# Логические выражения

Логические переменные

Для того, чтобы мы могли хранить данные логического типа, нам надо знать о логических переменных.

Логические данные хранятся в переменных типа bool (или boolean в Arduino).
Хранить они могут только два значения:

• «Верно» — это true;
• «Ложь» — это false;

По умолчанию C++ при выводе логических значений используются два значения:

• 1 для true;
• 0 для false;

**Логические операторы в C++.**

**Операторы сравнения**

Язык C++ имеет 5 различных операторов сравнения в своём арсенале.

Давайте разберём по порядку каждый из них:

• A < B — сравнивает две переменные и возвращает true, если A меньше B.
• A > B — возвращает true, если A строго больше B.
• A == B — проверяет на равенство переменные A и B.
• A != B — проверяет переменные A и B на неравенство.
• A >= B — нестрогое неравенство. Возвращает true, если A больше или равно B.
• A <= B — противно неравенству A > B.

Теперь разберём пример, чтобы закрепить теорию практикой:

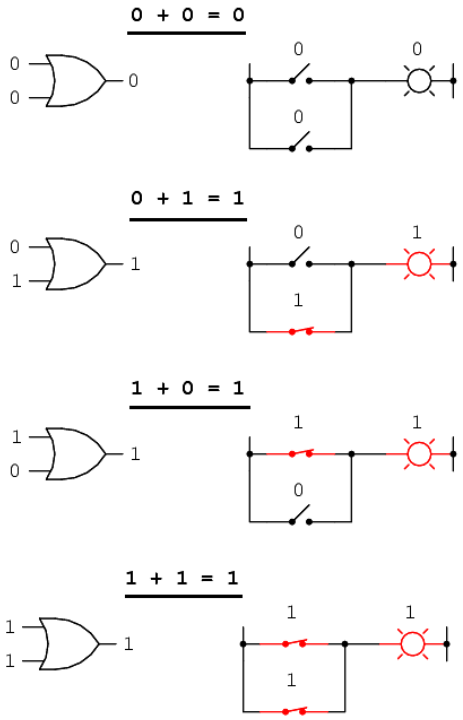
```
bool r; int a = 5, b = 7;

r = a > b; // false поскольку 5 меньше 7
r = a <= b; // true, поскольку это противно неравенству
r = a <= 5; // true, поскольку a равно 5
r = b == 9; // false, поскольку 7 != 9
```

**Логические операторы**

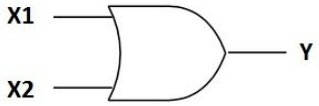
Рассмотрим следующий список операторов:

• A && B — эквивалент логического «И». Соответственно возвращает true, если A и B являются истиной.
• A || B — эквивалент логического «ИЛИ». Вернёт true если хотя бы одно из выражений является истинным.
• A xor B — этот оператор можно сравнить с «ТОЛЬКО ОДИН», соответственно вернёт true если A == true и B == false, или наоборот.
• !A — данный оператор инвертирует значение A. То есть, если A == true, то он вернёт false и наоборот.

## ИЛИ, OR, ||



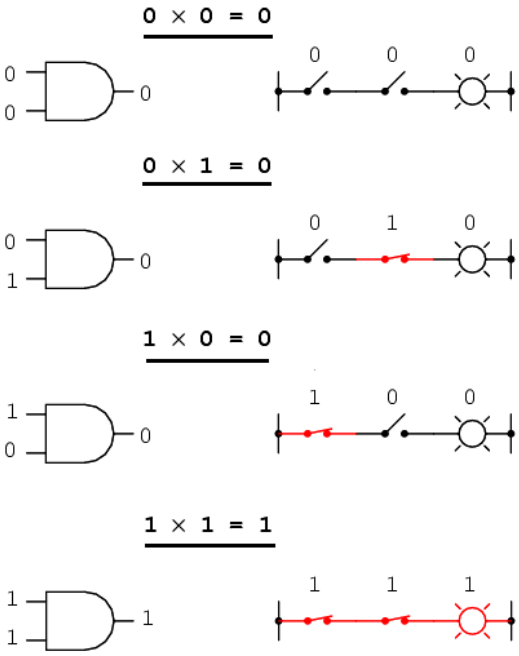

На рисунке слева продемонстрирована работа логического оператора ИЛИ ( OR, || )



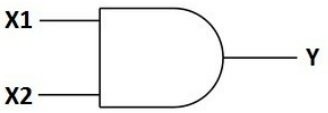

| Вход X1 | Вход X2 | Выход Y |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 1 | 1 |

Таблица истинности для элемента «2ИЛИ» показывает, что для появления на выходе логической единицы, достаточно чтобы логическая единица была на первом входе ИЛИ на втором входе. Если логические единицы будут сразу на двух входах, на выходе также будет единица.

## И, AND, &&



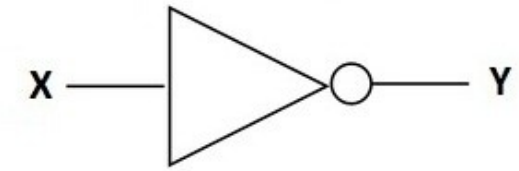

На рисунке слева продемонстрирована работа логического оператора И ( AND, && )




| Вход X1 | Вход X2 | Выход Y |
|---|---|---|
| 0 | 0 | 0 |
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 1 | 1 |

Таблица истинности для элемента 2И показывает, что на выходе элемента будет логическая единица лишь в том случае, если логические единицы будут одновременно на первом входе И на втором входе. В остальных трёх возможных случаях на выходе будет ноль.

**НЕ (NOT, !)**




| Вход X | Выход Y |
|---|---|
| 0 | 1 |
| 1 | 0 |

Таблица истинности для инвертора показывает, что высокий потенциал на входе даёт низкий потенциал на выходе и наоборот.

| Операнд 1 | Операнд 2 | && (И) | \|\| (ИЛИ) |
|---|---|---|---|
| true | true | true | true |
| true | false | false | true |
| false | true | false | true |
| false | false | false | false |

| Операция | Назначение | Пример |
|---|---|---|
| && | *Логическое И* | (R>2) && (R<20) |
| \|\| | *Логическое ИЛИ* | (L= =12) \|\| (L>15) |
| ! | *Логическое НЕ* | Tab != 13 |

Пример:

```
void setup()
{
   Serial.begin(9600);
   bool r; // Создаём переменную типа bool
   int a = 10, b = 7; // а также две переменные типа int

   r = (a < b) && (b == 7); // Возвращает 0 (False)
   Serial.println(r ? "true" : "false");

   r = a < b || b == 7; // Возвращает 1 (True)
   Serial.println(r ? "true" : "false");

   r = (a < b) xor (b == 7); // Возвращает 1 (True)
   Serial.println(r ? "true" : "false");

   r = !(a == 10 && (b <= 8 || true)); // Возвращает 0 (False)
   Serial.print(r ? "true" : "false");
}

void loop()
{
   // Тут ничего нет
}
```