Логические выражения

Логическое выражение — это утверждение или комбинация утверждений, которые могут быть истинными (true) или ложными (false).

Примеры логических выражений:

- `x > y` (x больше, чем y)
- `a == b` (a равно b)
- `(x > 5) and (y < 10)` (x больше 5 и y меньше 10)
- `(a != b) or (c == d)` (a не равно b или c равно d)

Логические выражения могут быть использованы в различных языках программирования, таких как Java, Python, C++ и других. Они играют важную роль в управлении потоком выполнения программы и помогают программистам создавать более гибкие и мощные программы.

Логические операции

Логические операции — это операции, которые выполняются над логическими выражениями и результатом их выполнения является новое логическое выражение.

В программировании, логические операции обычно выполняются над булевыми значениями (true/false).

Примеры логических операций:

- **Операция «И» (AND), конъюнкция**: обозначается символом && или * или ∧, возвращает true только если оба операнда равны true. Например, «сходить в магазин И купить молоко» — будет выполнено только если оба условия верны.
- **Операция «ИЛИ» (OR), дизъюнкция:** обозначается символом || или + или ∨, возвращает true, если хотя бы один из операндов равен true. Например, «если завтра будет солнечно ИЛИ я не буду занят — я поеду на пикник» — здесь выполнение произойдет если хотя бы одно из условий верно.
- **Операция «НЕ» (NOT), отрицание:** обозначается символом ! или ¬, инвертирует логическое значение операнда (true становится false, и наоборот). Например, «не пойду в кино» — значение станет true, если изначально мы планировали пойти в кино, а после применения NOT мы получим отрицание этого утверждения — что мы не пойдем.
- **Тождественное равенство** — это логическая операция, которая используется для проверки равенства двух логических выражений во всех возможных случаях. Она обозначается как «≡» или как «==» в некоторых языках программирования. Результат тождественного равенства является логическим значением истинности (true) или ложности (false). Например, выражение «x = y ≡ y > x» вернет значение false.
- **Импликация (следование)** — это логическая операция, которая определяет, как одно логическое выражение влияет на другое. Она обозначается как «→» или как «->» в некоторых языках программирования. Импликация говорит о том, что если одно выражение (предпосылка) является истинным, то другое выражение (заключение) также является истинным. Если же предпосылка ложна, то заключение может быть как истинным, так и ложным. Например, выражение «если

я поеду в отпуск, то я возьму с собой книги» может быть записано как «поездка в отпуск → взятие книг». Если я поеду в отпуск, то я обязательно возьму с собой книги. Но если я не поеду в отпуск, то я могу взять книги или не брать их в зависимости от других факторов.

- **Таблицы истинности для конъюнкции, дизъюнкции, импликации**
- Конъюнкция, дизъюнкция и импликация — это три основные бинарные логические операции. Бинарные логические операции работают с двумя логическими выражениями (условиями) и возвращают результат в виде одного логического значения — истинности или ложности.
- Таблицы истинности для этих операций выглядят следующим образом:
- **Конъюнкция (AND)**

A	B	A AND B
0	0	0
0	1	0
1	0	0
1	1	1

- В таблице истинности для конъюнкции, результат будет истинным только в том случае, если оба условия истинны.
- **Дизъюнкция (OR)**

A	B	A OR B
0	0	0
0	1	1
1	0	1
1	1	1

- В таблице истинности для дизъюнкции, результат будет истинным, если хотя бы одно из условий истинно.
- **Импликация (IF-THEN)**

A	B	A → B
0	0	1
0	1	1
1	0	0
1	1	1

Приоритет логических операций

Приоритет логических операций — это порядок, в котором операции выполняются в логическом выражении.

Если в выражении присутствуют несколько операций, то порядок их выполнения зависит от приоритета операций.

В общем случае, порядок выполнения операций в выражении определяется следующими правилами:

1. Сначала выполняются операции в скобках.
2. Затем выполняются операции логического отрицания (NOT).
3. Затем выполняются операции логического умножения (AND).
4. Затем выполняются операции логического сложения (OR).
5. В конце выполняются операции импликации (IF-THEN) и эквивалентности (IF AND ONLY IF).

Таким образом, операции в логическом выражении выполняются в порядке убывания приоритета.

Построение таблиц истинности логических выражений с помощью Python

Построение таблицы истинности для конъюнкции трех условий.

Для построения таблицы истинности для конъюнкции трех условий можно воспользоваться языком программирования Python и его возможностью работы с логическими операциями.

Для создания всех возможных комбинаций 0 и 1 (истинных и ложных значений), воспользуемся тройным вложенным циклом `for` Каждый цикл перебирает два возможных значения (`0` и `1`) для каждого из операндов: `x`, `y`, `z`

Для этого можно написать следующий код:

```
# выводим заголовок таблицы
print("a | b | c | a ∧ b ∧ c")
print("--|---|---|---------")

# перебираем все возможные значения переменных a, b и c
for a in range(2):
    for b in range(2):
        for c in range(2):

            # вычисляем значение выражения и выводим результат в таблицу
            result = a and b and c
            print(f"{int(a)} | {int(b)} | {int(c)} | {int(result)}")
```

В данном коде мы задаем начальные значения переменных a, b и c как False. Затем мы выводим заголовок таблицы истинности и перебираем все возможные значения переменных a, b и c, используя вложенные циклы. Для каждой комбинации значений переменных мы вычисляем значение выражения и выводим результат в таблицу. В конце каждой строки таблицы мы переходим на новую строку с помощью команды `print()`.

Результатом выполнения данного кода будет таблица истинности для конъюнкции трех условий:

```
a | b | c | a ∧ b ∧ c
--|---|---|---------
0 | 0 | 0 | 0
0 | 0 | 1 | 0
```

```
0 | 1 | 0 | 0
0 | 1 | 1 | 0
1 | 0 | 0 | 0
1 | 0 | 1 | 0
1 | 1 | 0 | 0
1 | 1 | 1 | 1
```

Как можно видеть из таблицы, значение конъюнкции трех условий будет равно 1 (True) только в том случае, когда все три условия истинны. В остальных случаях, значение выражения будет равно 0 (False).

Решение основано на принципах работы с булевыми значениями (True/False) и логическими операциями (and). При помощи циклов `for` и операторов `range` мы перебираем все возможные комбинации значений переменных a, b и c, и для каждой комбинации вычисляем значение выражения a ∧ b ∧ c. Результаты вычислений выводим в таблицу истинности с помощью команды `print()`.

Построение таблицы истинности для импликации в Python

Как обозначается импликация в Python?

Как мы ранее узнали, импликация — это логическая операция, которая означает «если…, то…». В Python, импликацию можно реализовать двумя способами:

1. Надежный, но длинный. Через предобразование: `A → B == not A or B`
2. Быстрый но ненадежный. Можно напутать с приоритетами операций, когда есть скобки. `A → B == A<=B`

Давайте построим таблицу истинности для импликации с помощью Python:

```
def implication(a, b):
    return (not a) or b

print("Таблица истинности для импликации:")
print("a | b | a -> b")
print("-"*13)
for a in [True, False]:
    for b in [True, False]:
        result = implication(a, b)
        print(int(a), "|", int(b), "|", int(result))
```

Здесь мы определяем функцию `implication`, которая принимает два аргумента `a` и `b` и возвращает значение импликации `a -> b`. Затем мы выводим заголовок таблицы истинности и используем два вложенных цикла `for`, чтобы перебрать все возможные комбинации значений `a` и `b`. Для каждой комбинации мы вычисляем результат функции `implication` и выводим значения `a`, `b` и результат в виде строки таблицы.

Результат выполнения программы будет следующим:

```
Таблица истинности для импликации:
a | b | a -> b
-------------
1 | 1 | 1
1 | 0 | 0
0 | 1 | 1
0 | 0 | 1
```

Здесь `1` означает `True`, а `0` — `False`. Как видим, значение импликации `a -> b` истинно только в тех случаях, когда `a` ложно или `a` и `b` истинны одновременно.

Решение заданий ЕГЭ с помощью Python

Типовое задание:

Логическая функция *F* задаётся выражением $(x \equiv z) \vee (x \rightarrow (y \wedge z))$.

Дан частично заполненный фрагмент, содержащий **неповторяющиеся** строки таблицы истинности функции *F*.

Определите, какому столбцу таблицы истинности соответствует каждая из переменных *x*, *y*, *z*.

Переменная 1	**Переменная 2**	**Переменная 3**	**Функция**
???	???	???	*F*
0	0		0
1			0

В ответе напишите буквы *x*, *y*, *z* в том порядке, в котором идут соответствующие им столбцы (сначала — буква, соответствующая первому столбцу; затем — буква, соответствующая второму столбцу, и т. д.). Буквы в ответе пишите подряд, никаких разделителей между буквами ставить не нужно.

Пример. Пусть задано выражение $x \rightarrow y$, зависящее от двух переменных *x* и *y*, и фрагмент таблицы истинности:

Переменная 1	**Переменная 2**	**Функция**
???	???	*F*
0	1	0

Тогда первому столбцу соответствует переменная *y*, а второму столбцу соответствует переменная *x*. В ответе нужно написать: *yx*.

Решение

Для построения таблицы истинности данного логического выражения можно воспользоваться языком программирования Python и его возможностью работы с логическими операциями. Для этого можно написать следующий код:

```
# выводим заголовок таблицы
print("x | y | z | (x ≡ z) ∨ (x → (y ∧ z))")
print("--|---|---|----------------------")

# перебираем все возможные значения переменных x, y и z
for x in range(2):
    for y in range(2):
        for z in range(2):
            # вычисляем значение выражения и выводим результат в таблицу
            result = (x == z) or (not x or (y and z))
```

```
#Фильтруем таблицу, печатаем только те значения, где результат = 0 (False)
            if not result:
                print(f"{x} | {y} | {z} | {int(result)}")
```

В данном коде мы задаем начальные значения переменных x, y и z как False. Затем мы выводим заголовок таблицы истинности и перебираем все возможные значения переменных x, y и z, используя вложенные циклы. Для каждой комбинации значений переменных мы вычисляем значение выражения и выводим результат в таблицу. В конце каждой строки таблицы мы переходим на новую строку с помощью команды `print()`.

Результатом выполнения данного кода будет таблица истинности для данного логического выражения:

```
x | y | z | (x ≡ z) ∨ (x → (y ∧ z))
--|---|---|-----------------------
1 | 0 | 0 | 0
1 | 1 | 0 | 0
```

Как можно видеть из таблицы, значение данного логического выражения будет равно 1 (True) только в тех случаях, когда x и z равны, или когда x равно True, а y и z также равны True. В остальных случаях, значение выражения будет равно 0 (False).