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 ПВ-223
Мелехов Артём Дмитриевич

Проверили:
ст. преп. Притчин Иван Сергеевич
асс. Черников Сергей Викторович

Белгород 2022 г.

Лабораторная работа №7 «Побитовые операции»

Цель работы: получение навыков работы с побитовыми операциями.

Содержание отчета:

Тема лабораторной работы

Цель лабораторной работы

Решения задач. Для каждой задачи указаны:

– Условие задачи.

– Тестовые данные.

– Исходный код функции и её спецификация.

Вывод.

Задача №1.

Условие:

Вывести восьмеричное представление записи числа x.

Тестовые данные:

Входные данные	Ожидаемый результат	Пояснение
1 (1_8)	1	Ввод числа непревышающего систему счисления.
8 (10_8)	10	Ввод числа больше 8.
17 (21_8)	21	Ввод числа больше 8.

Спецификация функции *countOctDigits*:

1. Заголовок: `long long countOctDigits(long long unsigned x)`.
2. Назначение: возвращает количество символов в восьмеричном представлении числа x.

Спецификация функции *printOct*:

1. Заголовок: `void printOct(long long unsigned x)`.
2. Назначение: выводит восьмеричную запись числа x.

```c
#define BINARY_DIGITS_IN_ONE_OCT_DIGIT 3U

long long countOctDigits(long long unsigned x) {
    long long count = 0;

    while (x != 0) {
        x >>= BINARY_DIGITS_IN_ONE_OCT_DIGIT;
        count++;
    }

    return count;
}

void printOct(long long unsigned x) {
    long long nOctDigits = countOctDigits(x);
    long long shift = (nOctDigits - 1) * BINARY_DIGITS_IN_ONE_OCT_DIGIT;

    while (shift >= 0) {
        long long octDigit = x >> shift & 7;

        printf("%lld", octDigit);

        shift -= BINARY_DIGITS_IN_ONE_OCT_DIGIT;
    }
}
```

Задача №2.

Условие:

Напишите функцию $deleteOctNumber$, которая удаляет цифру $digit$ в записи данного восьмеричного числа x. Вывод результата можно произвести в любой системе счисления.

Входные данные	Выходные данные
$3179_{10} = 110'001'101'011_2 = 6153_8$ $digit = 1$	$653_8 = 110'101'011_2 = 427_{10}$
$9_{10} = 1'001_2 = 11_8$ $digit = 1$	0_{10}
$37_{10} = 100'101_2 = 45_8$ $digit = 1$	$45_8 = 100'101_2 = 37_{10}$

Тестовые данные:

Входные данные	Ожидаемый результат	Пояснение
3179 (6153_8) 1	653_8	Цифра *digit* находится в числе x_8 единожды.
9 (11_8) 1	0_8	Число x_8 состоит из цифр *digit*.
37 (45_8) 1	45_8	Число x_8 не содержит цифры *digit*.

Спецификация функции $deleteOctNumber$:

1. Заголовок: `void deleteOctNumber(long long unsigned x, unsigned digit)`.
2. Назначение: выводит число *x* в восьмеричной записи без цифры *digit*.

```c
#define BINARY_DIGITS_IN_ONE_OCT_DIGIT 3U

long long countOctDigits(long long unsigned x) {
    long long count = 0;

    while (x != 0) {
        x >>= BINARY_DIGITS_IN_ONE_OCT_DIGIT;
        count++;
    }

    return count;
}

void deleteOctNumber(long long unsigned x, unsigned digit) {
    long long nOctDigits = countOctDigits(x);
    long long shift = (nOctDigits - 1) * BINARY_DIGITS_IN_ONE_OCT_DIGIT;
    long long unsigned count = 0;

    while (shift >= 0) {
        long long octDigit = x >> shift & 7;

        if (octDigit != digit)
            printf("%lld", octDigit);
        else
            count++;

        shift -= BINARY_DIGITS_IN_ONE_OCT_DIGIT;

        if (count == nOctDigits)
            printf("0");
    }
}
```

Задача №3.

Условие:

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.

Входные данные	Выходные данные
$77_{10} = 1001101_2$	$1001110_2 = 78_{10}$
$165_{10} = 10100101_2$	$1011010_2 = 90_{10}$

Тестовые данные:

Входные данные	Ожидаемый результат	Пояснение
77 (1001101_2)	1001110_2	Число состоящее из нечётного количества бит.
165 (10100101_2)	1011010_2	Число состоящее из нечётного количества бит.

Спецификация функции *swapPairBites*:

1. Заголовок: `void swapPairBites(long long unsigned x)`.
2. Назначение: выводит двоичную запись числа *x* с попарным изменением битов начиная с младшего (непарная старшая цифра остаётся без изменений).

```
void swapPairBites(long long unsigned x) {
    long long number = 0;
    long long unsigned shift = 0;
    int binDigit1, binDigit2;

    while (x >= 4) {
        binDigit1 = x & 1;
        x >>= 1;
        binDigit2 = x & 1;
        x >>= 1;
        number |= binDigit2 << shift;
        shift++;
        number |= binDigit1 << shift;
        shift++;
    }

    if (x == 2)
        number |= 1 << shift;
    else
        number |= x << shift;

    printf("%lld", number);
}
```

Задача №4.

Условие:

Напишите функцию $invertHex$, которая преобразует число x, переставляя в обратном порядке цифры в шестнадцатеричном представлении данного натурального числа.

Входные данные	Выходные данные
$77_{10} = 100'1101_2 = 4D_{16}$	$D4_{16} = 1101'0100_2 = 212_{10}$
$2732_{10} = 1010'1010'1100_2 = AAC_{16}$	$CAA_{16} = 1100'1010'1010_2 = 3242_{10}$

Тестовые данные:

Входные данные	Ожидаемый результат	Пояснение
77 ($4D_{16}$)	D4	Ввод «Пифагоровой тройки».
2732 (AAC_{16})	CAA	Ввод псевдослучайных чисел с нецелым ответом.

Спецификация функции *countHexDigits*:

1. Заголовок: `int countHexDigits(unsigned int x)`.
2. Назначение: возвращает количество элементов в 16-ричной записи числа *x*.

Спецификация функции $invertHex$:

1. Заголовок: `void invertHex(unsigned int x)`.
2. Назначение: выводит обратную запись 16-ричного числа *x*.

```c
#define BINARY_DIGITS_IN_ONE_HEX_DIGIT 4U

int countHexDigits(unsigned int x) {
    int count = 0;

    while (x != 0) {
        x >>= BINARY_DIGITS_IN_ONE_HEX_DIGIT;
        count++;
    }

    return count;
}

void invertHex(unsigned int x) {
    int nHexDigits = countHexDigits(x);
    int shift = (nHexDigits - 1) * BINARY_DIGITS_IN_ONE_HEX_DIGIT;

    while (shift >= 0) {
        int hexDigit = x & 15;

        if (hexDigit < 10)
            printf("%d", hexDigit);
        else
            printf("%c", (hexDigit - 10 + 'A'));

        shift -= BINARY_DIGITS_IN_ONE_HEX_DIGIT;
        x /= 16;
    }
}
```

Задача №5.

Условие:

Напишите функцию $isBinP\ oly$, которая возвращает значение ’истина’, если число x является палиндромом в двоичном представлении, иначе - ’ложь’.

Входные данные	Выходные данные
$27_{10} = 11011_2$	YES
$454_{10} = 111000110_2$	NO

Тестовые данные:

Входные данные	Ожидаемый результат	Пояснение
27 (11011_2)	true	Число является палиндромом в двоичной системе счисления.
454 (111000110_2)	false	Число не является палиндромом в двоичной системе счисления.

Спецификация функции *reverseX*:

1. Заголовок: `long long reverseX(long long x)` .
2. Назначение: возвращает реверс двоичного представления числа *x*.

Спецификация функции *isBinPoly*:

1. Заголовок: `bool isBinPoly(unsigned int x)`.
2. Назначение: возвращает значение ‘истина’, если число в двоичной системе счисления является палиндромом и ‘ложь’ в противоположном.

```
#include<stdbool.h>

long long reverseX(long long x) {
    long long number = 0;

    while (x != 0) {
        number <<= 1;

        long long binDigit = x & 1;

        x >>= 1;
        number |= binDigit;
    }

    return number;
}

bool isBinPoly(long long x) {
    return reverseX(x) == x;
}
```

Задача №6.

Условие:

Даны два двухбайтовых целых *sh1* и *sh2*. Получить целое число, последовательность четных битов которого представляет собой значение *sh1* , а последовательность нечетных – значение *sh2*.

Входные данные	Выходные данные
$sh_1 = 0000000000001100_2 = 12_{10}$ $sh_2 = 0000000000010010_2 = 18_{10}$	$00000000000000000000000110100100_2 = 420_{10}$
$sh_1 = 0111111100000000_2 = 32512_{10}$ $sh_2 = 0000000000000000_2 = 0_{10}$	$00101010101010100000000000000000_2 = 715784192_{10}$

Тестовые данные:

Входные данные	Ожидаемый результат	Пояснение
12 (1100_2) 18 (10010_2)	$00000000000000000000000110100100_2$	Ввод двух чисел.
32512 (111111100000000_2) 0 (0_2)	$00101010101010100000000000000000_2$	Ввод двух чисел.

Спецификация функции *sequenceOfTwoBin*:

1. Заголовок: `long long sequenceOfTwoBin(long long sh1, long long sh2)` .
2. Назначение: возвращает последовательность битов, где чётные позиции – число *sh1*, а нечётные – *sh2*.

```
#define BINARY_DIGITS_IN_ONE_BIN_DIGIT 1U

long long sequenceOfTwoBin(long long sh1, long long sh2) {
    char shift = 0;
    long long number = 0;
    char binDigit;

    while (shift < 32) {
        binDigit = sh1 & 1;
        sh1 >>= BINARY_DIGITS_IN_ONE_BIN_DIGIT;
        number |= binDigit << shift;
        shift += BINARY_DIGITS_IN_ONE_BIN_DIGIT;

        binDigit = sh2 & 1;
        sh2 >>= BINARY_DIGITS_IN_ONE_BIN_DIGIT;
        number |= binDigit << shift;
        shift += BINARY_DIGITS_IN_ONE_BIN_DIGIT;
    }

    return number;
}
```

Задача №7.

Условие:

Определить максимальную длину последовательности подряд идущих битов, равных единице в двоичном представлении данного целого числа.

Входные данные	Выходные данные
$x = 61454_{10} = 1111000000001110_2$	4
$x = 11_{10} = 1011_2$	2

Тестовые данные:

Входные данные	Ожидаемый результат	Пояснение
61454 (1111000000001110_2)	4	Ввод числа.
11 (1011_2)	2	Максимум объявляется несколько раз в процессе его поиска.

Спецификация функции *max2*:

1. Заголовок: `long long max2(long long a, long long b)`.
2. Назначение: возвращает максимальное из двух чисел.
3.

Спецификация функции *maximumNumberOfContractors*:

1. Заголовок: `long long maximumNumberOfContractors1(unsigned x)`.
2. Назначение: возвращает максимальную длину последовательности состоящей из единиц в двоичном представлении числа x.

```
long long max2(long long a, long long b) {
    return a > b ? a : b;
}

long long maximumNumberOfContractors1(long long x) {
    long long max = 0;
    long long count = 0;

    while (x != 0) {
        if (x & 1 == 1)
            count++;
        else {
            max = max2(max, count);
            count = 0;
        }

        x >>= 1;
    }

    return max2(max, count);
}
```

Вывод: в ходе работы получены навыки работы с побитовыми операциями.