Управление образования

Администрации Сергиево-Посадского района

Муниципальное бюджетное общеобразовательное учреждение

«Физико-математический лицей»

**«Битовые операции»**

консультация для учителей

Учитель: Перлова Н..В.

2018-2019уч.г.

Би́товаяопера́ция в программировании — некоторые операции над цепочками битов. В программировании, как правило, рассматриваются лишь некоторые виды этих операций: логические побитовые операции и битовые сдвиги. Битовые операции применяются в языках программирования и цифровой технике, изучаются в дискретной математике.

**Побитовое отрицание (NOT)**

**Побитовое отрицание** (или **побитовое НЕ**, или **дополнение**) — это унарная операция, действие которой эквивалентно применению логического отрицания к каждому биту двоичного представления операнда. Другими словами, на той позиции, где в двоичном представлении операнда был 0, в результате будет 1, и, наоборот, где была 1, там будет 0.

**Например:** значение R10011010 тогдаnotR 01100101

**Побитовое «И» (AND)**

**Побитовое «И»** — это бинарная операция, действие которой эквивалентно применению логического «И» к каждой паре битов, которые стоят на одинаковых позициях в двоичных представлениях операндов. Другими словами, если оба соответствующих бита операндов равны 1, результирующий двоичный разряд равен 1; если же хотя бы один бит из пары равен 0, результирующий двоичный разряд равен 0.

**Например:**0101 and 0011 получим 0001

Второй операнд в данном случае позволит нам выделить или обработать только нужные биты. В данном примере мы получим значение двух младших бит. Остальные сбрасываются в 0. Второй операнд по другому называется маска. **Маска** – константа, которая определяет область применения логической операции к битам многоразрядного числа. Побитовые операции позволяют манипулировать непосредственно двоичным содержимым памяти компьютера.

**С помощью операции «И» можно сбросить (установить в ноль) биты, для которых маска равна 0!**

**Побитовое «ИЛИ» (OR)**

**Побитовое «ИЛИ»** — это бинарная операция, действие которой эквивалентно применению логического «ИЛИ» к каждой паре битов, которые стоят на одинаковых позициях в двоичных представлениях операндов. Другими словами, если оба соответствующих бита операндов равны 0, двоичный разряд результата равен 0; если же хотя бы один бит из пары равен 1, двоичный разряд результата равен 1.

**Например:**0101 or 0011 получим 0111

**С помощью операции «ИЛИ» можно записать единицу в биты, для которых маска равна 1!**

**Исключающее «ИЛИ» (XOR)**

**Исключающее« ИЛИ»** (или **сложение по модулю 2**) — это бинарная операция, результат действия которой равен 1, если число складываемых единичных битов нечётно и равен 0, если чётно. Другими словами, если оба соответствующих бита операндов равны между собой, двоичный разряд результата равен 0; в противном случае, двоичный разряд результата равен 1.

**Например:**0101 xor 0011 получим 0110

Также данная операция может называться «инверсией по маске», то есть у исходного двоичного числа инвертируются биты, которые совпадают с 1 в маске.

**Би́товый сдвиг** — изменение позиций бит в машинном слове.

Большинство компьютеров не могут напрямую адресовать биты, которые содержатся группами по 8, 16, 32 или 64 бита в машинном слове. Для обеспечения работы с битами существует множество машинных инструкций, включающие различные типы сдвигов. Все сдвиги похожи друг на друга поведением средних битов, которые просто сдвигаются влево или вправо на определённую величину. Однако, поведение крайних битов, которые уходят из слова и которые появляются в слове, зависит от типа сдвига.

**Операции *сдвига*** в свою очередь, подразделяются на:

* логические сдвиги, которые имеют разновидности – сдвиг вправо, сдвиг влево, циклический сдвиг вправо, циклический сдвиг влево;
* арифметические сдвиги вправо и влево, выполнение которых зависит от знака и кода сдвигаемого числа.

**Сдвиг влево** выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (освобождающийся самый правый т.е. самый младший, разряд заполняется 0, а «выталкиваемый» разряд пропадает).

**Например:** код 11001110 после сдвига влево будет иметь вид 10011100.

**Сдвиг вправо** выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем разряде (в освобождающийся самый левый, т.е. самый старший, разряд заполняется 0, «выталкиваемый» разряд пропадает).

**Например:**код 11001110 после сдвига влево будет иметь вид 01100111.

**Циклический сдвигвлево** выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем младшем разряде (в освобождающийся самый правый, т.е. самый младший, разряд заносится значение старшего, т.е. самого левого разряда исходного кода).

**Например:**код 11001110 после сдвига влево будет иметь вид 10011101.

**Циклический сдвиг вправо** выполняется за счет установки в разряд значения, соответствующего исходному значению в ближайшем старшем разряде (в освобождающийся самый левый т.е. самый старший, разряд заполняется значение в самом младшем разряде исходного кода).

**Например:**код 11001110 после сдвига влево будет иметь вид 01100111.

**Арифметические сдвиги** обеспечивают выполнение умножения (сдвиги влево) или операции деления (сдвиги вправо) двоичных кодов на два, точно так же, как сдвиги вправо и влево десятичного числа обеспечиваю выполнение деления и умножение на 10.

Арифметические сдвиги влево двоичного прямого кода выполняются в зависимости от того, какое сдвигается число – положительное или отрицательное.

Если сдвигается положительное число, то сдвиг (вправо или влево) выполняется как соответствующий логический сдвиг (влево или вправо), с той лишь разницей, что предусматриваются средства определения факта переполнения при сдвиге влево, что реализуется и при всех других арифметических операциях. При любом сдвиге вправо предусматриваются средства для округления после завершения нужного количества сдвигов и средства обнаружения обнуления сдвигаемой величины после очередного сдвига.

Арифметические сдвиги влево положительных двоичных чисел выполняются не зависимо от используемого кода (прямого обратного, дополнительного).

**Например:**сдвиг вправо на 8 G:= $FF00; G:=Gshr 8; получим в G 00FF

сдвиг влево на 8 G:= $FF00; G:=Gshl 8; получим в G 0000

**Поразрядные операции в Паскале.**

1. {Ввести целое число n и вывести 2𝑛, используя битовые операции}

**var**x:integer;

**begin**

writeln('Введитечисло');

readln(x);

x:=x **shl**1;

writeln(x);

**end**.

1. {Вывести двоичное представление целого положительного числа, используя битовые операции}

**var**x:integer; s:string;

**begin**

writeln('Введитечисло');

readln(x);s:='';

**while** x>0 **do**

**begin**

**if** x **and** 1 =0 **then** s:='0' +s **else** s:='1' +s;

x:=x **shr**1;

**end**;

writeln(s);

**end**.

1. {Вывести двоичное представление целого положительного числа, используя битовые операции}

**var**x:integer; s:string;

**begin**

writeln('Введитечисло');

readln(x);s:='';

**while** x>0 **do**

**begin**

**if** x **and** 1 =0 **then** s:='0' +s **else** s:='1' +s;

x:=x **shr**1;

**end**;

writeln(s);

**end**.

Задачи для самостоятельного решения на битовые операции:

вариант 1

1.Найти количество единиц в двоичном представлении целого положительного числа.

2. Ввести целые числа n и i и вывести целое число, у которого i-й бит сброшен в 0, а все остальные биты совпадают с битами числа n на тех же позициях. Например, если введены 11 и 1, ответом будет 9.

3. Что будет выведено на экран в результате выполнения данной программы?

var a: byte=54; b: byte=108;

begin

writeln( byte(not(byte(b shl 1) and byte(b shr 1))) and (byte((a or b) shr 1) or byte((a and b) shl 1)));

end.

вариант 2

1. Найти номер старшего бита в двоичном представлении целого положительного числа.

2. Ввести целые числа n и i и вывести целое число, у которого i -й бит отличается от i -го бита числа n, а все остальные биты совпадают с битами числа n на тех же позициях. Например, если введены 5 и 1, ответом будет 4

3. Что будет выведено на экран в результате выполнения данной программы?

var a: byte=22; b: byte=220;

begin writeln( byte(not(byte(b shl 1) and byte(b shr 1))) and (byte((a or b) shr 1) or byte((a and b) shl 1))); end.

Используемые материалы:

1. <https://de.ifmo.ru/bk_netra/page.php?index=86&tutindex=5>
2. <https://yandex.ru/images/search?source=wiz&img_url=https%3A%2F%2Fcf.ppt-online.org%2Ffiles%2Fslide%2Fe%2FezQIBPJx3ukwClNRAYWEj97hMFnptX5Kc16GOo%2Fslide-71.jpg&p=2&text=%D1%81%D1%83%D0%BC%D0%BC%D0%B0%D1%82%D0%BE%D1%80&pos=79&lr=161980&rpt=simage>
3. <http://kpolyakov.spb.ru/school/probook/slides.htm>
4. <http://www.math.spbu.ru/user/nlebedin/tasks_bit.pdf>