

# **Операции отношения и логические операции**

# Операторы ветвления

В структурном программировании используется три основных конструкции:

- Следование (линейный алгоритм)
- Ветвление (разветвляющийся алгоритм)
- Цикл (циклический алгоритм)

Как и во многих других языках программирования, для реализации ветвления в Си используется две основных конструкции:

оператор выбора **switch**  
условный оператор **if – else**

1. Оператор выбора **switch** позволяет выбрать один из нескольких заранее определённых вариантов. В этом случае у каждого варианта должна иметься своя отличительная черта, по которой можно один вариант отличить от другого.
2. Условный оператор **if – else** позволяет выполнять те или иные команды в зависимости от выполнения некоторого условия, которое программист может задать самостоятельно.

# Операции отношения и логические операции

- Условный оператор выполняет те или иные действия в зависимости от выполнения условия, заданного программистом.
- Компьютер не очень-то сообразительный. В данном случае это проявляется в том, что он понимает только простые условия на да/нет.

- Самое простое, что умеет делать компьютер, – это сравнивать между собой числа и значения числовых выражений. Для этого предназначены **операции отношения**.

>	- больше
>=	- больше либо равно
<	- меньше
<=	- меньше либо равно
==	- равно
!=	- не равно

```
#include <stdio.h>

int main(void) {

    int a = 10, b = 5, c = 2;
    char ch = 'a';

    printf("%d\n", b > c );
    printf("%d\n", a == b*c );
    printf("%d\n", a != b*c );
    printf("%d\n", a < c );
    printf("%d\n", b >= c + 3 );
    printf("%d\n", -1 <= 2 );
    printf("%d\n", ch == 'a' );
    printf("%d\n", ch != 'a' );

    return 0;
}
```

Вы ещё помните, что компьютер понимает только нули и единицы?

Правило!

нуль это ложь, а единица – истина.

Любое число, кроме нуля, интерпретируется как истина.

На операцию сравнения обратите особое внимание.

Т.к. один знак равно = используется для оператора присваивания, то для операции сравнения используется два знака равенства ==.

Будьте внимательны и не забывайте об этом. Это частая ошибка, которую совершают начинающие программисты.



Кроме операций отношения есть ещё и логические операции. Они используются для того, чтобы из простых условий создавать сложные.

Уже немного испугались? На самом деле ничего сложного.

! - логическое НЕ.

&& - логическое И.

|| - логическое ИЛИ.

Оператор логическое **НЕ !(условие)** отрицает выражение, к которому он был применён. Если условие было истиной, оно становится ложью, а если было ложью, то становится истиной. Вот пример:

```
#include <stdio.h>

int main(void) {

    int a = 10, b = 5, c = 2;

    printf("%d\n", a == b*c );
    printf("%d\n", !(a == b*c) );
    printf("%d\n", a != b*c );
    printf("%d\n", !(a != b*c) );

    return 0;
}
```

**(условие1) && (условие2)** - оператор логическое И. Возвращает истину, если оба условия истинны. Иначе возвращает ЛОЖЬ.

```
#include <stdio.h>

int main(void) {

    printf("%d\n", 1 && 1 );
    printf("%d\n", 1 && 0 );
    printf("%d\n", 0 && 1 );
    printf("%d\n", 0 && 0 );

    return 0;
}
```

(условие1) || (условие2) - оператор логическое ИЛИ.  
Возвращает истину, если хотя бы одно из условий истина или оба условия сразу истинны. Иначе возвращает ложь.

```
#include <stdio.h>

int main(void) {

    printf("%d\n", 1 || 1 );
    printf("%d\n", 1 || 0 );
    printf("%d\n", 0 || 1 );
    printf("%d\n", 0 || 0 );

    return 0;
}
```

# Приоритет логических операций

Для логических операций тоже есть определённая очерёдность выполнения.

1. Операции в скобках
2. Логическая операция НЕ
3. Логическая операция И
4. Логическая операция ИЛИ

Включим логические операции в общий список приоритетов операций, который мы составляли для арифметических операций и математических функций.

- Операции в скобках: ()
- Вычисляются функции (например, `sqrt()`, `cos()` и др.)
- Логическая операция НЕ (!)
- Умножение, деление, остаток от деления (слева направо) (\*,/,%)
- Сложение, вычитание (+,-)
- Логическая операция И (&&)
- Логическая операция ИЛИ (||)
- Выполняется присваивание (=)

# Отметьте все логические выражения, значения которых **ИСТИНА**.

Проверка.

```
int main(int argc, char *argv[]) {  
    printf("%d", 1==1);  
  
    return 0;  
}
```

•  $8 > 3$

•  $1 == 1$

~~•  $1 < 1$~~

•  $8 \% 2 != 4$

•  $-1$

•  $1$

•  $2 < 33$

Отметьте все логические выражения,  
значения которых **ЛОЖЬ**.

- ~~• 1~~
- $2 > 3$
- $10 \neq 10$
- $1 == 0$
- ~~•  $8 \neq 2 = 0$~~
- 0
- ~~•  $2 > 3$~~



# Отметьте все логические выражения, значения которых **ИСТИНА**.

## Проверка

```
int main(int argc, char *argv[]) {  
    printf("%d", 1 && 0);  
  
    return 0;  
}
```

• 1 && 0

• 0 && 0

• 1 || 0

• 0 || 1

• 1 && 1

• 0 && 1

• 0 || 0

• 1 || 1

Отметьте все логические выражения,  
значения которых **ЛОЖЬ**.

0 || 0

0 || 1

1 || 0

1 || 1

Дано следующее логическое  
отношение:

$!(a \& \& b) \mid \mid (!a \& \& b)$

Выберите вариант(ы), в котором  
(ых) значения  
переменных **a** и **b** такие, что  
значение данного логического  
выражения **истина**.

$a = 0, b = -2$

$a = 1, b = 0$

$a = 0, b = 1$

$a = 0, b = 0$

$a = 1, b = 1$

# Проверим

```
int main(int argc, char *argv[]) {  
  
    int a = 0;  
    int b = 0;  
  
    printf("%d", !(a&&b) || (!a&&b));  
  
    return 0;  
}
```

Дано следующее логическое  
отношение:

$!(a \& \& b) \mid \mid (!a \& \& b)$

Выберите вариант(ы), в котором  
(ых) значения  
переменных **a** и **b** такие, что  
значение данного логического  
выражения **истина**.

$a = 0, b = -2$

$a = 1, b = 0$

$a = 0, b = 1$

$a = 0, b = 0$

$a = 1, b = 1$

# Сравнение чисел

Напишите программу сравнивающе две целых числа.

**Входные данные:**

Два целых числа  $x$ ,  $y$

**Выходные данные:**

1 -- если  $x = y$

0 -- если  $x \neq y$

```
#include <stdio.h>
```

```
int main() {
```

```
    printf("%d \n", 2== -2);
```

```
    return 0;
```

```
}
```