



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 №1
по дисциплине
«Структуры и алгоритмы обработки данных»

Тема. Поразрядные операции и их применение

Выполнил студент группы ИКБО-04-22

Основин А.И.

Принял старший преподаватель

Скворцова Л.А.

Москва 2023

СОДЕРЖАНИЕ

1	Задание 1.....	4
1.1	Постановка задачи	4
1.2	Упражнение 1	4
1.2.1	Условие задачи.....	4
1.2.2	Разработка решения задачи варианта	4
1.2.3	Реализация решения	4
1.3	Упражнение 2	5
1.3.1	Условие задачи.....	5
1.3.2	Разработка решения задачи варианта	5
1.3.3	Реализация решения	5
1.4	Упражнение 3	6
1.4.1	Условие задачи.....	6
1.4.2	Разработка решения задачи варианта	6
1.4.3	Реализация решения	6
1.5	Упражнение 4	7
1.5.1	Условие задачи.....	7
1.5.2	Разработка решения задачи варианта	7
1.5.3	Реализация решения	7
1.6	Упражнение 5	7
1.6.1	Условие задачи.....	7
1.6.2	Разработка решения задачи варианта	8
1.6.3	Реализация решения	8
2	Задание 2.....	9
2.1	Постановка задачи	9
2.2	Описание структуры данных	9
2.3	Алгоритм решения	9
2.4	Тестовый пример.....	9
2.5	Код программы.....	9
2.6	Результаты тестирования	11

3	Выводы	12
---	--------------	----

1 ЗАДАНИЕ 1

1.1 Постановка задачи

Выполнить упражнения по применению битовых операций по изменению значений битов в ячейке оперативной памяти, созданию маски для изменения значения ячейки. Создать выражения, содержащего поразрядные операции, для выполнения определенной операции над значением ячейки.

Номер варианта: 17.

Задания варианта:

1. Установить с третьего бита четыре слева в единицу;
2. Обнулить 12-й, 11-й и 1-й биты введенного числа;
3. Умножить введенное число на 32, используя побитовые операции;
4. Поделить введенное число на 32, используя побитовые операции;
5. Обнулить n -й бит введенного числа, используя маску, инициализированную единицей в старшем разряде.

1.2 Упражнение 1

1.2.1 Условие задачи

Установить с третьего бита четыре слева в единицу.

1.2.2 Разработка решения задачи варианта

Выражение:

1. `unsigned short mask = 0x78; num = num | mask`
2. `num = num | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6)`

Тестовый пример (`num = 0xA02`):

```
0000 1010 0000 0010 +
0000 0000 0111 1000
-----
0000 1010 0111 1010
```

1.2.3 Реализация решения

Код функции:

```
void task_1() {
    unsigned short num = 0xA02, mask = 0x78;
```

```

    cout << "Number 0xA02: " << bitset<sizeof(short) * 8>(num) <<
endl;

    num = num | mask;
    cout << "Number 0xA02 with 1-st mask:" << bitset<sizeof(short) * 8>(num) <<
endl;
    num = num | (1 << 3) | (1 << 4) | (1 << 5) | (1 << 6);
    cout << "Number 0xA02 with 2-nd mask:" << bitset<sizeof(short) * 8>(num);
}

```

Результат тестирования:

```

Number 0xA02: 0000101000000010
Number 0xA02 with 1-st mask:0000101001111010
Number 0xA02 with 2-nd mask:0000101001111010

```

Рисунок 1 – Результат теста 1.1

1.3 Упражнение 2

1.3.1 Условие задачи

Обнулить 12-й, 11-й и 1-й биты введённого числа.

1.3.2 Разработка решения задачи варианта

Выражение:

1. unsigned short mask = 0x1802; num = num & ~mask;
2. num = num & ~(1 << 12) & ~(1 << 11) & ~(1 << 1)

Тестовый пример (n = 0xFFFF):

```

~mask = ~(0001 1000 0000 0010) = 1110 0111 1111 1101
1111 1111 1111 1111 &
1110 0111 1111 1101
-----
1110 0111 1111 1101

```

1.3.3 Реализация решения

Код функции:

```

void task_2() {
    unsigned short num, mask = 0x1802;
    cout << "Input number: ";
    cin >> num;
    cout << " " << bitset<sizeof(short) * 8>(num) << endl;

    num = num & ~mask;
    cout << "Number with 1-st mask: " << bitset<sizeof(short) * 8>(num) << endl;
    num = num & ~(1 << 12) & ~(1 << 11) & ~(1 << 1);
    cout << "Number with 2-nd mask: " << bitset<sizeof(short) * 8>(num);
}

```

Результат тестирования:

```
Input number:          65536
                        1111111111111111
Number with 1-st mask: 1110011111111101
Number with 2-nd mask: 1110011111111101
```

Рисунок 2 – Результат теста 1.2

1.4 Упражнение 3

1.4.1 Условие задачи

Умножить введённое число на 32, используя побитовые операции.

1.4.2 Разработка решения задачи варианта

Выражение:

`unsigned short pos = 5; num = num << pos`

Тестовый пример (num = 10):

0000 0000 0000 1010 << 5

0000 0001 0100 0000

1.4.3 Реализация решения

Код функции:

```
void task_3() {
    unsigned short num, pos = 5;
    cout << "Input number: ";
    cin >> num;
    cout << " " << bitset<sizeof(short) * 8>(num) << endl;

    num = num << pos;
    cout << "Number with mask: " << bitset<sizeof(short) * 8>(num) << " " <<
num;
}
```

Результат тестирования:

```
Input number:          10
                        00000000000001010
Number with mask:      0000000101000000      320
```

Рисунок 3 – Результат теста 1.3

1.5 Упражнение 4

1.5.1 Условие задачи

Умножить введённое число на 32, используя побитовые операции.

1.5.2 Разработка решения задачи варианта

Выражение:

```
unsigned short pos = 5; num = num >> pos
```

Тестовый пример (num = 512):

```
0000 0010 0000 0000 >> 5
```

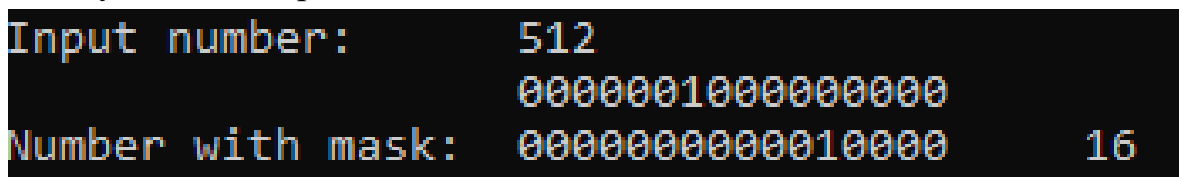
```
-----  
0000 0000 0001 0000
```

1.5.3 Реализация решения

Код функции:

```
void task_4() {  
    unsigned short num, pos = 5;  
    cout << "Input number: ";  
    cin >> num;  
    cout << " " << bitset<sizeof(short) * 8>(num) << endl;  
  
    num = num >> pos;  
    cout << "Number with mask: " << bitset<sizeof(short) * 8>(num) << " " <<  
    num;  
}
```

Результат тестирования:



```
Input number:      512  
                  0000000100000000  
Number with mask:  00000000000010000      16
```

Рисунок 4 – Результат теста 1.4

1.6 Упражнение 5

1.6.1 Условие задачи

Обнулить n-й бит введённого числа, используя маску, инициализированную единицей в старшем разряде.

1.6.2 Разработка решения задачи варианта

Выражение:

`unsigned short mask = 0x800; num = num & ~(mask >> 15 - pos)`

Тестовый пример (`num = 65535; pos = 3`):

$\sim(\text{mask} \gg 15 - \text{pos}) = \sim(1000\ 0000\ 0000\ 0000 \gg 15 - 3) = \sim(0000\ 0000\ 0000\ 1000) = 1111\ 1111\ 1111\ 0111$

1111 1111 1111 1111 &
1111 1101 1111 1111

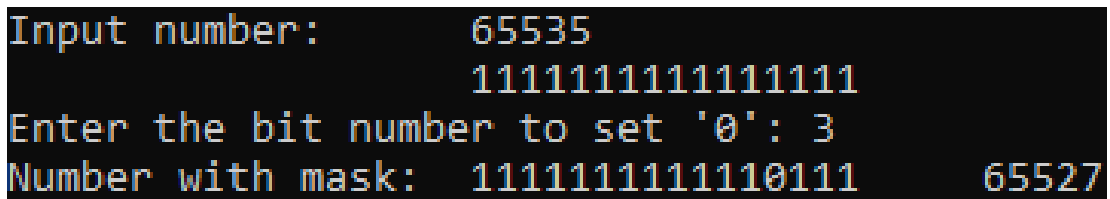
1111 1111 1111 0111

1.6.3 Реализация решения

Код функции:

```
void task_5() {  
    unsigned short num, pos, mask = 0x8000;  
    cout << "Input number: ";  
    cin >> num;  
    cout << " " << bitset<sizeof(short) * 8>(num) << endl;  
  
    cout << "Enter the bit number to set '0': ";  
    cin >> pos;  
  
    num = num & ~(mask >> ((sizeof(short) * 8 - 1) - pos));  
    cout << "Number with mask: " << bitset<sizeof(short) * 8>(num) << " " <<  
    num;  
}
```

Результат тестирования:



```
Input number:      65535  
                  1111111111111111  
Enter the bit number to set '0': 3  
Number with mask: 1111111111110111      65527
```

Рисунок 5 – Результат теста 1.5

2 ЗАДАНИЕ 2

2.1 Постановка задачи

Необходимо отсортировать массив из 10^7 семизначных чисел, затратив на это ~1 МБ оперативной памяти и малое количество времени (не более 10 секунд).

2.2 Описание структуры данных

Для решения поставленной задачи следует использовать битовый массив. Его можно реализовать несколькими способами: массивом типа `unsigned char` размером $[10^7 / 8] + 1$, с помощью класса `bitset` или с помощью класса `vector` из элементов типа `bool`: под значение типа `bool` выделяется 1 байт памяти, но в классе `vector` происходит оптимизация, в результате которой одно логическое значение занимает 1 бит.

2.3 Алгоритм решения

Задачу можно разбить на три подзадачи:

1. Создание битового массива с нулевыми исходными значениями;
2. Считывание целых чисел и установка в единицу соответствующих бит массива;
3. Формирование упорядоченного выходного массива путём последовательной проверки бит массива и вывода индексов тех бит, которые установлены в единицу.

2.4 Тестовый пример

Исходный массив: [15, 7, 9, 4, 6].

Битовый массив, сформированный на основе исходного: `char[10000000 / 8 + 1]`: [00001011, 01000001, 00000000, ..., 00000000].

Отсортированный массив, сформированный на основе битового: [4, 6, 7, 9, 15].

Код программы:

```
unsigned char* create_array(size_t size) {  
    unsigned char* numbers = new unsigned char[(size >> 3) + 1];  
    for (size_t i = 0; i <= (size >> 3); ++i) {  
        numbers[i] = 0;  
    }  
}
```

```

    return numbers;
}

void sort_array(unsigned char* &numbers, vector<int> input_nums) {
    for (size_t i = 0; i < input_nums.size(); ++i) {
        numbers[input_nums[i] >> 3] = numbers[input_nums[i] >> 3] | (1 <<
(input_nums[i] & 7));
    }
}

void get_nums(unsigned char* numbers, vector<int> &output_nums) {
    output_nums.clear();
    unsigned char mask = 1;
    size_t bit;

    for (size_t i = 0; i <= 10000000 >> 3; ++i) {
        bit = 0;
        while (numbers[i] != 0) {
            if ((numbers[i] & mask) != 0) {
                output_nums.push_back((i << 3) + bit);
            }
            numbers[i] >>= 1;
            ++bit;
        }
    }
}

void array_bit_sort() {
    size_t average, num;
    cout << "How many unique numbers we need sort? ";
    cin >> average;

    vector<int> input_nums(average);
    cout << "Enter numbers to sort:" << endl;
    for (size_t i = 0; i < average; ++i) {
        cin >> input_nums[i];
    }

    auto start = chrono::high_resolution_clock::now();
    unsigned char* numbers = create_array(10000000);
    sort_array(numbers, input_nums);
    get_nums(numbers, input_nums);
    auto end = chrono::high_resolution_clock::now();

    cout << "-----SORTED-----" << endl;
    for (int num : input_nums) {
        cout << num << " ";
    }

    chrono::duration<float> duration = end - start;
    cout << endl << "Time: " << setprecision(4) << duration.count();

    delete[] numbers;
}

```

2.5 Результаты тестирования

```
How many unique numbers we need sort? 100
Enter numbers to sort:
3219499 2979872 3981938 2784144 1655171 2065886 9355341 7285857 8987438 7832327 4452133 7329172 6424448 5911748 6693996
5144175 7844166 3928342 3178010 9125349 5731012 7162507 2857417 5601034 8358970 7769953 2706110 9763241 3070099 1402631
4378012 3224947 6459111 7273724 5921713 8945946 9134591 1323869 9282879 1127674 7552156 7470307 6307687 3345775 2208773
4439783 3054387 4500403 6983096 4903572 8729078 1885864 6373017 2329215 3118570 9851776 7774942 4616196 6339640 7166687
8261963 9933802 8444251 9960342 6889748 6749738 7945576 4949749 1346134 8756311 6495450 6462513 9606358 1534097 8085521
3154152 6175989 6148611 8822578 6767090 6381642 9900268 9272098 3242461 2133538 3439923 3299552 9374297 4675257 8269308
6080498 9035210 8234281 2387233 3576613 5810389 1909545 2348418 6391221 8421217
-----SORTED-----
1127674 1323869 1346134 1402631 1534097 1655171 1885864 1909545 2065886 2133538 2208773 2329215 2348418 2387233 2706110
2784144 2857417 2979872 3054387 3070099 3118570 3154152 3178010 3219499 3224947 3242461 3299552 3345775 3439923 3576613
3928342 3981938 4378012 4439783 4452133 4500403 4616196 4675257 4903572 4949749 5144175 5601034 5731012 5810389 5911748
5921713 6080498 6148611 6175989 6307687 6339640 6373017 6381642 6391221 6424448 6459111 6462513 6495450 6693996 6749738
6767090 6889748 6983096 7162507 7166687 7273724 7285857 7329172 7470307 7552156 7769953 7774942 7832327 7844166 7945576
8085521 8234281 8261963 8269308 8358970 8421217 8444251 8729078 8756311 8822578 8945946 8987438 9035210 9125349 9134591
9272098 9282879 9355341 9374297 9606358 9763241 9851776 9900268 9933802 9960342
Time: 0.01697
```

Рисунок 1 – Результат тестирования сортировки 100 чисел

```
2748 8627 6667 5158 2266 3446 7223 9188 7500 8277 7248 2844 3126 5813 5078 9664 6093 8738 2052 9955 9309 9577 6085 5882 9142 739
2 4720 6333 2242 2924 2497 5403 5753 6154 8036 9033 6400 1102 2780 3821 9302 6314 8812 5654 4462 8811 1938 3443 6403 3788 3312 47
04 4177 5870 2034 8626 3299 5235 5494 3063 7930 7484 2434 8704 1958 3215 1199 4890 9819 8897 4268 1460 9469 3335 8079 1610 9337 2
761 9613 3449 5801 6536 2150 8261 6711 7889 1796 8372 3141 6272 2557 8150 5560 7411 6987 5222 6126 8088 6996 2464 3943 5717 9687
6007 7845 6430 1920 5754 9733 2049 4425 8291 5968 7610 4172 5705 6190 1394 8249 7319 3644 1619 2692 2571 1015 6885 4238 3754 1676
4250 9735 5188 2962 9299 7264 4353 1435 7668 5209 2543 6695 7569 9357 5835 4068 2172 4009 4192 9775 3522 8687 5648 4456 7465 811
6 8255 5906 1529 4257 2966 3079 1436 4895 1950 3004 3544 6666 6948 7772 8517 1885 7155 1800 4594 7069 3383 4098 1617 6941 4253 86
54 7635 9931 3366 4465 6459 9211 3381 7910 9717 3236 4162 6528 1579 4690 6929 7891 1804 4432 8276 5089 2126 4702 1556 9778 2205 8
209 1414 2306 3221 5858 9957 5075 7528 9492 2736 3377 5439 6908 7482 2728 3685 2162 1368 6141 8658 8873 8173 9063 1750 5728 2795
6649 2998 5407 4874 6162 7575 3151 5558 9316 6247 7415 4674 6631 4468 9284 9421 6334 1130 3631 7330 7166 2588 9939 9529 4325 5350
8939 6420 9848 3343 1421 3046 2502 1851 8954 8419 5612 8921 7924 9198 9699 8280 3448 6687 2515 9240 2366 5744 7333 5285 4543 102
8 1160 5369 4669 1070 8427 1198 1168 5684 2916 7915 1912 1212 7268 2783 1831 9556 2113 2117 1238 3744 6406 5313 4686 6243 5405 1479 12
21 5039 5685 9656 1839 7869 6380 7846 9118 2285 1543 6432 5688 8229 5400 1773 3237 7614 8618 5887 5199 5127 3675 8735 3112 6418 1
722 6811 6060 1357 9280 9452 6052 2805 1061 6761 8523 1959 4628 7592 4027 4143 2282 8141 6689 7091 4185 2158 3650 1373 4790 2586
8339 2044 8543 9721 3757 2550 1101 9078 1861 8169 4864 6363 7372 6486 7534 7132 8083 1228 3953 2608 1569 5296 3289 4919 4375 4091
9791 7332 4215 2128 7514 6015 7656 9489 6896 3376 7576 1908 9148 6467 5722 5842 1936 6058 8581 2088 1810 6379 2421 6700 2955 477
9 5861 6879 3248 1056 8414 4324 8617 4559 5234 7014 4115 7728 2783 1831 9556 2113 2117 1238 3744 6406 5313 4686 6243 5405 1479 12
71 1155 7257 1143 9853 3623 6296 8577 5033 4397 2971 8031 5037 3791 1797 8023 5816 2912 3519 3881 2061 2701 8377 1973 4765 9540 4
347 3502 7637 9349 4049 3697 3651 6111 5975 9375 3669 1630 4389 3027 9696 2606 3574 3034 1194 3958 7953 2798 2903 3561 4860 1301 6219
5487 5859 7340 5976 5646 5629 2771 9317 3455 9537 2793 8879 4099 2811 7502 6607 3045 4872 6100 1998 9167 2664 3660 7008 9963 3833
2487 7313 8635 2925 2046 4129 2165 5356 9887 2326 1500 6592 4108 4833 6369 5865 5107 2709 8162 5478 6775 2593 4246 3779 8340 841
5 9849 4447 4426 5700 3971 5277 2630 3265 3625 4460 1029 8625 7866 4493 8934 1754 2362 4699 4486 8102 1022 1588 6870 3126 6830 38
14 4075 6055 1400 6362 9503 9591 5602 1346 5869 4627 2977 2679 9011 3952 9283 4112 6539 1345 5389 8284 1597 7701 5787 4206 2700 4
368 5302 1185 5137 1376 4270 5982 6777 6033 6117 3001 3698 9913 8588 2755 2785 4416 3131 4789 4564 8028 5759 8694 3593 8708 7368
7165 7064 2227 6367 2493 2706 1452 6462 7998 7679 8692 9993 8682 7164 6825 4734 7809 6690 6253 5783 5948 2840 2250 7886 1571 4616
6540 4829 3760 9228 6185 9526 9097 6876 6167 9403 4782 9241 6603 3261 3435 8889 3826 9460 6337 3867 5441 6809 6136 7159 1132 238
6 7729 4943 6180 8866 6557 5032 4861 6716 5310 4007 6164 9097 1405 5066 9970 6453 8035 9582 4533 1629 5777 7776 4479 1845 86
10 7354 9921 6043 9709 1467 8937 8774 1642 1809 8161 1215 5198 5312 5373 1949 5673 1699 5297 9706 6347 4969 8907 6822 4088 6294 3
214 6726 3361 4518 7991 5344 7061 4264 6724 7559 7242 3629 8689 5171 6477 8603 1034 4930 2789 5866 6984 6909 2147 8046 7935 1123
9312 9277 3748 1561 9770 3219 7899 8773 1657 7454 6484 8831 8299 1088 6545 5798 6847 8693 4837 2418 5060 1482 1230 4769 6960 8719
-----SORTED-----
1015 1022 1028 1029 1034 1050 1061 1070 1088 1101 1102 1123 1130 1132 1143 1155 1160 1168 1171 1185 1194 1198 1199 1212 1215 1218
1228 1230 1238 1271 1296 1301 1345 1346 1357 1368 1373 1376 1394 1400 1405 1414 1417 1421 1435 1436 1452 1460 1467 1479 1482 150
0 1529 1543 1545 1556 1561 1569 1571 1579 1588 1597 1610 1617 1619 1629 1630 1642 1657 1676 1699 1722 1750 1754 1773 1796 1797 18
00 1804 1809 1810 1831 1839 1845 1851 1861 1885 1908 1920 1936 1938 1949 1950 1958 1959 1973 1998 2034 2044 2046 2049 2052 2061 2
088 2113 2117 2126 2128 2134 2147 2150 2153 2156 2158 2162 2165 2172 2205 2227 2242 2250 2266 2269 2282 2285 2306 2326 2362 2366
2386 2418 2421 2434 2464 2487 2493 2497 2502 2515 2526 2545 2550 2556 2557 2571 2586 2588 2593 2606 2608 2630 2664 2679 2692 2700
2701 2706 2709 2728 2736 2748 2755 2761 2771 2780 2783 2785 2789 2793 2795 2798 2805 2811 2813 2840 2844 2903 2912 2916 2924 292
5 2955 2962 2966 2971 2977 2998 3001 3004 3021 3027 3034 3045 3046 3063 3069 3079 3102 3112 3126 3131 3141 3151 3214 3215 3219 32
21 3236 3237 3248 3261 3265 3289 3299 3312 3335 3343 3361 3366 3376 3377 3381 3383 3435 3443 3446 3448 3449 3455 3502 3519 3522 3
544 3561 3574 3575 3593 3623 3625 3629 3631 3644 3650 3651 3660 3669 3675 3685 3693 3698 3726 3744 3748 3754 3757 3760 3779 3788
3791 3814 3821 3826 3833 3847 3867 3881 3943 3947 3952 3953 3958 3971 4007 4009 4027 4036 4068 4075 4088 4091 4097 4098 4099 4108
4112 4115 4129 4143 4162 4172 4177 4185 4192 4206 4215 4238 4246 4250 4253 4257 4264 4268 4270 4324 4325 4347 4353 4368 4374 437
5 4379 4389 4397 4416 4425 4426 4432 4447 4456 4460 4462 4465 4468 4479 4488 4493 4504 4506 4518 4533 4543 4559 4564 4594 4616 46
27 4628 4630 4669 4674 4686 4690 4699 4702 4704 4712 4720 4765 4769 4779 4782 4789 4790 4798 4829 4833 4837 4860 4861 4864 4872 4
874 4890 4895 4919 4930 4943 4969 5032 5033 5037 5039 5060 5075 5078 5089 5107 5127 5137 5158 5171 5188 5198 5199 5208 5209 5222
5234 5235 5277 5282 5285 5296 5297 5302 5310 5312 5313 5344 5350 5356 5369 5373 5389 5400 5403 5405 5407 5439 5441 5478 5487 5494
5558 5560 5602 5612 5623 5629 5638 5646 5648 5654 5673 5684 5685 5688 5700 5705 5717 5722 5728 5744 5753 5754 5759 5765 5777 578
7 5798 5801 5813 5816 5835 5842 5858 5859 5861 5865 5866 5869 5870 5873 5882 5887 5906 5941 5948 5968 5975 5976 5982 6007 6015 60
33 6035 6043 6052 6055 6058 6060 6085 6093 6100 6111 6117 6126 6136 6141 6143 6154 6159 6162 6164 6167 6180 6185 6190 6219 6243 6
247 6253 6272 6294 6296 6314 6333 6334 6337 6347 6362 6363 6367 6369 6379 6380 6400 6403 6406 6418 6420 6430 6432 6453 6459 6462
6467 6477 6484 6486 6528 6536 6539 6540 6545 6557 6592 6603 6607 6621 6631 6649 6666 6667 6687 6689 6690 6695 6700 6711 6716 6724
6726 6761 6775 6777 6809 6811 6822 6824 6825 6830 6847 6870 6876 6879 6885 6896 6908 6909 6929 6941 6948 6960 6984 6987 6996 700
8 7014 7061 7064 7069 7091 7132 7155 7159 7164 7165 7166 7223 7236 7242 7248 7257 7264 7268 7313 7319 7330 7332 7333 7340 7351 73
54 7387 7372 7377 7390 7392 7411 7415 7454 7455 7465 7482 7484 7500 7502 7514 7528 7534 7559 7569 7575 7576 7583 7592 7610 7614 7
635 7637 7640 7656 7668 7679 7700 7701 7703 7728 7729 7741 7772 7775 7795 7809 7845 7846 7866 7869 7886 7889 7891 7899 7910 7912
7915 7919 7924 7930 7936 7953 7991 7998 8083 8028 8031 8035 8036 8046 8079 8083 8088 8102 8116 8141 8150 8161 8162 8169 8173 8209
8229 8249 8255 8261 8276 8277 8280 8284 8291 8299 8330 8340 8372 8377 8414 8415 8419 8427 8428 8436 8517 8523 8543 8577 8581 858
8 8603 8610 8617 8618 8625 8626 8627 8635 8654 8658 8682 8687 8689 8692 8693 8694 8704 8708 8719 8735 8738 8744 8773 8774 8811 88
12 8817 8831 8866 8873 8878 8879 8889 8897 8907 8921 8934 8937 8939 8954 8977 9011 9033 9063 9073 9078 9097 9101 9118 9142 9148 9
167 9188 9198 9211 9228 9240 9241 9277 9280 9283 9284 9299 9302 9309 9312 9316 9317 9337 9349 9357 9375 9403 9421 9452 9460 9469
9489 9492 9497 9503 9518 9526 9529 9537 9540 9556 9577 9582 9591 9613 9614 9656 9664 9687 9696 9699 9706 9709 9717 9721 9733 9735
9770 9775 9778 9791 9819 9848 9849 9853 9887 9913 9921 9931 9939 9955 9957 9963 9970 9993 9996
Time: 0.01681
```

Рисунок 2 – Результат сортировки 1000 чисел

3 ВЫВОДЫ

В ходе выполнения практической работы были изучены битовые операции в языке программирования C++, выполнены простые задачи с использованием битовых операций, также была реализована сортировка с использованием битового массива, работающая только для уникальных элементов.