



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 №5**  
**по дисциплине**  
**«Структуры и алгоритмы обработки данных»**

Тема. Работа с данными из файла

Выполнил студент группы ИКБО-43-23

Кощеев М. И.

Принял старший преподаватель

Рысин М.Л.

Москва 2024

## СОДЕРЖАНИЕ

1	ЦЕЛЬ.....	3
2	ХОД РАБОТЫ (5.1).....	4
2.1	Задание 1а .....	4
2.2	Задание 1б .....	5
2.3	Задание 1в .....	6
2.4	Задание 2а .....	7
2.5	Задание 2б .....	9
2.6	Задание 2в .....	11
2.7	Задание 3 (а и б).....	13
3	ХОД РАБОТЫ (5.2).....	15
3.1	Задание 1.1 .....	15
3.2	Задание 1.2 .....	18
3.3	Задание 1.3 .....	19
4	ВЫВОДЫ.....	21
5	ИСПОЛЬЗУЕМЫЙ ИСТОЧНИК .....	22

## **1 ЦЕЛЬ**

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

## 2 ХОД РАБОТЫ (5.1)

### 2.1 Задание 1a

Реализуйте вышеприведённый пример, проверьте правильность результата в том числе и на других значениях x.

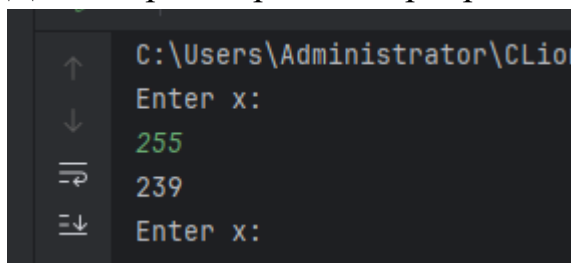
Алгоритм решения:

1. Побитовый сдвиг влево единичной маски на нужное количество бит-1
2. Инверсия
3. Поразрядное И числа и маски

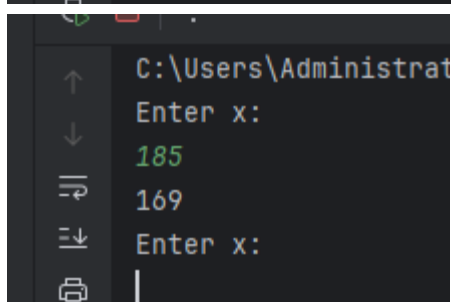
Код:

```
15
16 void task1_a() {
17     unsigned int x;
18     cout<<"Enter x: "<<endl;
19     cin>>x;
20     unsigned char maska=1;
21     x=x&(~(maska<<4));
22     cout<<x<<endl;
23 }
24
```

Демонстрация работы программы:



```
C:\Users\Administrator\CLion
Enter x:
255
239
Enter x:
```



```
C:\Users\Administrator
Enter x:
185
169
Enter x:
```

Тесты:

Входные данные (x)	Ожидаемый результат	Фактический результат
255	239	239
185	169	169

Вывод по заданию:

Был изучен способ преобразования числа с помощью битовых операций.

## 2.2 Задание 16

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

Алгоритм решения:

1. Побитовый сдвиг влево единичной маски на нужное количество бит-1
2. Поразрядное ИЛИ числа и маски

Код:

```
24
25 void task1_b() {
26     unsigned int x;
27     cout<<"Enter x: "<<endl;
28     cin>>x;
29     unsigned char maska=1;
30     x=x|((maska<<6));
31     cout<<x<<endl;
32 }
33
```

Демонстрация работы программы:

```
Enter x:
0
64
```

```
Enter x:
123
123
```

Тесты:

Входные данные (x)	Ожидаемый результат	Фактический результат
0	64	64
123	123	123

Вывод по заданию:

Реализовано увеличение числа битовыми операциями

### 2.3 Задание 1в

Реализуйте код листинга 1, объясните выводимый программой результат.

Алгоритм решения:

1. Установить значение маски  $2^{31}$
2. Вывод битового И x и маски для каждого значения маски ( $2^{31} - 2^{30} - 2^{29} - \dots - 2^0$ )

Код:

```
34 void task1_c() {
35
36     unsigned int x=25;
37     const int n=sizeof(int)*8;
38     unsigned maska=(1<<n-1);
39     cout<<"The initial maska: "<<bitset<n>(maska)<<endl;
40     cout<<"Result: "<<endl;
41     for(int i=1;i<=n;i++) {
42         cout<<((x&maska)>>(n-i));
43         maska=maska>>1;
44     }
45     cout<<endl;
46 }
47
```

Демонстрация работы программы:

```
The initial maska: 10000000000000000000000000000000
Result:
0000000000000000000000000000000011001
```

Тесты:

Входные данные (x)	Ожидаемый результат	Фактический результат
Без входных х данных	0000000000000000000000000001 1001	0000000000000000000000000001 1001

Вывод по заданию:

Реализовано создание битового массива с помощью класса bitset

## 2.4 Задание 2а

Реализуйте пример 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.

Алгоритм решения:

1. Создание битового массива на 8 бит с помощью unsigned char
2. Ввод чисел
3. Вывод введенных чисел
4. Битовое ИЛИ массива со сдвигом влево на  $i$ , где  $i$  изменяется от нуля до длины сортируемого массива
5. Цикл  $i$  от нуля до длины массива, если установлен бит, вывод  $i$

Код:

```
49 void task2_a() {
50     vector<int> numbers = {};
51     unsigned char bit_array = 0;
52
53     for (int i = 0; i < 8; i++) {
54         int number_to_push = 0;
55         cout << "Enter number between 0 and 7: " << endl;
56         cin >> number_to_push;
57         if (number_to_push < 0 || number_to_push > 7) {
58             cout << "Invalid number! Must be between 0 and 7." << endl;
59             i--;
60         }
61         else {
62             numbers.push_back(number_to_push);
63             bit_array |= (1 << number_to_push);
64         }
65     }
66     cout << "Input numbers: ";
67     for (int i = 0; i < numbers.size(); i++) {
68         cout << numbers[i] << " ";
69     }
70     cout << endl;
71     cout << "Sorted numbers: ";
72     for (int i = 0; i < 8; i++) {
73         if (bit_array & (1 << i)) {
74             cout << i << " ";
75         }
76     }
77     cout << endl;
78 }
```



Демонстрация работы программы:

```
Enter number between 0 and 7:
1
Enter number between 0 and 7:
0
Enter number between 0 and 7:
5
Enter number between 0 and 7:
7
Enter number between 0 and 7:
2
Enter number between 0 and 7:
4
Enter number between 0 and 7:
3
Enter number between 0 and 7:
4
Input numbers: 1 0 5 7 2 4 3 4
Sorted numbers: 0 1 2 3 4 5 7
```

Тесты:

Входные данные (x)	Ожидаемый результат	Фактический результат
1 0 5 7 2 4 3 4	0 1 2 3 4 5 6 7	0 1 2 3 4 5 6 7

Вывод по заданию:

Реализована битовая сортировка массива с помощью битового массива, созданного из переменной unsigned char

## 2.5 Задание 2б

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.

Алгоритм решения:

1. Создание битового массива на 64 бит с помощью unsigned long long
2. Генерация уникальных чисел от 0 до 63 включительно
3. Вывод сгенерированных чисел

4. Битовое ИЛИ массива со сдвигом влево на  $i$ , где  $i$  изменяется от нуля до длины сортируемого массива
5. Цикл  $i$  от нуля до длины массива, если установлен бит, вывод  $i$

Код:

```

80 void task2_b() {
81     vector<int> numbers;
82     unsigned long long bit_array = 0;
83     srand(static_cast<unsigned int>(time(0)));
84     while (numbers.size() < 64) {
85         int number_to_push = rand() % 64;
86         if (!(bit_array & (1ULL << number_to_push))) {
87             numbers.push_back(number_to_push);
88             bit_array |= (1ULL << number_to_push);
89         }
90     }
91     cout << "Input numbers: ";
92     for (int num : numbers) {
93         cout << num << " ";
94     }
95     cout << endl;
96     cout << "Sorted numbers: ";
97     for (int i = 0; i < 64; i++) {
98         if (bit_array & (1ULL << i)) {
99             cout << i << " ";
100         }
101     }
102     cout << endl;
103 }

```

Демонстрация работы программы:

```

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

```

Тесты:

Входные данные (x)	Ожидаемый результат	Фактический результат
52 8 57 53 8 62 58 14 40	0 2 3 6 7 8 9 10 11 12 13 14	0 2 3 6 7 8 9 10 11 12 13 14
19 6 35 17 12 35 63 61	15 16 17 18 19 20 21 22 23	15 16 17 18 19 20 21 22 23
63 61 61 10 46 62 62 61	24 25 26 27 29 30 31 32 34	24 25 26 27 29 30 31 32 34
7 31 51 51 19 90 26 7 58	35 38 39 40 41 42 43 44 45	35 38 39 40 41 42 43 44 45
40 50 56 61 32 41 51 2	46 47 48 49 50 51 52 53 54	46 47 48 49 50 51 52 53 54
23 54 13 59 43 43 25 48	55 56 57 58 59 61 62 63	55 56 57 58 59 61 62 63
47 47 30 25 27 41 11 10		
19 45 51 2 20		

Вывод по заданию:

Реализована битовая сортировка массива с помощью битового массива, созданного из переменной unsigned long long.

## 2.6 Задание 2в

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.

Алгоритм решения:

1. Создание битовых массивов на 8 бит (8 штук) с помощью
2. Генерация уникальных чисел от 0 до 63 включительно
3. Установка числа в определенный массив
4. Битовое ИЛИ массива со сдвигом влево на  $i$ , где  $i$  изменяется от нуля до длины сортируемого массива
5. Цикл  $i$  от нуля до длины массива, если установлен бит, вывод  $i$

Код:

```

105 void task2_c() {
106     vector<int> numbers;
107     unsigned char bit_array[8] = {0};
108     srand(static_cast<unsigned int>(time(0)));
109     for (int i = 0; i < 64; ) {
110         int number_to_push = rand() % 64;
111         int byte_index = number_to_push / 8;
112         int bit_index = number_to_push % 8;
113         if (!(bit_array[byte_index] & (1 << bit_index))) {
114             numbers.push_back(number_to_push);
115             bit_array[byte_index] |= (1 << bit_index);
116             i++;
117         }
118     }
119     cout << "Input numbers: ";
120     for (int num : numbers) {
121         cout << num << " ";
122     }
123     cout << endl;
124     cout << "Sorted numbers: ";
125     for (int i = 0; i < 64; i++) {
126         int byte_index = i / 8;
127         int bit_index = i % 8;
128         if (bit_array[byte_index] & (1 << bit_index)) {
129             cout << i << " ";
130         }
131     }
132     cout << endl;
133 }

```

Демонстрация работы программы:

```

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

```

Тесты:

Входные данные (x)	Ожидаемый результат	Фактический результат
52 8 57 53 8 62 58 14 40	0 2 3 6 7 8 9 10 11 12 13	0 2 3 6 7 8 9 10 11 12 13 14
19 6 35 17 12 35 63 61	14 15 16 17 18 19 20 21 22	15 16 17 18 19 20 21 22 23
63 61 61 10 46 62 62 61	23 24 25 26 27 29 30 31 32	24 25 26 27 29 30 31 32 34
7 31 51 51 19 9 0 26 7	34 35 38 39 40 41 42 43 44	35 38 39 40 41 42 43 44 45
58 40 50 56 61 32 41 51	45 46 47 48 49 50 51 52 53	46 47 48 49 50 51 52 53 54
2 23 54 13 59 43 43 25	54 55 56 57 58 59 61 62 63	55 56 57 58 59 61 62 63
48 47 47 30 25 27 41 11		
10 19 45 51 2 20		

## 2.7 Задание 3 (а и б)

Реализуйте задачу сортировки числового файла с заданными условиями. Добавьте в код возможность определения времени работы программы. Определите программно объём оперативной памяти, занимаемый битовым Массивом

Алгоритм решения:

1. Генерация входного файла чисел.
2. Замер времени
2. Создание битового массива с нулевыми исходными значениями.
3. Считывание целых чисел из файла и установка в 1 соответствующих бит массива.
4. Формирование упорядоченного выходного файла путём последовательной проверки бит массива и вывода в файл номеров (индексов) тех бит, которые установлены в 1
5. Остановка времени

Код:

```
135 void task3() {
136     ofstream file_for_input("input.txt");
137     int max_num = 10000000;
138     vector<int> numbers(max_num);
139     for (int i = 0; i < max_num; ++i) {
140         numbers[i] = i;
141     }
142     random_device rd;
143     mt19937 g(rd);
144     shuffle(numbers.begin(), numbers.end(), g);
145     for (const int& num : numbers) {
146         file_for_input << num << endl;
147     }
148     file_for_input.close();
149     cout << "File has been created" << endl;
150     auto start_time = high_resolution_clock::now();
151     ifstream input_file("input.txt");
152     ofstream output_file("output.txt");
153     vector<int> bit_array(max_num, 0);
154     int num;
155     while (input_file >> num) {
156         if (num >= 0 && num < max_num) {
157             bit_array[num] = 1;
158         }
159     }
160     input_file.close();
161     for (int i = 0; i < max_num; ++i) {
162         if (bit_array[i]) {
163             output_file << i << endl;
164         }
165     }
166     output_file.close();
167     auto end_time = high_resolution_clock::now();
168     duration<double> duration = end_time - start_time;
169     cout << "File has been created! Data has been sorted! Time: " << duration.count() << endl;
170     size_t memory_usage = bit_array.size() / 8;
171     cout << "Memory used by bit array: " << memory_usage << " bytes" << endl;
172 }
```

Демонстрация работы программы/тест:

```
File has been created
```

```
File has been created! Data has been sorted! Time: 18.185
```

```
Memory used by bit array: 1250000 bytes
```

## 3 ХОД РАБОТЫ (5.2)

### 3.1 Задание 1.1

Создать двоичный файл из записей

Алгоритм решения:

1. Создание txt файла, в котором будут храниться номера телефонов
2. Ввод количества номеров телефонов
3. Генерация номеров телефона
4. Исключение повторяющихся номеров телефона
5. Сортировка номеров телефона
6. Занесение номеров телефона в txt файл
7. Создание bin файла
8. Запись в bin файл данных из txt файла

Код 1 (создание txt файла):

```
40 void create_file(){
41     vector<string> used_phonenumbers;
42     srand(static_cast<unsigned int>(time(0)));
43     ofstream file("common_file.txt");
44     int num_of_strings=0;
45     cout<<"Enter number of phone numbers: "<<endl;
46     cin>>num_of_strings;
47     global_num_of_phones = num_of_strings;
48     cout<<endl;
49     for (int i = 0; i < num_of_strings; i++) {
50         int randomNumber1 = 10 + rand() % (99 - 10 + 1);
51         int randomNumber2 = 100 + rand() % (999 - 100 + 1);
52         int randomNumber3 = 1000 + rand() % (9000 - 10 + 1);
53         string phonenumbers="89"+to_string(randomNumber1)+to_string(randomNumber2)+to_string(randomNumber3);
54         if (find(first:used_phonenumbers.begin(), last:used_phonenumbers.end(), phonenumbers) == used_phonenumbers.end()) {
55             used_phonenumbers.push_back(phonenumbers);
56         }
57         else {
58             cout<<phonenumbers<<" has been used. DETERMINATE THIS!!!"<<endl;
59             i--;
60         }
61     }
62     sort(first:used_phonenumbers.begin(), last:used_phonenumbers.end());
63     for (int i = 0; i < used_phonenumbers.size(); i++) {
64         file<<used_phonenumbers[i]<<" "<<endl;
65     }
66     file.close();
67 }
```

## Код 2 (создание bin файла)

```
14 void mega_converter(const string &text_file, const string &binary_file) {
15     ifstream text_file(text_file);
16     ofstream binary_file(binary_file);
17
18     if (text_file.is_open() && binary_file.is_open()) {
19         string line;
20         while (getline([&text_file, [&]line)) {
21             size_t len = line.length();
22             for (int i = 0; i < len; i++) {
23                 if (line[i] == ' ') {
24                     binary_file << endl;
25                 }
26                 else {
27                     binary_file << line[i];
28                 }
29             }
30         }
31         text_file.close();
32         binary_file.close();
33         cout<<"Converted file successfully!"<<endl;
34     }
35     else {
36         cout<<"File could not be opened!"<<endl;
37     }
38 };
```



# Тестирование (bin файл со 100 записями)

Offset	+0	+1	+2	+3	+4	+5	+6	+7	+8	+9	+a	+b	+c	+d	+e	+f	Equivalent ASCII characters
00000000	38	39	31	31	31	33	32	33	34	32	33	0d	0e	38	39	31	8 9 1 1 1 3 2 3 4 2 3 8 9 1
00000010	31	33	31	31	34	34	32	39	0d	0e	38	39	31	31	38	30	1 3 1 1 4 4 2 9 8 9 1 1 8 0
00000020	30	37	32	33	39	0d	0e	38	39	31	33	35	30	35	38	32	0 7 2 3 9 8 9 1 3 5 0 5 8 2
00000030	38	34	0d	0e	38	39	31	33	37	38	33	36	32	35	33	0d	8 4 8 9 1 3 7 8 3 6 2 5 3
00000040	0e	38	39	31	33	38	34	32	37	31	31	32	0d	0e	38	39	8 9 1 3 8 4 2 7 1 1 2 8 9
00000050	31	33	38	39	32	38	32	38	35	0d	0e	38	39	31	34	38	1 3 8 9 2 8 2 8 5 8 9 1 4 8
00000060	38	31	38	30	34	39	0d	0e	38	39	31	35	38	32	39	31	8 1 8 0 4 9 8 9 1 5 8 2 9 1
00000070	38	33	34	0d	0e	38	39	31	35	39	38	38	35	38	32	30	8 3 4 8 9 1 5 9 8 8 5 8 2 0
00000080	0d	0e	38	39	31	38	36	36	34	37	34	33	37	0d	0e	38	8 9 1 8 6 6 4 7 4 3 7 8
00000090	39	32	30	37	36	38	33	32	35	38	0d	0e	38	39	32	30	9 2 0 7 6 8 3 2 5 8 8 9 2 0
000000a0	38	34	30	35	34	30	34	0d	0e	38	39	32	32	31	32	32	8 4 0 5 4 0 4 8 9 2 2 1 2 2
000000b0	39	32	31	37	0d	0e	38	39	32	32	38	33	32	37	37	35	9 2 1 7 8 9 2 2 8 3 2 7 7 5
000000c0	36	0d	0e	38	39	32	34	34	36	33	39	38	31	38	0d	0e	6 8 9 2 4 4 6 3 9 8 1 8
000000d0	38	39	32	35	34	37	34	36	37	33	30	0d	0e	38	39	32	8 9 2 5 4 7 4 6 7 3 0 8 9 2
000000e0	36	38	34	30	33	35	32	31	0d	0e	38	39	32	36	39	34	6 8 4 0 3 5 2 1 8 9 2 6 9 4
000000f0	33	36	37	33	38	0d	0e	38	39	32	37	37	31	36	37	38	3 6 7 3 8 8 9 2 7 7 1 6 7 8
00000100	39	33	0d	0e	38	39	32	38	32	35	36	38	30	32	33	0d	9 3 8 9 2 8 2 5 6 8 0 2 3
00000110	0e	38	39	32	38	34	39	36	31	32	30	32	0d	0e	38	39	8 9 2 8 4 9 6 1 2 0 2 8 9
00000120	32	38	36	36	37	31	37	30	30	0d	0e	38	39	33	31	33	2 8 6 6 7 1 7 0 0 8 9 3 1 3
00000130	34	31	39	30	35	37	0d	0e	38	39	33	31	39	36	37	32	4 1 9 0 5 7 8 9 3 1 9 6 7 2
00000140	31	39	32	0d	0e	38	39	33	31	39	38	30	37	37	39	39	1 9 2 8 9 3 1 9 8 0 7 7 9 9
00000150	0d	0e	38	39	33	33	32	38	39	34	38	31	36	0d	0e	38	8 9 3 3 2 8 9 4 8 1 6 8
00000160	39	33	33	33	32	34	38	34	39	38	0d	0e	38	39	33	33	9 3 3 3 2 4 8 4 9 8 8 9 3 3
00000170	35	32	39	38	37	37	38	0d	0e	38	39	33	33	37	37	30	5 2 9 8 7 7 8 8 9 3 3 7 7 0
00000180	34	32	31	34	0d	0e	38	39	33	34	32	35	36	39	30	30	4 2 1 4 8 9 3 4 2 5 6 9 0 0
00000190	33	0d	0e	38	39	33	34	38	34	34	37	30	36	35	0d	0e	3 8 9 3 4 8 4 4 7 0 6 5
000001a0	38	39	33	35	33	31	33	32	34	30	34	0d	0e	38	39	33	8 9 3 5 3 1 3 2 4 0 4 8 9 3
000001b0	37	34	31	35	39	37	34	34	0d	0e	38	39	33	38	36	32	7 4 1 5 9 7 4 4 8 9 3 8 6 2
000001c0	34	33	30	38	38	0d	0e	38	39	33	39	32	35	35	35	34	4 3 0 6 8 8 9 3 9 2 5 5 5 4
000001d0	33	33	0d	0e	38	39	33	39	32	38	30	33	34	36	35	0d	3 3 8 9 3 9 2 8 0 3 4 6 5
000001e0	0e	38	39	33	39	38	36	36	32	37	34	33	0d	0e	38	39	8 9 3 9 8 6 6 2 7 4 3 8 9
000001f0	34	30	35	37	39	36	34	37	34	0d	0e	38	39	34	30	37	4 0 5 7 9 6 4 7 4 8 9 4 0 7
00000200	35	39	39	38	32	30	0d	0e	38	39	34	31	37	34	35	31	5 9 9 8 2 0 8 9 4 1 7 4 5 1
00000210	32	31	32	0d	0e	38	39	34	32	39	30	33	34	35	31	33	2 1 2 8 9 4 2 9 0 3 4 5 1 3
00000220	0d	0e	38	39	34	34	34	36	35	35	36	31	33	0d	0e	38	8 9 4 4 4 6 5 5 6 1 3 8
00000230	39	34	37	36	39	34	31	30	37	38	0d	0e	38	39	34	38	9 4 7 6 9 4 1 0 7 8 8 9 4 8
00000240	35	39	33	32	35	38	39	0d	0e	38	39	34	39	38	31	35	5 9 3 2 5 8 9 8 9 4 9 8 1 5
00000250	34	35	38	36	0d	0e	38	39	35	30	33	30	37	39	36	36	4 5 8 6 8 9 5 0 3 0 7 9 6 6
00000260	30	0d	0e	38	39	35	30	34	39	38	33	35	30	30	0d	0e	0 8 9 5 0 4 9 8 3 5 0 0
00000270	38	39	35	31	38	34	34	37	36	37	37	0d	0e	38	39	35	8 9 5 1 8 4 4 7 6 7 7 8 9 5
00000280	33	31	31	32	34	34	30	0d	0e	38	39	35	33	33	39	39	3 1 1 2 4 4 0 8 8 9 5 3 3 9
00000290	35	33	36	32	32	0d	0e	38	39	35	33	38	33	30	37	36	5 3 6 2 2 8 9 5 3 8 3 0 7 6
000002a0	37	30	0d	0e	38	39	35	34	38	39	36	36	30	33	38	0d	7 0 8 9 5 4 8 9 6 6 0 3 8
000002b0	0e	38	39	35	35	35	32	39	38	31	37	34	0d	0e	38	39	8 9 5 5 5 2 9 8 1 7 4 8 9
000002c0	35	37	31	34	34	35	36	30	30	0d	0e	38	39	35	37	38	5 7 1 4 4 5 6 0 0 8 9 5 7 8
000002d0	33	32	38	37	35	0d	0e	38	39	35	38	38	37	33	34	34	3 2 2 8 7 5 8 9 5 8 8 7 3 4
000002e0	38	30	34	0d	0e	38	39	35	39	36	30	39	32	32	33	35	8 0 4 8 9 5 9 6 0 9 2 2 3 5
000002f0	0d	0e	38	39	36	30	35	31	33	32	34	32	32	0d	0e	38	8 9 6 0 5 1 3 2 4 2 2 8
00000300	39	36	30	37	38	33	39	38	31	30	0d	0e	38	39	36	30	9 6 0 7 8 3 9 8 1 0 8 9 6 0
00000310	38	30	38	32	33	36	35	0d	0e	38	39	36	30	38	32	33	8 0 8 2 3 6 5 8 9 6 0 8 2 3
00000320	36	37	32	37	0d	0e	38	39	36	30	38	39	39	34	36	35	6 7 2 7 8 9 6 0 8 9 9 4 6 5
00000330	32	0d	0e	38	39	36	32	32	39	36	36	33	34	37	0d	0e	2 8 9 6 2 2 9 6 6 3 4 7
00000340	38	39	36	32	33	38	36	32	39	37	38	0d	0e	38	39	36	8 9 6 2 3 8 6 2 9 7 8 8 9 6
00000350	34	34	35	30	36	30	31	33	0d	0e	38	39	36	34	36	36	4 4 5 0 6 0 1 3 8 9 6 4 6 6
00000360	35	35	36	30	34	0d	0e	38	39	36	34	39	37	33	31	34	5 5 6 0 4 8 9 6 4 9 7 3 1 4
00000370	33	31	0d	0e	38	39	36	35	31	34	32	39	39	34	37	0d	3 1 8 9 6 5 1 4 2 9 9 4 7
00000380	0e	38	39	36	37	32	36	38	39	34	30	35	0d	0e	38	39	8 9 6 7 2 6 8 9 4 0 5 8 9
00000390	36	39	37	30	33	31	35	37	31	0d	0e	38	39	37	30	36	6 9 7 0 3 1 5 7 1 8 9 7 0 6
000003a0	34	38	32	36	33	35	0d	0e	38	39	37	30	36	36	38	34	4 8 2 6 3 5 8 9 7 0 6 6 8 4
000003b0	37	37	32	0d	0e	38	39	37	33	31	35	32	31	36	30	36	7 7 2 8 9 7 3 1 5 2 1 6 0 6
000003c0	0d	0e	38	39	37	33	33	36	35	33	33	31	33	0d	0e	38	8 9 7 3 3 6 5 3 3 1 3 8
000003d0	39	37	34	35	38	32	37	36	32	33	0d	0e	38	39	37	35	9 7 4 5 8 2 7 6 2 3 8 9 7 5
000003e0	37	32	31	33	38	39	0d	0e	38	39	37	35	38	37	39	39	7 2 2 1 3 8 9 8 9 7 5 8 7 9
000003f0	36	30	32	37	0d	0e	38	39	37	37	32	37	33	39	33	31	6 0 2 7 8 9 7 7 2 7 3 9 3 1
00000400	32	0d	0e	38	39	37	38	38	35	31	34	31	36	37	0d	0e	2 8 9 7 8 8 5 1 4 1 6 7
00000410	38	39	38	31	39	38	39	32	39	32	34	0d	0e	38	39	38	8 9 8 1 9 8 9 2 9 2 4 8 9 8
00000420	35	32	35	36	33	37	39	37	0d	0e	38	39	38	37	39	39	5 2 5 6 3 7 9 7 8 9 8 7 9 9
00000430	35	38	33	33	30	0d	0e	38	39	38	38	31	33	33	32	36	5 8 3 3 0 8 9 8 8 1 3 3 2 6
00000440	36	33	0d	0e	38	39	38	39	32	37	37	35	37	34	30	0d	6 3 8 9 8 9 2 7 7 5 7 4 0
00000450	0e	38	39	38	39	34	34	31	32	37	30	32	0d	0e	38	39	8 9 8 9 4 4 1 2 7 0 2 8 9
00000460	38	39	36	33	35	37	31	34	36	0d	0e	38	39	38	39	36	8 9 6 3 5 7 1 4 6 8 9 8 9 6
00000470	34	39	39	38	31	34	0d	0e	38	39	38	39	36	36	33	38	4 9 9 8 1 4 8 9 8 9 6 6 3 8
00000480	32	37	33	0d	0e	38	39	39	30	36	37	34	32	32	35	37	2 7 3 8 9 9 0 6 7 4 2 2 5 7
00000490	0d	0e	38	39	39	31	32	38	36	38	36	32	38	0d	0e	38	8 9 9 1 2 8 6 8 6 2 8 8
000004a0	39	39	31	32	39	36	32	31	32	37	0d	0e	38	39	39	32	9 9 1 2 9 6 2 1 2 7 8 9 9 2
000004b0	35	31	37	37	37	34	33	0d	0e	38	39	39	32</				

### 3.2 Задание 1.2

Разработать программу поиска записи по ключу в бинарном файле с применением алгоритма линейного поиска.

Алгоритм решения:

1. Ввод ключа
2. Запуск таймера
3. Чтение bin файла
4. Проверка каждой строки по порядку
5. Вывод номера телефона (если он найден, если нет, вывод "Phone number not found")
6. Остановка таймера

Код:

```
int lin_file_search() {
    string key;
    cout<<"Linear search"<<endl;
    cout << "Enter phone number (key): " << endl;
    cin >> key;
    auto start_time = high_resolution_clock::now();
    ifstream file(s:"common_file.bin");
    for (int i = 0; i < global_num_of_phones; i++) {
        string line;
        getline([&]file, [&]line);
        if (line == key) {
            cout << "Phone number found!" << endl;
            cout << line << endl;
            auto end_time =high_resolution_clock::now();
            duration<double> elapsed = end_time - start_time;
            cout << "Sorting completed in: " << elapsed.count() << " seconds." << endl;
            return 0;
        }
    }
    cout << "Phone number not found!" << endl;
    auto end_time =high_resolution_clock::now();
    duration<double> elapsed = end_time - start_time;
    cout << "Sorting completed in: " << elapsed.count() << " seconds." << endl;
    return 0;
}
```

Кол-во данных	Лучший случай	Средний случай	Худший случай
100	0.0007849 с	0.0008606 с	0.0009228 с
1000	0.0005112 с	0.0012213 с	0.0018258 с
10 000	0.001096 с	0.001949 с	0.0021382 с

### 3.3 Задание 1.3

Разработать программу поиска записи по ключу в бинарном файле с применением алгоритма линейного поиска.

Алгоритм решения:

1. Ввод ключа
2. Запуск таймера
3. Чтение bin файла
4. Бинарный поиск с таблицей смещений
  - а. Загружается таблица смещений, чтобы быстро обращаться к нужным строкам в файле
  - б. Пока диапазон поиска не сократится до нуля
    - i. Рассчитывается середина
    - ii. Из таблицы смещений извлекается позиция записи
    - iii. Программа переходит по смещению в бинарном файле и читает строку
    - iv. Сравнение: если строка совпадает с ключом — телефон найден
    - v. Если строка меньше ключа — продолжаем поиск в правой половине, иначе — в левой
5. Остановка таймера

Код:

```
90 int bin_file_search() {
91     string key;
92     cout << "Binary search using offset table" << endl;
93     cout << "Enter phone number (key): " << endl;
94     cin >> key;
95     auto start_time = high_resolution_clock::now();
96     ifstream file(s:"common_file.bin", mode:ios::binary);
97     if (!file.is_open()) {
98         cerr << "Error opening binary file!" << endl;
99         return -1;
100     }
101     int left = 0;
102     int right = global_num_of_phones - 1;
103     const int record_size = 20;
104     char buffer[record_size + 1];
105     while (left <= right) {
106         int mid = left + (right - left) / 2;
107         streampos pos = offset_table[mid];
108         file.seekg(pos);
109         file.read(buffer, record_size);
110         buffer[record_size] = '\0';
111         string current_number(buffer);
112         current_number = current_number.substr(pos:0, n:current_number.find(c: '\0'));
113         if (current_number == key) {
114             cout << "Phone number found!" << endl;
115             cout << current_number << endl;
116             auto end_time = high_resolution_clock::now();
117             duration<double> elapsed = end_time - start_time;
118             cout << "Search completed in: " << elapsed.count() << " seconds." << endl;
119             return 0;
120         } else if (current_number < key) {
121             left = mid + 1;
122         } else {
123             right = mid - 1;
124         }
125     }
```

## Тестирование

Кол-во данных	Лучший случай	Средний случай	Худший случай
100	0.0002576 с	0.0011368 с	0.0012054 с
1000	0.0006375 с	0.0011223 с	0.0015271 с
10 000	0.0008211 с	0.0013974 с	0.0015912 с

## **4 ВЫВОДЫ**

В результате выполнения практической работы была освоена технология сортировки массивов неповторяющихся чисел с помощью битового массива. Были изучены битовые операции.

## **5 ИСПОЛЬЗУЕМЫЙ ИСТОЧНИК**

1. М.Л. РЫСИН, М.В. САРТАКОВ, М.Б. ТУМАНОВА Учебно-методическое пособие СиАОД часть 2

