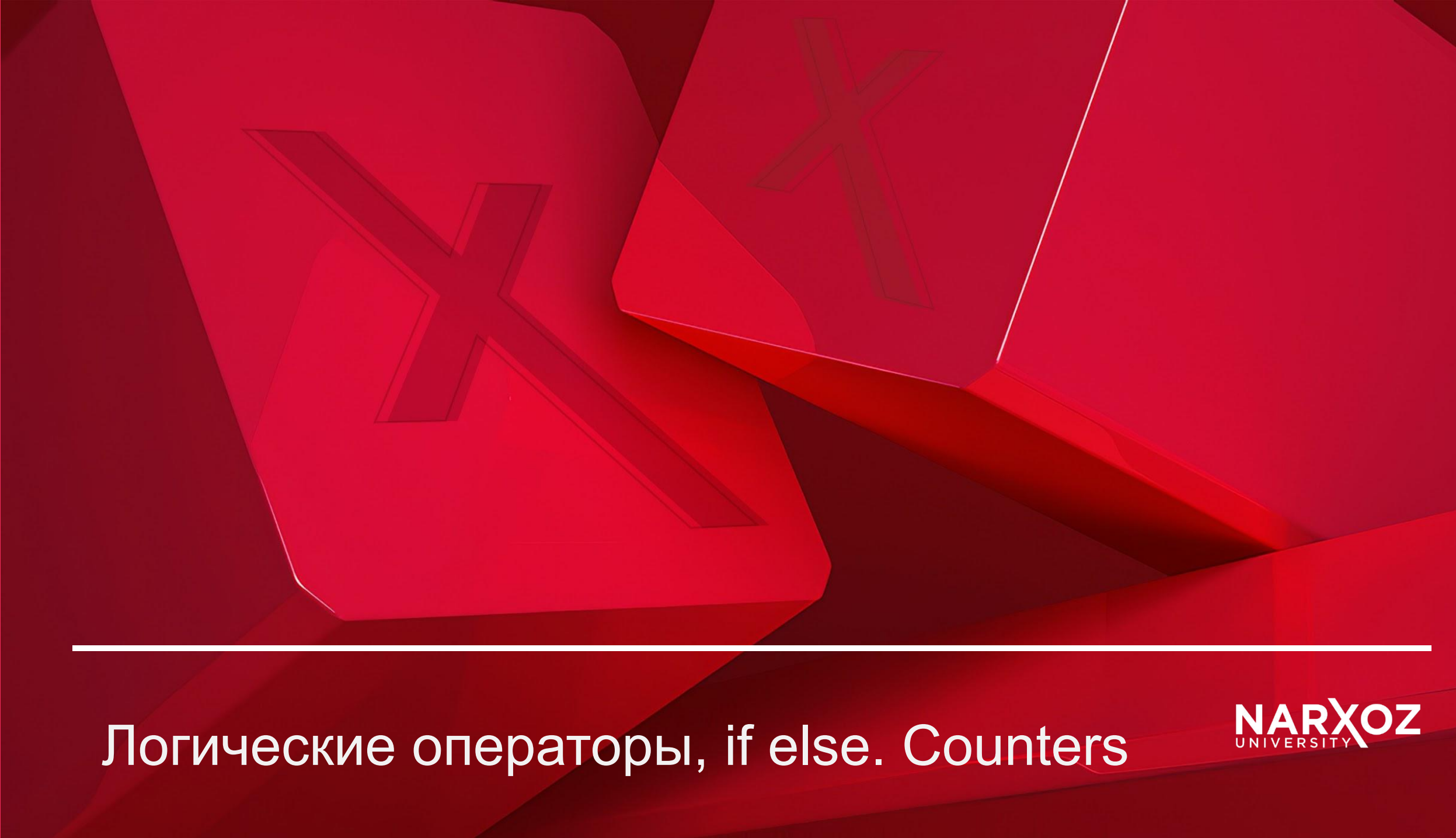

Логические операторы, if else. Counters

NARXOZ UNIVERSITY

Логические операторы

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

- Логические данные хранятся в переменных типа bool.
- Хранить они могут только два значения:
  - «Верно» — это true; значения: 1 для true;
  - «Лож» — это false; 0 для false;

Логические операторы &&, ||

Для комбинации сразу нескольких логических выражений мы должны использовать один или набор логических операторов.

1. A && B — эквивалент «И». Соответственно возвращает true, если A и B являются истиной.

2. A || B — эквивалент логического «ИЛИ». Вернет true ели хотя бы одно из выражений является истинным.

Логические операторы xor, !

3 A xor B — этот оператор можно сравнить с «ТОЛЬКО ОДИН», соответственно вернет true если A == true и B == false, или наоборот.

4 !A — данный оператор инвертирует значение A. То есть, если A == true, то он вернет false и наоборот.

Логический оператор И (&&)		
Левый операнд	**Правый операнд**	**Результат**
false	false	false
false	true	false
true	false	false
true	true	true

Логический оператор ИЛИ (||)		
Левый операнд	**Правый операнд**	**Результат**
false	false	false
false	true	true
true	false	true
true	true	true

Побитовое исключающее ИЛИ (XOR)		
Левый операнд	**Правый операнд**	**Результат**
false	false	false
false	true	true
true	false	true
true	true	false

Логический оператор НЕ (!)	
Операнд	**Результат**
true	false
false	true

Логические операторы &&, ||, xor, !
Примеры использования:

```cpp
#include <iostream>
using namespace std;

int main() {
    cout.setf(ios::boolalpha);

    bool r; // создаем переменную bool типа
    int a = 10, b = 7; // а также две переменные типа int

    r = (a < b) && (b == 7); // r равно false, поскольку a > b
    cout << "r = " << r << endl; // вывод результата

    r = a < b || b == 7; // r равен true
    cout << "r = " << r << endl; // вывод результата

    r = (a < b) xor (b == 7); // r равен true, поскольку только b == 7 верно
    cout << "r = " << r << endl; // вывод результата

    r = !(a == 10 && (b <= 8 || true)); // комбинируем целую кучу операторов
    cout << "r = " << r << endl; // и снова выводим результат

    return 0;
}
```

If else операторы условий

Операторы условий позволяют обработать несколько возможных сценариев построения печатной формы документа. В процессе формирования шаблона система проверяет заданные условия и на основе полученного результата выполняет подходящий фрагмент кода.
Условная конструкция в C++ всегда записывается в круглых скобках после оператора if . Внутри фигурных скобок указывается тело условия. Если условие выполнится, то начнется выполнение всех команд, которые находятся между фигурными скобками.
Как это работает: Когда оператор if-else исполняется, условие проверяется, и если оно возвращает True , тогда инструкции в блоке if исполняются. Но если возвращается False , тогда исполняются инструкции из блока else.

Встречаются ситуации, когда программе нужно выбрать, какую операцию ей выполнить, в зависимости от определенного условия.

К примеру, мы вводим с клавиатуры целое число. Если это число больше десяти, то программа должна выполнить одно действие, иначе — другое. Реализуем этот алгоритм на C++:

```
4  #include <iostream>
5  using namespace std;
6  int main() {
7      double num;
8      cout << "Введите произвольное число: ";
9      cin >> num;
10     if (num < 10) { // Если введенное число меньше 10.
11         cout << "Это число меньше 10." << endl;
12 } else { // иначе
13         cout << "Это число больше либо равно 10." << endl; }
14     return 0;
15 }
```

Если вы запустите эту программу, то при вводе числа, меньшего десяти, будет выводиться соответствующее сообщение.
Если введенное число окажется большим, либо равным десяти — отобразится другое сообщение.

Counters

В c++ есть так называемые счётчики или циклы "for", "while" и "do while". Для чего они нужны? Они нужны для того, чтобы выполнять некоторую часть кода по несколько раз.

Цикл For:
Этот цикл мы используем, когда мы знаем точное количество действий, которое должен выполнить код.

Скелет этого цикла такой:
for (объявление переменных; условие; инкремент/декремент счетчика) тело цикла;

Переменные, объявленные внутри цикла существуют только внутри цикла и использовать их вне этого скелета невозможно.

Цикл for в C++ выполняется в 3 шага:

Шаг №1: Объявление переменных. Как правило, здесь выполняется определение и инициализация счетчиков цикла, а точнее — одного счетчика цикла. Эта часть выполняется только один раз, когда цикл выполняется впервые.
Шаг №2: Условие. Если оно равно false, то цикл немедленно завершает свое выполнение. Если же условие равно true, то выполняется тело цикла.
Шаг №3: Инкремент/декремент счетчика цикла. Переменная увеличивается или уменьшается на единицу. После этого цикл возвращается к шагу №2.

Пример кода

Напишем программу, которая будет считать сумму всех чисел от 1 до 1000.

```
#include <iostream>
using namespace std;
int main() {
    int i; // счетчик цикл
    int sum = 0; // сумма чисел от 1 до 1000.
    for (i = 1; i <= 1000; i++) // задаем начальное значение 1, конечное 1000 и задаем шаг цикла - 1.
    {
    sum = sum + i;
    }
    cout << "Сумма чисел от 1 до 1000 = " << sum << endl;
    return 0;
}
```

Если мы скомпилируем этот код и запустим программу, то она покажет нам ответ: 500500. Это и есть сумма всех целых чисел от 1 до 1000.

Цикл while

Когда мы не знаем, сколько итераций должен произвести цикл, нам понадобится цикл while или do while. Синтаксис цикла while в C++ выглядит следующим образом.

```
while (Условие) {
    Тело цикла;
}
```

Данный цикл будет выполняться, пока условие, указанное в круглых скобках является истиной. Решим ту же задачу с помощью цикла while. Хотя здесь мы точно знаем, сколько итераций должен выполнить цикл, очень часто бывают ситуации, когда это значение неизвестно.

Ту же самую задачу мы можем решить и с этим циклом:

```cpp
#include <iostream>
using namespace std;
int main() {
int sum = 0; // инициализируем счетчик суммы.
while (i < 1000) {
    i++;
    sum += i;
}
cout << "Сумма чисел от 1 до 1000 = " << sum << endl;
return 0;
}
```

Цикл do while

Цикл do while очень похож на цикл while. Единственное их различие в том, что при выполнении цикла do while один проход цикла будет выполнен независимо от условия.

То есть, если условие неверно изначально, то цикл while выполняться не будет, а do while - выполнится один раз.