



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 № 1
«Поразрядные операции и их применение»
по дисциплине
«Структуры и алгоритмы обработки данных»

Выполнил студент группы *ИКБО-03-22*

Хохлинов Д.И.

Принял

Сорокин А.В.

Практическая
работа выполнена

«__» _____ 2023 г.

«Зачтено»

«__» _____ 2023 г.

Москва 2023

СОДЕРЖАНИЕ

1 ЗАДАНИЕ 1	3
1.1 Постановка задачи	3
1.2 Описание алгоритмов	4
1.3 Блок-схемы алгоритмов.....	5
1.4 Код программы.....	8
1.5 Тестирование	10
2 ЗАДАНИЕ 2	12
2.1 Постановка задачи	12
2.2 Описание структуры хранения битового массива в оперативной памяти	12
2.3 Описание алгоритмов	13
2.4 Блок-схемы алгоритмов.....	15
2.4 Код программы.....	17
2.5 Тестирование	20
3 ЗАКЛЮЧЕНИЕ	24
4 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	25

1 ЗАДАНИЕ 1

1.1 Постановка задачи

В соответствии с [2] постановка задачи такова:

Разработать программу, которая продемонстрирует выполнение упражнений варианта. Результаты выполнения упражнения выводить на монитор. Требования к упражнениям:

Упражнение 1 - Определить переменную целого типа, присвоить ей значение, используя константу в шестнадцатеричной системе счисления. Разработать оператор присваивания и его выражение, которое установит седьмой, пятый и тринадцатый биты исходного значения переменной в значение 1, используя соответствующую маску и поразрядную операцию.

Упражнение 2 - Определить переменную целого типа. Разработать оператор присваивания и его выражение, которое обнуляет четыре старших бита исходного значения переменной, используя соответствующую маску и поразрядную операцию. Значение в переменную вводится с клавиатуры.

Упражнение 3 - Определить переменную целого типа. Разработать оператор присваивания и выражение, которое умножает значение переменной на 512, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

Упражнение 4 - Определить переменную целого типа. Разработать оператор присваивания и выражение, которое делит значение переменной на 128, используя соответствующую поразрядную операцию. Изменяемое число вводится с клавиатуры.

Упражнение 5 - Определить переменную целого типа. Разработать оператор присваивания и выражение, которое устанавливает n-ый бит в 1, используя маску. Маска инициализируется единицей в старшем разряде. Изменяемое число вводится с клавиатуры.

1.2 Описание алгоритмов

В вышеперечисленных упражнениях будем использовать тип переменной `unsigned long` (32 бита на переменную). Для решения поставленной задачи будем использовать следующие алгоритмы:

Упражнение 1. Установим для x некоторое константное значение в 16-ичной СС, например, 89ABCDEF. Затем инициализируем переменную `mask` типа `unsigned long` выражением $(1 \ll 5) | (1 \ll 7) | (1 \ll 13)$. Оно представляет собой побитовую дизъюнкцию значений, полученных побитовым сдвигом числа 1 на 5, 7 и 13 разрядов влево. Для получения необходимого результата необходимо вычислить побитовую дизъюнкцию переменных x и `mask` и затем вывести ее на экран.

Упражнение 2. Введем число x с клавиатуры. Затем инициализируем переменную `mask`, установив все биты в 1 и выполнив побитовый сдвиг полученного значения на 4 разряда вправо. После этого необходимо вычислить побитовую конъюнкцию переменных x и `mask` и вывести ее на экран.

Например, пусть введено число $x = 2124975845$ (в двоичной СС число будет иметь вид $1111110101010001000111011100101_2$). Маска имеет вид $00001111111111111111111111111111_2$. После побитовой конъюнкции этих значений имеем результат $0000110101010001000111011100101_2 = 111709925$.

Упражнение 3. Введем число x с клавиатуры. Затем выполним побитовый сдвиг числа x на 9 разрядов влево (т.к. $2^9 = 512$) и выведем результат на экран.

Например, пусть введено число $17 = 10001_2$. Тогда результатом будет являться число $10001000000000_2 = 8704$.

Упражнение 4. Введем число x с клавиатуры. Затем выполним побитовый сдвиг числа x на 7 разрядов вправо (т.к. $2^7 = 128$) и выведем результат на экран.

Например, пусть введено число $548 = 1000100100_2$. Тогда результатом будет являться число $100_2 = 4$.

Упражнение 5. Введем число x с клавиатуры. Затем введем число n – номер разряда, который необходимо установить в 1. Число n должно находиться в диапазоне $[0; 31]$; если оно выходит за границы диапазона слева или справа, то принимаем его равным 0 или 31 соответственно. Инициализируем маску константой 80000000_{16} (что эквивалентно 32-разрядному двоичному числу, старший разряд которого равен 1, остальные разряды – 0). Затем выполним побитовый сдвиг маски вправо на $31 - n$ разрядов и вычислим побитовую дизъюнкцию переменных x и $mask$, после чего выведем ее на экран.

Например, пусть введено число $x = 4051 = 111111010011_2$ и число $n = 17$. Итоговая маска будет иметь вид $0000\ 0000\ 0000\ 0010\ 0000\ 0000\ 0000\ 0000_2$. Результатом побитовой дизъюнкции будет число $0000\ 0000\ 0000\ 0010\ 0000\ 1111\ 1101\ 0011_2 = 135123$.

1.3 Блок-схемы алгоритмов

Приведем блок-схемы описанных алгоритмов для упражнений 1-5 (рисунки 1-5):

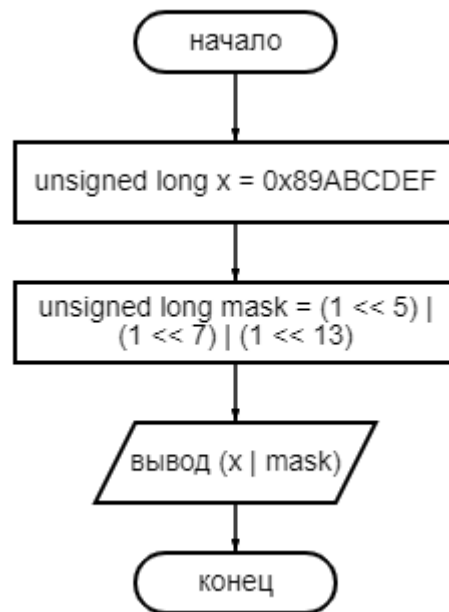


Рисунок 1 – Блок-схема алгоритма упражнения 1

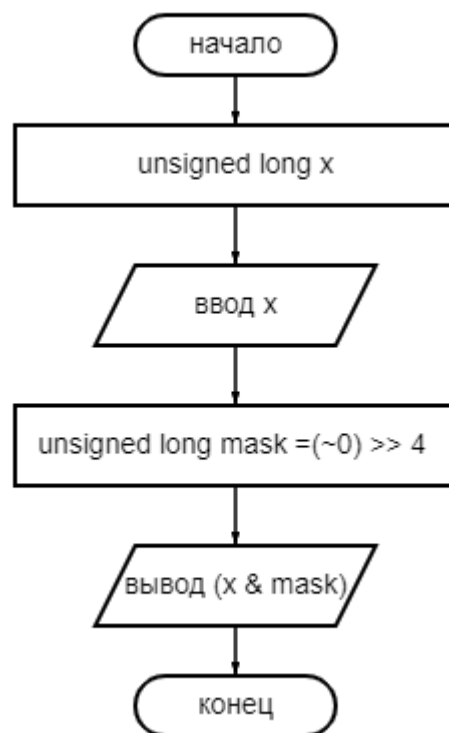


Рисунок 2 – Блок-схема алгоритма упражнения 2

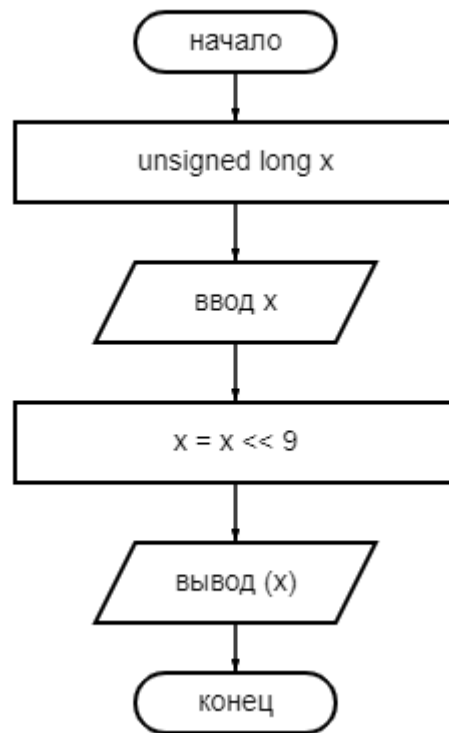


Рисунок 3 – Блок-схема алгоритма упражнения 3

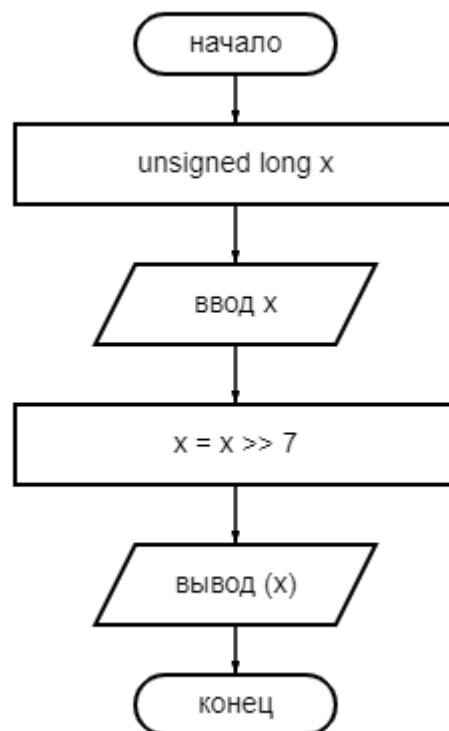


Рисунок 4 – Блок-схема алгоритма упражнения 4

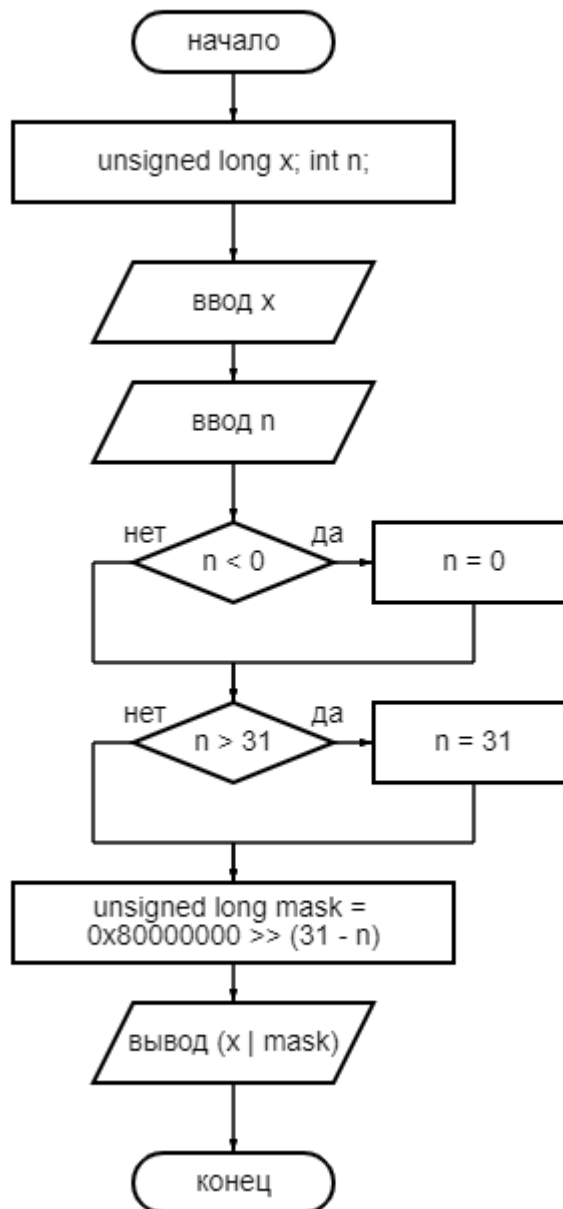


Рисунок 5 – Блок-схема алгоритма упражнения 5

1.4 Код программы

Реализуем алгоритмы, описанные выше, на языке C++. Также для удобства поместим код всех алгоритмов в одну функцию и добавим отображение значения переменной *x* в двоичной и десятичной СС до и после выполнения алгоритма (листинг 1):

Листинг 1. Код программы, используемой для решения задания 1

```
#include <iostream>
#include <string>
#include <bitset>
#include <locale.h>

using namespace std;

int main()
{
    setlocale(LC_ALL, "ru");
    cout << "Упражнение 1" << endl;
    unsigned long x = 0x89ABCDEF;
    unsigned long mask = 0;
    mask = (1 << 5) | (1 << 7) | (1 << 13);
    cout << "Значение x до применения маски: " << x << ", в двоичном  
виде: " << bitset<sizeof(x)*8>(x) << endl;
    cout << "Значение x после применения маски: " << (x | mask) << ",  
в двоичном виде: " << bitset<sizeof(x)*8>(x|mask) << endl;
    system("pause");
    system("cls");

    cout << "Упражнение 2" << endl;
    cout << "Введите x: ";
    cin >> x;
    mask = (~0);
    mask = mask >> 4;
    cout << "Значение x до применения маски: " << x << ", в двоичном  
виде: " << bitset<sizeof(x) * 8>(x) << endl;
    cout << "Значение x после применения маски: " << (x & mask) << ",  
в двоичном виде: " << bitset<sizeof(x) * 8>(x & mask) << endl;
    system("pause");
    system("cls");

    cout << "Упражнение 3" << endl;
    cout << "Введите x: ";
    cin >> x;
    cout << "Значение x до применения операции: " << x << ", в  
двоичном виде: " << bitset<sizeof(x) * 8>(x) << endl;
    x = x << 9;
    cout << "Значение x после применения операции: " << x << ", в  
двоичном виде: " << bitset<sizeof(x) * 8>(x) << endl;
    system("pause");
    system("cls");

    cout << "Упражнение 4" << endl;
    cout << "Введите x: ";
    cin >> x;
    cout << "Значение x до применения операции: " << x << ", в  
двоичном виде: " << bitset<sizeof(x) * 8>(x) << endl;
    x = x >> 7;
    cout << "Значение x после применения операции: " << x << ", в  
двоичном виде: " << bitset<sizeof(x) * 8>(x) << endl;
    system("pause");
    system("cls");
}
```

Продолжение листинга 1

```
cout << "Упражнение 5" << endl;
cout << "Введите x: ";
cin >> x;
cout << "Введите n (0-31): ";
int n;
cin >> n;
if (n < 0) n = 0;
if (n > 31) n = 31;
mask = 0x80000000;
mask = mask >> (31 - n);
cout << "Значение x до применения операции: " << x << ", в
двоичном виде: " << bitset<sizeof(x) * 8>(x) << endl;
cout << "Значение x после применения операции: " << (x | mask) <<
", в двоичном виде: " << bitset<sizeof(x) * 8>(x | mask) << endl;
}
```

1.5 Тестирование

Проведем тестирование кода, изложенного в листинге 1. Будем использовать тестовые данные, описанные в пункте 1.2 (рисунки 6-10):

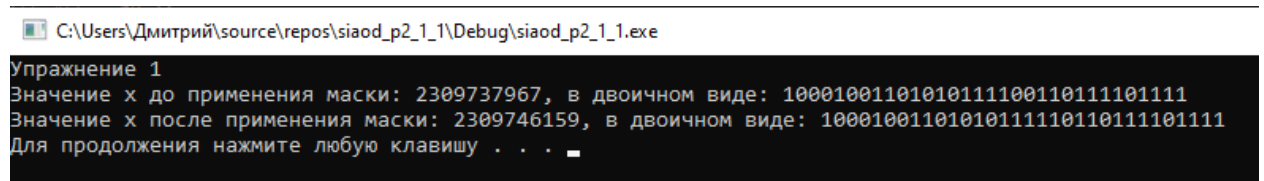


Рисунок 6 – Тестирование упражнения 1

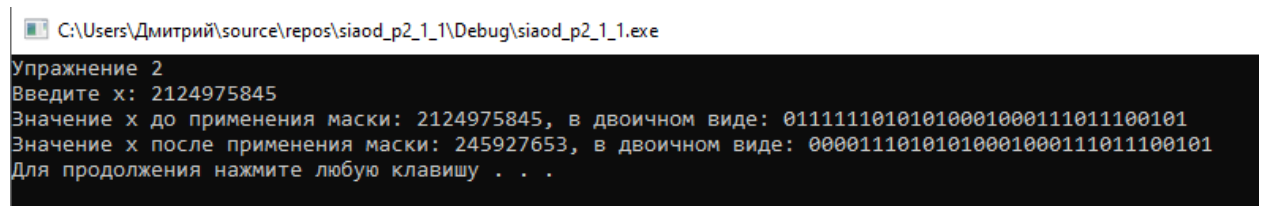


Рисунок 7 – Тестирование упражнения 2

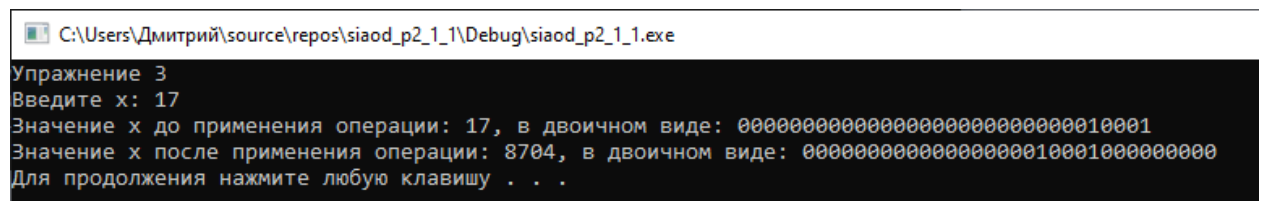


Рисунок 8 – Тестирование упражнения 3

2 ЗАДАНИЕ 2

2.1 Постановка задачи

В соответствии с [1]:

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.

2.2 Описание структуры хранения битового массива в оперативной памяти

Очевидно, что максимально допустимого объема ОЗУ не хватит для решения этой задачи (для представления битового массива длины 10^7 требуется $10^7 / 8 / 1024 / 1024 \approx 1,192$ МБ), поэтому возникают сомнения о правильности постановки задачи.

Для хранения массива битов в оперативной памяти используется массив типа `unsigned char` и длины $10\,000\,000 / 8 = 1\,250\,000$. Каждому целому числу от 0 до 9999999 соответствует свой бит в массиве, как показано на рисунке 11:

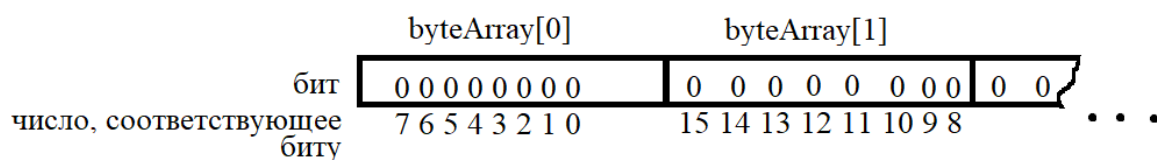


Рисунок 11 – Соответствие чисел битам в массиве

2.3 Описание алгоритмов

Для решения задания будем использовать следующий алгоритм.

Выделим память под битовый массив, реализованный с помощью массива типа `unsigned char` длиной 1 250 000, и обнулим его. Затем предложим пользователю способ ввода: ручной или автоматический – из файла.

В случае ручного ввода пользователь вводит неповторяющиеся числа в диапазоне $[0; 9\,999\,999]$, которые сразу же после ввода будем пытаться записать в битовый массив. Если бит, соответствующий очередному числу, равен 1, значит, данное число уже вводилось, и пользователь должен повторить ввод. Сигналом окончания ввода будет служить ввод числа -1.

В случае автоматического ввода пользователь вводит путь к файлу, в котором записаны целые числа в диапазоне $[0; 9\,999\,999]$, каждое – на своей строке. Эти числа будут считываться в массив, при этом проверка на уникальность числа не требуется, исходя из постановки задачи.

После сортировки каждый бит битового массива считывается: если он равен 1, то на экран выводится соответствующее число.

Дадим более подробное описание алгоритмов занесения числа в массив битов, сортировки массива, заданного файлом, и считывания данных из этого массива.

Для занесения числа в массив битов используем функцию `setNumber` с аргументами `n – int` и `data – указатель на массив типа unsigned char`:

Инициализируем переменную `int k = n`; создаем маску типа `unsigned char` выражением `1 << (8 - (k % 8) - 1)`. Затем, если побитовая конъюнкция элемента `data` с индексом `k / 8` и маски не равна 0, значит, вводимое число уже было записано в массив – тогда выполнение функции заканчивается и возвращается значение «ложь». В противном случае элементу `data` с

индексом $k / 8$ присваивается значение побитовой дизъюнкции элемента `data` с индексом $k / 8$ и маски и возвращается значение «истина».

Для вывода битового массива используем функцию `printNumbers` с аргументом `data` – указателем на массив типа `unsigned char`:

Инициализируем переменную `int n = 0`. Далее в цикле `for` перебираем значения переменной `i` от 0 до 1 250 000 (длина массива). Для каждого прохода цикла создаем объект класса `bitset<8> temp`, используя в качестве аргумента для конструктора `i`-ый элемент массива `data`. Затем объявляем цикл `for` для переменной `j` от 7 до 0 включительно; в теле этого цикла выводим переменную `n`, если `j`-ый бит объекта `temp` равен 1, и увеличиваем `n` на 1.

Для заполнения битового массива из файла используем функцию `insert` с аргументами `byte_massive` – указатель на массив типа `unsigned char` и `path` – строку, содержащую путь к файлу с числами:

Открываем файл с помощью объекта `fstream data_source`. Затем считываем каждую строку из файла `data_source` в переменную `temp` и вызываем функцию `setNumber` с аргументами `temp` и `byte_massive`. После достижения конца файла закрываем его.

2.4 Блок-схемы алгоритмов

Приведем блок-схемы для описанных алгоритмов (рисунки 12-14):

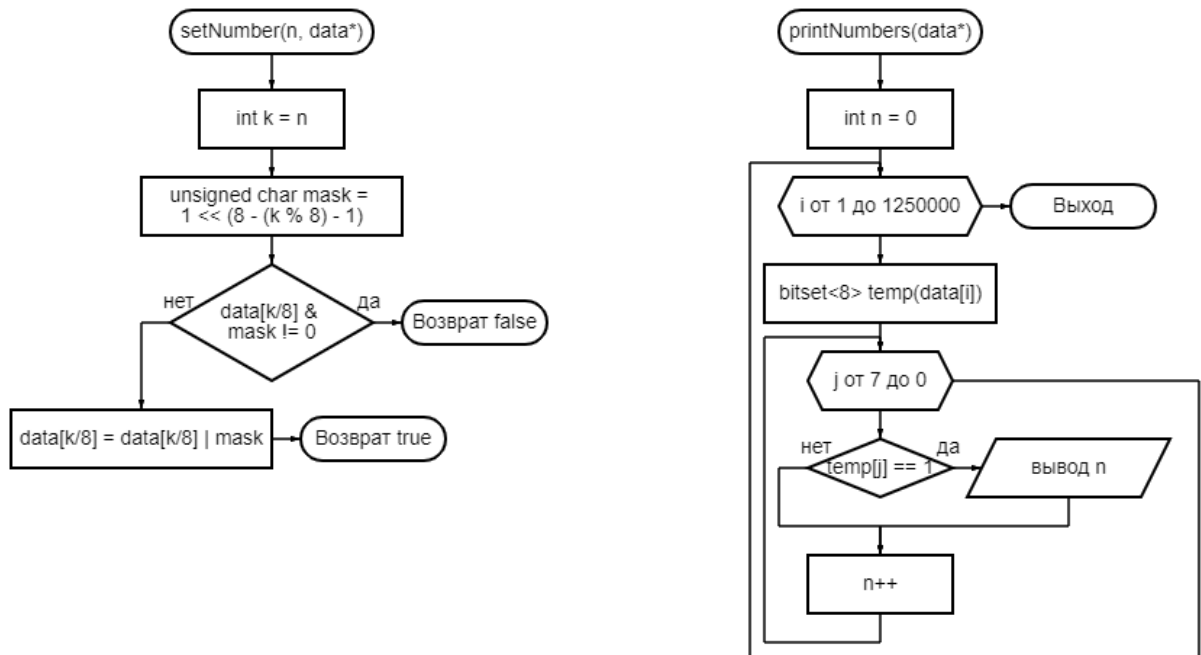


Рисунок 12 – Блок-схема алгоритмов функций `setNumber` и `printNumbers`

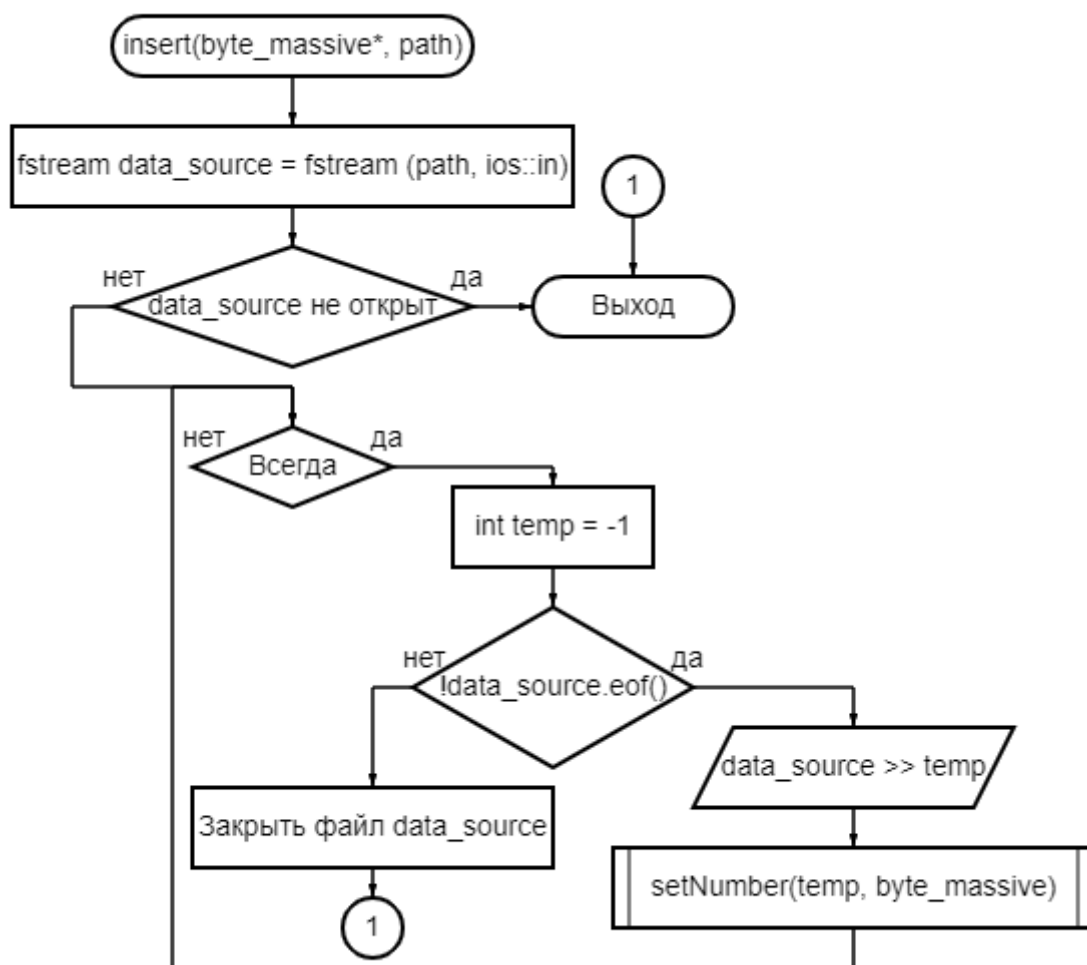


Рисунок 13 – Блок-схема алгоритма функции insert

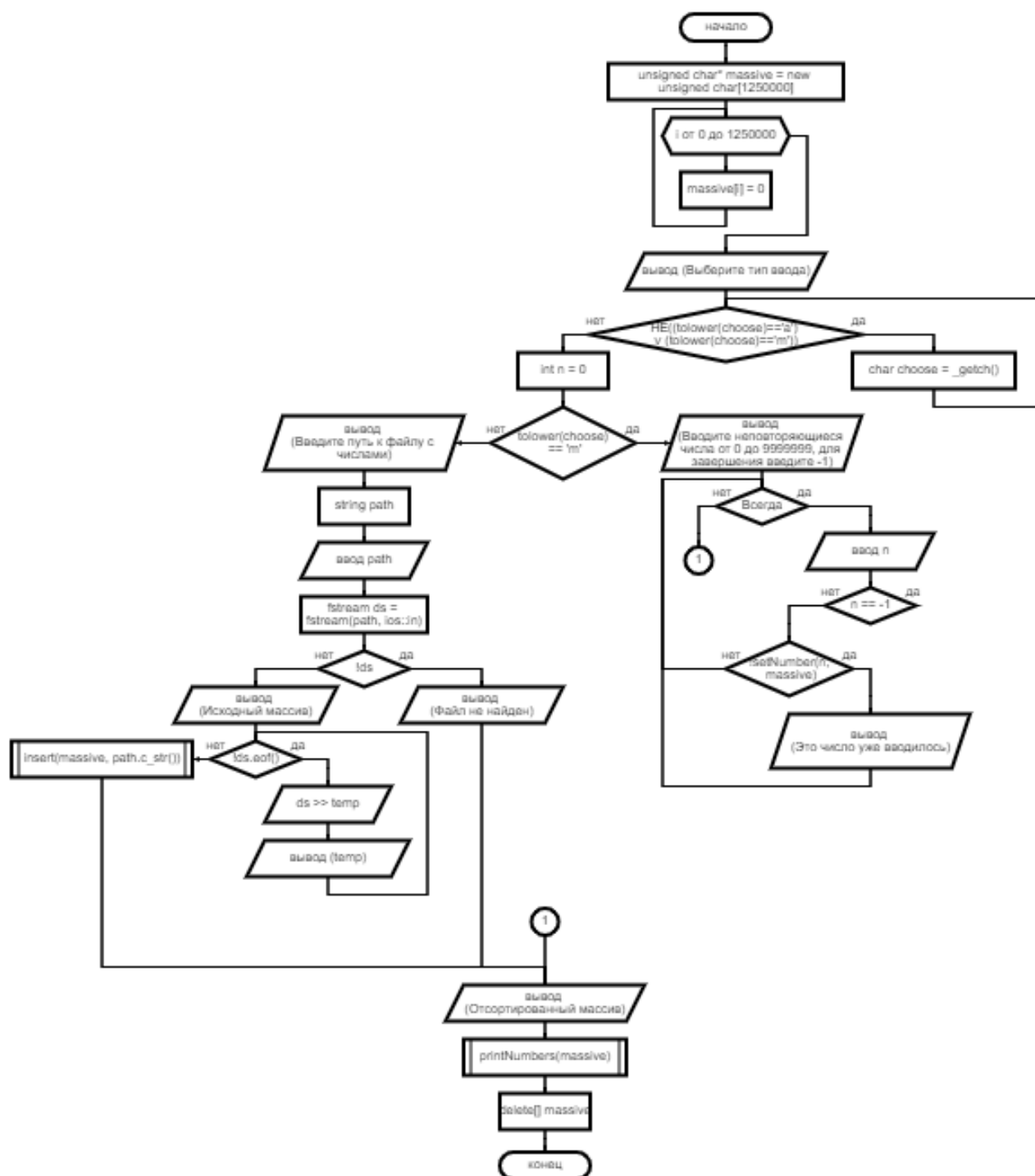


Рисунок 14 – Блок-схема алгоритма основной функции

2.4 Код программы

Реализуем алгоритмы, описанные выше, на языке C++. Также добавим счетчик времени, затраченного на сортировку, в функции insert (листинг 2):

Листинг 2. Код программы, используемой для решения задания 2

```
#include <iostream>
#include <locale.h>
#include <conio.h>
#include <ctime>
#include <cmath>
#include <bitset>
#include <fstream>
#include <chrono>

using namespace std;

const int massive_length = 10000000 / 8; //делим на 8 - количество
чисел в одном элементе char,
//чтобы получить необходимую длину массива для сортировки целых чисел
от 0 до 10^7

bool setNumber(int n, unsigned char* data)
{
    int k = n;
    unsigned char mask = 1 << (8 - (k % 8) - 1);
    if ((data[k / 8] & mask) != 0)
    {
        return false;
    }
    data[k / 8] = data[k / 8] | mask;
    return true;
}

void printNumbers(unsigned char* data)
{
    int n = 0;
    for (int i = 0; i < massive_length; i++)
    {
        bitset<8> temp(data[i]);
        for (int j = 7; j >= 0; j--)
        {
            if (temp[j] == 1)
            {
                cout << n << " ";
            }
            n++;
        }
    }
}

void insert(unsigned char* byte_massive, const char* path)
{
    fstream data_source = fstream(path, ios::in);
    if (!data_source) return;
    auto start = chrono::steady_clock::now();
    while(1)
    {
        int temp = -1;
        if (!data_source.eof())
        {
```

Продолжение листинга 2

```
        data_source >> temp;
        setNumber(temp, byte_massive);
    }
    else
    {
        break;
    }
}
auto end = chrono::steady_clock::now();
data_source.close();
auto duration = chrono::duration_cast<chrono::microseconds> (end
- start);
cout << "Время сортировки данных - " << duration.count() << "
мкс" << endl;
}


int main()
{
    setlocale(LC_ALL, "Russian");
    srand(time(NULL));
    unsigned char* massive = new unsigned char[massive_length];
    for (int i = 0; i < massive_length; i++)
    {
        massive[i] = 0;
    }
    char choose = ' ';
    cout << "Выберите тип ввода: [M] - ручной, [A] - автоматический
из файла: " << endl;
    while (!(tolower(choose) == 'm') || (tolower(choose) == 'a'))
    {
        choose = _getch();
    }
    int n = 0;
    switch(tolower(choose))
    {
    case 'm':
        cout << "Вводите не повторяющиеся целые числа в диапазоне
[0; 9999999], для завершения введите -1:" << endl;
        while (true)
        {
            cin >> n;
            if (n == -1) break;
            if (!setNumber(n, massive))
                cout << "Это число уже вводилось. Введите другое
или введите -1 для завершения." << endl;
        }
        break;
    case 'a':
        cout << "Введите путь к файлу с числами: " << endl;
        string path;
        getline(cin, path);
        fstream ds = fstream(path, ios::in);
        if (!ds) {
            cout << "Файл не найден" << endl;
        }
        else
```

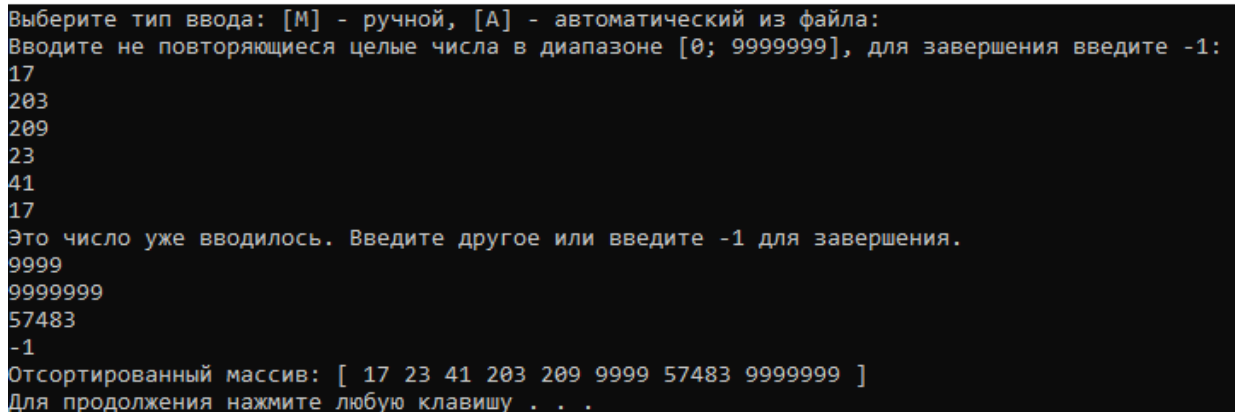
Продолжение листинга 2

```
        {
            cout << "Исходный массив: [ ";
            int temp;
            while (!ds.eof()) {
                ds >> temp;
                cout << temp << " ";
            }
            cout << "]" << endl;
            insert(massive, path.c_str());
        }
    }
    cout << "Отсортированный массив: [ ";
    printNumbers(massive);
    cout << "]" << endl;
    system("pause");
    delete[] massive;
}
```

2.5 Тестирование

Проведем тестирование написанной программы (рисунки 15-19):

 C:\Users\Дмитрий\source\repos\siaod_p2_1_2\Debug\siaod_p2_1_2.exe



```
Выберите тип ввода: [M] - ручной, [A] - автоматический из файла:
Вводите не повторяющиеся целые числа в диапазоне [0; 9999999], для завершения введите -1:
17
203
209
23
41
17
Это число уже вводилось. Введите другое или введите -1 для завершения.
9999
9999999
57483
-1
Отсортированный массив: [ 17 23 41 203 209 9999 57483 9999999 ]
Для продолжения нажмите любую клавишу . . .
```

Рисунок 15 – Пример ручного ввода чисел

```
Консоль отладки Microsoft Visual Studio
Выберите тип ввода: [M] - ручной, [A] - автоматический из файла:
Введите путь к файлу с числами:
nums_100.txt
Исходный массив: [ 3252572 6454606 9708644 9487603 4897801 6587511 6525569 6954892 7382133 2165063 1535698 4617744 60062
68 9855626 373494 1106728 179555 9974315 2425631 2169612 1923321 4322620 637261 9289894 6240328 8935701 2046733 1530347
7184810 7864 1602370 2854845 4231961 3032634 7111888 3576920 1261179 282539 8521130 1865067 6739269 5302130 5052806 5357
153 3230959 1035564 3277799 641016 3075823 7799195 1791333 1800563 1494512 6818692 1071112 72517 1660697 3437664 3488680
4216266 5035011 1453055 8706222 3221321 5074084 3716005 6331513 4376582 6961373 9782540 6496460 2925914 4005734 9712635
3500804 4477996 5042249 1069312 3932922 9428730 4205497 5588842 9560017 6240281 579505 2330392 8339453 5066329 9812951
8774156 6543413 561041 5616343 4116663 1396835 5553188 8039203 1217483 3823857 1890149 ]
Время сортировки данных - 246 мкс
Отсортированный массив: [ 7864 72517 179555 282539 373494 561041 579505 637261 641016 1035564 1069312 1071112 1106728 12
17483 1261179 1396835 1453055 1494512 1530347 1535698 1602370 1660697 1791333 1800563 1865067 1890149 1923321 2046733 21
50663 2169612 2330959 2425631 2854845 2925914 3032634 3075823 3221321 3230959 3252572 3277799 3437664 3488680 3500804 35
76920 3716005 3823857 3932922 4005734 4116663 4205497 4216266 4231961 4322620 4376582 4477996 4617744 4897801 5035011 50
42249 5052806 5066329 5074084 5302130 5357153 5553188 5588842 5616343 6006268 6240281 6240328 6331513 6454606 6496460 65
25569 6543413 6587511 6739269 6818692 6954892 6961373 7104810 7111888 7382133 7799195 8039203 8339453 8521130 8706222 87
74156 8935701 9289894 9428730 9487603 9560017 9708644 9712635 9782540 9812951 9855626 9974315 ]
Для продолжения нажмите любую клавишу . . .
```

Рисунок 16 – Пример сортировки 100 чисел

```
Консоль отладки Microsoft Visual Studio
Выберите тип ввода: [M] - ручной, [A] - автоматический из файла:
Введите путь к файлу с числами:
nums_1000.txt
Исходный массив: [ 1565130 6581193 3191516 9895540 7460133 8190176 8910464 6386869 3826357 9098285 9534022 9024813 96348
34 2040240 3545247 5553383 7023253 3310203 8827325 5304735 5794628 1019201 7996061 2138666 5912842 3001668 6732872 40444
87 4055620 7835772 7384057 2491417 6341147 4543450 8919189 5111051 664211 3241876 6935386 8953828 1364931 2217685 928061
2 7616142 3657892 1609865 6220028 8152571 6061936 643771 1753016 769063 5456295 4521747 4342495 6826346 2114553 2424508
4538270 1755625 8387885 6360898 7134791 9709672 4215437 2003109 3824486 5084947 7022802 1067195 4639346 9946374 8652194
8096122 4845591 7179415 745582 3409191 4160746 9989746 3864236 3857244 477468 2296088 9327073 9039521 493749 493113 1529
965 68419 8418024 4486433 7675021 823598 2222111 618830 6627975 2399221 9981648 8213371 6235468 1346815 1116430 263379 5
230765 7844366 6924363 4714349 8956197 344384 488419 6764275 5255090 2794548 5175880 5087882 9895231 3777665 3571428 221
1817 2590207 8776626 9699154 3332953 7705500 6764723 5371066 2164896 7767471 4225818 3009275 1164154 7718603 4651697 169
4152 3864246 3530206 6432884 8962951 4828596 1442468 6175124 4019991 7571397 7651458 6666476 177493 9780214 6488674 1933
700 9194061 5657899 1071435 7347806 1988508 1248208 9870402 704386 7409817 9422484 544986 1738217 7492903 1637177 150438
8 9550653 2281838 2659280 3625572 9782125 5780108 3574997 6021121 780267 5173152 8175636 6698622 7871759 6104987 1020053
2845124 9069407 5836506 961904 250662 197872 8027666 7576087 5896260 8156198 4969752 2097222 6103673 8902433 858568 29
33992 3117192 1430729 5313292 9529287 2903755 3309948 9022270 5080228 3516605 9539605 5191522 9561608 1344967 520930 836
9347 5083850 7830857 990501 4401467 377961 1904231 4523411 7470279 8473504 2631108 5468292 9187363 9300512 216594 340974
3 730227 6873802 3029367 7919756 8696943 8088262 7318764 8324165 6938340 4729994 9886817 6564596 8495831 2280030 9675170
8116444 1213120 9762482 7765052 8523579 8191271 8872958 6168546 3531522 4246007 8969442 2591384 5986494 3021293 9077464
1797560 3322548 4354894 5171941 4817135 1180019 3974295 4593862 9107188 1537529 3728412 441174 7271734 6715997 8734615
2322274 2199070 4630721 6698908 3725340 4533216 7282303 2435353 1769753 2073119 1712623 415107 2568217 3130748 2136664 3
099269 4576198 7508520 4065765 2938590 9418689 3946348 7380504 8363722 5941076 1441772 6453023 9573154 9205904 7312584 5
621342 4452279 118400 1367652 1063303 9233454 781129 3344669 7981291 7429914 3794461 6031045 4352675 5470622 5678892 29
0797 3185180 1077291 7277473 525425 3436534 7937510 3990448 7864642 2287944 4711331 9074273 5778332 5113815 700750 73041
43 9216477 7850171 8649231 8927447 4811406 2407677 1568331 5041176 741623 3870804 7684208 874753 6938272 3232967 1919798
3646162 7779728 3422117 9706415 5959718 3978035 3922636 5049016 6281442 5914379 5602812 3547015 1827693 9104318 6901639
4289738 7946435 5961374 8907130 4004947 8119684 8666158 1113258 1588941 9905380 7196548 3136464 7172165 5800127 124732
2040923 5030087 7837751 1063139 5076727 5497156 943410 3227612 9337624 3480903 7622146 1688419 9018707 376462 6301953 66
433 2578776 8811953 8230038 1052804 8317533 1296313 3041523 3804664 5468177 3059311 713616 1855458 7374590 9072192 21353
9 5677234 3739637 8720797 5096309 5271142 6425494 7096741 5727997 2889829 3356634 9222244 5034360 8158921 2991832 779884
3 6424055 5730909 6083145 2154261 7212517 2516180 1133087 267625 198703 7028359 9459434 6935307 2737477 9852411 2560236
6655389 3730786 3494715 4175764 833064 7125074 8023367 7345716 592438 6526679 8075423 4225226 9452612 3458778 658389 320
8618 6462132 5510284 1088177 1803064 1130091 8072702 543813 581556 5650104 7427438 4488768 4492170 8226798 7958060 42689
58 4140451 9508933 97455 4064708 1527034 1140022 2494207 3061690 5519152 7797063 1185182 865783 5522322 8058533 9105042
7605207 8420791 2816555 7314597 6028064 3558336 4588691 6441640 8187738 9600200 98702 6380815 3749092 4497039 8372359 54
38857 8453237 8359857 3229967 6472269 7684898 7975483 3839484 5794544 6993920 6269486 6980224 2372588 4348973 785731 628
0062 2757874 5880384 2667620 3376761 4060678 1837670 2567890 6917422 7531424 8641105 2846056 8674470 2898565 1622978 466
3830 3741513 9098091 3247720 1275253 8433125 264826 7295468 3571902 6088640 8929459 2267367 6239607 8182623 3777738 4175
25 3938569 6180010 3327661 6914483 4372378 1144761 6318946 1626752 3295867 2772873 3168839 4559828 24266 5292786 699191
1279358 1730892 1791355 8267430 3428662 8587320 750558 943194 6913079 3053134 8290538 6295481 6054089 7383386 1976910 86
08499 8427430 7388631 4392466 6754714 8779915 7684718 7976391 2275417 7709698 428243 1011064 4004865 6479745 6707423 924
9975 2620284 4594991 4680371 5084565 7201030 8133221 629132 9190963 3287554 7331043 1471851 1645976 2704209 2196006 7149
526 1641305 9455810 2114618 5908526 4264096 5632858 5913741 7387867 3722039 9678362 8288006 5488681 9171635 9394233 4384
098 2897625 6540097 7013626 4429514 1771052 5757651 3591106 6154068 5208850 2072183 9923787 1901313 521662 8011196 95129
44 600588 2279146 3506294 5314736 611284 5260003 8504697 6247973 7244793 4927633 3288304 5434491 3378557 4132895 7181675
1037047 6116394 1015452 2972784 1415840 4652468 9865557 9608529 9078068 3551453 685629 7773758 8557867 7905300 5518842
```

Рисунок 17 – Пример сортировки 1000 чисел, часть 1

```
Выбрать Консоль отладки Microsoft Visual Studio
69514 8826309 7751137 853235 2088183 6406086 2095537 5055010 6585376 4091590 7273535 791991 2909201 3990199 9035001 6793
60 4725966 8712782 471446 9982491 6794124 2143712 4637944 4496815 946442 29904 6531282 5952265 9689159 5910502 4980877 7
315861 553856 9678912 5569900 9742726 5473027 9031789 9038753 669856 7102410 7475821 5525920 8822832 4507384 8647210 984
4672 3300676 1909812 6316741 8130808 8900462 4039923 715985 3141030 1842515 900616 6689496 7079309 7728325 4969735 24967
90 4731523 7897349 1229543 2600449 3124479 7122956 3949113 3314101 7451288 5995228 3468085 2453863 7888471 6661533 18632
39 8935139 3415749 6590270 5369289 712735 1013975 4062792 2322258 2084830 2035605 7197393 9454313 2677639 6292293 582877
2 5901789 55028 5939557 1319533 8072141 2682254 9999591 6779758 4540017 961133 7439223 4871455 1515321 836826 8122042 33
85632 3278133 2763135 4345003 9693639 8455531 5577415 8273985 3501013 2604680 2221579 3699330 8951581 6957674 666047 594
9856 3360822 3858375 5352425 7609990 9782815 8290614 2491791 4469425 5600770 5352210 1759473 8411215 8711934 4053484 468
6772 4273850 53111 992060 2103757 8631290 2491106 2272482 9558028 1457975 353175 3837743 4134432 1767121 3583234 91215 8
523659 7044076 4712671 4590228 2713672 4775180 7360359 9280107 3298550 3688067 5959584 175567 1356254 7862680 3170742 77
36389 5237422 9700537 147963 9565831 9750726 2161443 3843793 7396088 35911 9787296 7938691 462181 2918525 3566482 496754
9 8782590 313912 1592250 801634 9456132 290573 8142669 9871896 6042786 2370202 8084963 7472014 9466916 3635804 1222215 4
47406 7274439 4129796 1322781 8186265 2555077 1969824 5977676 4863078 2418421 4278534 7891175 6025320 9643939 7849526 71
5613 968755 8790200 323288 602507 8633674 7055719 6593704 7803442 2577302 6630237 1588265 7479309 6230140 2485311 760202
5 6503585 1218380 1719899 3380019 9052773 831169 5473552 833278 2810182 9216401 4789771 9350229 9555470 6000114 8413961
1345477 9225670 1834920 574780 5938692 3614438 2460508 3442142 1648908 5908456 9784866 7044664 3474937 8672677 2735702 3
993438 3677349 3115537 3374380 2537469 8257647 3694022 8727656 386235 8515604 3269324 6148747 9124863 5661371 8258830 31
59069 5208042 4478720 7405392 9451108 3691076 9280252 6862848 9648682 2950851 5069273 3061676 5816836 5306221 388061 352
531 8184012 5722045 212876 373688 3474571 5779250 5553408 3361851 401618 1857205 9649140 1849016 3470515 1907324 5562737
2428408 5609626 3338263 6879010 6078973 4982497 5064605 730550 4975226 7223600 2179221 7936298 9977122 6131683 1846314
7247567 2053277 2036274 4492484 2336070 8988768 3678743 9425063 7708542 3414775 4047372 7083084 9426899 ]
Время сортировки данных - 2352 мкс
Отсортированный массив: [ 24266 29904 35911 53111 55028 66433 68419 69514 91215 97455 98702 118400 124732 147963 175567
177493 197872 198703 212876 213539 216594 250662 263379 264826 267625 290573 290797 313912 323288 344384 352531 353175 5
73688 376462 377961 386235 388061 401618 415107 417525 428243 441174 447406 462181 471446 477468 488419 493113 493749 52
0930 521662 525425 543813 544986 553856 574780 581556 592438 600588 602507 611284 618830 629132 643771 658389 664211 666
047 669856 679360 685629 699191 700750 704386 712735 713616 715613 715985 730227 730550 741623 745582 750558 769063 7802
67 785731 791991 801634 823598 831169 833064 833278 836826 853235 865783 874753 900616 943194 943410 946442 961133 96190
4 968755 990501 992060 1011064 1013975 1015452 1019201 1020053 1037047 1052804 1063139 1063303 1067195 1071435 1077291 1
088177 1113258 1116430 1130091 1133087 1140022 1144761 1164154 1180019 1185182 1213120 1218380 1222215 1229543 1248208 1
275253 1279358 1296313 1319533 1322781 1344967 1345477 1346815 1356254 1364931 1367652 1415840 1430729 1441772 1442468 1
457975 1471851 1504388 1515321 1527034 1529965 1537529 1565130 1568331 1588265 1588941 1592250 1609865 1622978 1626752 1
637177 1641305 1645976 1648908 1688419 1694152 1712623 1719899 1730892 1738217 1753016 1755625 1759473 1767121 1769753 1
771052 1791355 1797560 1803064 1827693 1834920 1837670 1842515 1846314 1849016 1855458 1857205 1863239 1901313 1904231 1
907324 1909812 1919798 1933700 1969824 1976910 1988508 2003109 2035605 2036274 2040240 2040923 2053277 2072183 2073119 2
084830 2088183 2095537 2097222 2103757 2114553 2114618 2136664 2138666 2143712 2154261 2161443 2164896 2179221 2196006 2
199070 2211817 2217685 2221579 2222111 2267367 2272482 2275417 2279146 2280030 2281838 2287944 2296088 2322258 2322274 2
336070 2370202 2372588 2399221 2407677 2418421 2424508 2428408 2435353 2453863 2460508 2485311 2491106 2491417 2491791 2
494207 2496790 2516180 2537469 2555077 2560236 2567890 2568217 2577302 2578776 2590207 2591384 2600449 2604680 2620284 2
631108 2659280 2667620 2677639 2682254 2704209 2713672 2735702 2737477 2757874 2763135 2772873 2794548 2810182 2816555 2
845124 2846056 2889829 2897625 2898565 2903755 2909201 2918525 2933992 2938590 2950851 2972784 2991832 3001668 3009275 3
021293 3029367 3041523 3053134 3059311 3061676 3061690 3099269 3115537 3117192 3124479 3130748 3136464 3141030 3159069 3
168839 3170742 3185180 3191516 3208618 3227612 3229967 3232967 3241876 3247720 3269324 3278133 3287554 3288304 3295867 3
298550 3300676 3309948 3310203 3314101 3322548 3327661 3332953 3338263 3344669 3356634 3360822 3361851 3374380 3376761 3
378557 3380019 3385632 3409191 3409743 3414775 3415749 3422117 3428662 3436534 3442142 3458778 3468085 3470515 3474571 3
474937 3480903 3494715 3501013 3506294 3516605 3530206 3531522 3545247 3547015 3551453 3558336 3566482 3571428 3571902 3
```

Рисунок 18 – Пример сортировки 1000 чисел, часть 2

Выбрать Консоль отладки Microsoft Visual Studio

574997	3583234	3591106	3614438	3625572	3635804	3646162	3657892	3677349	3678743	3688067	3691076	3694022	3699330	3722039	3
725340	3728412	3730786	3739637	3741513	3749092	3777665	3777738	3794461	3804664	3824486	3826357	3837743	3839484	3843793	3
857244	3858375	3864236	3864246	3870804	3922636	3938569	3946348	3949113	3974295	3978035	3990199	3990448	3993438	4004865	4
804947	4019991	4039923	4044487	4047372	4053484	4055620	4060678	4062792	4064708	4065765	4091590	4129796	4132895	4134432	4
140451	4160746	4175764	4215437	4225226	4225818	4246007	4264096	4268958	4273850	4278534	4289738	4342495	4345003	4348973	4
352675	4354894	4372378	4384098	4392466	4401467	4429514	4452279	4469425	4478720	4486433	4488768	4492170	4492484	4496815	4
497039	4507384	4521747	4523411	4533216	4538270	4540017	4543450	4559828	4576198	4588691	4590228	4593862	4594991	4630721	4
637944	4639346	4651697	4652468	4663830	4680371	4686772	4711331	4712671	4714349	4725966	4729994	4731523	4775180	4789771	4
811406	4817135	4828596	4845591	4863078	4871455	4927633	4967549	4969735	4969752	4975226	4980877	4982497	5030087	5034360	5
941176	5049016	5055010	5064605	5069273	5076727	5080228	5083850	5084565	5084947	5087882	5096309	5111051	5113815	5171941	5
173152	5175880	5191522	5208042	5208850	5230765	5237422	5255090	5260003	5271142	5292786	5304735	5306221	5313292	5314736	5
352210	5352425	5369289	5371066	5434491	5438857	5456295	5468177	5468292	5470622	5473027	5473552	5488681	5497156	5510284	5
518842	5519152	5522322	5525920	5553383	5553408	5562737	5569900	5577415	5600770	5602812	5609626	5621342	5632858	5650104	5
657899	5661371	5677234	5678892	5722045	5727997	5730909	5757651	5778332	5779250	5780108	5794544	5794628	5800127	5816836	5
828772	5836506	5880384	5896260	5901789	5908456	5908526	5910502	5912842	5913741	5914379	5938692	5939557	5941076	5949856	5
952265	5959584	5959718	5961374	5977676	5986494	5995228	6000114	6021121	6025320	6028064	6031045	6042786	6054089	6061936	6
978973	6083145	6088640	6103673	6104987	6116394	6131683	6148747	6154068	6168546	6175124	6180010	6220028	6230140	6235468	6
239607	6247973	6269486	6280062	6281442	6292293	6295481	6301953	6316741	6318946	6341147	6360898	6380815	6386869	6406086	6
424055	6425494	6432884	6441640	6453023	6462132	6472269	6479745	6488674	6503585	6526679	6531282	6540097	6564596	6581193	6
585376	6590270	6593704	6627975	6630237	6655389	6661533	6666476	6689496	6698622	6698908	6707423	6715997	6732872	6754714	6
764275	6764723	6779758	6794124	6826346	6862848	6873802	6879010	6901639	6913079	6914483	6917422	6924363	6935307	6935386	6
938272	6938340	6957674	6980224	6993920	7013626	7022802	7023253	7028359	7044076	7044664	7055719	7079309	7083084	7096741	7
102410	7122956	7125074	7134791	7149526	7172165	7179415	7181675	7196548	7197393	7201030	7212517	7223600	7244793	7247567	7
271734	7273535	7274439	7277473	7282303	7295468	7304143	7312584	7314597	7315861	7318764	7331043	7345716	7347806	7360359	7
374590	7380504	7383386	7384057	7387867	7388631	7396088	7405392	7409817	7427438	7429914	7439223	7451288	7460133	7470279	7
472014	7475821	7479309	7492903	7508520	7531424	7571397	7576087	7602025	7605207	7609990	7616142	7622146	7651458	7675021	7
684208	7684718	7684898	7705500	7708542	7709698	7718603	7728325	7736389	7751137	7765052	7767471	7773758	7779728	7797063	7
798843	7803442	7811629	7830857	7835772	7837751	7844366	7849526	7850171	7862680	7864642	7871759	7888471	7891175	7897349	7
905300	7919756	7936298	7937510	7938691	7946435	7958060	7975483	7976391	7981291	7996061	8011196	8023367	8027666	8058533	8
872141	8072702	8075423	8084963	8088262	8096122	8116444	8119684	8122042	8130808	8133221	8142669	8152571	8156198	8158921	8
175636	8182623	8184012	8186265	8187738	8190176	8191271	8213371	8226798	8230038	8257647	8258830	8267430	8273985	8288006	8
290538	8290614	8317533	8324165	8359857	8363722	8366937	8372359	8387885	8411215	8413961	8418024	8420791	8427430	8433125	8
453237	8455531	8473504	8495831	8504697	8515604	8523579	8523659	8557867	8585968	8587320	8608499	8631290	8633674	8641105	8
647210	8649231	8652194	8666158	8672677	8674470	8696943	8711934	8712782	8720797	8727656	8734615	8776626	8779915	8782590	8
790200	8811953	8822832	8826309	8827325	8872958	8900462	8902433	8907130	8910464	8919189	8927447	8929459	8935139	8951581	8
953828	8956197	8962951	8969442	8988768	9018707	9022270	9024813	9031789	9035001	9038753	9039521	9052773	9069407	9072192	9
874273	9077464	9078068	9098091	9098285	9104318	9105042	9107188	9124863	9171635	9187363	9190963	9194061	9205904	9216401	9
216477	9222244	9225670	9233454	9249975	9280107	9280252	9280612	9300512	9327073	9337624	9350229	9394233	9418689	9422484	9
425063	9426899	9451108	9452612	9454313	9455810	9456132	9459434	9466916	9512944	9529287	9534022	9539605	9550653	9555470	9
558028	9561608	9565831	9573154	9598933	9600200	9608529	9634834	9643939	9648682	9649140	9675170	9678362	9678912	9689159	9
693639	9699154	9700537	9706415	9709672	9742726	9750726	9762482	9780214	9782125	9782815	9784866	9787296	9844672	9852411	9
865557	9870402	9871896	9886817	9895231	9895540	9905380	9923787	9946374	9977122	9981648	9982491	9989746	9999591]	9

Рисунок 19 – Пример сортировки 1000 чисел, часть 3

По результатам тестирования программы сортировка с помощью битового массива выполнена корректно, значит, можно говорить о правильности составленных алгоритмов. На примере тестов с 100 и 1000 числами видно, что сложность сортировки с помощью битового массива равна $O(n)$, n – число элементов.

3 ЗАКЛЮЧЕНИЕ

В ходе практической работы были изучены битовые операции и способы их применения, в частности, сортировка чисел битовым массивом. Все написанные в рамках этой работы программы работают корректно, следовательно, можно говорить о правильности применения полученных знаний.

4 СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- 1) Методические указания к практической работе. URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F1126230%2Fmod_folder%2Fcontent%2F0%2F%D0%9F%D0%A0-2.1%20%28%D0%91%D0%B8%D1%82.%D0%BE%D0%BF%D0%B5%D1%80.%29.docx&forcedownload=1, дата обращения: 13.09.23
- 2) Задания для самостоятельной работы №1. URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F1137386%2Fmod_assign%2Fintroattachment%2F0%2F%D0%A1%D0%B8%D0%90%D0%9E%D0%94%20%D0%A1%D0%B0%D0%BC%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F%20%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%B0%201%20%28%D0%BF%D0%BE%D1%80%D0%B0%D0%B7%D1%80%D1%8F%D0%B4%D0%BD%D1%8B%D0%B5%20%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8%29.pdf&forcedownload=1, дата обращения: 13.09.23