

Содержание

1. Логические операции	2
1.1. Логическое И	2
1.1.1. Таблица истинности	2
1.1.2. Обозначения	2
1.1.3. Свойства	2
1.2. Логическое ИЛИ	3
1.2.1. Таблица истинности	3
1.2.2. Обозначения	3
1.2.3. Свойства	3
1.3. Отрицание	3
1.3.1. Таблица истинности	3
1.3.2. Обозначения	3
1.3.3. Свойства	3
1.4. Импликация	4
1.4.1. Таблица истинности	4
1.4.2. Обозначения	4
1.4.3. Свойства	4
1.5. Эквиваленция	4
1.5.1. Таблица истинности	4
1.5.2. Обозначения	4
1.5.3. Свойства	4
1.6. Исключающее ИЛИ	5
1.6.1. Таблица истинности	5
1.6.2. Обозначения	5
1.6.3. Свойства	5
1.7. Практическое применение	5

Список таблиц

1. Логическое И	2
2. Логическое ИЛИ	3
3. Отрицание	3
4. Импликация	4
5. Эквиваленция	4
6. Исключающее ИЛИ	5
7. Некоторые операции в языках программирования	5

1. Логические операции

Определение 1. Будем говорить, что операция является **сильной относительно значения**, если она принимает это значение **реже**, чем противоположное.

Определение 2. Будем говорить, что операция является **слабой относительно значения**, если она принимает это значение **чаще**, чем противоположное.

Пример: операция конъюнкции является сильной относительно единицы, поскольку она принимает её лишь в одном случае из четырёх возможных.

1.1. Логическое И

1.1.1. Таблица истинности

x	y	$x \& y$
0	0	0
0	1	0
1	0	0
1	1	1

Таблица 1. Логическое И

1.1.2. Обозначения

Операция также называется **конъюнкцией**, обозначается через \wedge , AND, И. В С-подобных языках и Java обозначается $\&$.

Замечание 1. Не путать с обозначением **исключающего ИЛИ** (1.6) в С-подобных языках (и Java)!

1.1.3. Свойства

1. Данная операция полностью эквивалентна обыкновенному умножению, что легко проверяется подстановкой.
2. Данная операция является слабой относительно нуля.

1.2. Логическое ИЛИ

1.2.1. Таблица истинности

x	y	$x \vee y$
0	0	0
0	1	1
1	0	1
1	1	1

Таблица 2. Логическое ИЛИ

1.2.2. Обозначения

Операция также называется **дизъюнкцией**, обозначается через $|$, OR, ИЛИ. В С-подобных языках и Java обозначается $|$.

1.2.3. Свойства

1. Данная операция является слабой относительно единицы.

1.3. Отрицание

1.3.1. Таблица истинности

x	$\neg x$
0	1
1	0

Таблица 3. Отрицание

1.3.2. Обозначения

Операция также называется **инверсией**. Имеет место обозначение \bar{x} . В С-подобных языках и Java обозначается \sim .

1.3.3. Свойства

1. Данная операция не является сильной или слабой.
2. Операция полностью эквивалентна арифметическому действию $\neg x := 1 - x$, что легко проверяется подстановкой.

1.4. Импликация

1.4.1. Таблица истинности

x	y	$x \rightarrow y$
0	0	1
0	1	1
1	0	0
1	1	1

Таблица 4. Импликация

1.4.2. Обозначения

Операция также называется **следованием**.

1.4.3. Свойства

1. Данная операция является слабой относительно единицы.
2. Операция полностью эквивалентна $\neg x \vee y$.

1.5. Эквиваленция

1.5.1. Таблица истинности

x	y	$x \leftrightarrow y$
0	0	1
0	1	0
1	0	0
1	1	1

Таблица 5. Эквиваленция

1.5.2. Обозначения

Операция также называется **равносильностью**, **тождественным равенством**. В ЕГЭ имеет место обозначение \equiv .

1.5.3. Свойства

1. Данная операция не является сильной.
2. Операция полностью эквивалентна $x \rightarrow y \& y \rightarrow x$.
3. Именованное “тождественное равенство” дано неслучайно: в самом деле, операция эквивалентна обыкновенному равенству

1.6. Исключающее ИЛИ

1.6.1. Таблица истинности

x	y	$x +_2 y$
0	0	0
0	1	1
1	0	1
1	1	0

Таблица 6. Исключающее ИЛИ

1.6.2. Обозначения

Обозначается XOR (eXclusive OR). В C-подобных языках и Java обозначается \wedge .

1.6.3. Свойства

1. Операция также называется **сложением по модулю 2**, поскольку является семантически эквивалентной ей: $x +_2 y := (x + y) \bmod 2$.
2. Данная операция не является сильной.
3. Операция является обратной к эквиваленции.

1.7. Практическое применение

Замечание 2. Языки программирования различают битовые и логические операции И, ИЛИ, НЕ, чем вносят некоторую путаницу.

Название	Логическая	Битовая
И	$\&\&$	$\&$
ИЛИ	$\ \ $	$\ $
НЕ	\sim	$!$

Таблица 7. Некоторые операции в языках программирования

Так, с точки зрения языка Си, аргументы **логического И** обрабатываются следующим образом: если аргумент не равен нулю, он воспринимается как истинностное значение. Ноль – как ложное. **Логического И** всегда возвращает **логический** результат – истину или ложь (в языке Си это 1 или 0, по умолчанию тип int, в Java – true или false, тип boolean).

Битовое И, однако, ведёт себя абсолютно иным образом: команда представляет аргументы в двоичном виде, делает поразрядную (побитовую) конъюнкцию и полученный результат переводит обратно в десятичный вид. Конечно же,

это реализуется иным образом, иначе эффективность операций была неоправданно низкой, однако человек бы делал то, что делает машина, именно таким образом (по крайней мере, наивную реализацию). **Битовое И** всегда возвращает **целочисленный** результат – целое число из разрешённого диапазона.

Пример:

- $123 \& 46 == \text{ИСТИНА И ИСТИНА} == \text{ИСТИНА}.$
- $123 \& 46 == 1111011_2 \text{ И } 0101110_2 == 0101010_2 == 41$

По аналогичному принципу работают остальные битовые операции.

Итого: **битовые** операции переводят целое число в двоичный формат и производят поразрядное применение **логической** операции в соответствии с построенными таблицами истинности.

Замечание 3. Всё вышесказанное в полной мере относится и к Java.

Замечание 4. Операция исключающего ИЛИ – только **битовая**! Операция импликации и иже с ними не реализована в языке, поскольку является выражимой через уже реализованные.

Замечание 5. Битовые операции поддерживаются подавляющим большинством современных процессоров на аппаратном уровне, поэтому их выполнение осуществляется очень быстро. В связи с этим многие операции, родственные битовым, такие как генерация подмножеств заданного множества, стараются осуществлять с помощью так называемых битовых хакков.