



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
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 № 5.1
«Битовые операции. Сортировка числового файла с помощью битового массива»
по дисциплине
«Структуры и алгоритмы обработки данных»

Выполнил студент группы *ИКБО-43-23*

Коротяев А. М.

Принял
Ассистент

Рысин М.Л.

Практическая
работа выполнена

«__» _____ 2024 г

«Зачтено»

«__» _____ 2024 г

Москва 2024

СОДЕРЖАНИЕ

1 ЦЕЛЬ РАБОТЫ	3
2 ЗАДАНИЕ №1	4
2.1 Формулировка задачи	4
2.2 Описание алгоритма	4
2.3 Код программы	5
2.4 Результаты тестирования	6
3 ЗАДАНИЕ №2	6
3.1 Формулировка задачи	6
3.2 Описание алгоритма	7
3.3 Код программы	8
3.4 Результаты тестирования	10
4 ЗАДАНИЕ №3	12
4.1 Формулировка задачи	12
4.2 Описание алгоритма	13
4.3 Код программы	14
4.4 Результаты тестирования	16
5 ВЫВОД	16

1 ЦЕЛЬ РАБОТЫ

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

2 ЗАДАНИЕ №1

2.1 Формулировка задачи

Задание №1.а. Реализуйте вышеприведенный пример, проверьте правильность результата в том числе и на других значениях x .

Задание №1.б. Реализуйте по аналогии с предыдущим примером установку 7-го бита числа в единицу.

Задание №1.в. Реализуйте код листинга 1, объясните выводимый программой результат.

Листинг 1.

```
1 //Битовые операции
2 #include <cstdlib>
3 #include <iostream>
4 #include <Windows.h>
5 #include <bitset>
6 using namespace std;
7
8 int main()
9 {
10     SetConsoleCP(1251);
11     SetConsoleOutputCP(1251);
12
13     unsigned int x = 25;
14     const int n = sizeof(int)*8; //32 - количество разрядов в числе типа int
15     unsigned maska = (1 << n - 1); //1 в старшем бите 32-разрядной сетки
16     cout << "Начальный вид маски: " << bitset<n>(maska) << endl;
17     cout << "Результат: ";
18     for (int i = 1; i <= n; i++) //32 раза - по количеству разрядов:
19     {
20         cout << ((x & maska) >> (n - i));
21         maska = maska >> 1; //смещение 1 в маске на разряд вправо
22     }
23     cout << endl;
24     system("pause");
25     return 0;
26 }
```

Рисунок 1 – Листинг 1

2.2 Описание алгоритма

Задание №1.а. Происходит сдвиг маски 00000001 влево на 4 позиции. Затем маска инвертируется к виду 11101111 и поразрядно умножается на число. Таким образом, мы устанавливаем 5-й бит числа в 0. После этого выводим получившееся число в десятичном виде.

Задание №1.б. Происходит сдвиг маски 00000001 влево на 6 позиций. Затем происходит операция поразрядного “ИЛИ” с маской и числом. Таким образом, мы устанавливаем 7-й бит числа в 1. После этого выводим получившееся число в десятичном виде.

Задание №1.в. Создается маска с единицей в старшем бите. Число побитового умножается на эту маску и сдвигается вправо на такое количество, чтобы бит, в котором у маски была 1 стал самым правым. Значение этого бита выводится на экран, а маска сдвигается вправо на 1 разряд. Этот цикл повторяется 32 раза (число бит в int) и поразрядно выводит битовое значение исходного числа.

2.3 Код программы

```

7 void Task_1a() {
8     unsigned char x = 255; // 11111111
9     unsigned char maska = 1; // 00000001
10
11     cout << "Исходное значение x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
12     x &= ~(maska << 4);
13     cout << "Результат x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
14
15     x = 127; // 01111111
16     cout << "Исходное значение x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
17     x &= ~(maska << 4);
18     cout << "Результат x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
19
20     x = 64; // 01000000
21     cout << "Исходное значение x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
22     x &= ~(maska << 4);
23     cout << "Результат x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
24 }

```

Рисунок 2 – Код задания №1а

```

26 void Task_1b() {
27     unsigned char x = 127;
28     unsigned char maska = 1;
29
30     cout << "Исходное значение x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
31     x |= (maska << 6);
32     cout << "Результат x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
33
34     x = 0; // 00000000
35     cout << "Исходное значение x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
36     x |= (maska << 6);
37     cout << "Результат x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
38
39     x = 59; // 00111011
40     cout << "Исходное значение x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
41     x |= (maska << 6);
42     cout << "Результат x: " << (int)x << " (" << bitset<8>(x) << ")" << "\n";
43 }

```

Рисунок 3 – Код задания №1б

работу программы.

Задание №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`.

Задание №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`.

3.2 Описание алгоритма

Задание №2.а. Инициализируется нулем “битовый массив” `unsigned char bitArray`. Пользователь вводит числа в диапазоне от 0 до 7, они записываются в битовый массив: происходит сдвиг единицы на количество позиций, равное введенному числу, и проводится побитовое ИЛИ с `bitArray`. Эта операция устанавливает 1 в бит, соответствующий введенному числу. После ввода всех чисел с помощью `for` проходится по “битовому массиву”. Если *i*-й бит равен 1, значит *i* число было введено пользователем, оно выводится на экран.

Задание №2.б. Алгоритм аналогичен №2.а., однако вместо `unsigned char` используется `unsigned long long` для увеличения вместимости “битового массива”.

Задание №2.в. Инициализируется нулем “битовый массив” `vector<unsigned char> bitArray`. Пользователь вводит числа в диапазоне от 0 до 7, для каждого введенного числа вычисляется индекс байта `byteIndex` (номер элемента в векторе `bitArray`) и индекс бита `bitIndex` внутри этого байта, с помощью побитовой операции ИЛИ и сдвига влево устанавливается бит с индексом `bitIndex` в байте `byteIndex` в значение 1. После ввода всех чисел с помощью `for` проходится по “битовому массиву”. Вычисляется индекс байта `byteIndex` и индекс бита `bitIndex`. Проверяется, установлен ли бит с индексом `bitIndex` в байте `byteIndex` в 1. Это делается с помощью сдвига вправо (`>>`) на `bitIndex` позиций и побитовой операции И (`&`) с числом 1. Если результат проверки равен 1, то число *i* выводится на экран.

2.3 Код программы

```
8 void Task_2a() {
9     unsigned char bitArray = 0;
10    int num, count;
11
12    cout << "Введите количество чисел (от 1 до 8): ";
13    cin >> count;
14
15    if (count < 1 || count > 8) {
16        cout << "Неверное количество чисел.\n";
17        return;
18    }
19
20    cout << "Введите числа от 0 до 7: ";
21    for (int i = 0; i < count; ++i) {
22        cin >> num;
23        if (num >= 0 && num <= 7) {
24            if ((bitArray >> num) & 1) {
25                cout << "Число " << num << " уже введено. Повторите ввод.\n";
26                --i;
27            }
28            else {
29                bitArray |= (1 << num);
30            }
31        }
32        else {
33            cout << "Неверный ввод. Введите число от 0 до 7" << "\n";
34            --i;
35        }
36    }
37
38    cout << "Отсортированные числа: ";
39    for (int i = 0; i < 8; ++i) {
40        if ((bitArray >> i) & 1) {
41            cout << i << " ";
42        }
43    }
44    cout << "\n";
45 }
```

Рисунок 6 – Код задания №2а


```

47 void Task_2b() {
48     unsigned long long bitArray = 0;
49     int num, count;
50
51     cout << "Введите количество чисел (от 1 до 64): ";
52     cin >> count;
53
54     if (count < 1 || count > 64) {
55         cout << "Неверное количество чисел.\n";
56         return;
57     }
58
59     cout << "Введите числа от 0 до 63: ";
60     for (int i = 0; i < count; ++i) {
61         cin >> num;
62         if (num >= 0 && num <= 63) {
63             if ((bitArray >> num) & 1) {
64                 cout << "Число " << num << " уже введено. Повторите ввод.\n";
65                 --i;
66             }
67             else {
68                 bitArray |= (1ULL << num);
69             }
70         }
71         else {
72             cout << "Неверный ввод. Введите число от 0 до 63" << "\n";
73             --i;
74         }
75     }
76
77     cout << "Отсортированные числа: ";
78     for (int i = 0; i < 64; ++i) {
79         if ((bitArray >> i) & 1) {
80             cout << i << " ";
81         }
82     }
83     cout << "\n";
84 }

```

Рисунок 7 – Код задания №26

```

86 void Task_2c() {
87     vector<unsigned char> bitArray(8);
88     int num, count;
89
90     cout << "Введите количество чисел (от 1 до 64): ";
91     cin >> count;
92
93     if (count < 1 || count > 64) {
94         cout << "Неверное количество чисел.\n";
95         return;
96     }
97
98     cout << "Введите числа от 0 до 63: ";
99     for (int i = 0; i < count; ++i) {
100         cin >> num;
101         if (num >= 0 && num <= 63) {
102             int byteIndex = num / 8;
103             int bitIndex = num % 8;
104             if ((bitArray[byteIndex] >> bitIndex) & 1) {
105                 cout << "Число " << num << " уже введено. Повторите ввод.\n";
106                 --i;
107             }
108             else {
109                 bitArray[byteIndex] |= (1 << bitIndex);
110             }
111         }
112         else {
113             cout << "Неверный ввод. Введите число от 0 до 63" << "\n";
114             --i;
115         }
116     }
117
118     cout << "Отсортированные числа: ";
119     for (int i = 0; i < 64; ++i)
120     {
121         int byteIndex = i / 8;
122         int bitIndex = i % 8;
123         if ((bitArray[byteIndex] >> bitIndex) & 1)
124         {
125             cout << i << " ";
126         }
127     }
128     cout << "\n";
129 }

```

Рисунок 8 – Код задания №2в

3.4 Результаты тестирования

```

Введите количество чисел (от 1 до 8): 4
Введите числа от 0 до 7: 7 1 4 3
Отсортированные числа: 1 3 4 7

```

Рисунок 9 – Тестирование 1 задания №2а

```
Введите количество чисел (от 1 до 8): 9
Неверное количество чисел.
```

Рисунок 10 – Тестирование 2 задания №2а

```
Введите количество чисел (от 1 до 8): 4
Введите числа от 0 до 7: 5 6 6 1
Число 6 уже введено. Повторите ввод.
2
Отсортированные числа: 1 2 5 6
```

Рисунок 11 – Тестирование 3 задания №2а

```
Введите количество чисел (от 1 до 64): 4
Введите числа от 0 до 63: 80
Неверный ввод. Введите число от 0 до 63
-1
Неверный ввод. Введите число от 0 до 63
3 4 1 69
Неверный ввод. Введите число от 0 до 63
56
Отсортированные числа: 1 3 4 56
```

Рисунок 12 – Тестирование 1 задания №2б

```
Введите количество чисел (от 1 до 64): -18
Неверное количество чисел.
```

Рисунок 13 – Тестирование 2 задания №2б

```
Введите количество чисел (от 1 до 64): 6
Введите числа от 0 до 63: 43
80
Неверный ввод. Введите число от 0 до 63
23
23
Число 23 уже введено. Повторите ввод.
5
3
8
37
Отсортированные числа: 3 5 8 23 37 43
```

Рисунок 14 – Тестирование 1 задания №2в

```
Введите количество чисел (от 1 до 64): 80
Неверное количество чисел.
```

Рисунок 15 – Тестирование 2 задания №2в

4 ЗАДАНИЕ №3

4.1 Формулировка задачи

Входные данные: файл, содержащий не более $n=107$ неотрицательных целых чисел, среди них нет повторяющихся.

Результат: упорядоченная по возрастанию последовательность исходных чисел в выходном файле

Время работы программы: ~ 10 с (до 1 мин. для систем малой вычислительной мощности).

Максимально допустимый объём ОЗУ для хранения данных: 1 МБ.

Задание №3.а. Реализуйте задачу сортировки числового файла с заданными условиями. Добавьте в код возможность определения времени работы программы.

В отчёт внесите результаты тестирования для наибольшего количества входных чисел, соответствующего битовому массиву длиной 1 МБ.

Задание №3.б. Определите программно объём оперативной памяти, занимаемый битовым массивом.

4.2 Описание алгоритма

Задание №3. Программа создает файл input.txt с уникальными случайными числами: генерируются случайные числа и добавляются в множество uniqueNumbers до тех пор, пока количество элементов множества не будет соответствовать необходимому. Для каждого числа вычисляется его позиция в битовом массиве и соответствующий бит устанавливается в 1. Программа учитывает, что в файле числа в диапазоне от 1000000 до 9999999, поэтому числу 1000000 будет соответствовать позиция 0 в “битовом массиве”. После программа создает файл output.txt и в него записывает в отсортированном виде. Функция учитывает диапазон, поэтому при наличии 1 в позиции 7 в output.txt будет записано число 1000007.

Перед заполнением “битового массива” программа запускает отсчет времени выполнения программы и заканчивает его после заполнения файла output.txt отсортированными числами. После этого выводится время сортировки чисел и размер “битового массива”.

4.3 Код программы

```
1  #include "task_3.h"
2  #include <iostream>
3  #include <fstream>
4  #include <vector>
5  #include <chrono>
6  #include <random>
7  #include <numeric>
8
9  namespace {
10     const size_t MAX_NUMBERS = 9000000; // Максимальное кол-во чисел в файле
11     const size_t BIT_ARRAY_SIZE = MAX_NUMBERS / 8; // Размер битового массива в байтах
12     const size_t MIN_VALUE = 1000000; // Минимальное число в файле
13     const size_t MAX_VALUE = 9999999; // Максимальное число в файле
14 }
15
16 void generateUniqueNumbersFile(const string& filename, size_t count) {
17     if (count > (MAX_VALUE - MIN_VALUE + 1)) {
18         cerr << "Ошибка: некорректный диапазон или количество чисел.\n";
19         return;
20     }
21
22     vector<int> numbers(MAX_VALUE - MIN_VALUE + 1);
23     iota(numbers.begin(), numbers.end(), MIN_VALUE);
24
25     // Перемешивание вектора
26     random_device rd;
27     mt19937 g(rd());
28     shuffle(numbers.begin(), numbers.end(), g);
29
30     numbers.resize(count);
31
32     ofstream outputFile(filename);
33     if (outputFile.is_open()) {
34         for (int number : numbers) {
35             outputFile << number << "\n";
36         }
37         outputFile.close();
38     }
39     else {
40         cerr << "Не удалось открыть файл для записи.\n";
41     }
42 }
43
```

Рисунок 16 – Код задания №3 (часть 1/3)

```

45 // Заполнение битового массива данными из файла
46 void fillBitArray(const string& filename, vector<unsigned char>& bitArray) {
47     ifstream inputFile(filename);
48     if (inputFile.is_open()) {
49         int number;
50         while (inputFile >> number) {
51             number -= MIN_VALUE;
52             int byteIndex = number / 8;
53             int bitIndex = number % 8;
54             bitArray[byteIndex] |= (1 << bitIndex);
55         }
56         inputFile.close();
57     }
58     else {
59         cerr << "Не удалось открыть входной файл.\n";
60         return;
61     }
62 }
63
64 // Формирование выходного файла с отсортированными числами
65 void generateOutputFile(const string& filename, const vector<unsigned char> bitArray) {
66     ofstream outputFile(filename);
67     if (outputFile.is_open()) {
68         for (int i = 0; i < BIT_ARRAY_SIZE; ++i) {
69             for (int j = 0; j < 8; ++j) {
70                 if ((bitArray[i] >> j) & 1) {
71                     outputFile << (i * 8 + j) + MIN_VALUE << "\n";
72                 }
73             }
74         }
75         outputFile.close();
76     }
77     else {
78         cerr << "Не удалось открыть выходной файл.\n";
79         return;
80     }
81 }

```

Рисунок 17 – Код задания №3 (часть 2/3)

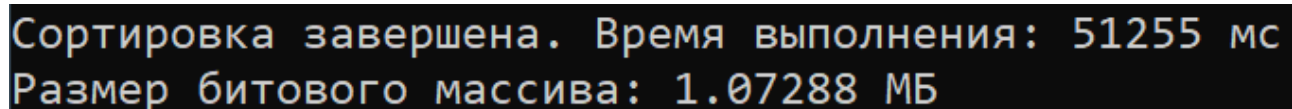
```

83 void Task_3(size_t numberCount) {
84     // Битовый массив
85     vector<unsigned char> bitArray(BIT_ARRAY_SIZE, 0);
86
87     generateUniqueNumbersFile("input.txt", numberCount);
88
89     auto start = chrono::high_resolution_clock::now();
90
91     fillBitArray("input.txt", bitArray);
92
93     generateOutputFile("output.txt", bitArray);
94
95     // Замер времени окончания сортировки
96     auto end = chrono::high_resolution_clock::now();
97     auto duration = chrono::duration_cast<chrono::milliseconds>(end - start);
98
99     cout << "Сортировка завершена. Время выполнения: " << duration.count() << " мс\n";
100     cout << "Размер битового массива: " << bitArray.size() / (1024.0 * 1024.0) << " МБ\n";
101 }

```

Рисунок 18 – Код задания №3 (часть 3/3)

4.4 Результаты тестирования



Сортировка завершена. Время выполнения: 51255 мс
Размер битового массива: 1.07288 МБ

Рисунок 19 – Тестирование задания №3

5 ВЫВОД

Были освоены приемы работы с битовым представлением беззнаковых целых чисел, реализован эффективный алгоритм внешней сортировки на основе битового массива. Тестирование показало, что все программы работают правильно.