

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего образования

**«Белгородский государственный технологический университет им.
В.Г. Шухова»**

(БГТУ им. В.Г. Шухова)

Кафедра программного обеспечения вычислительной техники и
автоматизированных систем

Лабораторная работа №7

По дисциплине: «Основы программирования»

Тема: «Побитовые операции»

Выполнил: студент группы ВТ-231

Борченко Александр Сергеевич

Проверили:

Черников Сергей Викторович

Новожен Никита Викторович

Белгород 2023

Цель работы: получение навыков работы с побитовыми операциями.

Содержание работы:

Задача №1: Вывести восьмеричное представление записи числа x	3
Задача №2: Напишите функцию <i>deleteOctNumber</i> , которая удаляет цифру <i>digit</i> в записи данного восьмеричного числа x . Вывод результата можно произвести в любой системе счисления.	4
Задача №3: Напишите функцию <i>swapPairBites</i> , которая меняет местами соседние цифры пар в двоичной записи данного натурального числа. Обмен начинается с младших разрядов. Непарная старшая цифра остается без изменения	5
Задача №4: Напишите функцию <i>invertHex</i> , которая преобразует число x , переставляя в обратном порядке цифры в шестнадцатеричном представлении данного натурального числа.....	6
Задача №5: Напишите функцию <i>isBinPoly</i> , которая возвращает значение 'истина', если число x является палиндромом в двоичном представлении, иначе - 'ложь'.	7
Задача №6: Даны два двухбайтовых целых $sh1$ и $sh2$. Получить целое число, последовательность четных битов которого представляет собой значение $sh1$, а последовательность нечетных – значение $sh2$	8
Задача №7: Определить максимальную длину последовательности подряд идущих битов, равных единице в двоичном представлении данного целого числа	9
Задача №8: ** Выполнить циклический сдвиг в двоичном представлении данного натурального числа x на k битов влево. Обратите внимание на механизм сдвига	10
Задача №9: ** Дано длинное целое неотрицательное число. Получить число, удалив каждую вторую цифру в двоичной записи данного числа, начиная со старших цифр	11
Задача №10: ** Дано целое неотрицательное число. Получить число перестановкой битов каждого байта данного числа в обратном порядке	12
Задача №11: **Пакет с монетами (1037A)	13

Задача №1: Вывести восьмеричное представление записи числа x.

Спецификация функции OctNumber:

1. Заголовок: int octNumber(int n)
2. Назначение: Вывод восьмеричного представления числа n.

Код задачи:

```
#include <stdio.h>

int octNumber(int n) {
    int octal_notation_number = 0, degree_number_10 = 1;
    while (n != 0) {
        int remainder = n & 7;
        octal_notation_number += remainder * degree_number_10;
        n >>= 3;
        degree_number_10 *= 10;
    }
    return octal_notation_number;
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("%d", octNumber(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
1024	2000
364	554

Задача №2: Напишите функцию *deleteOctNumber*, которая удаляет цифру *digit* в записи данного восьмеричного числа *x*. Вывод результата можно произвести в любой системе счисления.

Спецификация функции *deleteOctNumber*:

1. Заголовок: `int deleteOctNumber(int x, int digit)`
2. Назначение: Возвращает восьмеричное представление числа *x* у которого удалена цифра *digit*.

Код задачи:

```
#include <stdio.h>

int deleteOctNumber(int x, int digit) {
    int modified_number = 0;
    int position = 0;
    while (x != 0) {
        int last_oct_digit_number_x = x & 7;
        if (last_oct_digit_number_x != digit) {
            last_oct_digit_number_x <=<= position;
            position += 3; //+3 т.к с 8-ой с\с работаем
            modified_number |= last_oct_digit_number_x;
        }

        x >>= 3;
    }

    return modified_number;
}

int main()
{
    int n, digit;
    scanf("%d %d", &n, &digit);

    printf("%d", deleteOctNumber(n, digit));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
3179(10) = 110'001'101'011(2) = 6153(8) digit = 1	653(8) = 110'101'011 = 427(10)
9(10) = 1'001(2) = 11(8) digit = 1	0(10)

Задача №3: Напишите функцию *swapPairBites*, которая меняет местами соседние цифры пар в двоичной записи данного натурального числа. Обмен начинается с младших разрядов. Непарная старшая цифра остается без изменения.

Спецификация функции *swapPairBites*:

1. Заголовок: `int swapPairBites(int n)`
2. Назначение: Меняет местами соседние цифры пар в двоичной записи натурального числа `n`.

Код задачи:

```
#include <stdio.h>

int swapPairBites(int n) {
    int ammounts_bits_in_number = 0;
    int term_n = n;
    while (term_n > 0) {
        term_n >>= 1;
        ammounts_bits_in_number++;
    }
    int result;
    if (ammounts_bits_in_number % 2 == 0) {
        result = ((n & 0x55555555) << 1) | ((n & 0xAAAAAAAA) >> 1);
    }
    else {
        n = n - (1 << (ammounts_bits_in_number - 1));
        result = (((n & 0x55555555) << 1) | ((n & 0xAAAAAAAA) >> 1))
            + (1 << (ammounts_bits_in_number - 1));
    }
    return result;
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("%d", swapPairBites(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
6(10) = 110(2)	101(2) = 5(10)
77(10) = 1001101(2)	1001110(2) = 78(10)
165(10) = 10100101(2)	1011010(2) = 90(10)

Задача №4: Напишите функцию *invertHex*, которая преобразует число *x*, переставляя в обратном порядке цифры в шестнадцатеричном представлении данного натурального числа.

Спецификация функции *swapPairBites*:

1. Заголовок: `long long invertHex(long long n)`
2. Назначение: выводит число *n*, представляя в обратном порядке цифры в шестнадцатеричном представлении данного натурального числа.

Код задачи:

```
#define HEX_MASK 0xF
#include <stdio.h>

long long invertHex(long long n) {
    long long res = 0;
    while (n) {
        res <<= 4;
        res |= n & HEX_MASK;
        n >>= 4;
    }
    return res;
}

int main() {
    long long n;
    scanf("%lld", &n);

    printf("%lld", invertHex(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
77(10) = 100'1101(2) = 4D(16)	D4(16) = 1101'0100(2) = 212(10)
2732(10) = 1010'1010'1100(2) = AAC(16)	CAА(16) = 1100'1010'1010(2) = 3242(10)

Задача №5: Напишите функцию *isBinPoly*, которая возвращает значение 'истина', если число *x* является палиндромом в двоичном представлении, иначе - 'ложь'.

Спецификация функции *swapPairBites*:

1. Заголовок: `bool isBinPoly(int x)`
2. Назначение: возвращает значение "истина" если вводимое число *x* является палиндромом в двоичном представлении, иначе – "ложь".

Код задачи:

```
#include <stdio.h>
#include <stdbool.h>

bool isBinPoly(int x) {
    int reverseBinX = 0;
    int bin_x = x;

    while (bin_x != 0) {
        int lastDigit = bin_x & 1;
        reverseBinX <<= 1;
        reverseBinX |= lastDigit;
        bin_x >>= 1;
    }

    return x == reverseBinX;
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("%d", isBinPoly(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
27(10) = 11011(2)	1 (YES)
454(10) = 111000110(2)	0 (NO)

Задача №6: Даны два двухбайтовых целых $sh1$ и $sh2$. Получить целое число, последовательность четных битов которого представляет собой значение $sh1$, а последовательность нечетных – значение $sh2$.

Спецификация функции `getting_number`:

1. Заголовок: `int getting_number(int sh1, int sh2)`
2. Назначение: возвращает целое число последовательность битов которого представляет собой значение $sh1$, а последовательность нечётных $sh2$.

Код задачи:

```
#include <stdio.h>

int getting_number(int sh1, int sh2) {
    int new_num = 0, position_current_bit_in_new_num = 0;
    while (sh1 > 0 || sh2 > 0){
        new_num += (sh2 & 1) << position_current_bit_in_new_num++;
        sh2 >>= 1;
        new_num += (sh1 & 1) << position_current_bit_in_new_num++;
        sh1 >>= 1;
    }
    return new_num;
}

int main()
{
    int a, b;
    scanf("%d %d", &a, &b);

    printf("%d", getting_number(a, b));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
$sh1 = 0000000000001100(2) = 12(10)$ $sh2 = 0000000000010010(2) = 18(10)$	$000000000000000000000000110100100(2) = 420(10)$

Задача №7: Определить максимальную длину последовательности подряд идущих битов, равных единице в двоичном представлении данного целого числа.

Спецификация функции getMaxLength1:

1. Заголовок: int getMaxLength1(int x)
2. Назначение: возвращает десятичное число – максимальную длину последовательности идущих подряд единиц.

Код задачи:

```
#include <stdio.h>

int getMaxLength1(int x) {
    int count_units_in_number = 0;
    int max_length = 0;
    while (x != 0) {
        int digit = x & 1;
        if (digit == 1) {
            count_units_in_number++;

            if (count_units_in_number > max_length)
                max_length = count_units_in_number;
        } else {
            count_units_in_number = 0;
        }

        x >>= 1;
    }

    return max_length;
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("%d", getMaxLength1(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
1337(10) = 10100111001(2)	3(10)
61454(10) = 1111000000001110(2)	4(10)

Задача №8: ** Выполнить циклический сдвиг в двоичном представлении данного натурального числа x на k битов влево. Обратите внимание на механизм сдвига.

Спецификация функции cyclicShift:

1. Заголовок: long long cyclicShift(int n, int shift)
2. Назначение: возвращает число после циклического сдвига натурального числа $x(n)$ на $k(\text{shift})$ битов влево.

Код задачи:

```
#include <stdio.h>

int getBinLength(int n) {
    int lenght = 0;
    while (n != 0) {
        ++lenght;
        n >>= 1;
    }
    return lenght;
}

long long cyclicShift(int n, int shift) {
    int result = n;
    int lenght = getBinLength(result);
    int lenght_shift = shift;
    if (lenght != 0) {
        if (lenght_shift >= lenght)
            lenght_shift %= lenght;
        int position = lenght - 1;
        int mask = 1 << position;

        for (int i = 0; i < lenght_shift; ++i) {
            int value_bit_at_position = (n & mask) >> position;
            result &= ~mask;
            result <<= 1;
            result |= value_bit_at_position;
        }
    }
    return result;
}

int main()
{
    int x, k;
    scanf("%d %d", &x, &k);

    printf("%d", cyclicShift(x, k));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
$x = 27(10) = 11011(2)$ $k = 1$	$10111(2) = 23(10)$
$x = 30(10) = 11110(2)$ $k = 1$	$11101(2) = 29(10)$

Задача №9: ** Дано длинное целое неотрицательное число. Получить число, удалив каждую вторую цифру в двоичной записи данного числа, начиная со старших цифр.

Спецификация функции `getNumberWithDeleteSecondBits`:

1. Заголовок: `unsigned long long getNumberWithDeleteSecondBits(int n)`
2. Назначение: возвращает число x , у которого удалили каждую вторую цифру начиная со старших цифр.

Код задачи:

```
#define BIN_MASK 0x1
#include <stdio.h>

unsigned long long getNumberWithDeleteSecondBits(int n) { //по условию > 0
    unsigned long long num_with_delete_digits = 0;
    int reverse = 0;
    int len = 0;

    while (n != 0) {
        reverse <<= 1;
        reverse |= n & BIN_MASK;
        n >>= 1;
        ++len;
    }
    while (len > 0) {
        num_with_delete_digits <<= 1;
        num_with_delete_digits |= reverse & BIN_MASK;
        reverse >>= 2;
        len -= 2;
    }
    return num_with_delete_digits;
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("%d", getNumberWithDeleteSecondBits(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
$n = 40(10) = 101000(2)$	$110(2) = 6(10)$
$n = 3(10) = 11(2)$	$1(2) = 1(10)$
$n = 10(10) = 1010(2)$	$11(2) = 3(10)$

Задача №10: ** Дано целое неотрицательное число. Получить число перестановкой битов каждого байта данного числа в обратном порядке.

Спецификация функции shiftBits:

1. Заголовок: long long shiftBits(long long x)
2. Назначение: возвращает десятичное представление числа n после перестановки битов каждого байта данного числа в обратном порядке.

Код задачи:

```
#include <stdio.h>

long long shiftBits(long long x) {
    x = (x >> 8) | (x << 8);
    for (int i = 23; i >= 16; i--) {
        x &= ~(1 << i);
    }
    return x;
}

int main()
{
    int n;
    scanf("%d", &n);

    printf("%d", shiftBits(n));

    return 0;
}
```

Тестовые данные:

Входные данные	Выходные данные
x = 61455(10) = 1111000000001111(2)	111111110000(2) = 4080(10)
x = 43605(10) = 1010101001010101(2)	0101010110101010(2) = 21930(10)

Задача №11: **Пакет с монетами (1037A)

Код задачи:

```
#include <stdio.h>


int main()
{
    int ammount_coins;
    scanf("%d", &ammount_coins);

    int min_ammount_packages = 0;

    while (ammount_coins != 0) {
        ammount_coins >>= 1;
        min_ammount_packages++;
    }
    printf("%d", min_ammount_packages);

    return 0;
}
```

Вердикт тестовой системы:

Основное									
№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.	
232782939	Дорешивание: Sasha39-_-	1037A - 33	GNU C11	Полное решение	30 мс	8 КБ	2023-11-14 20:06:18	2023-11-14 20:06:18	 <input type="button" value="Сравнить"/>

Вывод: в ходе проведения лабораторной работы я час смеялся, потом плакал, а потом еще 3 дня разбирался, как работают побитовые операции. Теперь я осознал как работают побитовые операции на языке C, закрепил навыки написания функций и укрепил оставшуюся нервную систему.