

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

**"МИРЭА - Российский технологический университет"**

**РТУ МИРЭА**

Институт информационных технологий (ИТ)

Кафедра математического обеспечения и стандартизации информационных технологий (МОСИТ)

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №1**

**по дисциплине**

**«Структуры и алгоритмы обработки данных»**

Тема. Поразрядные операции и их применение

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-04-22 |  | Основин А.И. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

СОДЕРЖАНИЕ

[1 Задание 1 4](#_Toc145320325)

[1.1 Постановка задачи 4](#_Toc145320326)

[1.2 Упражнение 1 4](#_Toc145320327)

[1.2.1 Условие задачи 4](#_Toc145320328)

[1.2.2 Разработка решения задачи варианта 4](#_Toc145320329)

[1.2.3 Реализация решения 4](#_Toc145320330)

[1.3 Упражнение 2 5](#_Toc145320331)

[1.3.1 Условие задачи 5](#_Toc145320332)

[1.3.2 Разработка решения задачи варианта 5](#_Toc145320333)

[1.3.3 Реализация решения 5](#_Toc145320334)

[1.4 Упражнение 3 6](#_Toc145320335)

[1.4.1 Условие задачи 6](#_Toc145320336)

[1.4.2 Разработка решения задачи варианта 6](#_Toc145320337)

[1.4.3 Реализация решения 6](#_Toc145320338)

[1.5 Упражнение 4 7](#_Toc145320339)

[1.5.1 Условие задачи 7](#_Toc145320340)

[1.5.2 Разработка решения задачи варианта 7](#_Toc145320341)

[1.5.3 Реализация решения 7](#_Toc145320342)

[1.6 Упражнение 5 7](#_Toc145320343)

[1.6.1 Условие задачи 7](#_Toc145320344)

[1.6.2 Разработка решения задачи варианта 8](#_Toc145320345)

[1.6.3 Реализация решения 8](#_Toc145320346)

[2 Задание 2 9](#_Toc145320347)

[2.1 Постановка задачи 9](#_Toc145320348)

[2.2 Описание структуры данных 9](#_Toc145320349)

[2.3 Алгоритм решения 9](#_Toc145320350)

[2.4 Тестовый пример 9](#_Toc145320351)

[2.5 Код программы 9](#_Toc145320352)

[2.6 Результаты тестирования 11](#_Toc145320353)

[3 Выводы 12](#_Toc145320354)

# Задание 1

## Постановка задачи

Выполнить упражнения по применению битовых операций по изменению значений битов в ячейке оперативной памяти, созданию маски для изменения значения ячейки. Создать выражения, содержащего поразрядные операции, для выполнения определенной операции над значением ячейки.

Номер варианта: 17.

Задания варианта:

1. Установить с третьего бита четыре слева в единицу;
2. Обнулить 12-й, 11-й и 1-й биты введённого числа;
3. Умножить введённое число на 32, используя побитовые операции;
4. Поделить введённое число на 32, используя побитовые операции;
5. Обнулить n-й бит введённого числа, используя маску, инициализированную единицей в старшем разряде.

## Упражнение 1

### Условие задачи

Установить с третьего бита четыре слева в единицу.

### Разработка решения задачи варианта

Выражение:

1. unsigned short mask = 0x78; num = num | mask
2. num = num | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6)

Тестовый пример (num = 0xA02):

0000 1010 0000 0010 +

0000 0000 0111 1000

---------------------------

0000 1010 0111 1010

### Реализация решения

Код функции:

void task\_1() {

unsigned short num = 0xA02, mask = 0x78;

cout << "Number 0xA02: " << bitset<sizeof(short) \* 8>(num) << endl;

num = num | mask;

cout << "Number 0xA02 with 1-st mask:" << bitset<sizeof(short) \* 8>(num) << endl;

num = num | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);

cout << "Number 0xA02 with 2-nd mask:" << bitset<sizeof(short) \* 8>(num);

}

Результат тестирования:

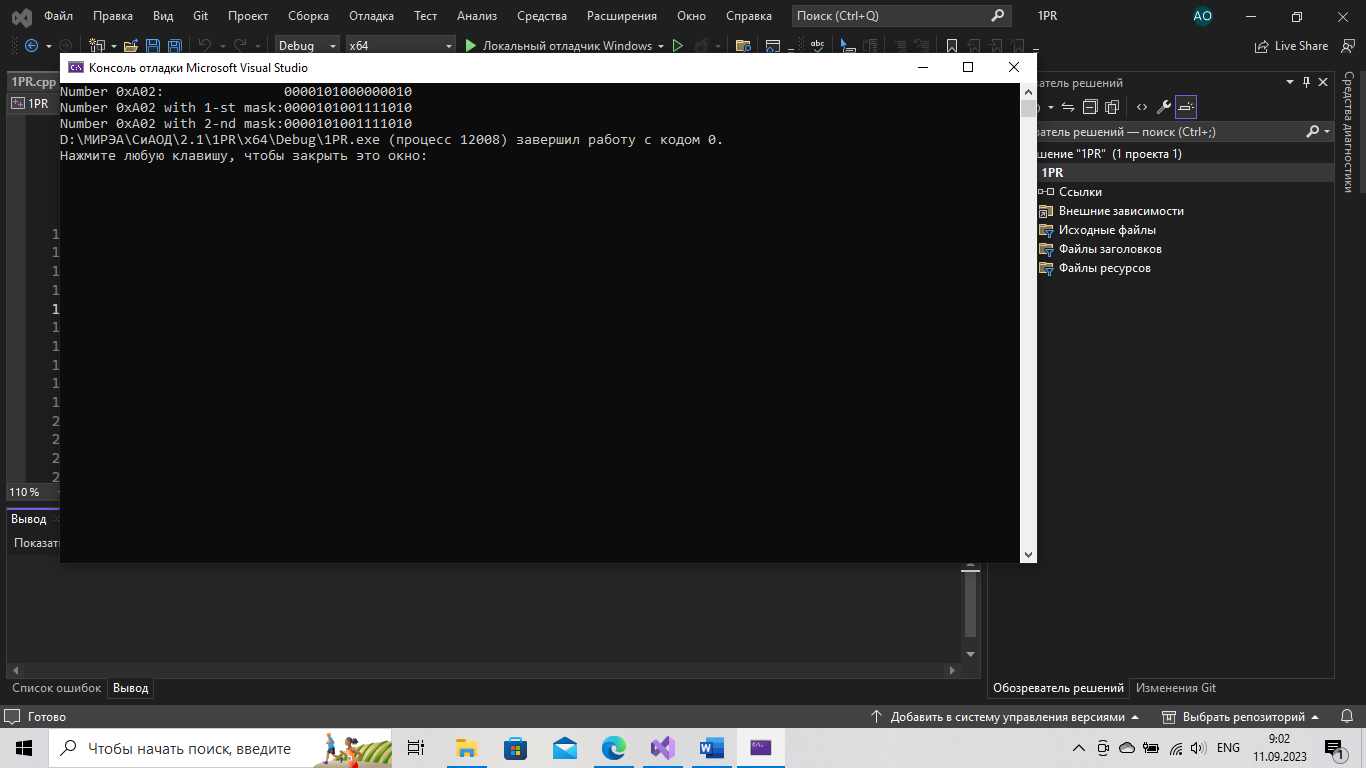


Рисунок 1 – Результат теста 1.1

## Упражнение 2

### Условие задачи

Обнулить 12-й, 11-й и 1-й биты введённого числа.

### Разработка решения задачи варианта

Выражение:

1. unsigned short mask = 0x1802; num = num & ~mask;
2. num = num & ~(1 << 12) & ~(1 << 11) & ~(1 << 1)

Тестовый пример (n = 0xFFFF):

~mask = ~(0001 1000 0000 0010) = 1110 0111 1111 1101

1111 1111 1111 1111 &

1110 0111 1111 1101

---------------------------

1110 0111 1111 1101

### Реализация решения

Код функции:

void task\_2() {

unsigned short num, mask = 0x1802;

cout << "Input number: ";

cin >> num;

cout << " " << bitset<sizeof(short) \* 8>(num) << endl;

num = num & ~mask;

cout << "Number with 1-st mask: " << bitset<sizeof(short) \* 8>(num) << endl;

num = num & ~(1 << 12) & ~(1 << 11) & ~(1 << 1);

cout << "Number with 2-nd mask: " << bitset<sizeof(short) \* 8>(num);

}

Результат тестирования:

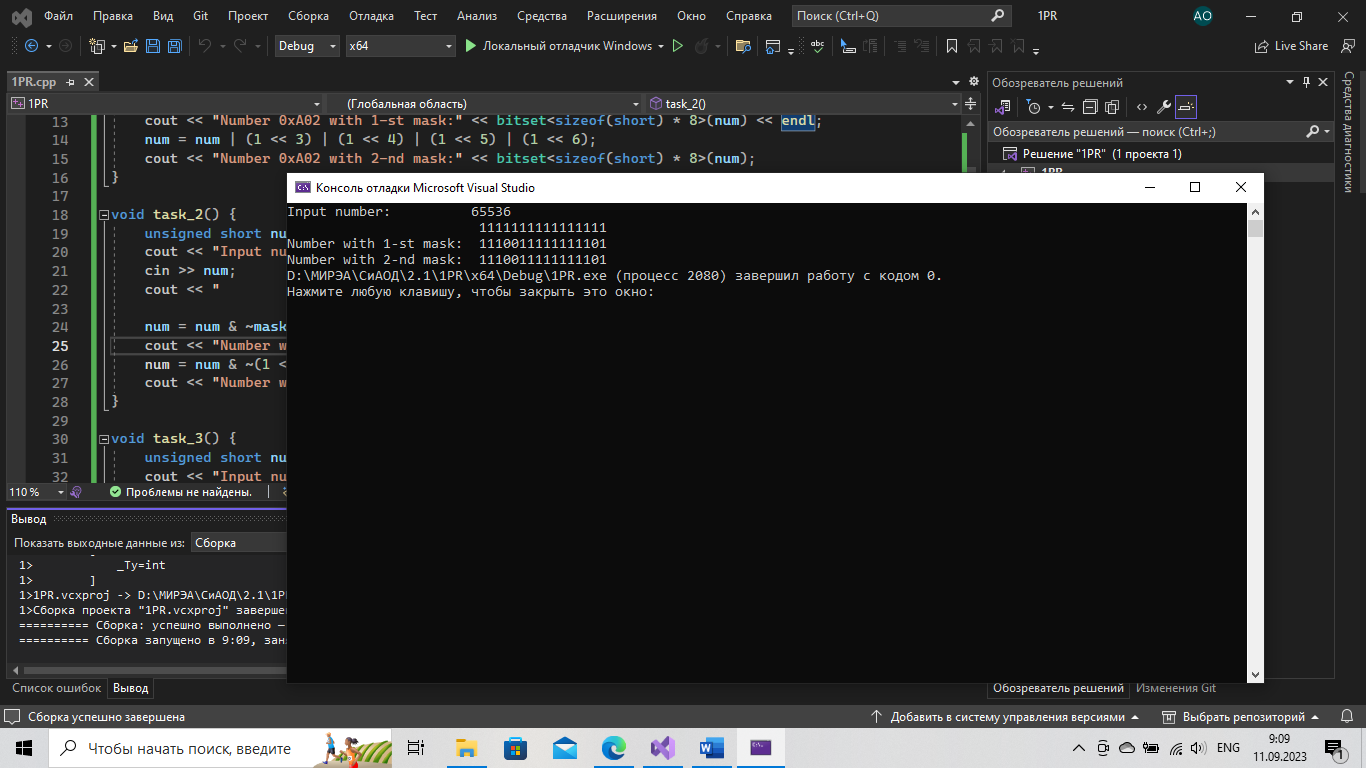


Рисунок 2 – Результат теста 1.2

## Упражнение 3

### Условие задачи

Умножить введённое число на 32, используя побитовые операции.

### Разработка решения задачи варианта

Выражение:

unsigned short pos = 5; num = num << pos

Тестовый пример (num = 10):

0000 0000 0000 1010 << 5

---------------------------

0000 0001 0100 0000

### Реализация решения

Код функции:

void task\_3() {

unsigned short num, pos = 5;

cout << "Input number: ";

cin >> num;

cout << " " << bitset<sizeof(short) \* 8>(num) << endl;

num = num << pos;

cout << "Number with mask: " << bitset<sizeof(short) \* 8>(num) << " " << num;

}

Результат тестирования:

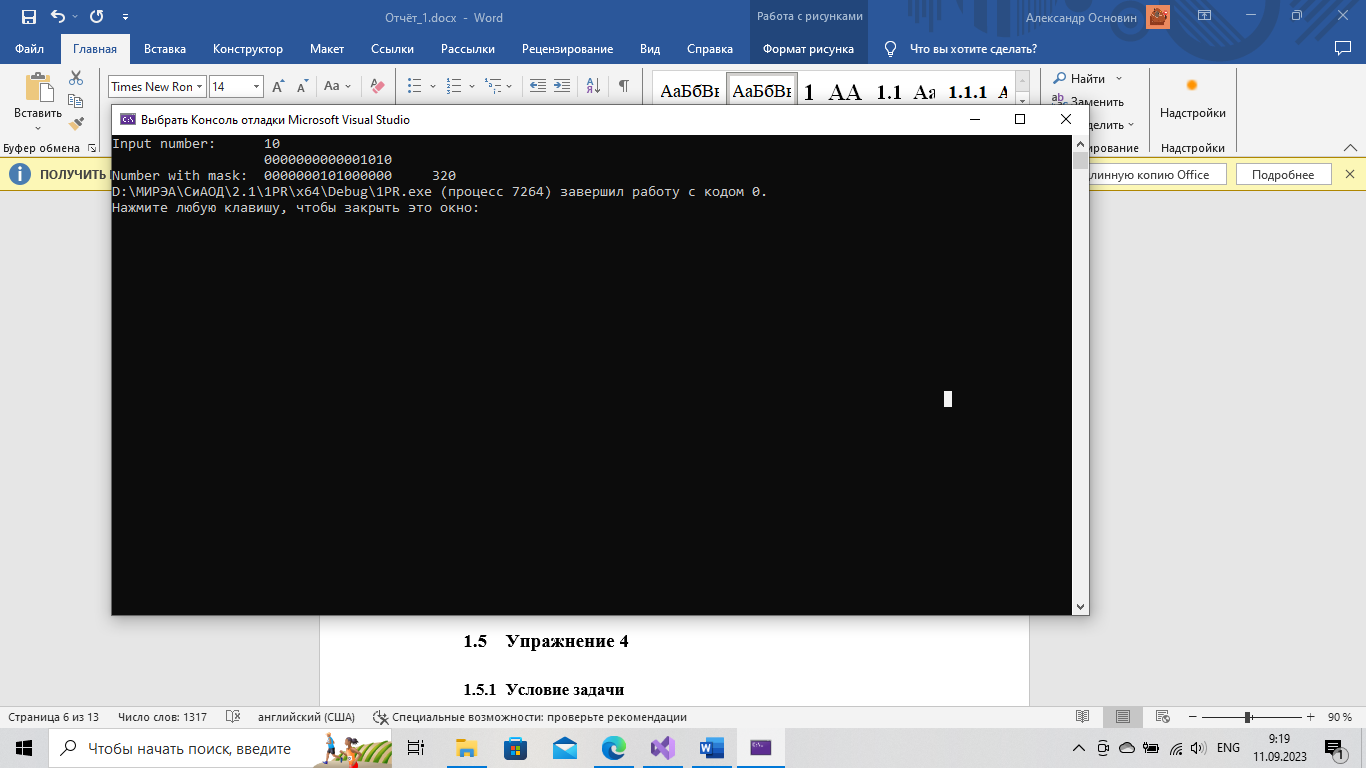


Рисунок 3 – Результат теста 1.3

## Упражнение 4

### Условие задачи

Умножить введённое число на 32, используя побитовые операции.

### Разработка решения задачи варианта

Выражение:

unsigned short pos = 5; num = num >> pos

Тестовый пример (num = 512):

0000 0010 0000 0000 >> 5

---------------------------

0000 0000 0001 0000

### Реализация решения

Код функции:

void task\_4() {

unsigned short num, pos = 5;

cout << "Input number: ";

cin >> num;

cout << " " << bitset<sizeof(short) \* 8>(num) << endl;

num = num >> pos;

cout << "Number with mask: " << bitset<sizeof(short) \* 8>(num) << " " << num;

}

Результат тестирования:

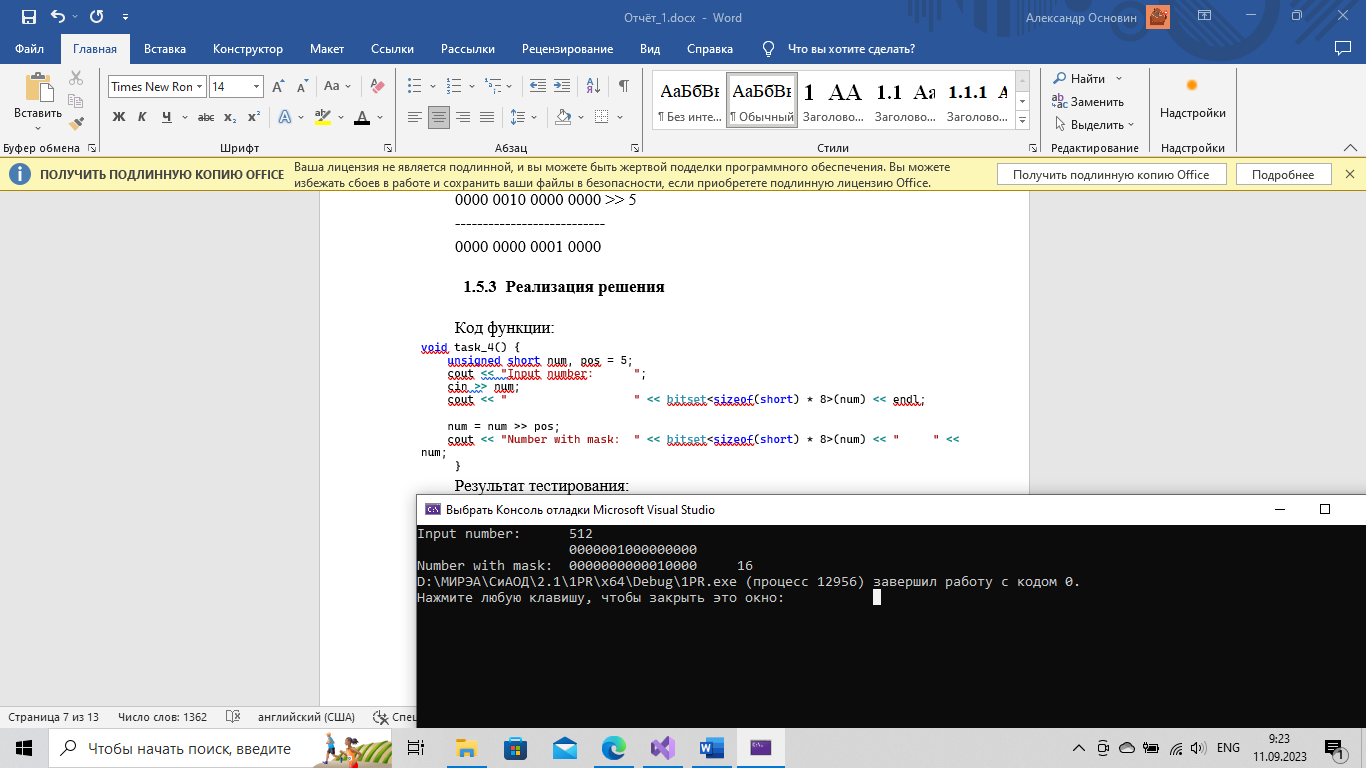


Рисунок 4 – Результат теста 1.4

## Упражнение 5

### Условие задачи

Обнулить n-й бит введённого числа, используя маску, инициализированную единицей в старшем разряде.

### Разработка решения задачи варианта

Выражение:

unsigned short mask = 0x800; num = num & ~(mask >> 15 - pos))

Тестовый пример (num = 65535; pos = 3):

~(mask >> 15 - pos)) = ~(1000 0000 0000 0000 >> 15 - 3) = ~(0000 0000 0000 1000) = 1111 1111 1111 0111

1111 1111 1111 1111 &

1111 1101 1111 1111

---------------------------

1111 1111 1111 0111

### Реализация решения

Код функции:

void task\_5() {

unsigned short num, pos, mask = 0x8000;

cout << "Input number: ";

cin >> num;

cout << " " << bitset<sizeof(short) \* 8>(num) << endl;

cout << "Enter the bit number to set '0': ";

cin >> pos;

num = num & ~(mask >> ((sizeof(short) \* 8 - 1) - pos));

cout << "Number with mask: " << bitset<sizeof(short) \* 8>(num) << " " << num;

}

Результат тестирования:

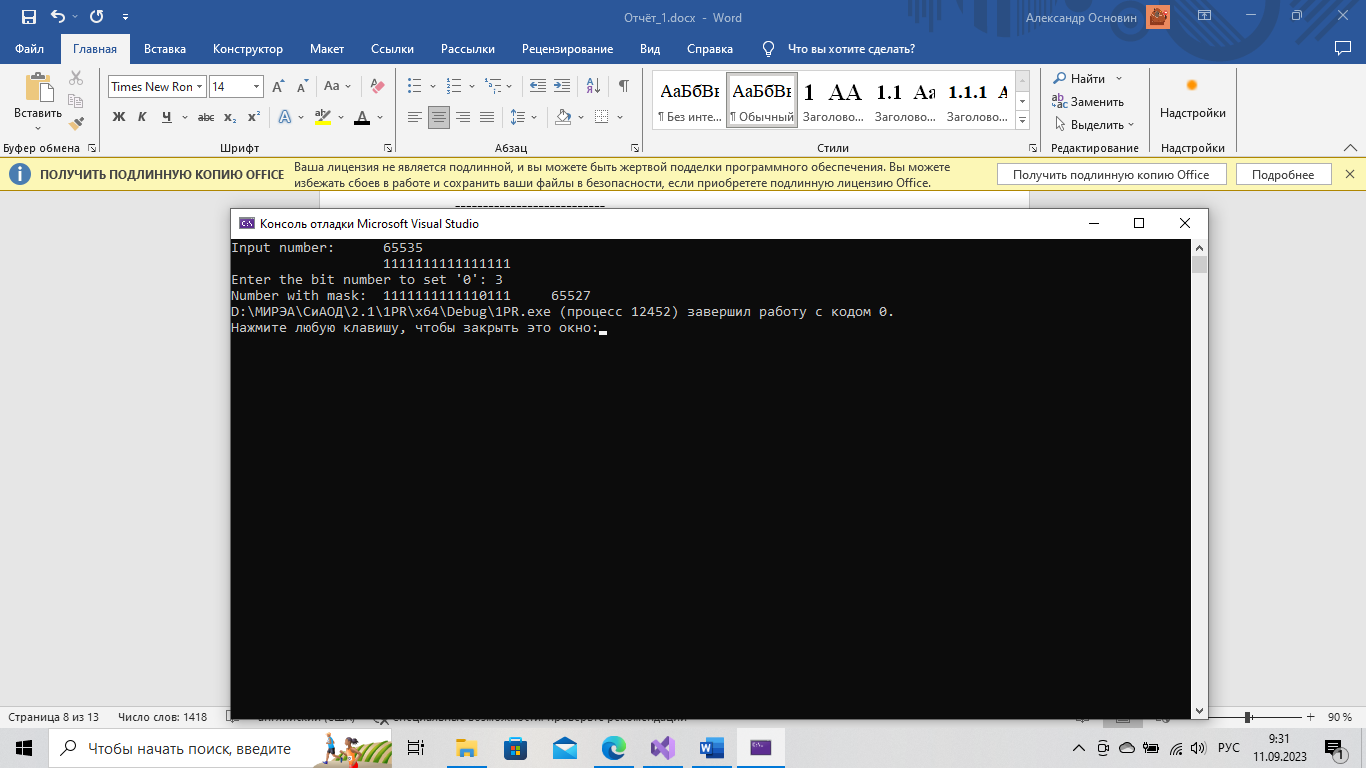


Рисунок 5 – Результат теста 1.5

# Задание 2

## Постановка задачи

Необходимо отсортировать массив из 107 семизначных чисел, затратив на это ~1 МБ оперативной памяти и малое количество времени (не более 10 секунд).

## Описание структуры данных

Для решения поставленной задачи следует использовать битовый массив. Его можно реализовать несколькими способами: массивом типа unsigned char размером [107 / 8] + 1, с помощью класса bitset или с помощью класса vector из элементов типа bool: под значение типа bool выделяется 1 байт памяти, но в классе vector происходит оптимизация, в результате которой одно логическое значение занимает 1 бит.

## Алгоритм решения

Задачу можно разбить на три подзадачи:

1. Создание битового массива с нулевыми исходными значениями;
2. Считывание целых чисел и установка в единицу соответствующих бит массива;
3. Формирование упорядоченного выходного массива путём последовательной проверки бит массива и вывода индексов тех бит, которые установлены с единицу.

## Тестовый пример

Исходный массив: [15, 7, 9, 4, 6].

Битовый массив, сформированный на основе исходного: char[10000000 / 8 + 1]: [00001011, 01000001, 00000000, …, 00000000].

Отсортированный массив, сформированный на основе битового: [4, 6, 7, 9, 15].

Код программы:

unsigned char\* create\_array(size\_t size) {

unsigned char\* numbers = new unsigned char[(size >> 3) + 1];

for (size\_t i = 0; i <= (size >> 3); ++i) {

numbers[i] = 0;

}

return numbers;

}

void sort\_array(unsigned char\* &numbers, vector<int>input\_nums) {

for (size\_t i = 0; i < input\_nums.size(); ++i) {

numbers[input\_nums[i] >> 3] = numbers[input\_nums[i] >> 3] | (1 << (input\_nums[i] & 7));

}

}

void get\_nums(unsigned char\* numbers, vector<int> &output\_nums) {

output\_nums.clear();

unsigned char mask = 1;

size\_t bit;

for (size\_t i = 0; i <= 10000000 >> 3; ++i) {

bit = 0;

while (numbers[i] != 0) {

if ((numbers[i] & mask) != 0) {

output\_nums.push\_back((i << 3) + bit);

}

numbers[i] >>= 1;

++bit;

}

}

}

void array\_bit\_sort() {

size\_t average, num;

cout << "How many unique numbers we need sort? ";

cin >> average;

vector<int> input\_nums(average);

cout << "Enter numbers to sort:" << endl;

for (size\_t i = 0; i < average; ++i) {

cin >> input\_nums[i];

}

auto start = chrono::high\_resolution\_clock::now();

unsigned char\* numbers = create\_array(10000000);

sort\_array(numbers, input\_nums);

get\_nums(numbers, input\_nums);

auto end = chrono::high\_resolution\_clock::now();

cout << "--------------------------SORTED--------------------------" << endl;

for (int num : input\_nums) {

cout << num << " ";

}

chrono::duration<float> duration = end - start;

cout << endl << "Time: " << setprecision(4) << duration.count();

delete[] numbers;

}

## Результаты тестирования

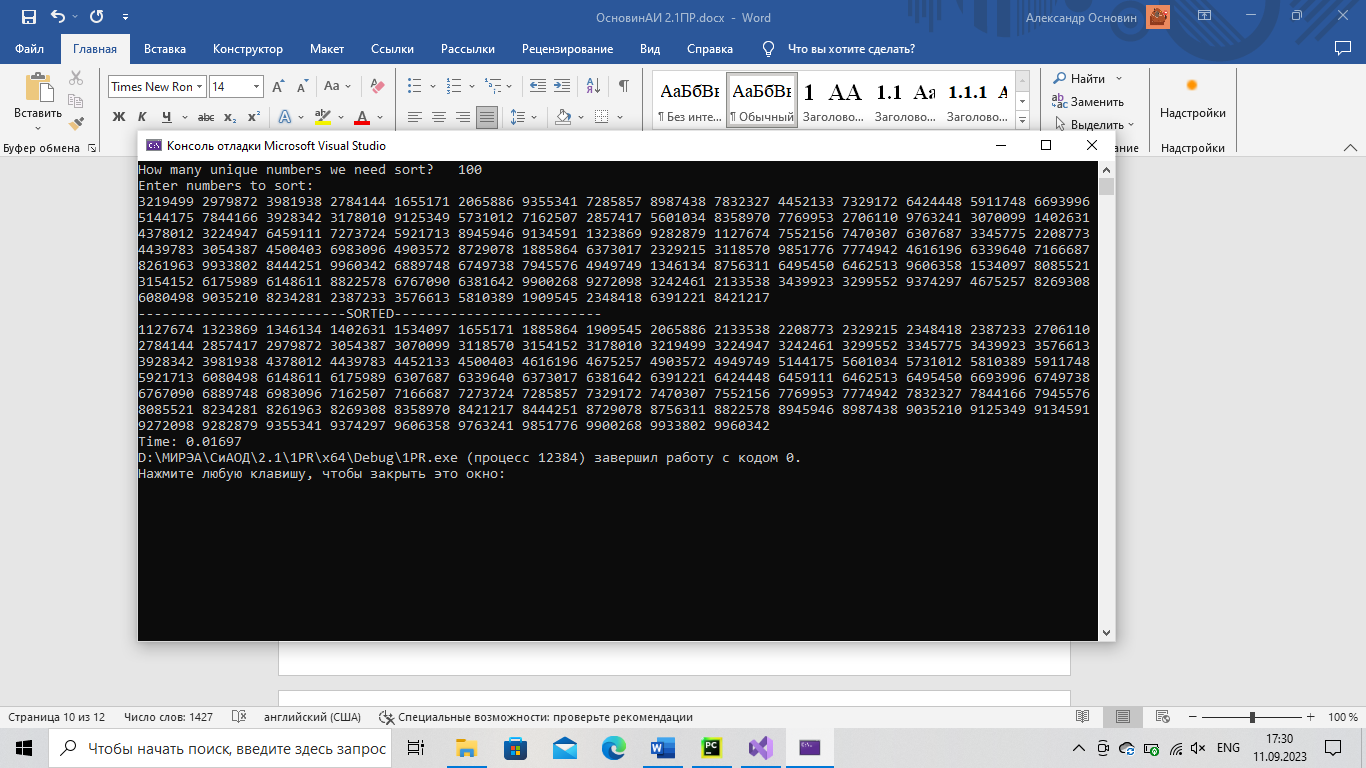


Рисунок 1 – Результат тестирования сортировки 100 чисел

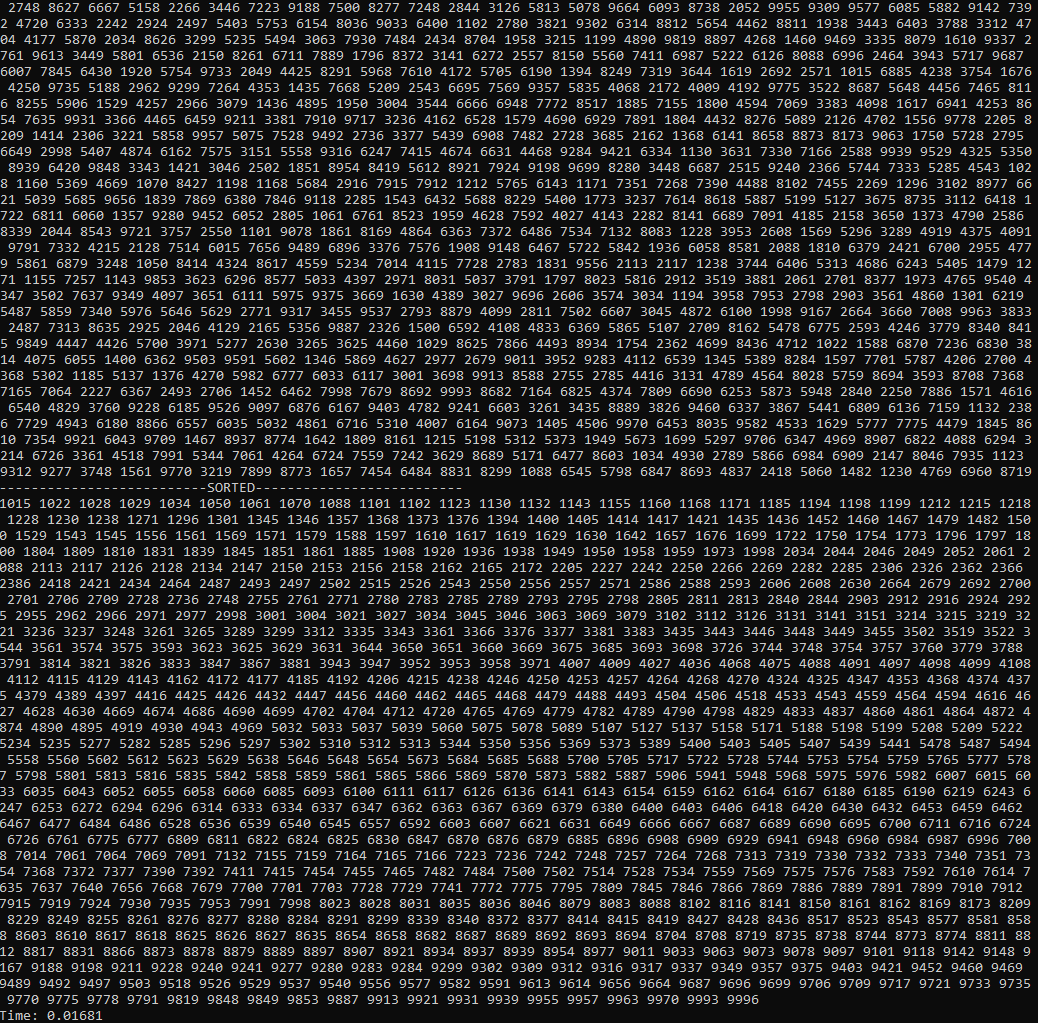


Рисунок 2 – Результат сортировки 1000 чисел

# Выводы

В ходе выполнения практической работы были изучены битовые операции в языке программирования C++, выполнены простые задачи с использованием битовых операций, также была реализована сортировка с использованием битового массива, работающая только для уникальных элементов.