое излучение, вполне сгодятся в роли таких источников. Однако применяются подобные устройства в приложениях сетевой безопасности крайне редко.

Источники энтропии (непредсказуемости)

Генераторы случайных чисел для своей работы обычно используют:

-  Счётчик тактов процессора
-  Текущее время, размер жёсткого диска, размер свободной памяти, номер процесса и имя компьютера
-  Шумы токов

Как получить СЧ в Java

- `Math.random()` даёт случайное вещественное число в диапазоне от 0 до 1 (не включая 1)
- Для получения диапазона от 0 до 100, можно умножить `Math.random() * 101`
- Для получения диапазона от 50 до 100, можно умножить `Math.random() * 51 + 50`
- Чтобы получить целый результат, используется приведение к типу `int`

Вариант №2

```
Random r = new Random();  
int x = r.nextInt(20); // числа от 0 до 19
```

Есть метод `nextDouble`.

Практика

- Загадать случайный возраст от 16 до 70 лет
- Загадать случайную температуру воздуха от -30 до +49
- Загадать случайный символ от A до Z
- Загадать случайную карту
- Загадать случайное время суток