



МИНОБРНАУКИ РОССИИ

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)

ОТЧЁТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ

«Работа с данными из файла»

**по дисциплине «Структуры и алгоритмы обработки данных (часть
2/2)»**

Выполнил студент группы ИКБО-41-23

Попов А.В.

Принял
Ассистент

Рысин М.Л.

Практические работы выполнены

«__»_____2024 г.

(подпись студента)

«Зачтено»

«__»_____2024 г.

(подпись преподавателя)

Москва 2024

СОДЕРЖАНИЕ

ЦЕЛЬ РАБОТЫ.....	3
ЗАДАНИЕ 1.....	4
Задание 1.а.....	4
Задание 1.б.....	5
Задание 1.в.....	6
ЗАДАНИЕ 2.....	7
Задание 2.а.....	7
Задание 2.б.....	8
Задание 2.в.....	10
ЗАДАНИЕ 3.....	12
Задание 3.а.....	12
Задание 3.б.....	15
ВЫВОД.....	16

ЦЕЛЬ РАБОТЫ

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

ЗАДАНИЕ 1

Задание 1.а

Задача:

Установить 5-й бит произвольного целого числа в 0.

Описание алгоритма:

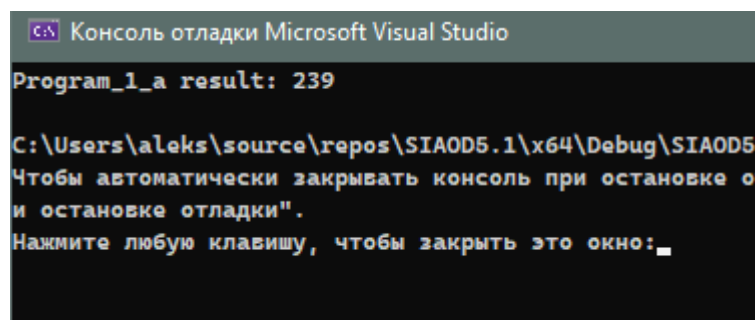
Для выполнения данной задачи нужно создать маску, равную единице, после при помощи побитового перемещения сдвигаем всё на 4 позиции влево, инверсируем и побитово перемножаем результат на число x.

Код программы:

```
int Program_1_a(int input) {  
    unsigned char x = (char)input;  
    unsigned char mask = 1;  
    x = x & ~(mask << 4);  
    return x;  
}
```

Рисунок 1 — Реализованный код для задачи 1.а

Результат тестирования:



Консоль отладки Microsoft Visual Studio

Program_1_a result: 239

C:\Users\aleks\source\repos\SIAOD5.1\x64\Debug\SIAOD5

Чтобы автоматически закрывать консоль при остановке о

и остановке отладки".

Нажмите любую клавишу, чтобы закрыть это окно: _

Рисунок 2 — Вывод программы для входного значения «255»

Задание 1.6

Задача:

Реализовать по аналогии с предыдущим примером установку 7-го бита числа в единицу.

Описание алгоритма:

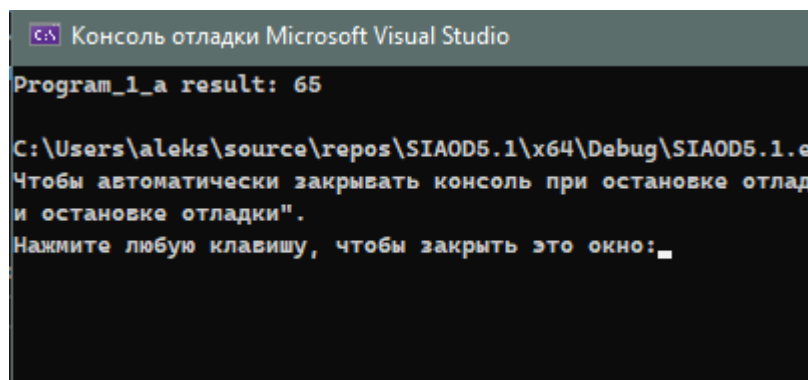
Для выполнения данной задачи нужно создать маску, равную единице, после при помощи побитового перемещения сдвигаем всё на 6 позиций влево, инверсируем и побитово перемножаем результат на число x. Таким образом мы гарантированно получаем 0 в 7-ом бите и далее прибавляем к числу маску в её исходном виде.

Код программы:

```
int Program_1_a(int input) {  
    unsigned char x = (char)input;  
    unsigned char mask = 1;  
    x = x & ~(mask << 6);  
    x = x | (mask << 6);  
    return x;  
}
```

Рисунок 3 - Реализованный код для задачи 1.6

Результат тестирования:



```
Консоль отладки Microsoft Visual Studio  
Program_1_a result: 65  
C:\Users\aleks\source\repos\SIAOD5.1\x64\Debug\SIAOD5.1.exe  
Чтобы автоматически закрывать консоль при остановке отладки  
и остановке отладки".  
Нажмите любую клавишу, чтобы закрыть это окно: _
```

Рисунок 4 — Вывод программы для входного значения «1»

Задание 1.в

Задача:

Реализовать код листинга 1, объясните выводимый программой результат.

Описание алгоритма:

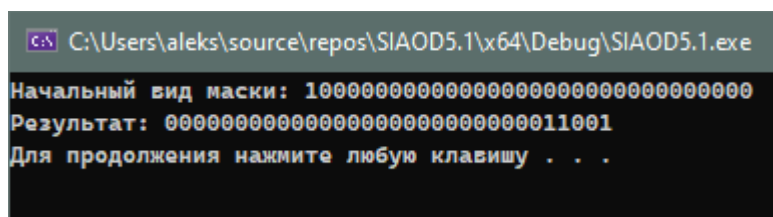
Алгоритм выводит побитовое представление числа x (в данном примере 25) в виде 32-битного числа. Для этого используется маска с единицей в старшем бите, которая последовательно сдвигается вправо, проверяя каждый бит числа x . Для каждого бита выводится его значение (0 или 1), начиная с самого старшего.

Код программы:

```
int Program_1_c(int input) {
    unsigned int x = input;
    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;
}
```

Рисунок 5 — Код листинга 1

Результат тестирования:



```
C:\Users\aleks\source\repos\SIAOD5.1\x64\Debug\SIAOD5.1.exe
Начальный вид маски: 10000000000000000000000000000000
Результат: 000000000000000000000000000011001
Для продолжения нажмите любую клавишу . . .
```

Рисунок 6 — Вывод программы для входного значения «25»

ЗАДАНИЕ 2

Задание 2.а

Задача:

Сортировка не более 8-и неповторяющихся чисел в диапазоне [0-7] типа данных unsigned char.

Описание алгоритма:

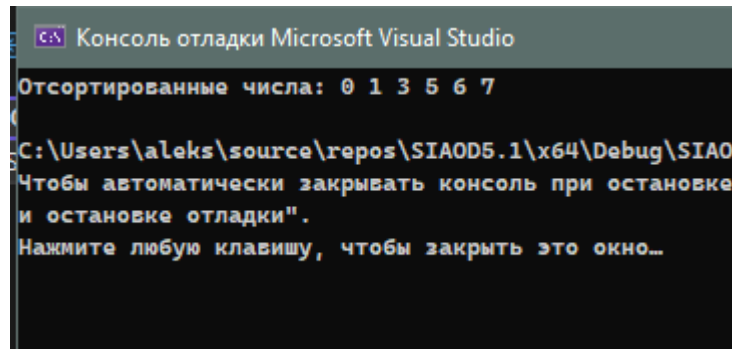
Алгоритм сортировки неповторяющихся чисел в диапазоне [0-7] с использованием битового массива основывается на использовании битовой маски. Изначально создается переменная типа unsigned char, которая будет служить битовой маской, где каждый бит соответствует одному числу из диапазона от 0 до 7. Для заполнения битовой маски перебираем все числа из входного массива. Если число присутствует, устанавливаем соответствующий бит в маске в 1 с помощью операции побитового сдвига и побитового ИЛИ. После заполнения битовой маски, чтобы получить отсортированные числа, последовательно проверяем каждый бит маски от 0 до 7. Если бит установлен в 1, выводим соответствующее число, используя операцию побитового И.

Код программы:

```
void Program_2_a(unsigned char* arr, int size) {
    unsigned char bit_mask = 0;
    for (int i = 0; i < size; i++) {
        bit_mask |= (1 << arr[i]);
    }
    cout << "Отсортированные числа: ";
    for (int i = 0; i < 8; i++) {
        if (bit_mask & (1 << i)) {
            cout << i << " ";
        }
    }
    cout << endl;
}
```

Рисунок 7 — Реализованный код задания 2.а

Результат тестирования:



```
Консоль отладки Microsoft Visual Studio
Отсортированные числа: 0 1 3 5 6 7
C:\Users\aleks\source\repos\SIAOD5.1\x64\Debug\SIAO
Чтобы автоматически закрывать консоль при остановке
и остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рисунок 8 — Вывод программы для входного массива «{ 5, 3, 7, 1, 6, 0 }»

Задание 2.б

Задача:

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

Описание алгоритма:

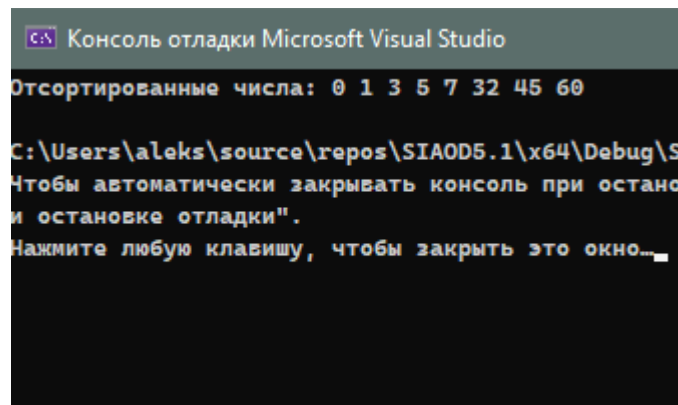
Для выполнения данного задания большая часть кода переходит с прошлого задания и меняются лишь некоторые переменные. Размер изменяется на 64, диапазон увеличивается до 63 и будет использовано 1ULL для правильной работы программы (ULL — это суффикс, который означает, что число будет представлено как unsigned long long).

Код программы:

```
void Program_2_b(unsigned char* arr, int size) {
    unsigned long long bit_mask = 0;
    for (int i = 0; i < size; i++) {
        bit_mask |= (1ULL << arr[i]);
    }
    cout << "Отсортированные числа: ";
    for (int i = 0; i < 64; i++) {
        if (bit_mask & (1ULL << i)) {
            cout << i << " ";
        }
    }
    cout << endl;
}
```

Рисунок 9 - Реализованный код задания 2.б

Результат тестирования:



The screenshot shows the 'Консоль отладки Microsoft Visual Studio' (Visual Studio Debug Console). The output of the program is displayed in yellow text on a black background: 'Отсортированные числа: 0 1 3 5 7 32 45 60'. Below this, the console shows the file path 'C:\Users\aleks\source\repos\SIAOD5.1\x64\Debug\S...' and a message: 'Чтобы автоматически закрывать консоль при остановке отладки. Нажмите любую клавишу, чтобы закрыть это окно...'.

Рисунок 10 — Вывод программы для входного массива «{ 5, 3, 60, 1, 45, 7, 32, 0 }»

Задание 2.в

Задача:

Исправить 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

Описание алгоритма:

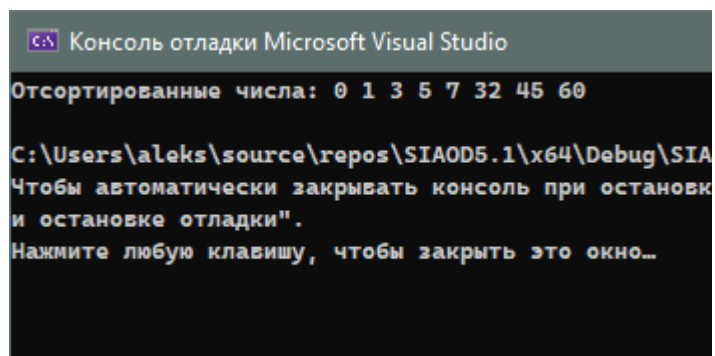
Программа выполняет сортировку чисел в диапазоне от 0 до 63 с использованием битового массива, представленного как линейный массив из 8 элементов типа unsigned char. Каждый элемент массива хранит 8 битов, что в сумме дает 64 бита, достаточные для представления всех чисел в диапазоне. Сначала программа инициализирует массив битовой маски, где все биты установлены в 0, затем для каждого числа из входного массива вычисляется байт, в котором оно должно находиться (делением числа на 8), и битовая позиция внутри этого байта (остатком от деления на 8). Соответствующий бит устанавливается в 1. После того как все числа обработаны, программа проходит по битовой маске и проверяет каждый бит. Если бит установлен в 1, программа выводит индекс этого бита.

Код программы:

```
void Program_2_c(unsigned char* arr, int size) {
    unsigned char bit_mask[8] = { 0 };
    for (int i = 0; i < size; i++) {
        int index = arr[i] / 8;
        int bit_position = arr[i] % 8;
        bit_mask[index] |= (1 << bit_position);
    }
    cout << "Отсортированные числа: ";
    for (int i = 0; i < 64; i++) {
        int index = i / 8;
        int bit_position = i % 8;
        if (bit_mask[index] & (1 << bit_position)) {
            cout << i << " ";
        }
    }
    cout << endl;
}
```

Рисунок 11 - Реализованный код задания 2.в

Результат тестирования:



Консоль отладки Microsoft Visual Studio

Отсортированные числа: 0 1 3 5 7 32 45 60

C:\Users\aleks\source\repos\SIAOD5.1\x64\Debug\SIA
Чтобы автоматически закрывать консоль при остановк
и остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

Рисунок 12 — Вывод программы для входного

массива «{ 7, 0, 45, 1, 60, 5, 32, 3 }»

ЗАДАНИЕ 3

Задание 3.а

Задача:

Реализовать задачу сортировки числового файла, содержащего не более $n=10^7$ неотрицательных чисел, среди которых нет повторяющихся. Результатом должна быть упорядоченная по возрастанию последовательность исходных чисел в выходном файле.

Описание алгоритма:

Сначала определяется максимальное значение, которое составляет $10^7 - 1$.
1. На основе этого значения создается битовый массив, где каждый бит соответствует числу в данном диапазоне. Размер массива рассчитывается как $((\text{максимальное значение} / 8) + 1)$, что позволяет учесть все возможные числа, поскольку один байт может хранить 8 битов. Затем программа открывает входной файл с уникальными числами и считывает каждое число. Для каждого числа вызывается лямбда-функция, которая устанавливает соответствующий бит в битовом массиве. Установка бита осуществляется путем деления числа на 8 для определения индекса байта и взятия остатка от деления на 8 для определения индекса бита внутри байта. После того как все числа были прочитаны и биты установлены, программа открывает новый файл для записи отсортированных чисел. Она проходит по битовому массиву, проверяя каждый бит. Если бит установлен, программа записывает соответствующее число в выходной файл. В конце выполнения программы пользователю выводится сообщение о том, что файл был успешно отсортирован и записан, а также время, затраченное на выполнение операции.

Код программы:

```
void Program_3_a(const string& input_file, const string& output_file, int max_value) {
    vector<unsigned char> bit_array((max_value / 8) + 1, 0); // битовый массив

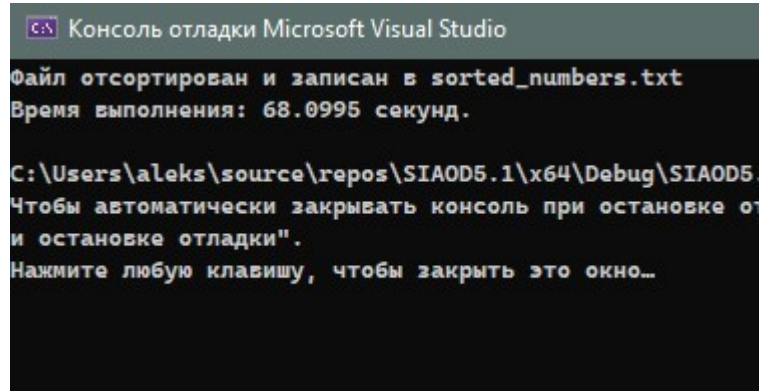
    auto set_bit = [&bit_array](int num) { // lambda-функция для установки 1 в битовом массиве
        int byte_index = num / 8;
        int bit_index = num % 8;
        bit_array[byte_index] |= (1 << bit_index);
    };

    ifstream infile(input_file); // чтение файла
    int number;
    while (infile >> number) {
        set_bit(number);
    }
    infile.close();

    ofstream outfile(output_file); // сортировка и запись
    for (int i = 0; i <= max_value; i++) {
        int byte_index = i / 8;
        int bit_index = i % 8;
        if (bit_array[byte_index] & (1 << bit_index)) {
            outfile << i << "\n";
        }
    }
    outfile.close();
}
```

Рисунок 13 - Реализованный код задания 3.а

Результат тестирования:



Консоль отладки Microsoft Visual Studio

Файл отсортирован и записан в sorted_numbers.txt
Время выполнения: 68.0995 секунд.

C:\Users\aleks\source\repos\SIA0D5.1\x64\Debug\SIA0D5.
Чтобы автоматически закрывать консоль при остановке от
и остановке отладки".
Нажмите любую клавишу, чтобы закрыть это окно...

Рисунок 14 - Вывод программы

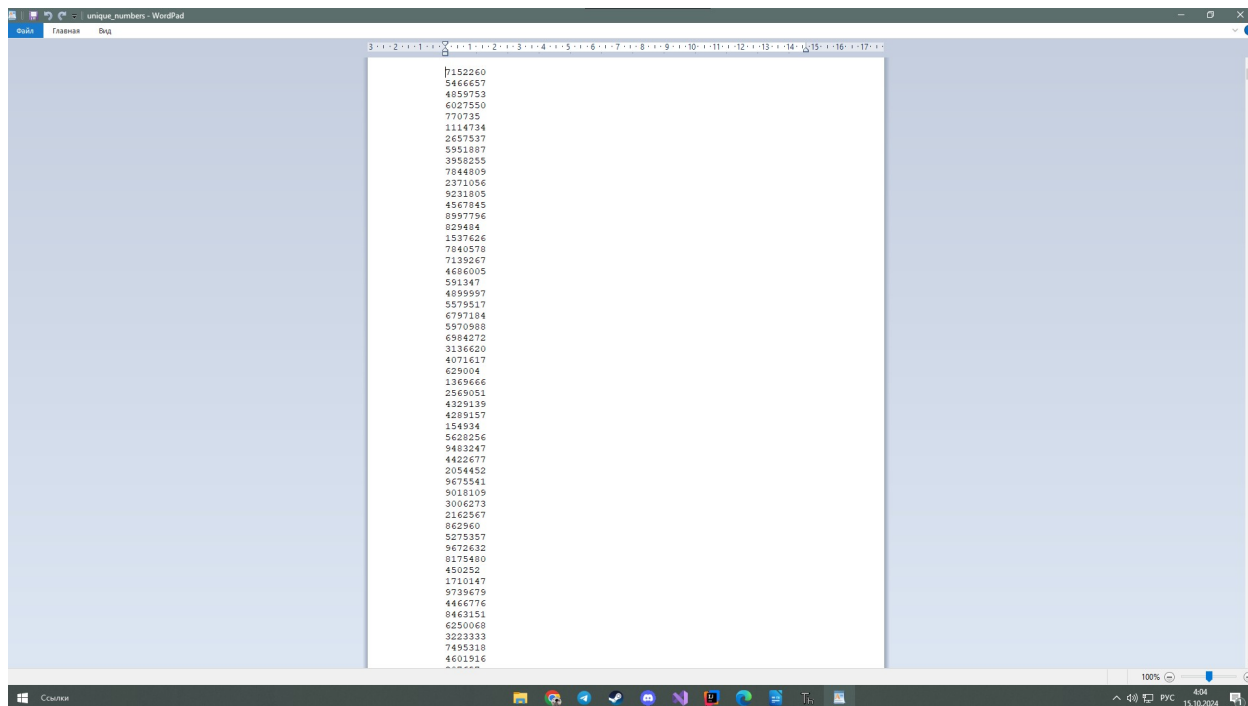


Рисунок 15 — Неотсортированный файл

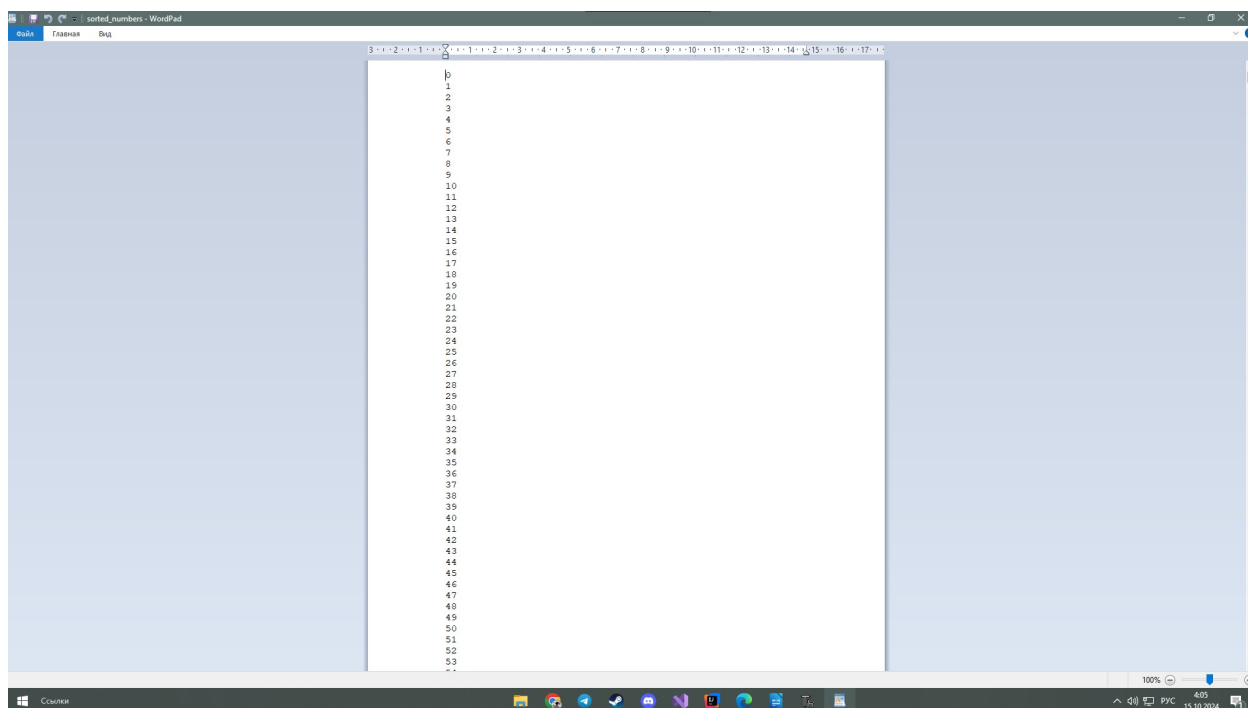


Рисунок 16 — Отсортированный файл

Задание 3.б

Задача:

Определить программно объём оперативной памяти, занимаемый битовым массивом.

Описание алгоритма:

Объём памяти, занимаемый массивом, можно рассчитать по формуле:

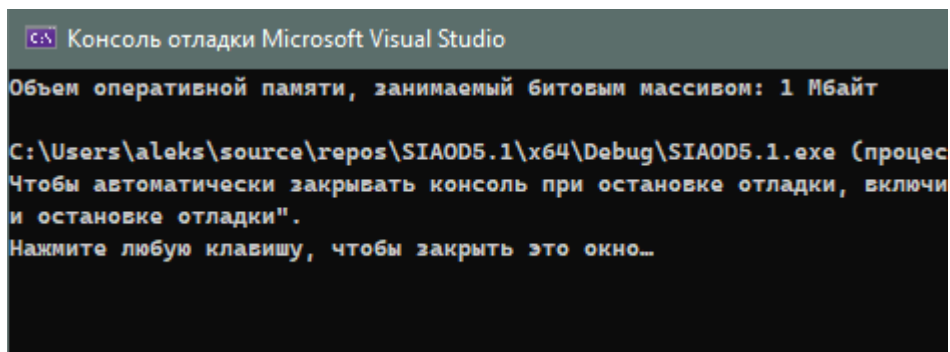
Объём памяти (в байтах) = Размер массива × Размер одного элемента

Код программы:

```
int Program_3_b() {  
    int max_value = 10000000 - 1;  
    vector<unsigned char> bit_array((max_value / 8) + 1, 0);  
    // size_t - это встроенный тип данных, который используется для представления размеров и количества объектов в памяти.  
    size_t memory_size = bit_array.size() * sizeof(unsigned char);  
    cout << "Объем оперативной памяти, занимаемый битовым массивом: " << memory_size / (1024 * 1024) << " Мбайт" << endl;  
    return 0;  
}
```

Рисунок 17 - Реализованный код задания 3.б

Результат тестирования:



```
Консоль отладки Microsoft Visual Studio  
Объем оперативной памяти, занимаемый битовым массивом: 1 Мбайт  
C:\Users\aleks\source\repos\SIAOD5.1\x64\Debug\SIAOD5.1.exe (процес  
Чтобы автоматически закрывать консоль при остановке отладки, включи  
и остановке отладки".  
Нажмите любую клавишу, чтобы закрыть это окно...
```

Рисунок 18 - Вывод программы

ВЫВОД

Были освоены приёмы работы с битовым представлением беззнаковых целых чисел и реализован эффективный алгоритм внешней сортировки на основе битового массива.