Эта лекция посвящена битовым операциям. Вначале вспомним основные операции математической логики.

Первая из них — операция AND, или операция И. У нее два логических аргумента, и в C++ она обозначается амперсандом «&». Ее значение истинно, если оба аргумента истинны, и ложно во всех остальных случаях. На рис. 1 — таблица истинности для операции AND в зависимости от значений ее аргументов x и y. Если x и y — нули, т.е. ложны, значение x AND y тоже ложно, если x принимает ложное значение, а y — истинное, то x AND y ложно, и т.д.

AND		
x	y	$x \,\&\, y$
0	0	0
0	1	0
1	0	0
1	1	1

Рис. 1. Операция AND

Следующая операция — OR, то есть ИЛИ. Это тоже операция двух аргументов, в C++ она обозначается вертикальной чертой «|». Значение операции OR ложно тогда и только тогда, когда значения обоих аргументов ложны (см. рис. 2).

Операция XOR — это «исключающее ИЛИ». Она обозначается галочкой «^». Значение операции XOR истинно, если значения аргументов различны, и ложно, если значения аргументов совпадают. Таблица истинности операции XOR представлена на рис. 3.

Помимо логических выражений, эти операции можно применять к целым числам, например, типа int. В частности, справедливы следующие равенства:

$$13 \,\&\, 14 = 12,$$
$$13 \,|\, 14 = 15,$$
$$13 \,\hat{}\, 14 = 3.$$

Выясним, как получаются такие значения. Для этого нужно вспомнить, что числа в памяти компьютера представляются в двоичной системе счисления.

OR		
x	y	$x \mid y$
0	0	0
0	1	1
1	0	1
1	1	1

Рис. 2. Операция OR

XOR		
x	y	x ^ y
0	0	0
0	1	1
1	0	1
1	1	0

Рис. 3. Операция XOR

Привычная нам запись чисел — это десятичная система счисления. Например,

$$2175 = 2 \cdot 10^3 + 1 \cdot 10^2 + 7 \cdot 10 + 1.$$

Запись числа в десятичной системе счисления состоит из цифр от 0 до 9, последняя цифра — это число единиц, предпоследняя — число десятков, следующая — число сотен ($100 = 10^2$) и т.д. Фактически мы имеем разложение числа по степеням числа 10.

В двоичной системе счисления две цифры — 0 и 1. Двоичную запись чисел обозначают нижним индексом 2:

$$1101_2 = 1 \cdot 2^3 + 1 \cdot 2^2 + 0 \cdot 2 + 1 = 13.$$

Последняя цифра числа в двоичной системе счисления — это количество единиц, предпоследняя — количество двоек, следующая — количество четвёрок и т.д.

Тип int — это 32-битный тип. Первый бит — это знак, для положительных чисел он равен 0, для отрицательных — 1. Мы будем рассматривать только положительные числа, потому что отрицательные задаются немного по-другому. Значит, в первом бите содержится 0, а последние биты содержат представление числа в двоичной системе счисления. На рис. 4 представлен пример для числа 13.

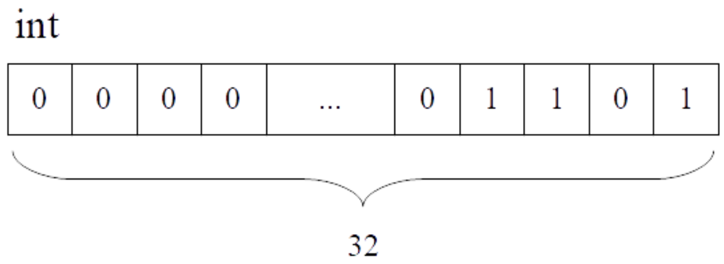

Рис. 4. Представление числа 13 в типе int

Пусть у нас есть два числа в таком представлении. Операции будут применяться к ним по битам: отдельно для первого бита, отдельно — для второго и т.д. в соответствии с таблицей истинности. Для операции AND единицы будут получаться только в тех разрядах, в которых две единицы. В остальных разрядах будут нули, в том числе в начальных битах (см. пример на рис. 5).

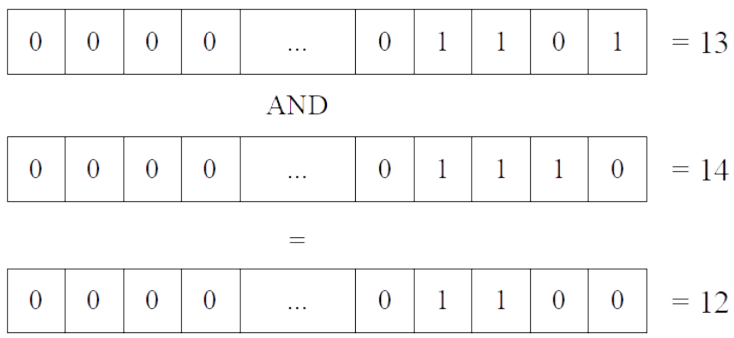

Рис. 5

Упражнение 4.1.1
Вычислите 23 ^ 26. Дайте ответ в десятичной системе счисления.

Рассмотрим еще две операции — битовый сдвиг влево shift left и битовый сдвиг вправо shift right. В C++ для них используются обозначения, как для чтения и записи в поток (см. рис. 6), только эти операции применяются к целым числам. Им даются два аргумента — число x и величина сдвига y. При выполнении этих операций происходит сдвиг битового представления

shift left
$x << y$
сдвиг влево на y бит

shift right
$x >> y$
сдвиг вправо на y бит

Рис. 6. Битовые сдвиги

числа x (например, в 32-битном типе int) на y бит влево или вправо. При этом возникают новые биты с одного из концов. Они заполняются нулями.

На рис. 7 показаны несколько примеров. Внизу под двоичными числами приведены эти же числа в десятичной системе счисления. Первый пример — сдвиг влево на 2 бита числа 13. При этом в конце добавляются два нуля и получается 52. Следующий пример — сдвиг числа 13 вправо на 3 бита. Последние 3 бита уничтожаются, а в начале добавляются три нуля. Получается число 1. Можно заметить, что сдвиг влево — это умножение на степень двойки. В первом случае 13 умножается на 4, во втором — 6 умножается на 8. Действительно, это аналогично приписке к числу в десятичной системе нескольких нулей в конце. При этом число умножится на степень числа 10. Аналогично сдвиг вправо — это деление на степень двойки. Если нацело не делится — происходит деление с остатком, младшие биты отбрасываются. Например, если число 13 разделить с остатком на 8, получится 1.

$00\ldots001101_2 << 2 = 00\ldots110100_2$ 13 52	$00\ldots001101_2 >> 3 = 00\ldots000001_2$ 13 1
$00\ldots000110_2 << 3 = 00\ldots110000_2$ 6 48	$00\ldots101000_2 >> 1 = 00\ldots010100_2$ 40 20

Рис. 7. Битовые сдвиги: примеры

Таким образом, мы пришли к выводу, что справедливы следующие формулы:

$$x << y = x \cdot 2^y,$$
$$x >> y = x/2^y.$$

Строго говоря, эти формулы верны не при любых x и y, а только если сдвиг y достаточно маленький. При этом битовые сдвиги работают быстрее умножения и деления, они более естественны для двоичных чисел. Умножение и деление реализованы через большое количество битовых операций, поэтому они медленнее. Особенно это становится заметно, если в программе выполняется много умножений и делений на степени двойки. Можно заменить их на битовые сдвиги.

Упражнение 4.1.2
Чему равно значение выражения $x \,\hat{}\, ((x >> y) << y)$?