**Министерство науки и высшего образования Российской Федерации**
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Факультет безопасности информационных технологий**

**Дисциплина:**

**«Программирование»**

# ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«*Поразрядные и логические операции*»

Вариант 7–14

**Выполнил:**

студент гр. N3146

Новосельский Андрей Сергеевич

(подпись)

**Проверил:**

**Преподаватель по**

**Программированию факультета БИТ**

Грозов Владимир Андреевич

_____

(отметка о выполнении)

_____

(подпись)

Санкт-Петербург

2025 г.

**СОДЕРЖАНИЕ**

## 1 ЗАДАНИЕ

Вариант 7-14. Программа получает на вход целое беззнаковое число типа uint64_t. Назовем триплетом группу из трех битов. В каждом третьем триплете, начиная с младшего, изменить порядок следования битов на обратный.

## 2 MAKEFILE

Листинг 1 – Исходный код Makefile

```
.PHONY: all clean debug

APP=prg1asnN3146
CFLAGS=-Wall -Wextra -Werror

all: $(APP)

$(APP): $(APP).c
gcc -o $(APP) $(CFLAGS) $(APP).c

debug: $(APP).c
gcc -o $(APP) $(CFLAGS) -g $(APP).c

clean:
rm $(APP)
```

## 3 ПРИМЕРЫ РАБОТЫ ПРОГРАММЫ

Рисунок 1 – Пример работы программы с заданным числом

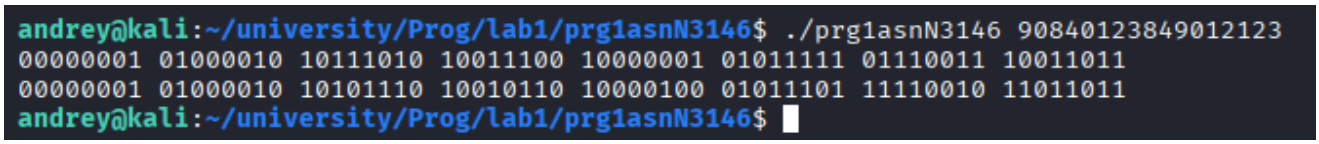



Рисунок 2 – Пример работы программы без заданного числа (со случайным значением)

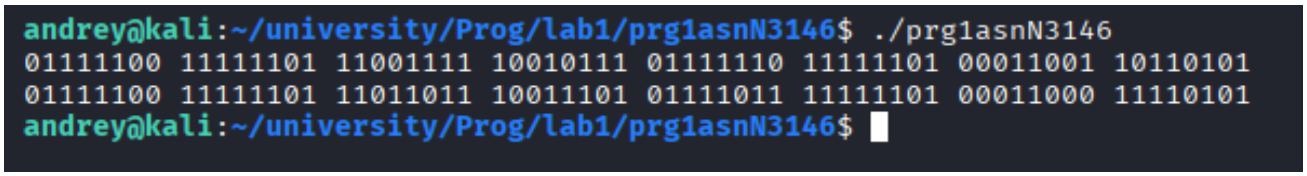



Рисунок 3 – Пример ошибки при вводе некорректного значения

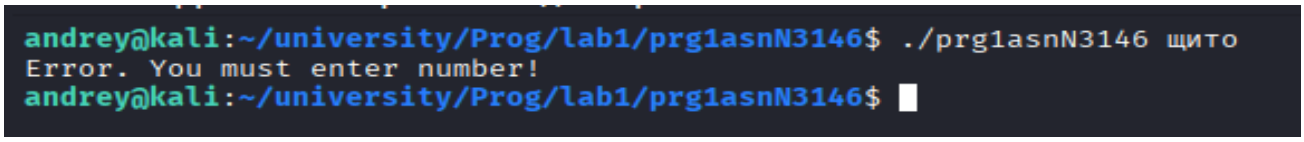
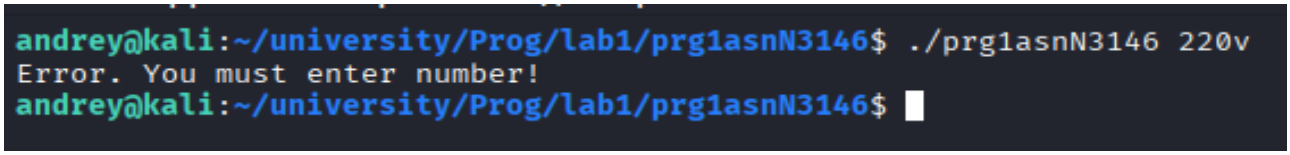
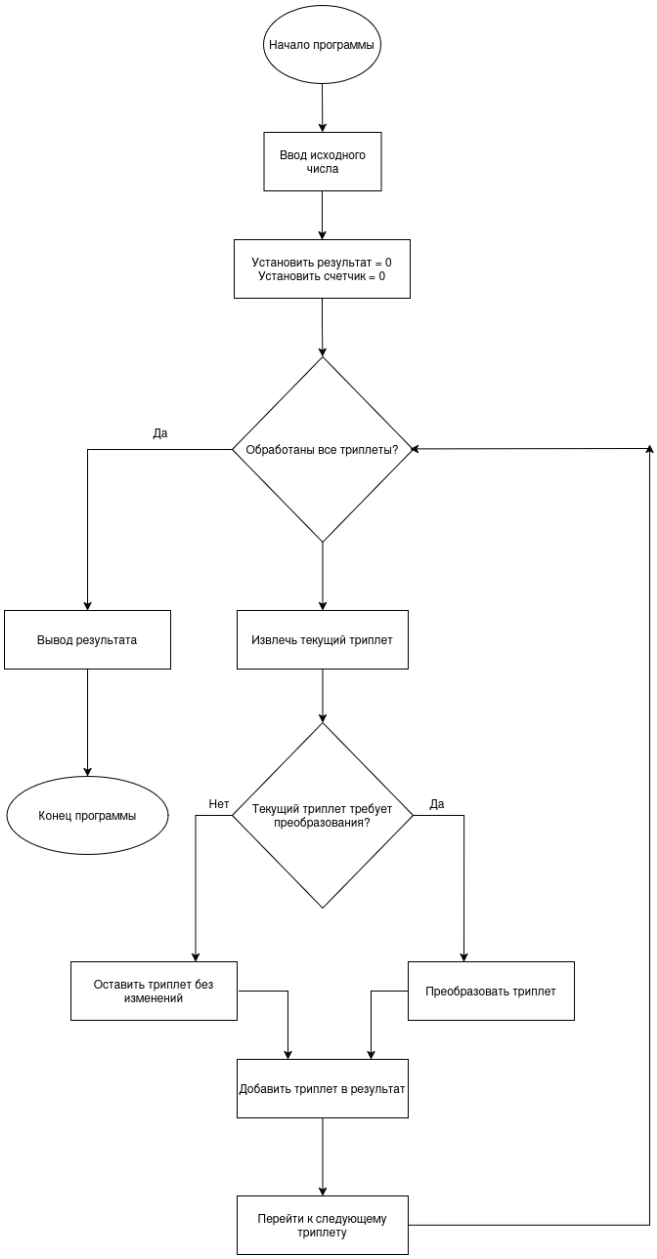



Рисунок 4 – Пример ошибки при вводе некорректного значения



## 4 БЛОК-СХЕМА АЛГОРИТМА ПРЕОБРАЗОВАНИЯ

Рисунок 5 – Блок-схема алгоритма преобразования



## 5 ИСХОДНЫЙ КОД ПРОГРАММЫ

Листинг 2 – Исходный код программы prg1asnN3146

```c
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <time.h>

uint64_t is_number(char* argv[])
{
    uint64_t number;
    char symbol;
    int result = sscanf(argv[1], "%lu%c", &number, &symbol);

    if (result == 1) {
        return number;
    } else {
        printf("Error. You must enter number!\n");
        exit(EXIT_FAILURE);
    }
}
```

```c
uint64_t check_available_input(int argc, char* argv[])
{
    if (argc > 2) {
        printf("Error. You must enter only one argument!\n");
        exit(EXIT_FAILURE);
    }

    if (argc == 1) {
        srand(time(NULL));

        uint64_t number =
            ((uint64_t) rand()) << 48 |
            ((uint64_t) rand()) << 32 |
            ((uint64_t) rand()) << 16 |
            ((uint64_t) rand());

        return number;
    }

    uint64_t number = is_number(argv);
    return number;
}
void get_binary_number(uint64_t number)
{
    int bits = sizeof(number) * 8;

    // Проходим по всем битам от старшего к младшему
    for (int i = bits - 1; i >= 0; i--) {
        // Получаем 1-й бит с помощью сдвига и маски
        int bit = (number >> i) & 1;
        printf("%d", bit);

        // Добавляем пробел для удобства чтения каждые 8 бит
        if (i % 8 == 0 && i != 0) printf(" ");
    }
    printf("\n");

}
```

```c
uint64_t reverse_triplets(uint64_t original)
{
    uint64_t result = 0;

    // Проходим по всем триплетам
    for (int i = 0; i < 21; i++) {

        // Извлекаем текущий триплет
        uint64_t triplet = (original >> (i * 3)) & 0x7;

        // Проверяем, нужно ли инвертировать порядок битов
        if ((i + 1) % 3 == 0) {
            // Инвертируем порядок битов в триплете
            uint64_t reversed_triplet = 0;
            reversed_triplet |= (triplet & 0x1) << 2; // Младший бит становится старшим
            reversed_triplet |= (triplet & 0x2);      // Средний бит остается на месте
            reversed_triplet |= (triplet & 0x4) >> 2; // Старший бит становится младшим

            triplet = reversed_triplet;
        }

        // Добавляем обработанный триплет в результат
        result |= (triplet << (i * 3));
    }

    return result;
}
```

```c
int main(int argc, char* argv[])
{
    uint64_t number = check_available_input(argc, argv);
    get_binary_number(number);
    number = reverse_triplets(number);
    get_binary_number(number);

    return 0;
}
```

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ 19.701–90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения: дата введения 1992-01-01. – Москва: Стандартинформ, 2010. – 24 с.;