



МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение  
высшего образования

«МИРЭА – Российский технологический университет»

**РТУ МИРЭА**

---

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных  
технологий

## **ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 2.1**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

**Тема: «Сортировка числового файла с помощью битового массива»**

Выполнил студент группы ИКБО-30-22

Сенькевич Г.Д.

Принял преподаватель

Красников С.А.

Лабораторная работа выполнена

«\_\_»\_\_\_\_\_202\_\_ г.

*(подпись студента)*

«Зачтено»

«\_\_»\_\_\_\_\_202\_\_ г.

*(подпись руководителя)*

Москва 2023

## 1. Цель работы

Освоить приёмы работы с битовым представлением беззнаковых целых чисел, реализовать эффективный алгоритм внешней сортировки на основе битового массива.

## 2. Ход работы

### 2.1. Формулировка задачи

#### Задание 1:

**1.а.** Реализуйте вышеприведённый пример, проверьте правильность результата в том числе и на других значениях  $x$ .

**1.б.** Реализуйте по аналогии с предыдущим примером установку 7-го бита числа в единицу.

**1.в.** Реализуйте код листинга 1, объясните выводимый программой результат.

#### Задание 2:

**2.а.** Реализуйте вышеописанный пример с вводом произвольного набора до 8-ми чисел (со значениями от 0 до 7) и его сортировкой битовым массивом в виде числа типа `unsigned char`. Проверьте работу программы.

Если количество чисел в исходной последовательности больше 8 и/или значения превосходят 7, можно подобрать тип беззнакового числа для битового массива с подходящим размером разрядной сетки – до 64 в типе `unsigned long long` (см. табл. 1).

**2.б.** Адаптируйте вышеприведённый пример для набора из 64-х чисел (со значениями от 0 до 63) с битовым массивом в виде числа типа `unsigned long long`.

Если количество чисел и/или их значения превосходят возможности разрядной сетки одного беззнакового целого числа, то можно организовать линейный массив (вектор) таких чисел, который в памяти ЭВМ будет представлен **одной непрерывной битовой последовательностью**.

**2.в.** Исправьте программу задания 2.б, чтобы для сортировки набора из 64-х чисел использовалось не одно число типа `unsigned long long`, а линейный массив чисел типа `unsigned char`.

## 2.2. Математическая модель решения

Для решения задачи были использованы такие операции как битовый сдвиг и побитовые логические операции.

### 1. Битовый сдвиг:

- Левый битовый сдвиг (<<): Эта операция сдвигает биты числа влево на указанное количество позиций.
- Правый битовый сдвиг (>>): Эта операция сдвигает биты числа вправо на указанное количество позиций.

### 2. Побитовые логические операции:

- Побитовое И (&): Эта операция выполняет логическое И между каждой парой битов двух чисел. Результат равен 1, только если оба бита равны 1.
- Побитовое ИЛИ (|): Эта операция выполняет логическое ИЛИ между каждой парой битов двух чисел. Результат равен 1, если хотя бы один из битов равен 1.

С помощью этих операций программист получает возможность взаимодействовать с отдельными битами чисел, что позволяет реализовать сортировку с использованием битового массива.

Сортировка с использованием битового массива — это метод, который позволяет упорядочить набор элементов путем использования битов для отображения присутствия или отсутствия каждого элемента в отсортированном массиве. Вот как это происходит:

1. **Инициализация битового массива:** создается битовый массив, который будет использоваться для отслеживания присутствия элементов в исходном массиве. Размер битового массива определяется максимальным значением элементов в исходном массиве.
2. **Установка битов:** для каждого элемента в исходном массиве происходит установка соответствующего бита в битовом массиве. Например, если элемент в исходном массиве равен 5, то бит с индексом 5 в битовом массиве устанавливается в 1, что указывает на то, что элемент присутствует в исходном массиве.
3. **Извлечение элементов:** после того как все элементы из исходного массива были обработаны и битовый массив правильно настроен, можно начать извлечение элементов в отсортированном порядке. Для этого проходят по битовому массиву и извлекают элементы,

соответствующие установленным битам. Эти элементы извлекаются в порядке возрастания индексов битов.

4. **Сортированный массив:** получается отсортированный массив, в котором элементы упорядочены по возрастанию.

### 2.3. Код программы

Решение задания 1.а. приведено в листинге 1. Тестирование на других значениях *x* приводится в разделе 2.4.

Листинг 1 — Решение задания 1.а.

```
#include <iostream>

int main()
{
    unsigned char x = 255;
    unsigned char mask = 1;
    x = x & ~(mask << 4);
    std::cout << (int)x;
    return 0;
}
```

Решение задания 1.б. приведено в листинге 2. Для того чтобы установить 7 бит числа в единицу нам нужно сдвинуть маску (изначально равную 1) на 6 позиций влево и провести с ней побитовое «или». Таким образом 7 биту присвоится значение операции «или» от 1 и значения, которое там находилось, что в любом случае даст 1.

Листинг 2 — Решение задания 1.б.

```
#include <iostream>

int main()
{
    unsigned char x = 0;
    unsigned char mask = 1;
    x = x | (mask << 6);
    std::cout << (int)x;
    return 0;
}
```

Решение задания 1.в. приведено в листинге 3. Этот код выводит биты числа 25 с помощью маски. Единичный бит маски последовательно сдвигается из крайней левой позиции направо, и на каждой итерации цикла с помощью маски происходит получение очередного бита числа  $x$  и вывод его на экран.

Листинг 3 — Решение задания 1.в.

```
#include <cstdlib>
#include <iostream>
#include <Windows.h>
#include <bitset>

using namespace std;

int main()
{
    SetConsoleCP(1251);
    SetConsoleOutputCP(1251);

    unsigned int x = 25;
    const int n = sizeof(int) * 8;
    unsigned maska = (1 << n - 1);
    cout << "Начальный вид маски: " << bitset<n> (maska) << endl;
    cout << "Результат: ";

    for (int i = 1; i <= n; i++)
    {
        cout << ((x & maska) >> (n - i));
        maska = maska >> 1;
    }
    cout << endl;

    system("pause");
    return 0;
}
```

Решение задания 2.а. приведено в листинге 4. В первом цикле происходит последовательное считывание чисел и запись их в битовый массив (запись единиц на позиции с номером считанных чисел). Во втором цикле происходит последовательная проверка битов битового массива и вывод позиций единичных битов. Таким образом на экран выводится отсортированный массив.

#### Листинг 4 — Решение задания 2.а.

```
#include <iostream>
//#include <bitset>

typedef unsigned char byte;

int main()
{
    const size_t size = 8;
    byte bit_array = 0;

    for (int i = 0; i < size; i++)
    {
        int buff = 0;
        std::cin >> buff;

        byte mask = 1;
        bit_array = bit_array | (mask << buff);
    }

    //std::cout << std::bitset<size> (bit_array);

    byte mask = 1;
    for (int i = 0; i < size; i++)
    {
        if (bit_array & (mask << i))
        {
            std::cout << i << ' ';
        }
    }
    std::cout << std::endl;

    return 0;
}0;
}
```

Решение задачи 2.б. приведено в листинге 5. Решение аналогично решению задания 2.а., с отличием в размере массива и границах допустимых значений, а соответственно и типе переменной, используемой для представления битового массива (64 бита вместо 8).

## Листинг 5 — Решение задания 2.б.

```
#include <iostream>
//#include <bitset>

typedef unsigned long long ull;

int main()
{
    const size_t size = 64;
    ull bit_array = 0;

    for (int i = 0; i < size; i++)
    {
        ull buff = 0;
        std::cin >> buff;

        ull mask = 1;
        bit_array = bit_array | (mask << buff);
    }

    //std::cout << std::bitset<size> (bit_array);

    ull mask = 1;
    for (int i = 0; i < size; i++)
    {
        if (bit_array & (mask << i))
        {
            std::cout << i << ' ';
        }
    }
    std::cout << std::endl;

    return 0;
}
```

Решение задания 2.в. приведено в листинге 6. Решение аналогично решениям заданий 2.а. и 2.б. Отличие заключается в том что битовый массив реализуется не через одну числовую переменную, а через массив 1-байтовых числовых переменных, каждая из которых представляет собой отдельный битовый массив. Это создаёт накладные расходы для записи числа в массив, так как сначала нужно вычислить индекс элемента, в который нужно записать число, а только потом позицию бита, в который нужно установить

единицу. Последующий вывод чисел из массива также сопровождается дополнительными вычислениями.

#### Листинг 6 — Решение задания 2.в.

```
#include <iostream>

typedef unsigned long long ull;
typedef unsigned char byte;

int main()
{
    const size_t size = 64;
    const size_t byte_size = sizeof(byte) * 8;
    const size_t byte_array_size = size / byte_size;
    byte* byte_array = new byte[byte_array_size];
    for (int i = 0; i < byte_array_size; i++)
    {
        byte_array[i] = 0;
    }

    for (int i = 0; i < size; i++)
    {
        ull buff = 0;
        std::cin >> buff;

        size_t index = buff / byte_size;
        byte shift = buff % byte_size;
        byte mask = 1;
        byte_array[index] = byte_array[index] | (mask << shift);
    }

    byte mask = 1;
    for (int i = 0; i < byte_array_size; i++)
    {
        for (int j = 0; j < byte_size; j++)
        {
            if (byte_array[i] & (mask << j))
            {
                std::cout << i * byte_size + j << ' ';
            }
        }
    }

    std::cout << std::endl;

    delete[] byte_array;
    return 0;
}
```



```
}
```

Решение задач 3.а. и 3.б. приведено в листинге 7. Решение аналогично решению задания 2.в. с отличием в том что размер массива вместо 64 бит составляет 1 Мб, а чтение и вывод информации происходят с помощью файловых потоков, а не стандартных. Также замеряется время, затрачиваемое на чтение информации из файла, записи этой информации в битовый массив (что одновременно является и сортировкой этих данных) и записи отсортированного массива в выходной файл.

Листинг 7 — Решение заданий 3.а. и 3.б.

```
#include <iostream>
#include <fstream>
#include <string>
#include <chrono>

typedef unsigned long long ull;
typedef unsigned char byte;

int main()
{
    const size_t size = 8388608; // 1 Mb in bits
    const size_t byte_size = sizeof(byte) * 8;
    const size_t byte_array_size = size / byte_size;
    byte* byte_array = new byte[byte_array_size];
    for (int i = 0; i < byte_array_size; i++)
    {
        byte_array[i] = 0;
    }

    std::cout << "Enter filename: ";
    std::string filename;
    std::cin >> filename;
    std::ifstream input_stream(filename);

    auto start = std::chrono::high_resolution_clock::now();

    ull buff = 0;
    while (input_stream >> buff)
    {
        size_t index = buff / byte_size;
        byte shift = buff % byte_size;
```

```

        byte mask = 1;
        byte_array[index] = byte_array[index] | (mask << shift);
    }

    std::ofstream output_stream("out.txt");
    byte mask = 1;
    for (int i = 0; i < byte_array_size; i++)
    {
        for (int j = 0; j < byte_size; j++)
        {
            if (byte_array[i] & (mask << j))
            {
                output_stream << i * byte_size + j << ' ';
            }
        }
    }

    auto stop = std::chrono::high_resolution_clock::now();

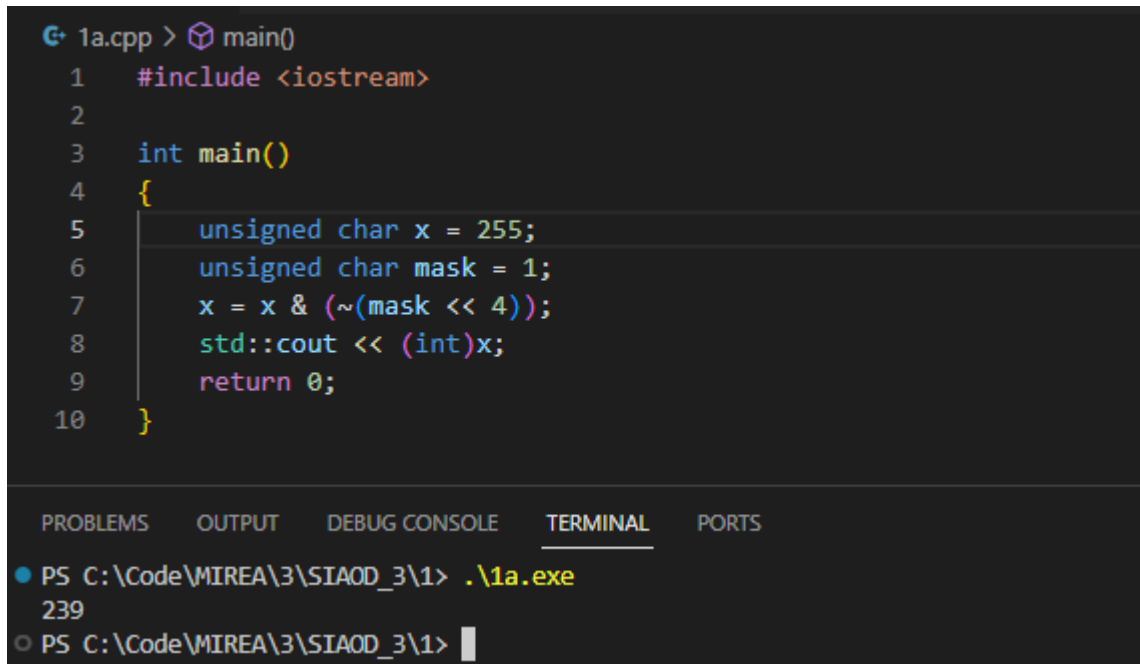
    auto duration =
std::chrono::duration_cast<std::chrono::milliseconds>(stop -
start);
    std::cout << "Reading, sorting and writing took " <<
duration.count() / 1000.0 << " seconds\n";

    delete[] byte_array;
    return 0;
}

```

## 2.4. Результаты тестирования

Результаты тестирования решения задания 1.а. представлены на рисунках 1 и 2.



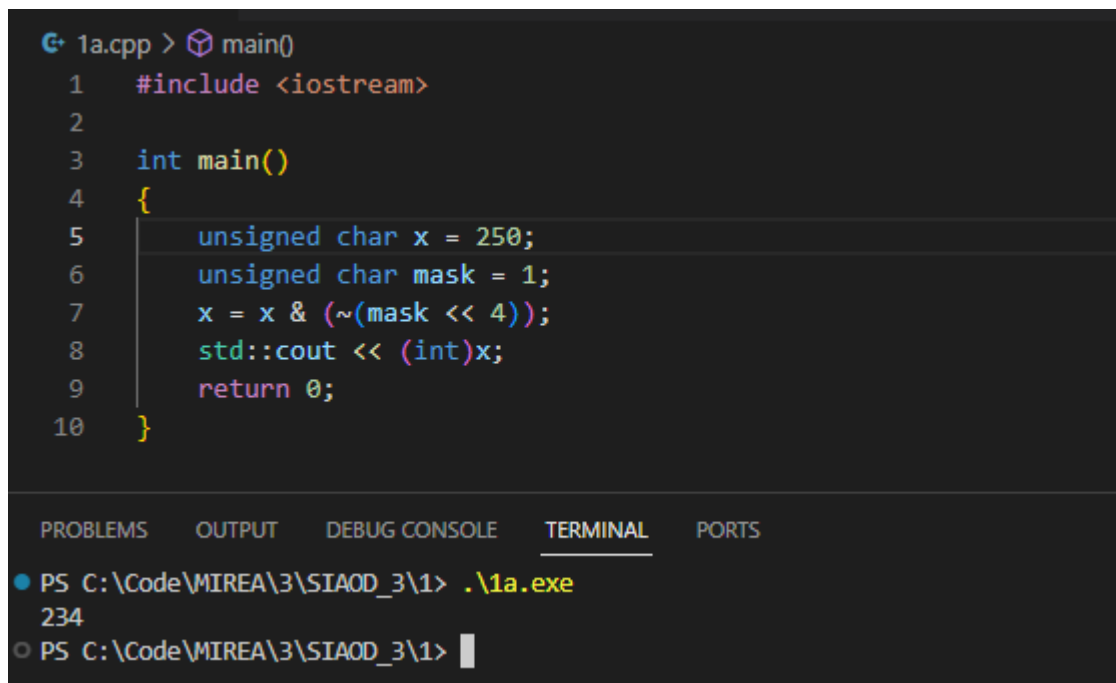
```
1a.cpp > main()
1  #include <iostream>
2
3  int main()
4  {
5      unsigned char x = 255;
6      unsigned char mask = 1;
7      x = x & ~(mask << 4);
8      std::cout << (int)x;
9      return 0;
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Code\MIREA\3\SIAOD\_3\1> .\1a.exe  
239

○ PS C:\Code\MIREA\3\SIAOD\_3\1> █

Рисунок 1 — Тестирование решения задания 1.а.



```
1a.cpp > main()
1  #include <iostream>
2
3  int main()
4  {
5      unsigned char x = 250;
6      unsigned char mask = 1;
7      x = x & ~(mask << 4);
8      std::cout << (int)x;
9      return 0;
10 }
```

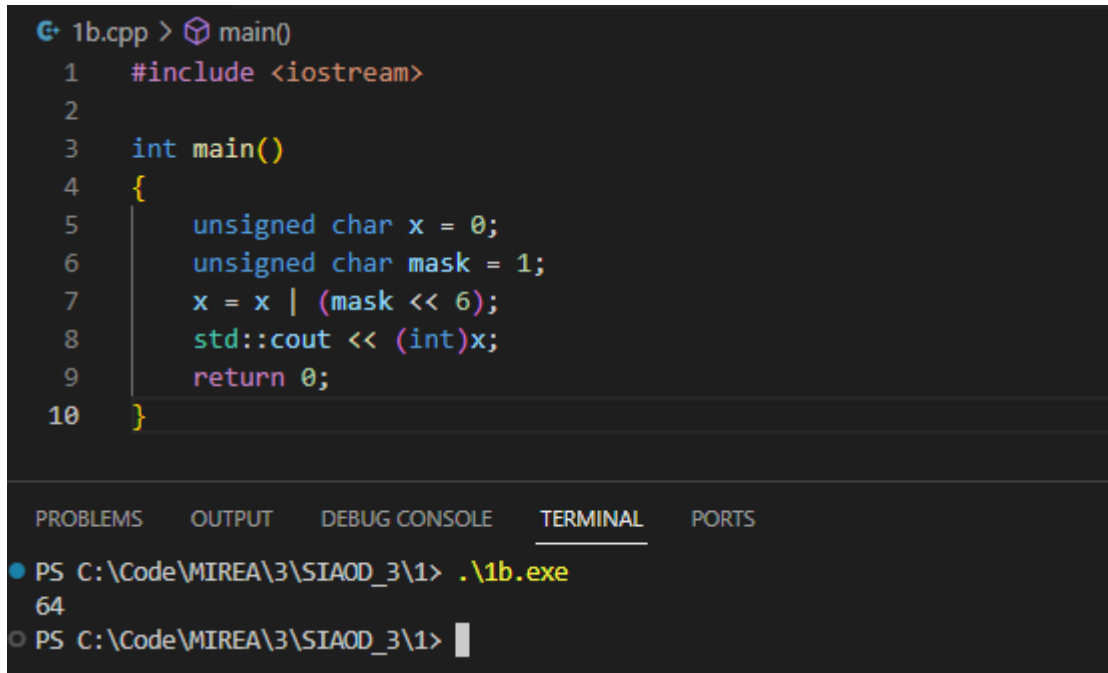
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

● PS C:\Code\MIREA\3\SIAOD\_3\1> .\1a.exe  
234

○ PS C:\Code\MIREA\3\SIAOD\_3\1> █

Рисунок 2 — Тестирование решения задания 1.а.

Результаты тестирования решения задания 1.б. представлены на рисунке 3.



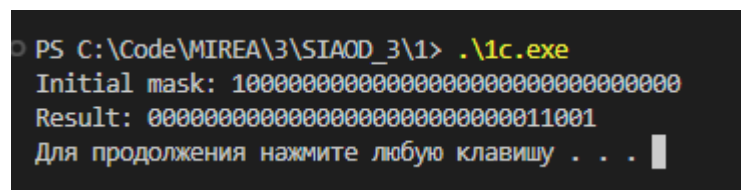
```
1b.cpp > main()
1  #include <iostream>
2
3  int main()
4  {
5      unsigned char x = 0;
6      unsigned char mask = 1;
7      x = x | (mask << 6);
8      std::cout << (int)x;
9      return 0;
10 }
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Code\MIREA\3\SIAOD_3\1> .\1b.exe
64
PS C:\Code\MIREA\3\SIAOD_3\1>
```

Рисунок 3 — Тестирование решения задания 1.б.

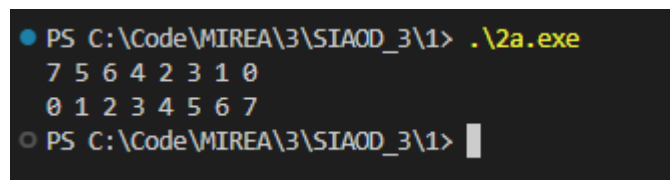
Результаты тестирования решения задания 1.в. представлены на рисунке 4.



```
PS C:\Code\MIREA\3\SIAOD_3\1> .\1c.exe
Initial mask: 10000000000000000000000000000000
Result: 000000000000000000000000000011001
Для продолжения нажмите любую клавишу . . .
```

Рисунок 4 — Тестирование решения задания 1.в.

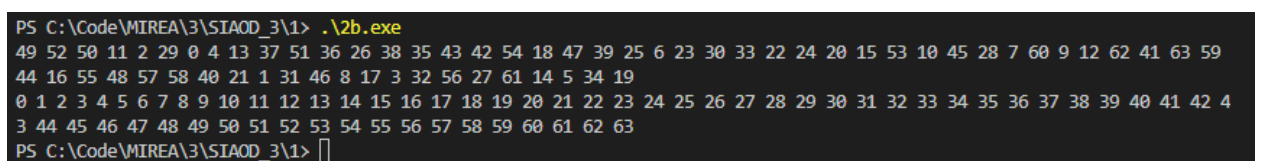
Результаты тестирования решения задания 2.а. представлены на рисунке 5.



```
PS C:\Code\MIREA\3\SIAOD_3\1> .\2a.exe
7 5 6 4 2 3 1 0
0 1 2 3 4 5 6 7
PS C:\Code\MIREA\3\SIAOD_3\1>
```

Рисунок 5 — Тестирование решения задания 2.а.

Результаты тестирования решения задания 2.б. представлены на рисунке 6.



```
PS C:\Code\MIREA\3\SIAOD_3\1> .\2b.exe
49 52 50 11 2 29 0 4 13 37 51 36 26 38 35 43 42 54 18 47 39 25 6 23 30 33 22 24 20 15 53 10 45 28 7 60 9 12 62 41 63 59
44 16 55 48 57 58 40 21 1 31 46 8 17 3 32 56 27 61 14 5 34 19
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 4
3 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
PS C:\Code\MIREA\3\SIAOD_3\1>
```

Рисунок 6 — Тестирование решения задания 2.б.

Результаты тестирования решения задания 2.в. представлены на рисунке 7.

```
PS C:\Code\MIREA\3\SIAOD_3\1> .\2c.exe
49 52 50 11 2 29 0 4 13 37 51 36 26 38 35 43 42 54 18 47 39 25 6 23 30 33 22 24 20 15 53 10 45 28 7 60 9 12 62 41 63 59
44 16 55 48 57 58 40 21 1 31 46 8 17 3 32 56 27 61 14 5 34 19
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 4
3 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
PS C:\Code\MIREA\3\SIAOD_3\1>
```

Рисунок 7 — Тестирование решения задания 2.в.

Для тестирования решения заданий 3.а. и 3.б. был написан скрипт, генерирующий входной файл, состоящий из 8388608 (это количество бит в одном мегабайте) чисел в диапазоне от 0 до 8388608, записанных в случайном порядке. Скрипт представлен на рисунке 8. Результаты тестирования программы представлены на рисунках 9 и 10. Выводимое время — это время, затраченное на чтение данных из файла, запись их в битовый массив и последующий вывод из массива в выходной файл.

```
generate_file.py > ...
1  import random
2
3  ceil = 8388608
4
5  data = [i for i in range(ceil)]
6  random.shuffle(data)
7
8  out = open("inp.txt", 'w')
9  out.write(" ".join(map(str, data)))
10 out.close()
11
```

Рисунок 8 — Скрипт, генерирующий входные данные для заданий 3.а. и 3.б.

```
● PS C:\Code\MIREA\3\SIAOD_3\1> .\3a.exe
Enter filename: inp.txt
Reading, sorting and writing took 3.101 seconds
○ PS C:\Code\MIREA\3\SIAOD_3\1> |
```

Рисунок 9 — Тестирование решения заданий 3.а. и 3.б.

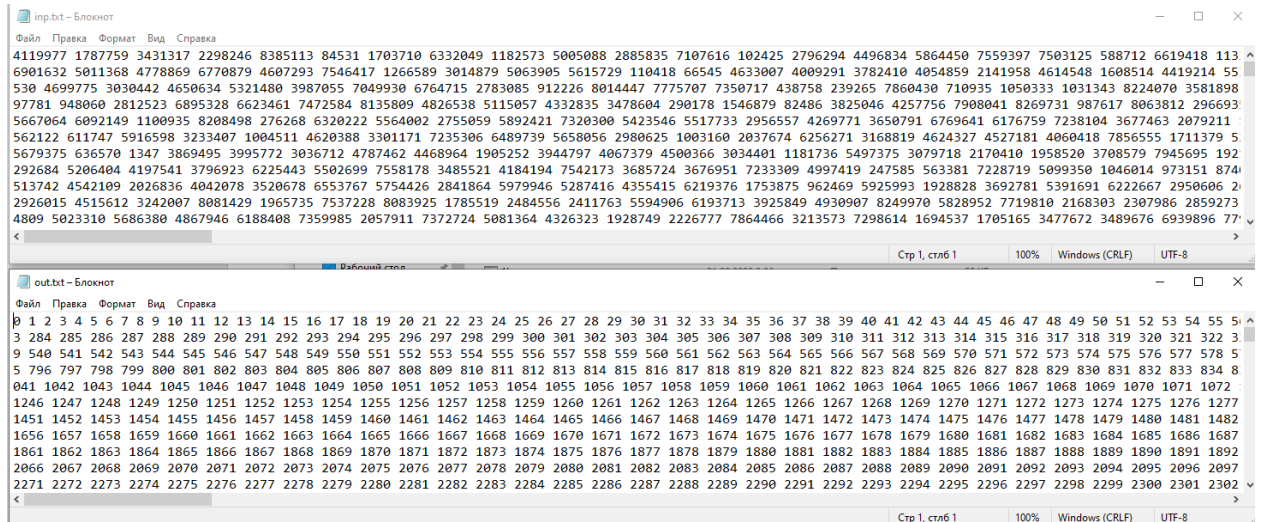


Рисунок 9 — Тестирование решения заданий 3.а. и 3.б.

Тестирование показало, что все программы работают правильно, корректно решая поставленные задачи.

### 3. Выводы

В результате выполнения работы я:

1. Освоил приёмы работы с битовым представлением беззнаковых целых чисел;
2. Реализовать эффективный алгоритм внешней сортировки на основе битового массива.

### 4. Список литературы

1. Страуструп Б. Программирование. Принципы и практика с использованием C++. 2-е изд., 2016.
2. Документация по языку C++ [Электронный ресурс]. URL: <https://docs.microsoft.com/ruru/cpp/cpp/> (дата обращения 01.09.2021).
3. Курс: Структуры и алгоритмы обработки данных. Часть 2 [Электронный ресурс]. URL: <https://online-edu.mirea.ru/course/view.php?id=4020> (дата обращения 01.09.2021).