



МИНОБРНАУКИ РОССИИ
*Федеральное государственное бюджетное образовательное учреждение высшего
образования*
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №5.1

Тема:

Битовые операции. Сортировка числового файла с помощью битового массива.
Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Васильев Б.А.

Группа: ИКБО-20-23

Москва 2024

СОДЕРЖАНИЕ

ЗАДАНИЕ 1	3
Формулировка задачи	3
Математическая модель решения (описание алгоритма)	4
Коды программ	5
Результаты тестирования	7
ЗАДАНИЕ 2	9
Формулировка задачи	9
Математическая модель решения (описание алгоритма)	9
Коды программ	11
Результаты тестирования	13
ЗАДАНИЕ 3	15
Формулировка задачи	15
Математическая модель решения (описание алгоритма)	16
Коды программ	17
Результаты тестирования	18
ВЫВОД.....	19
СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ	19

ЦЕЛЬ РАБОТЫ

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

ЗАДАНИЕ 1

Формулировка задачи

Определить делится ли число на каждую из своих цифр.

1.а. Реализуйте нижеприведённый пример с рисунка 1, 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 с рисунка 2, объясните выводимый программой результат.

```
unsigned char x=255;           //8-разрядное двоичное число 11111111
unsigned char maska = 1;       //1=00000001 – 8-разрядная маска
x = x & (~ (maska<<4));        //результат x=239
```

Рисунок 1 – Установка 5-го бита произвольного целого числа в 0

```
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 2 – к задаче 1.в

Математическая модель решения (описание алгоритма)

В заданиях 1.а. и 1.б. происходит установка n-го бита целого числа в 0. Сначала у пользователя запрашивается целое число от 0 до 255 (максимальное число представимое восемью двоичными разрядами). Затем создаётся маска вида 00000001. Далее с помощью побитового сдвига влево на n разрядов, получаем маску с единицей в том разряде, в котором у исходного числа требуется выставить ноль, инвертируем маску и с помощью операции “побитовое И” получаем необходимый результат.

Коды программ

Реализуем алгоритмы на языке программирования C++ (рис. 3-5)

```
1  > #include <iostream>
2  > #include <limits>
3  > #include <bitset>
4
5  > using namespace std;
6
7  > int main()
8  {
9      const int size = sizeof(unsigned char) * 8;
10     int input;
11     > while (true)
12     {
13         cout << "Provide a number between 0 and 255: ";
14         if (cin >> input && input >= 0 && input <= 255) break;
15
16         cin.clear();
17         cin.ignore(numeric_limits<streamsize>::max(), '\n');
18         cout << "Invalid input, try again\n";
19     }
20
21     unsigned char x = static_cast<unsigned char>(input);
22     cout << "Binary representation: " << bitset<size>(x);
23     unsigned char maska = 1;
24     x = x & (~(maska << 4));
25     cout << "\nResult with 5th bit set to 0: " << (int) x;
26     cout << "\nBinary representation: " << bitset<size>(x);
27
28     return 0;
29 }
```

Рисунок 3 – Код к задаче 1.а

```

1  #include <iostream>
2  #include <limits>
3  #include <bitset>
4
5  using namespace std;
6
7  int main()
8  {
9      const int size = sizeof(unsigned char) * 8;
10     int input;
11     while (true)
12     {
13         cout << "Provide a number between 0 and 255: ";
14         if (cin >> input && input >= 0 && input <= 255) break;
15
16         cin.clear();
17         cin.ignore(numeric_limits<streamsize>::max(), '\n');
18         cout << "Invalid input, try again\n";
19     }
20
21     unsigned char x = static_cast<unsigned char>(input);
22     cout << "Binary representation: " << bitset<size>(x);
23     unsigned char maska = 1;
24     x = x | (maska << 6);
25     cout << "\nResult with 7th bit set to 1: " << (int) x;
26     cout << "\nBinary representation: " << bitset<size>(x);
27
28     return 0;
29 }

```

Рисунок 4 – Код к задаче 1.6

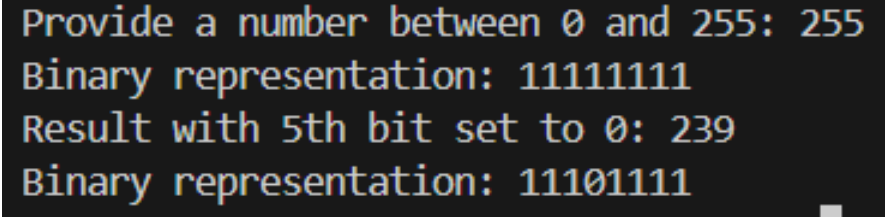
```

1  #include <cstdlib>
2  #include <iostream>
3  #include <Windows.h>
4  #include <bitset>
5  using namespace std;
6
7  int main()
8  {
9      setlocale(LC_ALL, "ru_RU.UTF-8");
10
11     SetConsoleCP(1251);
12     SetConsoleOutputCP(1251);
13
14     unsigned int x = 25; // сохраняется в виде 00000000 00000000 00000000 00011001
15     const int n = sizeof(int)*8; // = 32 количество разрядов в числе типа int
16     unsigned maska = (1 << n - 1); // 1 в старшем бите 32-разрядной сетки
17     cout << "Начальный вид маски: " << bitset<n>(maska) << endl;
18     cout << "Результат: ";
19     for (int i = 1; i <= n; i++) // 32 раза по количеству разрядов
20     {
21         cout << ((x & maska) >> (n - i));
22         /*побитовое И над числом x и маской, смещение результата на n-i разряда вправо,
23         т.е. i-й разряд окажется самым правым в разрядной сетке, остальные биты будут нулевыми,
24         а значит просто будет выведено значение i-го бита*/
25         maska = maska >> 1; // смещаем 1 на один разряд правее в маске
26     }
27     cout << endl; // в итоге получаем на выходе двоичную запись исходного числа, в нашем случае числа 25
28     system("pause");
29     return 0;

```

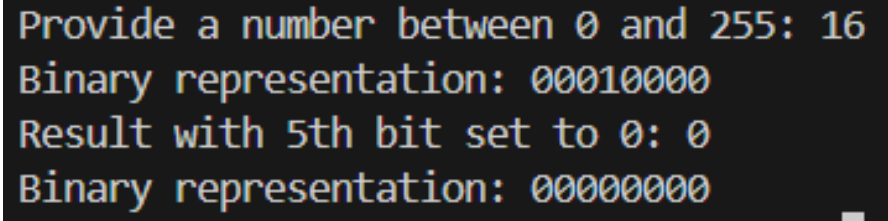
Рисунок 5 – Код к задаче 1.в и его объяснение

Результаты тестирования



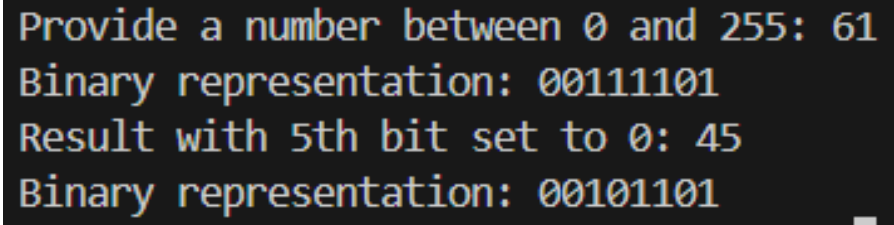
```
Provide a number between 0 and 255: 255  
Binary representation: 11111111  
Result with 5th bit set to 0: 239  
Binary representation: 11101111
```

Рисунок 6 – Тестирование кода к задаче 1.а (часть 1)



```
Provide a number between 0 and 255: 16  
Binary representation: 00010000  
Result with 5th bit set to 0: 0  
Binary representation: 00000000
```

Рисунок 7 – Тестирование кода к задаче 1.а (часть 2)



```
Provide a number between 0 and 255: 61  
Binary representation: 00111101  
Result with 5th bit set to 0: 45  
Binary representation: 00101101
```

Рисунок 8 – Тестирование кода к задаче 1.а (часть 3)

```
Provide a number between 0 and 255: 0
Binary representation: 00000000
Result with 7th bit set to 1: 64
Binary representation: 01000000
```

Рисунок 9 – Тестирование кода к задаче 1.б (часть 1)

```
Provide a number between 0 and 255: 191
Binary representation: 10111111
Result with 7th bit set to 1: 255
Binary representation: 11111111
```

Рисунок 10 – Тестирование кода к задаче 1.б (часть 2)

```
Provide a number between 0 and 255: 165
Binary representation: 10100101
Result with 7th bit set to 1: 229
Binary representation: 11100101
```

Рисунок 11 – Тестирование кода к задаче 1.б (часть 3)

```
Начальный вид маски: 10000000000000000000000000000000
Результат: 0000000000000000000000000000000011001
Press any key to continue . . .
```

Рисунок 12 – Тестирование кода к задаче 1.в

Тестирование показало, что программы работают исправно (рис. 6-12).

ЗАДАНИЕ 2

Формулировка задачи

Определить делится ли число на каждую из своих цифр.

2.а. Реализуйте сортировку последовательности чисел с помощью двоичного массива 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.

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`.

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`.

Математическая модель решения (описание алгоритма)

Алгоритм решения задания 2.а:

1. Ввод размера массива:

- Программа просит у пользователя ввести размер массива, который должен быть в пределах от 1 до 8 включительно.
- Если введённое значение не является числом или находится вне указанного диапазона, программа выводит сообщение об ошибке и повторяет запрос.

2. Ввод уникальных чисел:

- После успешного ввода размера массива программа запрашивает у пользователя ввести уникальные числа в диапазоне от 0 до 7.
- Если введённое число не соответствует условиям (например, не число или вне диапазона), программа выводит сообщение об ошибке и повторяет запрос.
- Введённые числа используются для установки соответствующих битов в переменной `bit_array`.

3. Установка битов:

- Программа использует побитовое ИЛИ с присвоением (`|=`) для установки битов в переменной `bit_array` на основе введённых чисел. Единица из битовой маски перемещается влево на количество разрядов равное введённому числу, а затем идёт установка в единицу соответствующего разряда в переменной `bit_array`.

4. Вывод отсортированного массива:

- Программа проходит по всем 8 битам переменной `bit_array`, начиная с младшего, и проверяет, установлены ли они в единицу.
- Если бит установлен, программа выводит соответствующее число.

Алгоритм решения задания 2.б аналогичен предыдущему:

1. **Ввод размера массива:**
 - Программа просит у пользователя ввести размер массива, который должен быть в пределах от 1 до 64 включительно.
 - Если введённое значение не является числом или находится вне указанного диапазона, программа выводит сообщение об ошибке и повторяет запрос.
2. **Ввод уникальных чисел:**
 - После успешного ввода размера массива программа запрашивает у пользователя ввести уникальные числа в диапазоне от 1 до 63.
 - Если введённое число не соответствует условиям (например, не число или вне диапазона), программа выводит сообщение об ошибке и повторяет запрос.
 - Введённые числа используются для установки соответствующих битов в переменной `bit_array`.
3. **Установка битов:**
 - Программа использует побитовое ИЛИ с присвоением (`|=`) для установки битов в переменной `bit_array` на основе введённых чисел. Единица из битовой маски перемещается влево на количество разрядов равное введённому числу, а затем идёт установка в единицу соответствующего разряда в переменной `bit_array`.
4. **Вывод отсортированного массива:**
 - Программа проходит по всем 64 битам переменной `bit_array` и проверяет, установлены ли они в единицу.
 - Если бит установлен, программа выводит соответствующее число.

Алгоритм решения задания 2.в:

1. **Инициализация данных:**
 - Создаётся вектор целых чисел `numbers` размером 64 для хранения вводимых значений.
 - Переменная `max_num` используется для определения наибольшего числа из введённых.
2. **Ввод чисел:**
 - Программа просит пользователя ввести 64 числа.
 - Если очередное введённое число превосходит значение переменной `max_num`, то происходит обновление значения этой переменной.
3. **Подготовка битового массива:**
 - Вычисляется количество байтов, необходимых для хранения битовой маски, используя наибольшее число из введённых. Это количество байтов хранится в `bytes_size`.
 - Массив `bytes` инициализируется нулями. Размер массива `bytes` равен `bytes_size`.
4. **Установка битов:**
 - Для каждого введённого числа определяется соответствующий байт и устанавливается соответствующий бит в этом байте. Это делается с помощью побитового ИЛИ с присвоением (`|=`) и сдвига маски (`>>`).
5. **Вывод отсортированной последовательности:**
 - Происходит проход по каждому биту каждого байта из массива `bytes`. Если очередной бит установлен в 1, то индекс этого бита из сквозной нумерации выводится в консоль.

Коды программ

Реализуем алгоритмы на языке программирования C++ (рис. 13-16)

```
int main()
{
    unsigned char bit_array = 0, one_mask = 1; // создание битового массива заполненного нулями и маски вида 00000001
    size_t SIZE;
    while (true) // запрос пользователя ввести количество чисел в последовательности
    {
        cout << "Enter array size between 1 and 8: ";
        if (cin >> SIZE && SIZE >= 1 && SIZE <= 8) break; // проверка введённого пользователем значения

        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Invalid input, try again\n";
    }

    cout << "Enter an array of " << SIZE << " unique numbers between 0 and 7: "; // запрос пользователя ввести последовательность чисел
    int curr_number;
    for (int i = 0; i < SIZE; i++)
    {
        while (true)
        {
            if (cin >> curr_number && curr_number >= 0 && curr_number <= 7) break; // проверка введённого пользователем значения

            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid number, try again\n";
        }
        bit_array |= (one_mask << curr_number); // установка соответствующего бита в массиве в единицу
    }

    cout << "Sorted array:"; // проход по всем битам массива и вывод отсортированной последовательности
    for (int i = 0; i < 8; i++)
    {
        if (bit_array & (one_mask << i))
        {
            cout << " " << i;
        }
    }
}
```

Рисунок 13 – Код к задаче 2.а

```

int main()
{
    unsigned long long bit_array = 0, one_mask = 1; // создание битового массива заполненного нулями и маски вида 000...001
    size_t MAX_NUMBER = sizeof(unsigned long long) * 8 - 1; // подсчёт индекса самого левого бита (самого старшего) битового массива
    size_t SIZE;
    while (true) // // запрос пользователя ввести количество чисел в последовательности
    {
        cout << "Enter array size between 1 and " << MAX_NUMBER + 1 << ": ";
        if (cin >> SIZE && SIZE >= 1 && SIZE <= MAX_NUMBER + 1) break; // проверка введённого пользователем значения

        cin.clear();
        cin.ignore(numeric_limits<streamsize>::max(), '\n');
        cout << "Invalid input, try again\n";
    }

    cout << "Enter an array of " << SIZE << " unique numbers between 0 and " << MAX_NUMBER << ": "; // запрос пользователя ввести последовательность чисел
    int curr_number;
    for (int i = 0; i < SIZE; i++)
    {
        while (true)
        {
            if (cin >> curr_number && curr_number >= 0 && curr_number <= MAX_NUMBER) break; // проверка введённого пользователем значения

            cin.clear();
            cin.ignore(numeric_limits<streamsize>::max(), '\n');
            cout << "Invalid number, try again\n";
        }
        bit_array |= (one_mask << curr_number); // установка соответствующего бита в массиве в единицу
    }

    cout << "Sorted array:";
    for (int i = 0; i < MAX_NUMBER + 1; i++)
    {
        if (bit_array & (one_mask << i))
        {
            cout << " " << i;
        }
    }
}

```

Рисунок 14 – Код к задаче 2.б

```

int main()
{
    const size_t SIZE = 64;
    vector<int> numbers(SIZE); // создание вектора целочисленных чисел
    int max_num = -1;

    cout << "Enter an array of " << SIZE << " numbers: "; // запрос пользователя ввести последовательность из 64 чисел
    for (auto &el: numbers)
    {
        cin >> el; // сохранение введённого числа в вектор
        if (el > max_num) max_num = el; // обновление значения максимального введённого числа
    }

    size_t bytes_size = (max_num / 8) + 1; // подсчёт необходимого количества байтов для последующих операций
    unsigned char bytes[bytes_size]; // создание массива из байтов
    for (int i = 0; i < bytes_size; i++)
    {
        bytes[i] = 0; // установка каждого байта в нулевое значение (а соответственно и обнуление всех битов)
    }

    for (auto &el: numbers) // проход по вектору введённых чисел
    {
        bytes[el / 8] |= (1 << (el%8)); // установка нужного бита нужного бита в 1 в соответствии с числом из вектора
    }

    cout << "\nBit array (Left byte has the most significant bit): "; // вывод массива байтов в двоичном виде
    for (int i = 0; i < bytes_size; i++)
    {
        cout << bitset<8>(bytes[bytes_size - 1 - i]) << " ";
    }
}

```

Рисунок 15 – Код к задаче 2.в (часть 1)

```

cout << "\n\nSorted array: ";
unsigned char mask;
for (int i = 0; i < bytes_size; i++) // проход по всем байтам из массива
{
    mask = 1; // создание маски вида 00000001
    for (int j = 0; j < 8; j++) // проход по всем битам байта
    {
        if (bytes[i] & mask) // если текущий бит установлен в единицу
        {
            cout << i*8 + j << " "; // вывод соответствующего числа
        }
        mask <<= 1; // сдвиг единицы в маске на одну позицию влево
    }
}
}

```

Рисунок 16 – Код к задаче 2.в (часть 2)

Результаты тестирования

```

Enter array size between 1 and 8: 3
Enter an array of 3 unique numbers between 0 and 7: 7 2 5
Sorted array: 2 5 7

```

Рисунок 17 – Тестирование кода к задаче 2.а (часть 1)

```

Enter array size between 1 and 8: 8
Enter an array of 8 unique numbers between 0 and 7: 5 1 3 7 2 0 4 6
Sorted array: 0 1 2 3 4 5 6 7

```

Рисунок 18 – Тестирование кода к задаче 2.а (часть 2)

```

Enter array size between 1 and 64: 3
Enter an array of 3 unique numbers between 0 and 63: 52 14 39
Sorted array: 14 39 52

```

Рисунок 19 – Тестирование кода к задаче 2.б (часть 1)

```

Enter array size between 1 and 64: 64
Enter an array of 64 unique numbers between 0 and 63: 56 28 24 63 25 52
58 45 32 10 9 31 22 12 36 14 61 60 6 43 29 18 26 5 40 16 50 11 2 48 3 37
54 0 46 8 21 19 55 62 30 44 33 47 42 53 15 39 20 59 7 51 23 41 35 38 17
34 13 49 27 4 57 1
Sorted array: 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 43 44 45 46
47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63

```

Рисунок 20 – Тестирование кода к задаче 2.б (часть 2)

```

Enter array size between 1 and 64: 32
Enter an array of 32 unique numbers between 0 and 63: 26 35 28 12 49 5 38 27 44 37 50 56 52 61 31 30 1 48 7 43
20 17 2 45 4 62 10 46 53 63 11 18
Sorted array: 1 2 4 5 7 10 11 12 17 18 20 26 27 28 30 31 35 37 38 43 44 45 46 48 49 50 52 53 56 61 62 63

```

Рисунок 21 – Тестирование кода к задаче 2.б (часть 3)

```

Enter an array of 64 numbers: 199 236 5 147 185 111 61 223 246 141 159 44 123 4 215 222 2
24 7 171 73 227 151 32 157 194 203 220 84 36 46 99 59 198 219 231 43 20 45 3 237 54 95 83
72 65 193 55 216 200 71 10 137 63 121 183 174 209 57 25 89 62 103 90 206

Bit array (Left byte has the most significant bit): 01000000 00110000 10001001 11011001 1
0000010 01001001 11000110 00000010 10000000 01001000 00000000 10100000 10001000 00100010
00000000 00001010 00000000 10000000 10001000 10000110 00011000 00000011 10000010 11101010
11000000 01111000 00010001 00000010 00010000 00000100 10111000

Sorted array: 3 4 5 7 10 20 25 32 36 43 44 45 46 54 55 57 59 61 62 63 65 71 72 73 83 84 8
9 90 95 99 103 111 121 123 137 141 147 151 157 159 171 174 183 185 193 194 198 199 200 20
3 206 209 215 216 219 220 222 223 224 227 231 236 237 246

```

Рисунок 22 – Тестирование кода к задаче 2.в (часть 1)

```

Enter an array of 64 numbers: 8 5 52 35 20 43 14 23 59 1 19 15 9 4 46 49 61 16 60 27 32
6 26 36 63 54 31 57 50 40 13 56 39 3 38 11 44 37 17 21 29 12 28 0 58 24 34 10 18 41 51
62 48 30 64 53 42 47 7 33 45 2 55 25

Bit array (Left byte has the most significant bit): 00000001 11111111 11111111 11111111
11111111 11111111 10111111 11111111 11111111

Sorted array: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 23 24 25 26 27 28
29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57
58 59 60 61 62 63 64

```

Рисунок 23 – Тестирование кода к задаче 2.в (часть 2)

Тестирование показало, что программы работают исправно (рис.17-23).

ЗАДАНИЕ 3

Формулировка задачи

Входные данные: файл, содержащий не более $n=10^7$ 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.

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.

3.б. Определите программно объём оперативной памяти, занимаемый битовым массивом

Математическая модель решения (описание алгоритма)

Алгоритм решения задания 3.а и 3.б:

1. **Входные данные:**
 - Считываем набор целых чисел из файла. Необходимо определить, какие числа встречаются, и сохранить эту информацию в ОЗУ.
2. **Хранение чисел в виде битов:**
 - Используем динамический массив (вектор) байтов. Каждый байт в векторе содержит 8 битов, и каждый бит отвечает за информацию о наличии одного конкретного числа.
 - Размер вектора увеличивается по мере необходимости: если текущее число превышает количество уже доступных бит для его хранения, вектор автоматически расширяется, добавляя новые байты.
3. **Привязка чисел к битам:**
 - Для каждого числа N определяем, в каком байте и в каком бите этого байта оно должно храниться:
 1. Индекс нужного байта находим как целую часть от деления N на 8.
 2. Индекс нужного бита внутри байта находим как остаток от деления N на 8.
 - Устанавливаем соответствующий бит в единицу с помощью битовой маски и операций побитового ИЛИ с присваиванием ($|=$) и сдвига ($<<$).
4. **Установка битов:**
 - Для каждого введенного числа определяется соответствующий байт и устанавливается соответствующий бит в этом байте. Это делается с помощью побитового ИЛИ с присвоением ($|=$) и сдвига маски ($<<$).
5. **Сохранение отсортированной последовательности:**
 - Закрываем файл со входными данными, он нам больше не потребуется.
 - Открываем файл для записи.
 - Проходим последовательно по каждому биту каждого байта, начиная с самого младшего, если значение двоичного разряда равно 1, то записываем индекс этого бита в файл.

Коды программ

Реализуем алгоритм на языке программирования C++ (рис. 25-26)

```
1  #include <vector>
2  #include <iostream>
3  #include <fstream>
4  #include <chrono>
5
6  using namespace std;
7
8  int main()
9  {
10     vector<unsigned char> bytes;
11     size_t byte_size = sizeof(unsigned char) * 8;
12     string input_file_name;
13     cout << "Enter input file name: ";
14     cin >> input_file_name;
15     auto start_time = chrono::high_resolution_clock::now(); // начало отсчёта времени выполнения
16     ifstream input_file(input_file_name);
17     if (input_file.is_open()) // проверка успешного открытия файла для считывания
18     {
19         int num;
20         while (input_file >> num) // считывается каждое число из файла
21         {
22             // если текущего количества байтов в векторе не хватает чтобы отметить число, добавить ещё один байт
23             while (num >= (bytes.size() * byte_size))
24             {
25                 bytes.push_back(0);
26             }
27             bytes[num / byte_size] |= (1 << (num % byte_size)); // установка соответствующего бита в единицу
28         }
29         input_file.close(); // закрываем файл
30     }
31     else
32     {
33         cerr << "Error reading input file";
34         return 1;
35     }
36 }
```

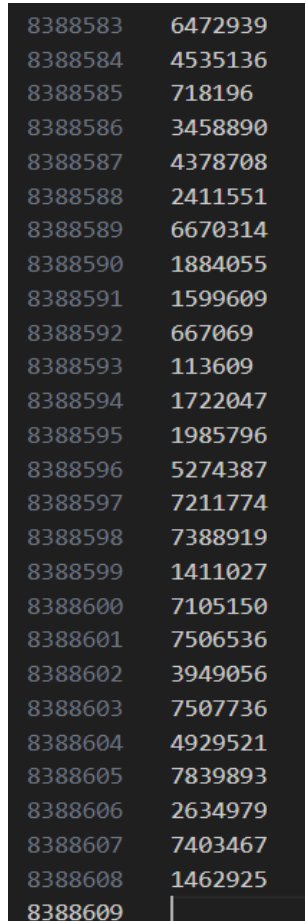
Рисунок 24 – Код к задаче 3.а и 3.б (часть 1)

```
37     unsigned char mask;
38     ofstream output_file("output.txt");
39     if (output_file.is_open()) // проверка успешного создания и открытия выходного файла
40     {
41         for (int i = 0; i < bytes.size(); i++) // проход по всем байтам
42         {
43             mask = 1; // создание маски вида 00000001
44             for (int j = 0; j < byte_size; j++) // проход по всем битам
45             {
46                 if (bytes[i] & mask) // если текущий бит установлен в 1
47                 {
48                     output_file << i*byte_size + j << "\n"; // выписывание соответствующего числа
49                 }
50                 mask <<= 1;
51             }
52         }
53         output_file.close(); // сохраняем и закрываем файл
54     }
55     else
56     {
57         cerr << "Error creating output file";
58         return -1;
59     }
60     auto end_time = chrono::high_resolution_clock::now(); // конец времени отсчёта выполнения
61     auto time_delta = chrono::duration_cast<chrono::milliseconds>(end_time - start_time); // подсчёт времени выполнения
62     cout << "Completion time in milliseconds: " << time_delta.count(); // вывод времени выполнения
63     cout << "\nRAM utilization in bits: " << bytes.size() * byte_size; // вывод затраченного количества ОЗУ
64 }
```

Рисунок 25 – Код к задаче 3.а и 3.б (часть 2)

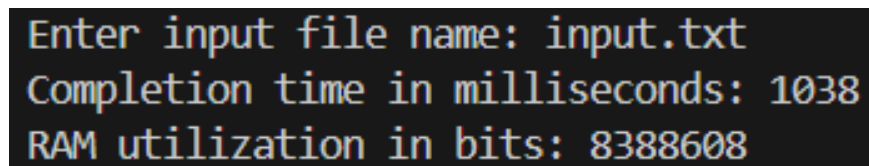
Результаты тестирования

Ограничение длины битового массива в 1 МБ позволяет обработать файл входных данных с неповторяющимися целыми неотрицательными числами, каждое из которых не превосходит по значению число 8 388 607, т.к. $1 \text{ МБ} = 1024 * 1 \text{ КБ} = 1024 * 1024 * 1 \text{ байт} = 1024 * 1024 * 8 \text{ бит} = 8\,388\,608 \text{ бит}$. Создадим файл из 8 388 608 чисел (рис. 27) и проверим нашу программу на нём (рис. 28).



8388583	6472939
8388584	4535136
8388585	718196
8388586	3458890
8388587	4378708
8388588	2411551
8388589	6670314
8388590	1884055
8388591	1599609
8388592	667069
8388593	113609
8388594	1722047
8388595	1985796
8388596	5274387
8388597	7211774
8388598	7388919
8388599	1411027
8388600	7105150
8388601	7506536
8388602	3949056
8388603	7507736
8388604	4929521
8388605	7839893
8388606	2634979
8388607	7403467
8388608	1462925
8388609	

Рисунок 27 – Фрагмент файла входных данных



```
Enter input file name: input.txt
Completion time in milliseconds: 1038
RAM utilization in bits: 8388608
```

Рисунок 28 – Тестирование кода к задаче 3.а и 3.б

Как видно, программа работает корректно и укладывается в поставленные ограничения по времени работы и количеству используемой памяти.

ВЫВОД

В результате выполнения работы освоены приёмы работы с битовым представлением беззнаковых целых чисел, а также реализован эффективный алгоритм внешней сортировки на основе битового массива.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Рысин, М. Л. Введение в структуры и алгоритмы обработки данных : учебное пособие / М. Л. Рысин, М. В. Сартаков, М. Б. Туманова. — Москва : РТУ МИРЭА, 2022 — Часть 2 : Поиск в тексте. Нелинейные структуры данных. Кодирование информации. Алгоритмические стратегии — 2022. — 111 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: <https://e.lanbook.com/book/310826> (дата обращения: 10.09.2024).
2. Документация по языку C++ [Электронный ресурс]. URL: <https://docs.microsoft.com/ruru/cpp/cpp/> (дата обращения 10.09.2024).