

Условия и циклы — конспект темы

Условный оператор if

Условный оператор `if` показывает, что команда выполнится при соблюдении некоторого условия или не выполнится, если условие не соблюдено:

```
int a, b;
cin >> a >> b;
if (a == b)
    cout << "equal"s << endl;
```

Ветка `else` указывает на команду, которая будет выполнена, если условие не соблюдается. Она всегда в паре с оператором `if`:

```
int a, b;
cin >> a >> b;
if (a == b)
    cout << "equal"s << endl;
else
    cout << "not equal"s << endl;
```

Когда при выполнении условия после `if` или `else` должно быть несколько действий, эти действия нужно заключить внутри фигурных скобок:

```
int a, b;
cin >> a >> b;
if (a == b) {
    // Фигурные скобки нужны, когда надо выполнить несколько команд
    cout << "equal"s << endl;
    cout << a << endl;
} else {
    cout << "not equal"s << endl;
    cout << a << " " << b << endl;
}
```

В C++ отступы не определяют вложенность команд.

```
int a = -1;
if (a >= 0)
    if (a > 0)
        cout << "positive"s << endl;
else
    cout << "negative"s << endl;
```

Когда условий несколько, их можно соединить, не используя фигурные скобки. Для этого нужна конструкция `else if`:

```
if ( <условие 1> ) {
    // ...
} else if ( <условие 2> ) {
    // ...
} else {
    // ...
}
```

Логические операции: сравнение

Результат операции сравнения — логическое выражение. Логическими называют выражения, результат которых равен `true` или `false`.

Операторы сравнения целых чисел:

- `==` (равно),
- `!=` (неравно),
- `<` (меньше),
- `<=` (меньше или равно),
- `>` (больше),
- `>=` (больше или равно).

Целые числа можно сравнивать с вещественными. Перед сравнением целое число неявно будет преобразовано в соответствующее ему вещественное:

```
int x = 1;
double y = 1.2;
x < 1;
```

```
y >= 2.5;
x == y;
```

Строки сравниваются лексикографически — строка `a` меньше строки `b`, потому что буква “f” стоит в английском алфавите раньше буквы “w”:

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

int main() {
    string a = "fire"s;
    string b = "water"s;
    if (a < b) {
        cout << "a is less than b"s << endl;
    }
}
```

На экране появится сообщение “`a` is less than `b`”.

Регистр символов имеет значение: например, строки “apple” и “APPLE” — разные.

Результат сравнения и других логических операций относится к типу данных `bool`.

У переменных типа `bool` два возможных значения: `true` и `false`.

При выводе булевы значения выводятся цифрами:

```
cout << true << endl // 1
      << false << endl; // 0
```

Если хотите вывести значение словами, включите режим `boolalpha`. Чтобы отключить `boolalpha`, напишите `noboolalpha`:

```
cout << boolalpha
      << true << endl // true
      << false << endl // false
      << noboolalpha
      << true << endl // 1
      << false << endl; // 0
```

Тройных сравнений в C++ не бывает.

Логические операции «и», «или», «не»

- `&&` (конъюнкция) — логическая операция «и». Выражение `a && b` возвращает `true`, если истинно и `a`, и `b`. В противном случае вернёт `false`.
- `||` (дизъюнкция) — логическая операция «или». Выражение `a || b` возвращает `true`, если истинно хотя бы одно из `a` и `b`. В противном случае вернёт `false`.
- `!` (отрицание) — логическая операция «не». Выражение `!a` возвращает `false`, если `a` истинно. В противном случае вернёт `true`.

Истинность `&&`

<code>a</code>	<code>b</code>	<code>a && b</code>
false	false	false
false	true	false
true	false	false
true	true	true

Истинность `||`

<code>a</code>	<code>b</code>	<code>a b</code>
false	false	false
false	true	true
true	false	true
true	true	true

Истинность `!`

<code>a</code>	<code>!a</code>
false	true
true	false

Операции `&&` и `||` вычисляются слева направо. Когда по значению левого аргумента можно определить значение всего выражения, правый аргумент не вычисляется.

Циклы while и do-while

Чтобы программа выполняла действие, пока есть некоторое условие, используют цикл `while`:

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

int main() {
    int n;
    cin >> n;
    int sum = 0;
    int i = 1;
    while (i <= n) {
        sum += i;
        ++i;
    }
    cout << sum << endl;
}
```

В цикле `while` условие продолжения цикла проверяется в самом начале, поэтому тело цикла может не выполниться ни разу, если условие будет изначально ложным.

В цикле `do-while` тело цикла выполняется хотя бы один раз, так как условие продолжения цикла выполняется в конце.

Цикл for

Цикл `for` используется, когда вы точно знаете, сколько раз действие должно повториться.

```
for (int i = 0; i != 3; i += 1) {
    cout << "Check the fridge"s << endl;
}
```

В записи шага цикла часто используют короткие варианты для `i += 1`:

- `++i` — префиксный инкремент,
- `i++` — постфиксный инкремент.

Выход из цикла

Чтобы программа перешла на следующий шаг цикла, применяют оператор

`continue`:

```
string str = "Drawing indices for fun and profit"s;
int num_iters = 0;
// считаем с клавиатуры, сколько раз мы бы хотели повторить цикл
cin >> num_iters;

for (int i = 0; i < num_iters; ++i) {
    int index1, index2;
    // считаем индексы
    cin >> index1 >> index2;

    // если index1 отрицательный или больше, чем длина нашей строки,
    // то продолжать этот шаг цикла невозможно
    if (index1 < 0 || index1 >= str.size()) {
        // скомандовав continue, программист просит перейти на следующую итерацию,
        // не заканчивая текущую
        continue;
    }

    // аналогичная логика для второго индекса
    if (index2 < 0 || index2 >= str.size()) {
        continue;
    }
    // выведем результат сравнения символов строки по указанным индексам
    cout << (str[index1] == str[index2]) << endl;
}
```

Чтобы выйти из цикла досрочно, используют оператор `break`:

```
#include <iostream>
#include <string>

using namespace std;

int main() {
    string animal;
    // считаем название животного
    cin >> animal;

    for (int i = 0; i < animal.size(); ++i) {
        // если текущая буква строки - а,
        if (animal[i] == 'a') {
            // то выведем индекс i на экран и закончим цикл
            cout << i << endl;
            break;
        }
    }
}
```

```
    cout << "Yes!"s << endl;  
}
```