C++. Операторы и выражения

ОПЕРАТОРЫ И ОПЕРАНДЫ • ВЫРАЖЕНИЯ • АРИФМЕТИЧЕСКИЕ И ЛОГИЧЕСКИЕ ОПЕРАТОРЫ • ОПЕРАТОРЫ ПРИСВАИВАНИЯ И СРАВНЕНИЯ

Операторы и операнды

Оператор

Оператор - конструкция в языках программирования, аналогичная по записи математическим операциям, то есть специальный способ записи некоторых действий.

Операнд

Операнд - данные, которые обрабатываются оператором.

Выражение

Выражение - комбинация *переменных*, *констант* и *операций*, приводящих к вычислению некого конечного значения.

Оператор присваивания

ПРАВИЛА ПРИСВАИВАНИЯ В C++

Оператор присваивания

Оператор присваивания записывается символом **=** (равно) и необходим для инициализации переменных новыми значениями.

Правила присваивания

Результат вычисления выражения, стоящего *справа* от знака присваивания *возвращается* переменной / оператору, стоящему *слева* от знака присваивания.

Пример работы оператора присваивания

```
int a = 10;
int b = 20;
int c = a + b;  // c = 30
```

Арифметические операторы

СЛОЖЕНИЕ • ВЫЧИТАНИЕ • УМНОЖЕНИЕ • ДЕЛЕНИЕ • ОСТАТОК ОТ ДЕЛЕНИЯ • ИНКРЕМЕНТ • ДЕКРЕМЕНТ

Стандартные арифметические операторы

C++ поддерживает 4 базовых арифметических операции, известных всем с самого раннего школьного возраста:

1. Сложение (+)
2. Вычитание (-)
3. Умножение (*)
4. Деление (/)

Арифметические операторы в действии

```
int    a = 5 + 5;     // 10
int    b = a - 15;    // -5
float  c = 5 * 3.14; // 15.7
float  d = c / 7.62; // 2.060...
```

Круглые скобки

Для управления *приоритетом* выполнения операций или имитации вычисления дробей в C++ существуют оператор **круглые скобки**. Их действие аналогично действию в линейной алгебре.

Круглые скобки в действии

```
int (a + b) * (c + d);
/*
  Сначала выполнится сложение
  в скобках, а потом умножение
*/
```

Оператор остатка от деления

Операция **остатка от деления** (%) применяется только к *целым числам* типа char, short, int и long. Результатом этой операции является *остаток*, получаемый при делении её левого операнда на правый.

Операция остатка от деления в действии

```
6 % 8;   // 6
7 % 8;   // 7
8 % 8;   // 0
9 % 8;   // 1
10 % 8;  // 2
```

Инкремент и декремент

Инкремент (++) - унарная операция, увеличивающая значение операнда на 1.

Декремент (--) - унарная операция, уменьшающая значения операнда на 1.

Префиксные и постфиксные формы записи инкремента / декремента

- Префиксная форма записи - операция перед операндом: сначала выполняется изменение значения операнда, а потом все остальные операции в выражении;
- Постфиксная форма записи - операция после операнда; сначала выполняются все операции в выражении, после чего операнды изменяют своё значение.

Комбинированные операторы присваивания

Расширенные операторы присваивания

Иногда очень удобно комбинировать арифметические операторы с оператором присваивания.

- Увеличить (+=)
- Уменьшить (-=)
- Умножить (*=)
- Разделить (/=)
- Остаток от деления (%=)

Пример комбинированного оператора присваивания

```
int a = 10;
    a += 15; // a = a + 15;
```

Использование комбинированного оператора

Комбинированные операторы можно использовать только с инициализированными заранее переменными.

Операторы сравнения

СРАВНЕНИЕ • НЕРАВЕНСТВО • БОЛЬШЕ ЧЕМ • МЕНЬШЕ ЧЕМ • БОЛЬШЕ ЧЕМ ИЛИ РАВНО • МЕНЬШЕ ЧЕМ ИЛИ РАВНО

Зачем нужны операторы сравнения?

Операторы сравнения сравнивают между собой значения двух операндов. Результатом сравнения является значение истина (true) или ложь (false).

Операторы сравнения

- Равно (==)
- Неравно (!=)
- Больше чем (>)
- Меньше чем (<)
- Больше чем или равно (>=)
- Меньше чем или равно (<=)

Примеры операторов сравнения (без ответов)

```
5 == 5;
5 == 10;
5 != 5;
5 != 10;
5 >  5;
5 >  10;
```

```
5 >= 5;
5 >= 10;
5 <  5;
5 <  10;
5 <= 5;
5 <= 10;
```

Примеры операторов сравнения (с ответами)

```
5 == 5;  /* true  */ 5 >= 5;  /* true  */
5 == 10; /* false */ 5 >= 10; /* false */
5 != 5;  /* false */ 5 < 5;   /* false */
5 != 10; /* true  */ 5 < 10;  /* true  */
5 >  5;  /* false */ 5 <= 5;  /* true  */
5 >  10; /* false */ 5 <= 10; /* true  */
```

Логические операторы

И • ИЛИ • ИСКЛЮЧАЮЩЕЕ ИЛИ • НЕ

Зачем нужны логические операции?

Логические операции составляют основной инструмент для построения булевой логики. Логические операции позволяют производить действия над булевыми переменными, то есть переменными, принимающими только два значения - *истина* и *ложь*.

Логические операторы

- Логическое И (`&&`)
- Логическое ИЛИ (`||`)
- Исключающее ИЛИ (`xor`)
- Логическое НЕ (`!`)

Логические И

Логическое И возвращает *истину* только в том случае, если оба его операнда - *истинны*.

Логические ИЛИ

Логическое ИЛИ возвращает *истину* в двух случаях:

1. Оба операнда - *истинны;*
2. Один из операндов - *истина*.

Исключающее ИЛИ

Исключающее ИЛИ возвращает *истину* **только** в том случае, если один из операндов - *истина*.

Логическое НЕ

Логическое НЕ (отрицание) - отрицает текущее состояние булевой переменной:

- **НЕ** *истина* = *ложь*
- **НЕ** *ложь* = *истина*

Таблица значения логических операторов

Значения операндов		Результат операции		
X	Y	not X	X and Y	X or Y
False	False	True	False	False
False	True	True	False	True
True	False	False	False	True
True	True	False	True	True

Приоритеты операторов

Таблица приоритетов операций

Тип операций	Операции	Приоритет
Унарные	!, ++, --, +, -	Высший
Арифметические	Мультипликативные *, / , %	
	Аддитивные +, -	
Отношения	Неравенства <,>,<=,=>	
	Равенства ==, !=	
Логические	И &&	
	ИЛИ ||	
Условная	?:	
Присваивания	=, +=, -=, *=, /=, %=	Низший

Задача 1: цийры числа в обратной порядке

Дано целое четырёхзначное число (к примеру 9876). Необходимо написать программу, которая с помощью использования арифметических операций выведет на экран цифры этого числа в обратном порядке. То есть, по завершению работы программы на жкране должно появится число 6789.

Задача 2: депозитный калькулятор

Пользователь вводит сумму депозита и количество месяцев хранения денег в банке. Необходимо провести расчёт и вывести на экран прибыль с депозита в месяц, за весь срок депозита и общую сумму к выплате в конце срока. Процентная ставка указывается в коде программы.