



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

Раев Н.Д.

Принял
Ассистент

Рысин М.Л.

Практические работы выполнены

«__»_____2024 г.

(подпись студента)

«Зачтено»

«__»_____2024 г.

(подпись преподавателя)

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:

```
11 int Program_1_a(int input) {  
12     unsigned char x = (char)input;  
13     unsigned char mask = 1;  
14     x = x & ~(mask << 4);  
15     return x;  
16 }
```

Рисунок 1 — Реализованный код для задачи 1.а

Результат тестирования:

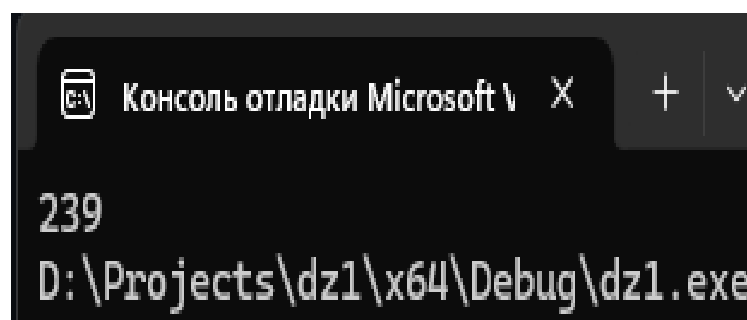


Рисунок 2 — Вывод программы для входного значения «255»

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:

```
18 int Program_1_b(int input) {  
19     unsigned char x = (char)input;  
20     unsigned char mask = 1;  
21     x = x & ~(mask << 6);  
22     x = x | (mask << 6);  
23     return x;  
24 }
```

Рисунок 3 - Реализованный код для задачи 1.6

Результат тестирования:

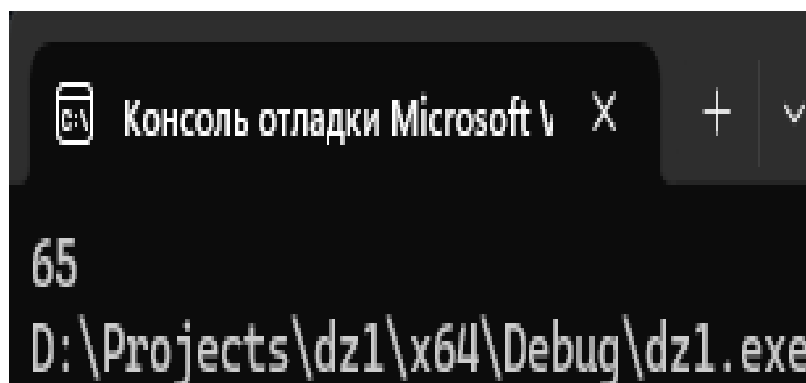


Рисунок 4 — Вывод программы для входного значения «1»

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:

```
26 int Program_1_c(int input) {
27     unsigned int x = input;
28     const int n = sizeof(int) * 8;
29     unsigned maska = (1 << (n - 1));
30     cout << "Изначальная маска: " << bitset<n>(maska) << endl;
31     cout << "Итого: ";
32     for (int i = 1; i <= n; i++) {
33         cout << ((x & maska) >> (n - i));
34         maska = maska >> 1;
35     }
36     cout << endl;
37     system("pause");
38     return 0;
39 }
```

Рисунок 5 — Код листинга 1

Результат тестирования:


```
41 void Program_2_a(unsigned char* arr, int size) {  
42     unsigned char bit_mask = 0;  
43     for (int i = 0; i < size; i++) {  
44         bit_mask |= (1 << arr[i]);  
45     }  
46     cout << "Отсортированные числа: ";  
47     for (int i = 0; i < 8; i++) {  
48         if (bit_mask & (1 << i)) {  
49             cout << i << " ";  
50         }  
51     }  
52     cout << endl;  
53 }
```

Рисунок 7 — Реализованный код задания 2.а

Результат тестирования:

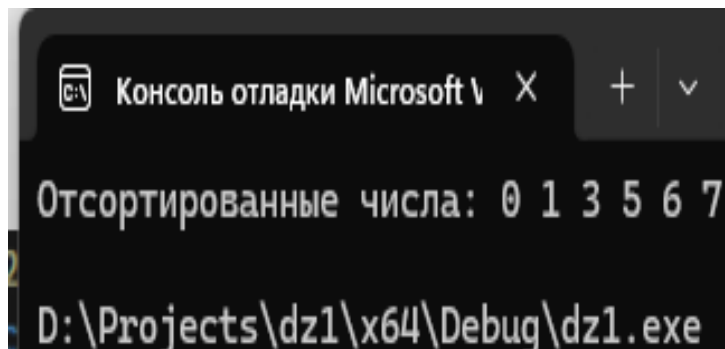


Рисунок 8 — Вывод программы для входного массива «{ 5, 3, 7, 1, 6, 0 }»

Задание 2.6

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:

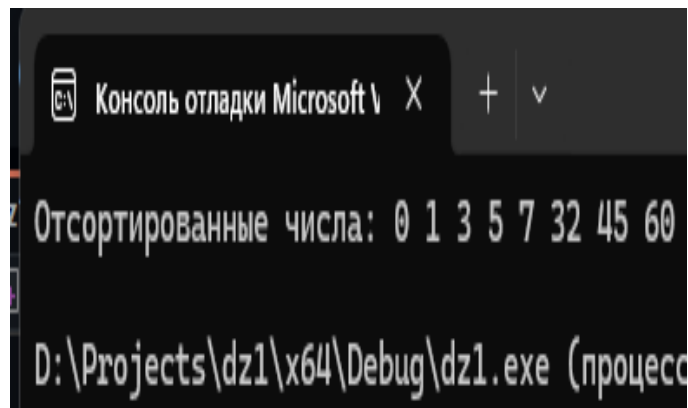
Для выполнения данного задания большая часть кода переходит с прошлого задания и меняются лишь некоторые переменные. Размер изменяется на 64, диапазон увеличивается до 63 и будет использовано 1ULL для правильной работы программы (ULL — это суффикс, который означает, что число будет представлено как unsigned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

Код программы:

```
55 void Program_2_b(unsigned char* arr, int size) {  
56     unsigned long long bit_mask = 0;  
57     for (int i = 0; i < size; i++) {  
58         bit_mask |= (1ULL << arr[i]);  
59     }  
60     cout << "Отсортированные числа: ";  
61     for (int i = 0; i < 64; i++) {  
62         if (bit_mask & (1ULL << i)) {  
63             cout << i << " ";  
64         }  
65     }  
66     cout << endl;  
67 }
```

Рисунок 9 - Реализованный код задания 2.б

Результат тестирования:



Консоль отладки Microsoft Visual Studio. Вывод программы: "Отсортированные числа: 0 1 3 5 7 32 45 60". В строке состояния внизу указано: "D:\Projects\dz1\x64\Debug\dz1.exe (процесс)".

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:

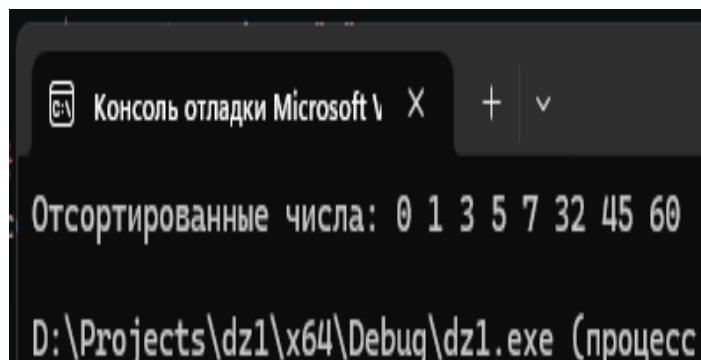
Программа выполняет сортировку чисел в диапазоне от 0 до 63 с использованием битового массива, представленного как линейный массив из 8 элементов типа unsigned char. Каждый элемент массива хранит 8 битов, что в сумме дает 64 бита, достаточные для представления всех чисел в диапазоне. Сначала программа 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:

```
69 void Program_2_c(unsigned char* arr, int size) {
70     unsigned char bit_mask[8] = { 0 };
71     for (int i = 0; i < size; i++) {
72         int index = arr[i] / 8;
73         int bit_position = arr[i] % 8;
74         bit_mask[index] |= (1 << bit_position);
75     }
76     cout << "Отсортированные числа: ";
77     for (int i = 0; i < 64; i++) {
78         int index = i / 8;
79         int bit_position = i % 8;
80         if (bit_mask[index] & (1 << bit_position)) {
81             cout << i << " ";
82         }
83     }
84     cout << endl;
85 }
```

Рисунок 11 - Реализованный код задания 2.в

Результат тестирования:



Консоль отладки Microsoft Visual Studio

Отсортированные числа: 0 1 3 5 7 32 45 60

D:\Projects\dz1\x64\Debug\dz1.exe (процесс

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:

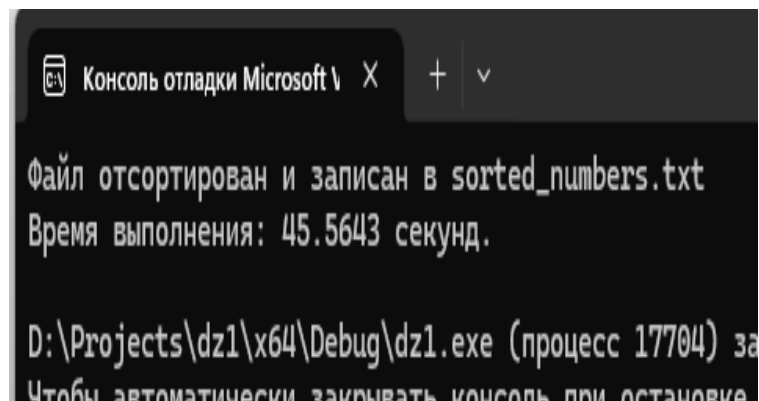
Сначала определяется максимальное значение, которое составляет $10^7 - 1$.
1. На основе этого значения создается битовый массив, где каждый бит соответствует числу в данном диапазоне. Размер массива рассчитывается как $((\text{максимальное значение} / 8) + 1)$, что позволяет учесть все возможные числа, 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:

```
86
87 void Program_3_a(const string& input_file, const string& output_file, int max_value) {
88     vector<unsigned char> bit_array((max_value / 8) + 1, 0); // битовый массив
89
90     auto set_bit = [&bit_array](int num) {
91         int byte_index = num / 8;
92         int bit_index = num % 8;
93         bit_array[byte_index] |= (1 << bit_index);
94     };
95
96     ifstream infile(input_file);
97     int number;
98     while (infile >> number) {
99         set_bit(number);
100     }
101     infile.close();
102
103     ofstream outfile(output_file);
104     for (int i = 0; i <= max_value; i++) {
105         int byte_index = i / 8;
106         int bit_index = i % 8;
107         if (bit_array[byte_index] & (1 << bit_index)) {
108             outfile << i << "\n";
109         }
110     }
111     outfile.close();
112 }
```

Рисунок 13 - Реализованный код задания 3.а

Результат тестирования:



Консоль отладки Microsoft V X + v

Файл отсортирован и записан в sorted_numbers.txt
Время выполнения: 45.5643 секунд.

D:\Projects\dz1\x64\Debug\dz1.exe (процесс 17704) за
Чтобы автоматически закрывать консоль при остановке

Рисунок 14 - Вывод программы



Рисунок 15 — Неотсортированный файл

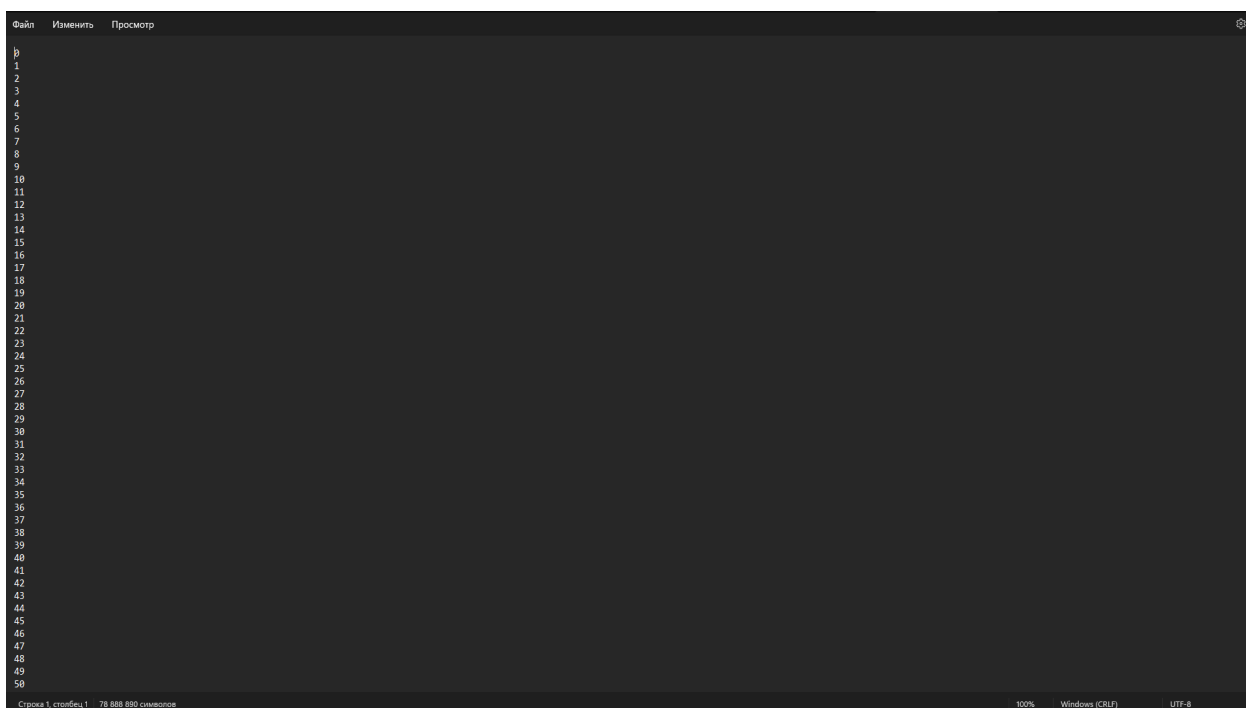


Рисунок 16 — Отсортированный файл

Задание 3.6

Задача:

Определить программно объём оперативной памяти, занимаемый битовым массивом.

Описание алгоритма:

Объём памяти, занимаемый массивом, можно рассчитать по формуле:

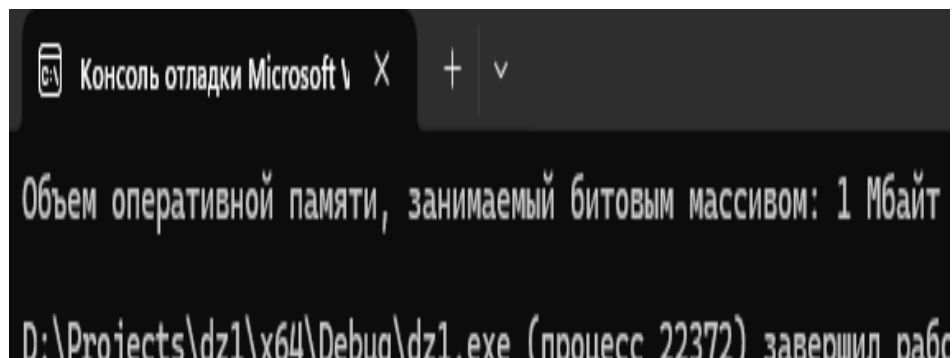
Объём памяти (в байтах) = Размер массива × Размер одного элемента

Код программы:

```
114 int Program_3_b() {
115     int max_value = 10000000 - 1;
116     vector<unsigned char> bit_array((max_value / 8) + 1, 0);
117     size_t memory_size = bit_array.size() * sizeof(unsigned char);
118     cout << "Объём оперативной памяти, занимаемый битовым массивом: " << memory_size / (1024 * 1024) << " Мбайт" << endl;
119     return 0;
120 }
121
```

Рисунок 17 - Реализованный код задания 3.6

Результат тестирования:



Консоль отладки Microsoft Visual Studio. Вывод программы: "Объём оперативной памяти, занимаемый битовым массивом: 1 Мбайт". Внизу строка: "D:\Projects\dz1\x64\Debug\dz1.exe (процесс 22372) завершил работу".

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.