Лабораторная работа № 14. Битовые операции

При организации вычислительных процессов в компьютерах используются **десятичная** система счисления (с/с), **двоичная**, **восьмеричная** и **шестнадцатеричная**.

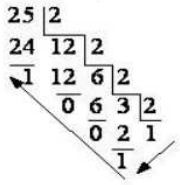
Пусть надо перевести число $a_1 \ldots a_n$, где $a_1, \ldots, a_n$ – цифры этого числа, из системы счисления с основанием **k** в десятичную систему счисления. Данное число можно представить в виде: $a_1 \cdot k^{n-1} + a_2 \cdot k^{n-2} + \ldots + a_n \cdot k^0$. Вычисление суммы даст нужный результат.

Пример перевода из *двоичной* в *десятичную* систему счисления:

$110011_{(2\ с/с)} = 1 \cdot 10^{101} + 1 \cdot 10^{100} + 0 \cdot 10^{11} + 0 \cdot 10^{10} + 1 \cdot 10^{1} + 1 \cdot 1^{0}{}_{(2\ с/с)} =$

$= 1 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0{}_{(10\ с/с)} = 32 + 16 + 2 + 1 = 51_{(10\ с/с)}.$

При переводе из *десятичной* системы счисления в *двоичную*, исходное число делится на основание с/с, т. е. на число 2, фиксируется остаток от деления и частное. Затем частное нужно снова разделить на основание с/с и зафиксировать остаток от деления. Процесс деления частных продолжать до тех пор, пока частное не станет меньше основания с/с. Все полученные в процессе деления остатки от деления и последнее частное будут образовывать цифры нужного результата в обратном порядке.



Например:

$25_{(10\ с/с)} = 11001_{(2\ с/с)} = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 25_{(10\ с/с)}$.

Десятичная с/с	Двоичная с/с	Восьмеричная с/с	Шестнадцатеричная с/с
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	10000	20	10
17	10001	21	11

На языке C/C++ предусмотрены битовые операции для работы с отдельными битами. Их **нельзя** применять к переменным **вещественного** типа. Основные битовые операции:

- **AND (и)** **&** если какой-то бит в одном из операндов равен 0, то результирующий бит тоже будет равен 0;
- **OR (или)** **|** если какой-то бит в одном из операндов равен 1, то результирующий бит тоже будет 1;
- **XOR** (исключающее **или**) **^** результирующий бит равен 1, если сравниваемые биты различны;
- **NOT (не)** **~** меняются все биты на противоположные;
- **сдвиг влево** **<<** удваивается значение; **сдвиг вправо** **>>** – значение уменьшается в два раза.

Задание	Краткие теоретические сведения
1. Изучить использование битовых операций и маскирования числа, опробовав программу, записанную в правой части, с различными исходными числами.	*П* ***#include <iostream>*** ***using namespace std;*** ***void main()*** ***{ setlocale(LC_CTYPE, "Russian");*** ***unsigned int value; int i;*** ***const unsigned int mask = 1 << 31;*** ***cout << "Введите целое число ";*** ***cin >> value;*** ***cout << "Двоичный вид: ";*** ***for (i = 1; i <= 32; i++)*** ***{ putchar(mask & value ? '1' : '0');*** ***value <<= 1;*** ***if (i % 8 == 0) putchar(' ');*** ***}*** ***}*** *ример* программы, печатающей тридцатидвухразрядное двоичное представление целого числа, введенного с клавиатуры. Здесь используется маскирование всех битов числа, за исключением текущего, выводимого на печать. Поскольку представление содержит 32 бита, то маска имеет вид 10000000 00000000 00000000 00000000, т. е. 1 << 31. Последовательно применяется маска и сдвигается число на разряд влево.
2. Выполнить программу, преставленную в правой части. Ознакомиться с результатом. Опробовать программу, изменяя различные биты различных чисел.	*П* ***#include <iostream>*** ***using namespace std;*** ***void main()*** ***{ setlocale(LC_CTYPE, "Russian");*** ***int A = 150; char tmp[33];*** ***_itoa_s(A, tmp, 2);*** ***cout << " Число A: " << tmp << endl;*** ***_itoa_s(0x24, tmp, 2);*** ***cout << " Маска для A: " << tmp << endl;*** ***_itoa_s(A | 0x24, tmp, 2);*** ***cout << " Результат: " << tmp << endl << endl;*** ***}*** *ример*. Установить в единицу каждый **третий** по порядку бит числа **A**, считая справа. Здесь для вывода двоичного представления числа используется стандартная функция: **_itoa_s (**число ввода, строка вывода, основание с/с**)**.
3. В программе, записанной в правой части, используются различные битовые операции. Внести изменения в программу с тем, чтобы проверялось число на кратность четырем.	*П* ***#include <iostream>*** ***using namespace std;*** ***void main()*** ***{ setlocale(LC_CTYPE, "Russian");*** ***int A, i; char tmp[33];*** ***cout << "Введите число "; cin >> A;*** ***_itoa_s(A, tmp, 2);*** ***cout << "Число в двоичном виде = " << tmp << endl;*** ***if ((A & 7) == 0)*** ***cout << "Число кратно 8" << endl;*** ***else*** ***cout << "Число не кратно 8" << endl;*** ***}*** *ример*. Пусть имеется некоторое целое число. Вывести его двоичное представление и проверить, кратно ли оно восьми.

4. В правой части приведен пример программы, демонстрирующей использование битовых операций. Проанализировать текст программы и написать пояснения.

```
#include <iostream>
using namespace std;
void main()
{
setlocale(LC_CTYPE, "Russian");
char tmp[33];
int A, B, maskA = 14;
int maskB = ~maskA >> 1;
cout << "Первое число="; cin >> A;
cout << "Второе число="; cin >> B;
_itoa_s(A, tmp, 2); cout << "A=" << tmp << endl;
_itoa_s(B, tmp, 2); cout << "B=" << tmp << endl;
_itoa_s(maskA, tmp, 2);
cout << "Маска для A: " << tmp << endl;
_itoa_s((A & maskA) >> 1, tmp, 2);
cout << "Выделенные биты A: " << tmp << endl;
_itoa_s(maskB, tmp, 2);
cout << "Маска для B: " << tmp << endl;
_itoa_s(B & maskB, tmp, 2);
cout << " Очищены биты в B: " << tmp << endl;
_itoa_s(((B & maskB) | ((A & maskA) >> 1)), tmp, 2);
cout << " Результат B=" << tmp << endl;
}
```

Пример. Извлечь 3 бита числа **A**, начиная с первого справа и вставить их в число **B**, начиная с нулевого.
Нумерация битов начинается с нуля.

5. В соответствии со своим вариантом разработать программы, использующие битовые операции для решения задач, представленных в таблице. Отсчет битов в числах везде начинается с правой стороны с нуля.

Результаты одной из программ представить в **Отладчике**.

№ варианта	Условия задач
1	1. Ввести целое **A** и посчитать, сколько нулей в числе, начиная с третьего бита по 13, включая эти биты. 2. В числе **A** инвертировать **n** битов вправо от позиции **p**, заменить ими **m** битов числа **B**, начиная с позиции **q**.
2	1. Извлечь 5 битов числа **A**, начиная со второго и вставить их в число **B**, начиная с третьего бита. 2. В числе **A** установить в единицу **n** битов вправо от позиции **p**.
3	1. Ввести целое число **A**. Инвертировать все биты со 2 по 14, включая эти биты. Вывести результат. 2. В числе **A** инвертировать **n** битов влево от позиции **p**, заменить ими **m** битов числа **B**, начиная с позиции **q**.
4	1. Используя битовые операции проверить, кратно ли четырем число **A**. 2. В числе **A** установить в единицу **n** битов влево от позиции **p**.
5	1. Определить, насколько в числе **A** больше значащих битов, равных единице, чем битов, равных нулю. 2. В числе **A** установить в единицу **n** битов вправо от позиции **p**, заменить ими **m** битов числа **B**, начиная с позиции **q**.
6	1. Установить в единицу каждый второй бит целого числа **A**. 2. Извлечь 3 бита числа **A**, начиная с позиции **n**, и вставить их в число **B**, начиная с позиции **m**.

7	1. Извлечь 4 бита числа **A**, начиная с пятого, и добавить их к числу **B** справа. 2. В числе **A** установить в единицу **n** битов влево от позиции **p**, заменить ими **m** битов числа **B**, начиная с позиции **q**.
8	1. Установить в ноль каждый третий бит числа **A**. 2. Извлечь 3 бита числа **A**, начиная с позиции **n**, и вставить их в число **B**, начиная с позиции **m**.
9	1. Извлечь 5 битов числа **A**, начиная с третьего, и вставить их в число **B**, начиная со 2. 2. В числе **A** установить в ноль **n** битов вправо от позиции **p**.
10	1. Вывести 6 битов числа **A**, начиная со 2-го. 2. Инвертировать в числе **A** **n** битов влево от позиции **p**.
11	1. Используя битовые операции проверить, кратно ли шестнадцати число **A**. 2. В числе **A** установить в ноль **n** битов влево от позиции **p**, заменить ими **m** битов числа **B**, начиная с позиции **q**.
12	1. Ввести целое число **A**. Инвертировать все биты с 4 по 8, включая эти биты. Вывести полученное число. 2. В числе **A** установить в ноль **n** битов вправо от позиции **p**, заменить ими **m** битов числа **B**, начиная с позиции **q**.
13	1. Ввести целое число **A**. Извлечь 2 бита числа **A**, начиная с пятого и вставить их в число **B**, начиная также с пятого бита. 2. В числе **A** инвертировать n битов вправо от позиции **p**.
14	1. Ввести целое число **A** и посчитать, сколько единиц в числе с 5 бита по 10 бит, включая эти биты. 2. Извлечь 3 бита числа **A**, начиная с позиции **n**, и вставить их в число **B**, начиная с позиции **m**.
15	1. Используя битовые операции проверить, кратно ли двум число **A**. 2. В числе **A** установить в ноль **n** битов влево от позиции **p.**
16	1. Ввести целое число **A**. Извлечь 3 бита числа **A**, начиная со второго и вставить их в число **B**, начиная с первого бита. 2. Установить в единицу два бита числа **A**, начиная с четвертого.

6. К номеру своего варианта прибавить 1 и написать программы для новых исходных данных (для варианта 16 перейти к варианту 1).

7. Дополнительные задания.

1. С помощью сдвига единицы на **n** битов вычислить два в степени **n**.

2. Даны два неравных числа **n** и **m**, не превосходящие 31. Вычислить **2n + 2m**.

вывести на экран двоичное представление восьмеричного числа.

3. Дано целое число **A** и натуральное число **i**. Вывести число, которое получается из числа **A** установкой значения **i**-го бита в ноль.

4. Дано целое число **A** и натуральное число **n**. Вывести число, которое состоит только из **n** последних бит числа **A** (т. е. обнулить все биты числа **A**, кроме последних **n**).

5. Дано целое число **A** и натуральное число **i**. Вывести значение **i**-го бита числа **A**.

6. Дано число от 0 до 255 типа **unsigned char**. Вывести его в битовой форме: старшие биты слева, младшие – справа.