

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: ст. группы ПВ-223  
Пахомов Владислав Андреевич

Проверили:  
Притчин Иван Сергеевич  
Черников Сергей Викторович

Код-ревьюер: ст. группы ПВ-223  
Голуцкий Георгий Юрьевич

Белгород 2022 г.

# Лабораторная работа № 7

Вариант №1 ( $10 \bmod 10 + 1 = 1$ )

## Содержание отчёта:

- Тема лабораторной работы
- Цель лабораторной работы
- Решения задач
  - Условие задачи
  - Тестовые данные
  - Исходный код функции и её спецификация
- Работа над ошибками (код-ревью)
- Вывод по работе.

**Тема лабораторной работы:** Побитовые операции

**Цель лабораторной работы:** получение навыков работы с побитовыми операциями.

## Решения задач:

### 1. Задача №1

Вывести восьмеричное представление записи числа  $x$ .

Входные данные	Выходные данные
1 0	0
2 1437	2635
3 2739128	12345670
4 18446744073709551615	17777777777777777777

1. Заголовок: `void printOct(unsigned long long x)`
2. Назначение: выводит восьмеричное представление числа  $x$

Входные данные	Выходные данные
1 $x = 0_{10} = 0_8 = 0_2$ , $at = 63$	0
2 $x = 3_{10} = 3_8 = 11_2$ , $at = 0$	3
3 $x = 71263_{10} = 213\ 137_8 =$ $1'0001'0110'0101'1111_2$ , $at =$	1 57
4 $x = 9223372036854775808_{10} =$ $1'000'000'000'000'000'000'000_8$ $= 1000'0000'0000'0000'0000$ $0000'0000'0000'0000'0000'0000$ $0000'0000'0000'0000_2$ , $at =$	1 = 0

3. Заголовок: `char getOctDigit(unsigned long long x, int at)`
4. Назначение: возвращает цифру в восьмеричном виде числа  $x$  на позиции  $at$  с левого конца числа в двоичном коде.

```

#include <stdio.h>
#include "lfunc.h"

#define OCT_BLOCK_BINARY_SIZE 3
#define OCT_FULL_BLOCK 711
#define LONG_LONG_BIT_SIZE (sizeof(unsigned long long) << 3)

/*
 * Заголовок: char getOctDigit(unsigned long long x, int at)
 * Назначение: возвращает цифру в восьмеричном виде числа x на позиции at с левого
 *             конца числа в двоичном коде.
 */
char getOctDigit(unsigned long long x, int at) {
    return (x >> (LONG_LONG_BIT_SIZE - 1 - at)) & OCT_FULL_BLOCK;
}

/*
 * Заголовок: void printOct(unsigned long long x)
 * Назначение: выводит восьмеричное представление числа x
 */
void printOct(unsigned long long x) {
    if (x == 0) {
        printf("0\n");
        return;
    }

    int digitBit = 0;
    char digit;

    while (!(digit = getOctDigit(x, digitBit)))
        digitBit += OCT_BLOCK_BINARY_SIZE;

    while (digitBit < LONG_LONG_BIT_SIZE) {
        char currentDigit = getOctDigit(x, digitBit);
        digitBit += OCT_BLOCK_BINARY_SIZE;

        printf("%d", currentDigit);
    }

    printf("\n");
}

```

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics/blob/c/lab7/sharedfuncs/src/lfunc.c>

## 2. Задача №4

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

Входные данные	Выходные данные
1 $0_{10} = 0_2 = 0_{16}$	$0_{10} = 0_2 = 0_{16}$
2 $62_{10} = 11'1110_2 = 3E_{16}$	$227_{10} = 1110'0011_2 = E3_{16}$
3 $51172_{10} =$ $1100'0111'1110'0100_2 =$ $C7E4_{16}$	$20092_{10} =$ $100'1110'0111'1100_2 =$ $4E7C_{16}$
4 $1311768467463790320_{10} =$ $0001'0010'0011'0100$ $0101'0110'0111'1000$ $1001'1010'1011'1100$ $1101'1110'1111'0000_2 =$ $123456789ABCDEF0_{16}$	$1147797409030816545_{10} =$ $1111'1110'1101'1100$ $1011'1010'1001'1000$ $0111'0110'0101'0100$ $0011'0010'0001_2 =$ $FEDCBA987654321_{16}$
5 $18446744073709551615_{10} =$ $1111'1111'1111'1111$ $1111'1111'1111'1111$ $1111'1111'1111'1111$ $1111'1111'1111'1111_2 =$ $FFFFFFFFFFFFFFFF_{16}$	$18446744073709551615_{10} =$ $1111'1111'1111'1111$ $1111'1111'1111'1111$ $1111'1111'1111'1111$ $1111'1111'1111'1111_2 =$ $FFFFFFFFFFFFFFFF_{16}$

1. Заголовок: `void invertHex(unsigned long long *x)`
2. Назначение: преобразует значение по адресу `x` переставляя в обратном порядке цифры значения по адресу `x` в обратном порядке в шестнадцатеричном представлении.

```
#include "4func.h"

#define HEX_FULL_BLOCK 15
#define HEX_BLOCK_LEN 4

/*
 * Заголовок: void invertHex(unsigned long long *x)
 * Назначение: преобразует значение по адресу x переставляя в обратном порядке цифры
 * значения по адресу x в обратном порядке в шестнадцатеричном
 * представлении.
 */
void invertHex(unsigned long long *x) {
    unsigned long long reversedX = 0;

    while (*x) {
        reversedX = (reversedX << HEX_BLOCK_LEN) + (*x & HEX_FULL_BLOCK);
        *x >>= HEX_BLOCK_LEN;
    }

    *x = reversedX;
}
```

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics/blob/c/lab7/sharedfuncs/src/4func.c>

### 3. Задача №7

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

Входные данные	Выходные данные
1 $x = 0_{10} = 0_2$	0
2 $x = 1_{10} = 1_2$	1
3 $x = 511_{10} = 1'1111'1111_2$	9
4 $x = 633_{10} = 10'0111'1001_2$	3
5 $x =$ 18446744073709551615 <sub>10</sub> = 1111'1111'1111'1111 1111'1111'1111'1111 1111'1111'1111'1111 1111'1111'1111'1111 <sub>2</sub>	64

1. Заголовок: `int getMaximumLength(unsigned long long x)`
2. Назначение: возвращает максимальное количество подряд идущих единиц в двоичной записи числа `x`.

```
#include "7func.h"

/*
 * Заголовок: int getMaximumLength(unsigned long long x)
 * Назначение: возвращает максимальное количество подряд идущих единиц в двоичной
 * записи числа x.
 */
int getMaximumLength(unsigned long long x) {
    int maxLen = 0;
    int currLen = 0;

    while (x) {
        if (x & 1) {
            currLen += 1;

            if (currLen > maxLen)
                maxLen = currLen;
        } else
            currLen = 0;

        x >>= 1;
    }

    return maxLen;
}
```

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics/blob/c/lab7/sharedfuncs/src/7func.c>

#### 4. Задача №8

**\*\* Выполнить циклический сдвиг в двоичном представлении данного натурального числа  $x$  на  $k$  битов влево.**

Входные данные	Выходные данные
1 $x = 0_{10} = 0_2$ $k = 100$	$0_{10} = 0_2$
2 $x = 1_{10} = 1_2$ $k = 100$	$1_{10} = 1_2$
3 $x = 34_{10} = 10'0010_2$ $k = 0$	$34_{10} = 10'0010_2$
4 $x = 34_{10} = 10'0010_2$ $k = 1$	$5_{10} = 101_2$
5 $x = 253_{10} = 1111'1101_2$ $k = 5$	$191_{10} = 1011'1111_2$
5 $x = 23_{10} = 1'0111_2$ $k = 3$	$15_{10} = 1111_2$

1. Заголовок: `void cycleShift(unsigned long long *pX, int k)`
2. Назначение: выполняет циклический сдвиг числа по адресу `pX` на `k` влево.

```
#include "8func.h"

/*
 * Заголовок: void cycleShift(unsigned long long *pX, int k)
 * Назначение: выполняет циклический сдвиг числа по адресу pX на k влево.
 */
void cycleShift(unsigned long long *pX, int k) {
    if (*pX == 0)
        return;

    while (k--) {
        unsigned long long copyX = *pX;
        *pX = 0;
        int pow = 1;

        while (copyX & (~1)) {
            *pX += (copyX & 1) << pow;
            copyX >>= 1;
            pow++;
        }

        *pX += 1;
    }
}
```

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics/blob/c/lab7/sharedfuncs/src/8func.c>

P.S. по тестовым данным пособия невозможно определить точно механизм работы циклического сдвига произвольного размера. Так, например, переносится ли ноль при циклическом сдвиге? (прим.  $x = 10111_2$ ,  $k = 3$ ; `cycleShift(x, k)` ->  $x = 1111_2$  или `cycleShift(x, k)` ->  $x = 11101_2$ ?).

## 5. Задача №9

\*\* Дано длинное целое неотрицательное число. Получить число, удалив каждую вторую цифру в двоичной записи данного числа, начиная со старших цифр.

Входные данные	Выходные данные
1 $x = 253_{10} = 1111'1101_2$	$14_{10} = 1110_2$
2 $x = 467_{10} = 1'1101'0011_2$	$29_{10} = 1'1101_2$

1. Заголовок: `unsigned long long removeEverySecondDigit(unsigned long long x)`
2. Назначение: возвращает преобразованное число `x`, в котором удаляется каждая вторая цифра в двоичной записи числа, начиная со старших цифр.

```
#include "9func.h"

/*
 * Заголовок: unsigned long long removeEverySecondDigit(unsigned long long x)
 * Назначение: возвращает преобразованное число x, в котором удаляется каждая вторая
 *             цифра в двоичной записи числа, начиная со старших цифр.
 */
unsigned long long removeEverySecondDigit(unsigned long long x) {
    unsigned long long variantA = 0, variantB = 0;

    int i;
    for (i = 0; x; x >>= 1, i++) {
        unsigned long long digit = x & 1;

        if (i & 1)
            variantA += digit << (i >> 1);
        else
            variantB += digit << (i >> 1);
    }

    if (i & 1)
        return variantB;
    else
        return variantA;
}
```

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics/blob/c/lab7/sharedfuncs/src/9func.c>

## 6. Задача №10

**\*\* Дано целое неотрицательное число. Получить число перестановкой битов каждого байта данного числа в обратном порядке.**

[illegible]

1. Заголовок: `unsigned long long getInvertedByte(unsigned long long x)`
2. Назначение: возвращает преобразованное значение `x` переставляя в обратном порядке байты значения `x` в обратном порядке

```
#include "10func.h"

#define BYTE_FULL_BLOCK 255
#define BYTE_BLOCK_LEN 8

/*
 * Заголовок: unsigned long long getInvertedByte(unsigned long long x)
 * Назначение: возвращает преобразованное значение x переставляя в обратном порядке
 *             байты значения x в обратном порядке
 */
unsigned long long getInvertedByte(unsigned long long x) {
    unsigned long long reversedByteNumber = 0;

    while (x) {
        reversedByteNumber = (reversedByteNumber << BYTE_BLOCK_LEN) +
                             (x & BYTE_FULL_BLOCK);
        x >>= BYTE_BLOCK_LEN;
    }

    return reversedByteNumber;
}
```

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics/blob/c/lab7/sharedfuncs/src/10func.c>



## 7. Пакеты с монетами (1037A)

```
#include <stdio.h>


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

    int cashBags = 0;

    while (cash != 0) {
        // Ну надо же хотя бы один побитовый оператор вставить
        // По побитовым операциям же лаба `\_(\`)/`
        cash >>= 1;
        cashBags++;
    }

    printf("%d", cashBags);

    return 0;
}
```

№	Отправитель	Задача	Язык	Вердикт	Время	Память	Отослано	Протест.		
181974575	Дорешивание: VladOS4052	<a href="#">1037A</a> - 33	GNU C11	Полное решение	15 мс	12 КБ	2022-11- 21 18:33:36	2022-11- 21 18:33:36		<input type="button" value="Сравнить"/>

Ссылка на репозиторий: <https://github.com/IAmProgrammist/programming-and-algorithmization-basics/blob/c/lab7/1037A.c>

## **Код-ревью:**

Упростить первую задачу

## **Работа над ошибками (код-ревью)**

Упростил

**Вывод:** в ходе выполнения лабораторной работы получены навыки работы с побитовыми операциями.