

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное автономное образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ
ПРЕПОДАВАТЕЛЬ

ассистент

должность, уч. степень, звание

подпись, дата

Синёв Н. И.

инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №7

ПОРАЗРЯДНАЯ ОБРАБОТКА ЦЕЛЫХ ЧИСЕЛ

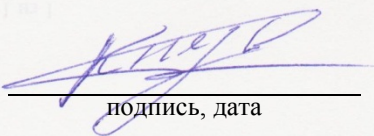
КУРС 1

по курсу: ОСНОВЫ ПРОГРАММИРОВАНИЯ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. №

1941



подпись, дата

Князюк Р.А.

инициалы, фамилия

Санкт-Петербург

2020

1. Постановка задачи.

В длинном целом числе N поменять байты в зеркальном порядке.

2. Формализация.

- **Список переменных в данной программе:**

- Переменные типа **long long (8 байт)**:
 - test_1* – переменная, хранящая обрабатываемое число;
 - temp_1, temp_2* – переменные для перестановки байтов;
 - mask_1* – для считывания левого байта (в процессе выполнения кода сдвигается вправо на 8 бит(1 байт));
 - mask_2* – для считывания самого правого байта (в процессе выполнения кода сдвигается влево на 8 бит(1 байт));
 - support_mask* – для удаления 1 при сдвиге вправо;
- Переменные типа **int**:
 - i* – счетчик, используемый в циклах для работы;

- Все данные вводятся с клавиатуры.
- Для удобства просмотра данных в консоли изменим ее размер перед выполнением программы.
- Алгоритм для вывода числа в двоичном виде будет отдельной подпрограммой.
- В качестве длинного целого числа будем использовать переменную **long long** (знакового типа);
- Так как необходимо поменять байты в зеркальном порядке, воспользуемся двумя масками. *mask_1=0xff00000000000000* (первый байт заполнен единицами, остальные нулями), *mask_2=0x00000000000000ff* (последний байт заполнен 1, а остальные 0), *support_mask=0x00ffffffffffff*. Маски нужны для извлечения байтов с правой и левой стороны
- Обработку числа организуем циклом for (так как известно кол-во итераций), количество итераций равно 4(количеству байтов типа **long long** деленное на 2).
- В цикле, сначала через побитовое умножение получаем 8 и 1 байт и сохраняем их в *temp_1, temp_2*. После меняем местами полученные байты сдвигом. Обнуляем байты в числе, после чего складываем с числом наши выделенные байты. Таким образом мы переставили 1 и 8 байт. Аналогично для 2 и 7, 3 и 6, 4 и 5.
- Для выделения необходимых байтов будем сдвигать маски. *mask_1* вправо, *mask_2* влево.
- После ввода числа, программа выведет его в двоичном формате и начнет обработку.

- В конце программа должны вывести обработанное число в двоичном формате;
- Операционная система: Windows 7 Максимальная
- Написание кода: Visual Studio Code
- Компилятор: MinGW GCC-8.2.0-5
- Запуск программы через *.exe

Таблица 1 – Тестовые примеры

Пример	№1	№2
Данные	1280 (00000000 00000000 00000000 00000000 00000000 00000000 00000101 00000000)	675539944105574400 (00001001 01100000 00000000 00000000 00000000 00000000 00000000 00000000)
Результат	00000000 00000101 00000000 00000000 00000000 00000000 00000000 00000000	00000000 00000000 00000000 00000000 00000000 00000000 01100000 00001001

3. Блок схемы программы.

а. Подпрограмма вывода числа в двоичном виде.

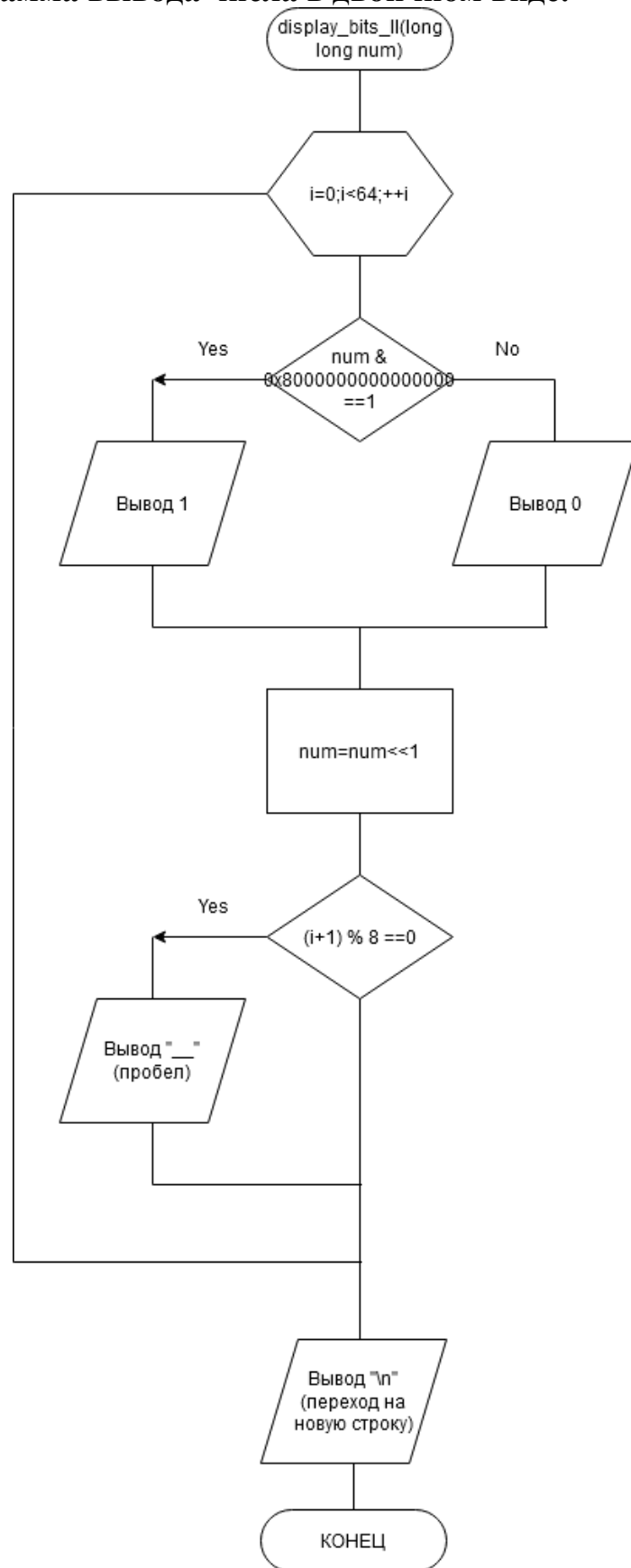


Рисунок 1 – подпрограммы вывода
`display_bits_ll(long long num)`

в. Основная программа.

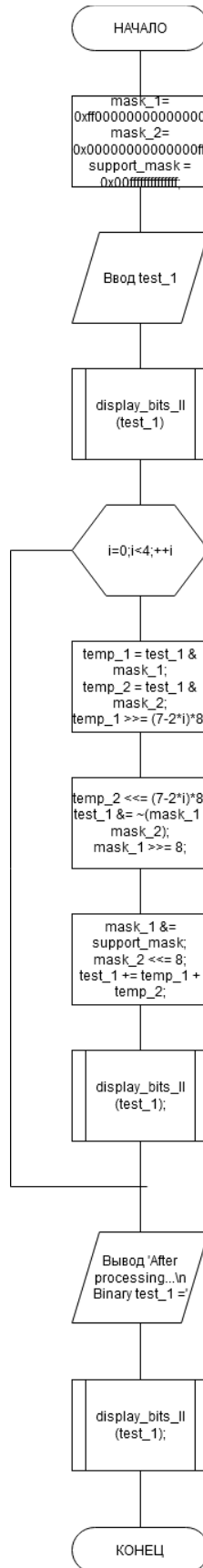


Рисунок 2 – Блок-схема основной программы

4. Код программы на Си.

```
#include <stdio.h>
#include <stdlib.h>
//-----
//-----
void display_bits_ll(long long num){
    int i;
    for(i=0;i<64;++i){
        //0x8000000000000000 в двоичной версии 100...000 (63 нуля)
        printf("%d", (num & 0x8000000000000000) ? 1: 0);
        num<<=1;
        if((i+1) % 8 == 0)
            printf(" ");
    }
    printf("\n");
}
//-----
//-----
int main(void){
    //объявляем переменные
    long long test_1,temp_1,temp_2,
        mask_1=0xff00000000000000,
        mask_2=0x00000000000000ff,
        //support_mask необходимо для сдвига вправо
        support_mask=0x00ffffffffffffff;
    int i;

    //меняем размер консоли
    system("mode con cols=110 lines=25");

    //ввод данных
    printf("Enter test number.\ntest_1 (long long) = ");
    scanf("%lld",&test_1);

    //вывод в двоичном виде
    printf("Binary test_1 = ");
    display_bits_ll(test_1);

    //обработка числа
    for(i=0;i<4;++i){
        temp_1 = test_1 & mask_1;
        temp_2 = test_1 & mask_2;
        temp_1 >>= (7-2*i)*8;
        temp_2 <<= (7-2*i)*8;
        test_1 &= ~(mask_1 | mask_2);
        mask_1 >>= 8;
        mask_1 &= support_mask;
        mask_2 <<= 8;
        test_1 += temp_1 + temp_2;
        display_bits_ll(test_1);
    }
```

```
}  
  
//ВЫВОДИМ В ДВОИЧНОМ ВИДЕ  
printf("After processing...\nBinary test_1 = ");  
display_bits_ll(test_1);  
  
system("pause");  
return 0;  
}
```

5. Проверка работы программы на тестовых примерах.

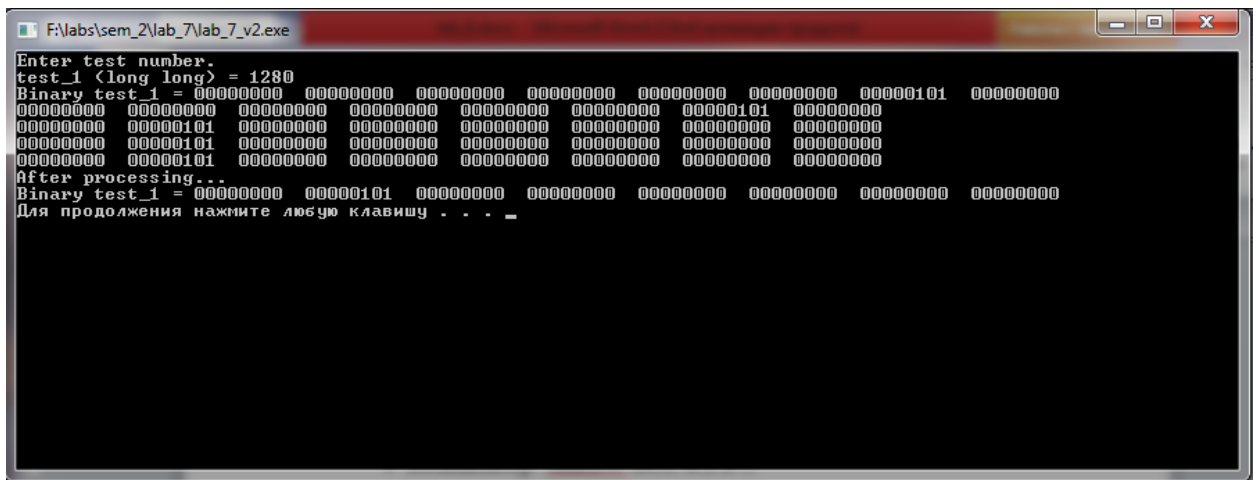


Рисунок 3 – Тестовый пример 1

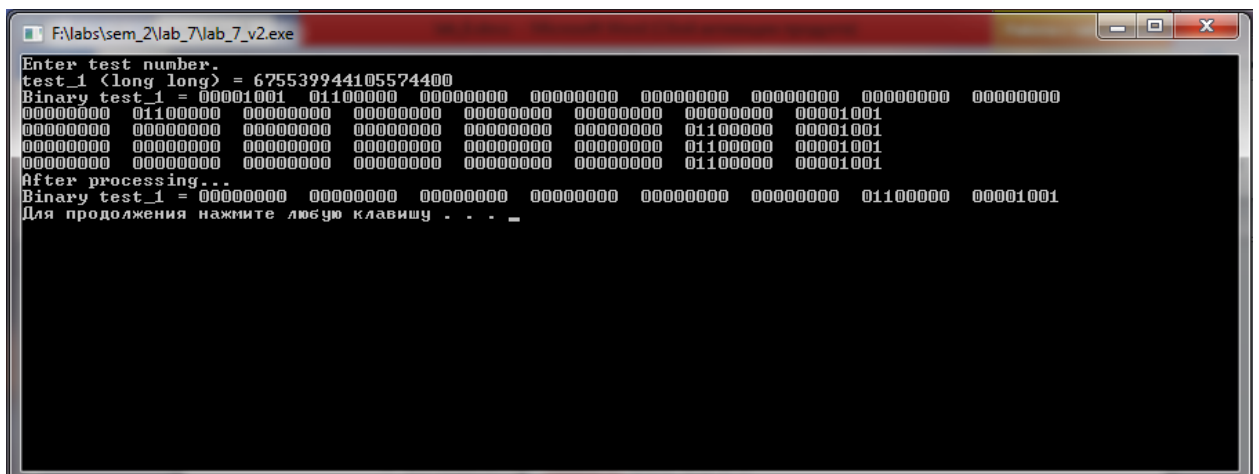


Рисунок 4 – Тестовый пример 2

6. Вывод.

Тестовые примеры (5 пункт отчета) доказывают правильность работы алгоритма и программы. Таким образом, программа способна в длинном числе зеркально поменять порядок байтов, используя поразрядную обработку чисел с использованием масок.