



Модуль: Условные операторы на C++.

Темы:

Условный оператор. Простые и составные условия. Вложенные условия. Оператор выбора switch



Разветвляющийся алгоритм

Разветвляющийся алгоритм – алгоритм, содержащий хотя бы одно **условие**, в результате проверки которого может осуществляться **разделение** на несколько **параллельных ветвей** алгоритма.



Условие – логическое выражение, принимающее одно из двух значений: «истина» или «ложь».

Пример из жизни:

Если на улице дождь — взять зонт, **иначе** — идти без него.

В языке C++ условные операторы позволяют управлять потоком исполнения программы, принимая решения, что и когда выполнять.

Условный оператор if

Условный оператор if – оператор, реализующий разветвляющийся алгоритм.

Полный условный оператор позволяет выполнить **один** из двух блоков кода: если условие истинно — выполняется **первая ветка**, иначе — **вторая**.

```
// Полный условный оператор
int age = 16;

if (age >= 18) {
    cout << "Доступ разрешён" << endl;
} else {
    cout << "Доступ запрещён" << endl;
}

// Неполный условный оператор
int x = 10;

if (x > 5) {
    cout << "x больше 5" << endl;
}
```

Неполный условный оператор - конструкция, которая выполняет блок кода **только** если **условие истинно**. Если условие **ложно** — программа **просто** продолжает выполнение.

Условия if-else

```
if (x > 0) {  
    cout << "Число положительное";  
} else if (x < 0) {  
    cout << "Число отрицательное";  
} else {  
    cout << "Ноль";  
}
```

Позволяют выполнять разные действия в зависимости от условий.

- **if проверяет условие** и выполняет код, если оно истинно.
- **else if** — дополнительная проверка, если **первое условие ложно**.
- **else** выполняется, если **все предыдущие условия ложны**.

Составное условие

```
int age = 20;  
if (age >= 18 && age <= 30) {  
    cout << "Молодой взрослый";  
}
```

- Логическое выражение, которое объединяет два или более простых условий с помощью **логических операторов (&&, ||, !)**.
- Условие сработает **только**, если возраст *и больше или равен 18, и меньше или равен 30*.

! – «не» (операция отрицания, not)

&& - «и» (логическое умножение, конъюнкция, and)

|| - «или» (логическое сложение, дизъюнкция, or)

Логические выражения

Условия в C++ основываются на **логических выражениях**, результатом которых является **булевское значение**: true (истина) или false (ложь).

Для построения условий используются:

Оператор	Название	Пример
==	равно	a == b
!=	не равно	a != b
<	меньше	a < b
>	больше	a > b
<=	меньше или равно	a <= b
>=	больше или равно	a >= b

Вложенные условия

Условие **внутри другого** условия используется, когда выбор зависит от **нескольких уровней проверки**.

```
int score = 85;
if (score >= 50) {
    if (score >= 80) {
        cout << "Отлично!";
    } else {
        cout << "Пройдено";
    }
} else {
    cout << "Провалено";
}
```

Сначала проверяется, сдал ли человек ($\text{score} \geq 50$), потом — насколько хорошо.

Если $\text{score} \geq 80$, то человек сдал на отлично

Иначе, то человек сдал на удовлетворительно

Тернарный оператор

Тернарный оператор в некотором роде похож на конструкцию if-else. Он принимает **три операнда** в следующем виде:

```
операнд1? операнд2 : операнд3
```



- **Первый** операнд представляет **условие**.
- Если это условие **верно** (равно true), тогда выбирается/выполняется **второй операнд**, который помещается после **символа ?**.
- Если условие **не верно**, тогда выбирается/выполняется **третий операнд**, который помещается **после двоеточия**.

Пример на C++

- Здесь **первым операндом** тернарного оператора является условие **$a > b$** .
- **Если** это условие **верно**, то возвращается **второй операнд** - результат выражения **$a - b$** .

```
#include <iostream>
using namespace std;
int main()
{
    int a = 5;
    int b = 8;
    int c = a > b ? a - b : a + b;

    cout << "c = " << c << endl;    // c = 13
}
```

- Если условие не верно, то возвращается третий операнд - **$a + b$** .
- И возвращенный операнд присваивается переменной **c**.

Оператор switch

Оператор **switch** используется, чтобы выполнить **один из нескольких блоков** кода в зависимости от значения **одной переменной**. Это альтернатива множественным if...else.

```
switch(выражение)
{
    case значение_1: инструкции_1;
    case значение_2: инструкции_2;
    .....
    case значение_N: инструкции_N;

    default: инструкции;
}
```

- Внутри оператора **switch** в скобках пишется сравниваемое выражение.
- Значение этого выражения **последовательно** сравнивается со значениями операторов **case**.
- И если **совпадение будет найдено**, то будет выполняться **инструкция** этого блока **case**.
- В конце конструкции switch может стоять блок **default**. Он необязателен и **выполняется** в том случае, если **значение после switch не соответствует** ни одному из операторов case.

Пример на C++

```
int day = 3;
switch (day) {
    case 1: cout << "Понедельник"; break;
    case 2: cout << "Вторник"; break;
    case 3: cout << "Среда"; break;
    default: cout << "Неизвестный день";
}
```

- **switch** сравнивает **одно значение** с разными вариантами (case)
- **break** нужен, чтобы выйти из switch **после выполнения** нужного блока
- **default** — как else, срабатывает, если **ни один** case не подошёл
- Работает только с **целочисленными** типами (int, char, enum)

Switch не подходит для:

- Проверки диапазонов ($x > 5$)
- Условий с логикой ($\&\&$, \parallel)