



Урок 9. C++: Логические значения и сложные условия

Часть 1. Материалы ученика

Программирование на Python и C++ · 8 класс

Урок 9 из 33 · 45 минут · Модуль «Ветвления. Логические переменные. Сложные условия»

Уровень: начинающий программист; C++17.

Программа: <https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

Алгоритм знаком по Python, но запись другая: `&&`, `||` и `!`. Перенесём смысл условий, не копируя механически синтаксис.

Результат урока: Записать сложное условие C++, проверить таблицу истинности и безопасный порядок вычисления при нулевом делителе.

01. Проблемная ситуация

0-5 минут. Для отрезка $-2 \leq x \leq 3$ ученик написал `-2 <= x <= 3`. Проверьте идею на $x=10$: почему в C++ это не проверка двух границ? Сначала запишите на русском два требования, затем соедините их логическим И.

02. Понятия и разобранный пример

5-14 минут: 4 минуты на понятия, 3 на пример, 2 на объяснение способа партнёру.

Сравнения соединяются операторами `&&` и `||`

В C++ тип `bool` имеет значения `true` и `false`. Для логических операндов `&&` означает И, `||` — ИЛИ, `!` — отрицание. Отрезок записывают `x >= -2 && x <= 3`. Запись `-2 <= x <= 3` вычисляет сначала одно сравнение, затем сравнивает его логическое значение, преобразованное в 0 или 1, с 3; нужного смысла она не имеет.

`std::boolalpha` печатает слова `true/false` вместо 1/0. Это формат вывода, а не изменение логики. Вне отрезка `x < -2 || x > 3`. Проверки обеих границ важны: $x=-2$ и $x=3$ входят, $x=-3$ и $x=4$ не входят.

Встроенные `&&` и `||` вычисляются слева направо с коротким замыканием. Если левая часть `&&` ложна, правая не вычисляется. Поэтому `d!=0 && n%d==0` безопасно для нулевого `d`; обратный порядок не защищает. Не заменяйте `&&` одиночным `&`: у побитовой операции другой смысл и нет такой гарантии остановки.

Отступы набирайте обычными пробелами: 4 на уровень. При копировании с веб-страницы замените неразрывные пробелы обычными; полный код с обычными пробелами есть в PDF.

При вводе 0 вывод `true false`; при 10 — `false true`. Для -2 и 3 вывод тоже `true false`. Ввод должен содержать целое, представимое `int`; при неудачном чтении печатается `error`. Дополнительные токены после первого числа здесь не анализируются.

```
#include <iostream>
```

```
int main() {
    int x;
    if (!(std::cin >> x)) { std::cout << "error\n"; return 0; }
    bool inside = x >= -2 && x <= 3;
    bool outside = x < -2 || x > 3;
    std::cout << std::boolalpha << inside << " " << outside << "\n";
}
```



A=false, B=false	A&&B=false; A B=false
A=false, B=true	A&&B=false; A B=true
A=true, B=false	A&&B=false; A B=true
A=true, B=true	A&&B=true; A B=true

Отрезок $-2 \dots 3$: `inside = x >= -2 && x <= 3`; `outside` — противоположное логическое значение.

03. Практика 1. Проверяем материал

Таблица логического запрета

14–25 минут. Учебный редактор разрешает изменение, если файл не заблокирован и не занят. Два входа `locked, busy` задаются целыми 0 или 1. Выведите разрешение словом `true/false`, иные входы отвергните.

locked, busy	Разрешено: <code>!locked && !busy</code>
false,false	true
false,true	false
true,false	false
true,true	false

2 минуты — таблица; 4 — программа; 3 — четыре сочетания и ошибочный вход 2 0; 2 — объяснение `!(locked||busy)`. Продукт: исходник и журнал пяти тестов. Критерии: валидация 0/1, отрицания, `&&`, вывод `boolalpha`. Это модель, не подключение к реальным файлам.

04. Практика 2. Перенос способа

Короткое вычисление защищает остаток

25–33 минуты. Напишите проверку делимости `n` на `d`: оба целые $0 \dots 100$, при `d=0` результат `false`. За 2 минуты прогноз (18,3),(18,4),(0,7),(18,0); за 4 код; за 2 запуск. Сначала проверьте диапазон, затем короткое условие. Критерии: отсутствие остатка от деления на ноль, четыре ответа, объяснение порядка.

05. Обсуждение и границы вывода

33–40 минут. Ученик предлагает заменить `&&` на `&` «чтобы короче». Объясните, почему сходный результат на `true/false` не даёт права менять оператор в защите `n%d`. Сопоставьте Python `and` и C++ `&&`: что переносится, а что нужно записать иначе?

06. Индивидуальная проверка

40–45 минут. Запишите C++-условие вне отрезка $5 \dots 8$, найдите результат для 5 и 9 и объясните, почему `d!=0` должно стоять слева в проверке делимости. Четыре элемента — четыре балла.



Часть 2. Ответы и материалы учителя

Практика 1: разбор и критерии

Только 0 0 даёт true, остальные допустимые пары false; 2 0 — error. Условия `!locked&&!busy` и `!(locked||busy)` равносильны на bool. Проверка диапазона нужна до интерпретации числа как логического признака.

```
#include <iostream>
```

```
int main() {
    int locked, busy;
    if (!(std::cin >> locked >> busy)
        || locked<0 || locked>1 || busy<0 || busy>1) {
        std::cout << "error\n"; return 0;
    }
    bool allowed = !locked && !busy;
    std::cout << std::boolalpha << allowed << "\n";
}
```

Практика 2: разбор и критерии

true,false,true,false. Для $n=0, d=7$ остаток равен нулю. При $d=0$ правая часть `&&` не выполняется. Отрицательные и превышающие 100 входы дают error, а не математическую классификацию.

```
#include <iostream>
```

```
int main() {
    int n,d;
    if (!(std::cin >> n >> d) || n<0 || n>100
        || d<0 || d>100) {
        std::cout << "error\n"; return 0;
    }
    bool divides = d!=0 && n%d==0;
    std::cout << std::boolalpha << divides << "\n";
}
```

Обсуждение: возможный вывод

Оба логических оператора дают короткое вычисление; побитовое `&` не заменяет эту гарантию и может вычислить опасную правую часть. Переносим смысл и порядок проверки, а не символы. Сначала ненулевой делитель, затем остаток; Python пишет `and`, C++ `&&`.

Индивидуальная проверка: ответ и критерии

$x<5 || x>8$; для 5 false; для 9 true; левая ложная часть `&&` останавливает вычисление до `n%d`. 4 балла.

Учителю: подготовка и план на 45 минут

Подготовьте компьютер, редактор, папку с правом записи, распечатку ученической части PDF, бумагу и ручку. Компилятор G++ с поддержкой C++17 и редактор. Сборка: `g++ -std=c++17 -Wall -Wextra -pedantic main.cpp -o lesson`. Запуск `./lesson` в Linux/macOS или `.\lesson.exe` в Windows (при сборке задайте `-o lesson.exe`). Установку и проверку запуска учитель выполняет заранее. Все программы, данные и ожидаемые ответы даны; регистрация детей и интернет на занятии не нужны.

Уровень — начинающий программист, 8 класс. Предпосылки: bool, сравнения, if, целое деление и остаток. В паре автор прогнозирует и вводит код, проверяющий ведёт журнал, затем роли меняются. Без компьютера выполните трассировку полного кода на бумаге и сопоставьте с готовыми результатами; учитель демонстрирует подготовленный запуск. Бумажный вариант не считается фактическим запуском ученика. Перед практикой учитель выдаёт готовый файл разобранного примера, чтобы дети тратили время на изменение и проверку, а не длинное перепечатывание. Матрица программы предназначена учителю.



Минуты	Работа	Результат
0–5	Проблема	Вопрос и первое решение
5–14	Понятия и пример	Способ проверки
14–25	Практика 1	Готовый продукт и проверка
25–33	Практика 2	Перенос способа
33–40	Обсуждение	Аргумент и ограничение
40–45	Индивидуальная проверка	Самостоятельный ответ

07. Соответствие исходной программе

Пункт программы	Действие ученика	Проверяемый результат
Ветвления. Логические переменные. Сложные условия	Разбирает пример и создаёт две подпрограммы	Исходники, прогнозы и протокол проверки
Составление и отладка программ	Сравнивает фактический результат с контрактом	Объяснение границ и самостоятельный ответ

Источники и права на изображения

Исходная программа курса

<https://schoolotzyv.ru/blog/python-cpp-8-klass-programma>

C++: встроенное логическое И и порядок вычисления

<https://eel.is/c++draft/expr.log.and>

C++: встроенное логическое ИЛИ

<https://eel.is/c++draft/expr.log.or>

Тексты, учебные данные, реплики, программы и схемы подготовлены для уроков портала «Школы России». Числовые серии — учебные модели, не результаты проведённых классом опытов. Схемы авторские, сторонние фотографии не используются. Методические ссылки приведены для учителя; ученик получает все материалы здесь. Проверка первичных источников: 18.10.2026.