

Лекция 4. Операторы и выражения Арифметические, логические, побитовые операторы. Приоритет операций, приведение типов.

Операторы и выражения

1. Введение

Каждая программа на C++ состоит из **инструкций**, которые выполняют определённые действия.

Основным инструментом для описания этих действий являются **операторы**.

Оператор — это символ или комбинация символов, выполняющая операцию над одним или несколькими операндами (значениями, переменными, выражениями).

Выражение (expression) — это комбинация операторов, переменных, констант и функций, которая возвращает некоторое значение.

Примеры выражений:

```
a + b * c
x = y - 2
(a > b) && (b > 0)
```

Операторы — это "грамматика" языка C++, через которую описываются вычисления, условия и логика работы программы.

2. Классификация операторов в C++

Категория операторов	Примеры
Арифметические	+, -, *, /, %
Инкремент/декремент	++, --
Логические	&&, ^
Побитовые	&, ^
Операторы сравнения	==, !=, <, >, <=, >=
Присваивания	=, +=, -=, *=, /=, %=
Тернарный (условный) ? :	
Преобразования типов	(int), static_cast<double>(x)
Прочие	sizeof, ,, typeid, new, delete

3. Арифметические операторы

Эти операторы выполняют стандартные математические действия над числами.

Оператор	Назначение	Пример	Результат
+	Сложение	5 + 3	8
-	Вычитание	10 - 4	6
*	Умножение	2 * 6	12
/	Деление	10 / 2	5
%	Остаток от деления (для целых)	10 % 3	1

⚠ При делении целых чисел результат всегда будет целым:

$$7 / 2 = 3.$$

Чтобы получить дробный результат, хотя бы один операнд должен быть вещественным:

$$7.0 / 2 = 3.5.$$

3.1. Инкремент и декремент

- ++ — увеличивает значение на 1;
- -- — уменьшает значение на 1.

Существуют две формы:

- **Префиксная:** ++x — сначала увеличивает, потом используется.
- **Постфиксная:** x++ — сначала используется, потом увеличивает.

Пример:

```
int a = 5;
cout << ++a; // 6
cout << a++; // 6 (потом a становится 7)
```

4. Логические операторы

Используются для логических выражений (возвращают true или false).

Оператор	Описание	Пример	Результат
&&	И (AND)	(x > 0 && y > 0)	true, если оба истинны
	ИЛИ (OR)		

Оператор	Описание	Пример	Результат
!	НЕ (NOT)	!true	false

Пример:

```
int a = 10, b = 5;
bool result = (a > b && b > 0); // true
```

💡 Особенность:

C++ использует *короткое замыкание (short-circuit)*:

- в выражении A && B, если A == false, B не вычисляется;
- в A || B, если A == true, B не вычисляется.

5. Побитовые операторы

Побитовые операции выполняются **над отдельными битами** числовых типов (int, char, long и т.д.).

Используются в системном программировании, обработке флагов, шифровании и т.п.

Оператор	Назначение	Пример (a=5, b=3)	Бинарная операция	Результат
&	Побитовое И	a & b	0101 & 0011	0001 (1)
	Побитовое ИЛИ	a b	0101 0011	0111 (7)
^	Исключающее ИЛИ (XOR)	a ^ b	0101 ^ 0011	0110 (6)
~	Побитовое НЕ (инверсия)	~a	~0101	инверсия всех бит 1010 (10)
<<	Сдвиг влево	a << 1	0101 → 1010	10
>>	Сдвиг вправо	a >> 1	0101 → 0010	2

Пример:

```
int a = 5, b = 3;
cout << (a & b) << endl; // 1
cout << (a | b) << endl; // 7
cout << (a ^ b) << endl; // 6
cout << (a << 1) << endl; // 10
```

💡 Применение:

- Работа с битовыми флагами;
- Управление портами ввода/вывода;
- Оптимизация хранения данных.

6. Операторы сравнения

Операторы сравнения возвращают логическое значение `true` или `false`.

Оператор	Значение	Пример	Результат
<code>==</code>	равно	<code>5 == 5</code>	<code>true</code>
<code>!=</code>	не равно	<code>5 != 3</code>	<code>true</code>
<code><</code>	меньше	<code>2 < 4</code>	<code>true</code>
<code>></code>	больше	<code>6 > 3</code>	<code>true</code>
<code><=</code>	меньше или равно	<code>5 <= 5</code>	<code>true</code>
<code>>=</code>	больше или равно	<code>5 >= 2</code>	<code>true</code>

7. Операторы присваивания

Используются для сохранения значений в переменных.

Оператор	Эквивалент	Пример
<code>=</code>	<code>a = b</code>	<code>a = 5;</code>
<code>+=</code>	<code>a = a + b</code>	<code>a += 2;</code>
<code>-=</code>	<code>a = a - b</code>	<code>a -= 1;</code>
<code>*=</code>	<code>a = a * b</code>	<code>a *= 3;</code>
<code>/=</code>	<code>a = a / b</code>	<code>a /= 2;</code>
<code>%=</code>	<code>a = a % b</code>	<code>a %= 2;</code>

Пример:

```
int x = 10;
x += 5; // x = 15
x *= 2; // x = 30
```

8. Приоритет операторов и порядок вычислений

Когда в выражении несколько операторов, C++ использует **приоритет** и **ассоциативность** для определения порядка.

Приоритет	Операторы	Направление
1 (высокий)	(), [], ++, --	слева направо
2	*, /, %	слева направо
3	+, -	слева направо
4	<, >, <=, >=	слева направо
5	==, !=	слева направо
6	&&	слева направо
7	`	
8	=, +=, -=, /=, %=	справа налево

Пример:

```
int x = 2 + 3 * 4; // сначала 3*4=12, потом 2+12=14
int y = (2 + 3) * 4; // 5*4=20
```

9. Приведение типов (Type Conversion)

В C++ типы данных могут преобразовываться **неявно** или **явно**.

9.1. Неявное преобразование (implicit cast)

Происходит автоматически, если это безопасно:

```
int a = 5;
double b = a; // int → double (автоматически)
```

9.2. Явное преобразование (explicit cast)

Программист явно указывает, в какой тип преобразовать:

```
int a = 7, b = 2;
double result = (double)a / b; // 3.5
```

Современный безопасный вариант:

```
double result = static_cast<double>(a) / b;
```

10. Примеры

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

int main() {
```

```

    int a = 7, b = 3;

    cout << "Арифметика: a+b=" << a+b << ", a*b=" <<
a*b << endl;
    cout << "Сравнение: a>b -> " << (a > b) << endl;
    cout << "Логика: (a>0 && b>0) -> " << (a > 0 && b >
0) << endl;
    cout << "Побитовые: a|b -> " << (a | b) << endl;

    double c = static_cast<double>(a) / b;
    cout << "Приведение типов: a/b = " << c << endl;

    return 0;
}

```

Вывод:

```

Арифметика: a+b=10, a*b=21
Сравнение: a>b -> 1
Логика: (a>0 && b>0) -> 1
Побитовые: a|b -> 7
Приведение типов: a/b = 2.33333

```

11. Заключение

- Операторы — основа всех выражений в C++.
 - Знание приоритета операций и типов данных необходимо для корректных вычислений.
 - Логические и побитовые операторы позволяют эффективно управлять логикой и данными на уровне битов.
 - Приведение типов помогает контролировать точность вычислений и предотвращать ошибки.
-

12. Вопросы для самопроверки

1. Что такое оператор и выражение?
2. Назовите основные арифметические операторы в C++.
3. Чем отличаются префиксная и постфиксная формы ++?
4. Каковы особенности логических операторов &&, || и !?
5. Что делает оператор % и для каких типов он применим?
6. Что выполняют побитовые операторы &, |, ^?
7. Как влияет приоритет операторов на порядок вычислений?
8. В чём разница между неявным и явным приведением типов?

9. Как безопасно привести тип `int` к `double` в современном C++?
10. Какие ошибки могут возникнуть при делении целых чисел?