

Тема:
«Работа с данными из файла»

Группа: ИКБО-10-23

Москва 2024

Цель:

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

1.1 Формулировка задачи

- a) Реализуйте вышеприведённый пример, проверьте правильность результата в том числе и на других значениях x .
- b) Реализуйте по аналогии с предыдущим примером установку 7-го бита числа в единицу.
- c) Реализуйте код листинга 1, объясните выводимый программой результат.

1.2 Описание алгоритма

Необходимо установить n -й бит произвольного числа в 0. Для этого необходимо создать маску, в которой все биты кроме n -го будут установлены в 1, тогда при побитовой конъюнкции с любым числом n -й бит обратится в 0, а все остальные сохранят исходное значение.

Для установки n -го бита числа в 1 понадобится уже маска, инвертированная по отношению к описанной выше. Все биты маски кроме n -го должны быть установлены в 0, тогда при побитовой дизъюнкции с любым числом n -й бит обратится в 1, а остальные сохранят исходное значение.

В коде листинга 1, представленного на рисунке 3, описывается алгоритм представления двоичной записи числа. Для получения n -го бита числа производится поразрядная конъюнкция с маской, в которой только n -й бит установлен в 1, а остальные в 0. В итоге единственный сохраненный бит смещается побитовым сдвигом на первую позицию и выводится. В маске установленный в 1 бит смещается на один разряд вправо и цикл повторяется. По итогу в консоли вывода оказываются выведены все битовые разряды числа по очереди.

1.3 Исходный код программ

На рисунках 1-3 представлен исходный код программ 1.а и 1.б, а также код листинга 1.

```

ex_5_1 > G+ a.cpp > main()
1  #include <iostream>
2  #include <cmath>
3  #include <bitset>
4
5  using namespace std;
6
7  int main() {
8      unsigned char x = 255;
9      unsigned char mask = 1;
10     cout << "Исходное число: " << +x << endl;
11     x = x & (~ (mask << 4));
12     cout << "Число 0 в 5-м бите: " << +x;
13     return 0;
14 }

```

Рисунок 1 — Исходный код 1.а

```

ex_5_1 > G+ main.cpp > ...
1  #include <iostream>
2  #include <cmath>
3
4  using namespace std;
5
6  int main() {
7      unsigned char x=32;
8      unsigned char maska = 1;
9      cout << "Исходное число: " << +x << endl;
10     x = x | (maska<<6);
11     cout << "Полученное число 1 в 7-м бите: " << +x;
12
13     return 0;
14 }
15

```

Рисунок 2 - Исходный код 1.б

```

1  #include <iostream>
2  #include <cmath>
3  #include <bitset>
4
5  using namespace std;
6
7  int main()
8  {
9      unsigned int x = 25;
10     const int n = 32;
11     unsigned int mask = (1 << n - 1);
12     cout << "Initial view of the mask: " << bitset<n>(mask) << endl;
13     cout << "Result: ";
14     for (int i = 1; i <= n; ++i)
15     {
16         cout << ((x & mask) >> (n - i));
17         mask = mask >> 1;
18     }
19     cout << endl;
20
21     return 0;
22 }

```

Рисунок 3 — Исходный код 1.в

1.4 Результаты тестирования

На рисунках 4-5 представлены результаты тестирования программы 1.а. На рисунке 6 представлен результат тестирования программы 1.б. На рисунке 7 представлен результат тестирования программы листинга 1.

```

Исходное число: 255
Число с 0 в 5-м бите: 239

```

Рисунок 4 — Результат тестирования 1.а

```

Исходное число: 127
Число с 0 в 5-м бите: 111

```

Рисунок 5 — Результат тестирования 1.а

```

Исходное число: 32
Полученное число с 1 в 7-м бите: 96

```

Рисунок 6 — Результат тестирования 1.б

```

Исходное число: 25
Начальный вид маски: 10000000000000000000000000000000
Результат: 0000000000000000000000000000000011001

```

Рисунок 7 — Результат тестирования 1.в

2.1 Формулировка задачи

- a) Реализуйте вышеописанный 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.
- b) 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`.
- c) 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`.

2.2 Описание алгоритма

Для сортировки при помощи битового массива каждый бит этого массива представляется как индикатор наличия в последовательности числа, равного номеру этого бита в массиве. Для вывода же отсортированной последовательности необходимо поочередно пройти по всем битам массива и вывести номера тех, которые установлены в 1.

Если же количество чисел n , соответственно, максимальное из них превосходит битовый размер типа данных, необходимо создать массив. Номер элемента массива, в который необходимо провести запись, можно вычислять как результат целочисленного деления записываемого числа на битовый размер одного элемента, а номер бита в этом элементе — как результат вычисления остатка. Для вывода уже отсортированной последовательности действия будут аналогичными.

2.3 Исходный код программ

На рисунках 8-10 представлен исходный коды программ 2.а, 2.б и 2.в.

```

ex_5_1 > 2_a.cpp > ...
1  #include <iostream>
2  #include <cmath>
3  #include <bitset>
4
5  using namespace std;
6
7  int main() {
8      cout << "Введите количество чисел: ";
9      int n;
10     cin >> n;
11     unsigned char bit_arr = 0;
12     int num;
13     unsigned char bit_num = 1;
14     for (int i = 0; i < n; ++i) {
15         cin >> num;
16         bit_arr = (bit_arr | (bit_num << num));
17     }
18     cout << "Отсортированный массив:";
19     for (int i = 0; i < sizeof(bit_arr) * 8; ++i) {
20         if (bit_arr & (bit_num << i)) {
21             cout << " " << i;
22         }
23     }
24     return 0;
25 }

```

Рисунок 8 — Исходный код 2.а

```

ex_5_1 > 2_b.cpp > ...
1  #include <iostream>
2  #include <cmath>
3  #include <bitset>
4
5  using namespace std;
6
7  int main() {
8      cout << "Введите количество чисел: ";
9      int n;
10     cin >> n;
11     unsigned long long bit_arr = 0;
12     int num;
13     unsigned long long bit_num = 1;
14     for (int i = 0; i < n; ++i) {
15         cin >> num;
16         bit_arr = (bit_arr | (bit_num << num));
17     }
18     cout << "Отсортированный массив:";
19     for (int i = 0; i < sizeof(bit_arr) * 8; ++i) {
20         if (bit_arr & (bit_num << i)) {
21             cout << " " << i;
22         }
23     }
24     return 0;
25 }

```

Рисунок 9 - Исходный код 2.б

```

ex_5.1 > 2.cpp > ...
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6      cout << "Введите количество чисел: ";
7      int n;
8      cin >> n;
9      unsigned char bit_arr[8];
10     for (int i = 0; i < 8; ++i) {
11         bit_arr[i] = 0;
12     }
13     int num;
14     unsigned char bit_one = 1;
15     cout << "Введите " << n << " чисел: ";
16     for (int i = 0; i < n; ++i) {
17         cin >> num;
18         bit_arr[(7 - num / 8)] = bit_arr[(7 - num / 8)] | (bit_one << (num % 8));
19     }
20     cout << endl;
21     cout << "Отсортированный список:";
22     for (int i = 0; i < 64; ++i) {
23         if (bit_arr[(7 - i / 8)] & (bit_one << (i % 8))) {
24             cout << " " << i;
25         }
26     }
27     return 0;
28 }

```

Рисунок 10 — Исходный код 2.в

2.4 Результаты тестирования

На рисунках 11-13 представлены результаты тестирования программы 2.а, 2.б и 2.в.

```

Введите количество чисел: 5
7 1 3 2 4
Отсортированный массив: 1 2 3 4 7

```

Рисунок 11 — Результат тестирования 2.а

```

Введите количество чисел: 10
43 3 1 60 7 39 12 4 5 45
Отсортированный массив: 1 3 4 5 7 12 39 43 45 60

```

Рисунок 12 — Результат тестирования 2.б

```

Введите количество чисел: 8
Введите 8 чисел: 10 63 1 0 35 48 3 30
Отсортированный список: 0 1 3 10 30 35 48 63

```

Рисунок 13 — Результат тестирования 2.в

3.1 Формулировка задачи

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

3.2 Описание алгоритма

Для генерации последовательности случайных неповторяющихся чисел сначала создаётся массив длиной n и заполняется числами от 0 до n , после чего поочередно каждый элемент массива меняется местами с другим элементом со случайным индексом.

Для сортировки при помощи битового массива каждый бит этого массива представляется как индикатор наличия в последовательности числа, равного номеру этого бита в массиве. Для вывода же отсортированной последовательности необходимо поочередно пройти по всем битам массива и вывести номера тех, которые установлены в 1.

Номер элемента массива, в который необходимо провести запись, можно вычислять как результат целочисленного деления записываемого числа на битовый размер одного элемента, а номер бита в этом элементе — как результат вычисления остатка. Для вывода уже отсортированной последовательности действия будут аналогичными.

Размер же массива вычисляется как количество чисел, разделенное на битовый размер одного элемента массива и округленное вверх.

3.3 Исходный код программы

На рисунке 14 представлен исходный коды программ 3.а.

```

41 void bit_sort(int ln)
42 {
43     ifstream A;
44     A.open("A_input.txt");
45
46     int arr_ln = ceil(ln / 8.0);
47     int num;
48     unsigned char bit_one = 1;
49     unsigned char *bit_array = new unsigned char[arr_ln];
50     for (int i = 0; i < arr_ln; ++i) {
51         bit_array[i] = 0;
52     }
53
54     while (!A.eof())
55     {
56         A >> num;
57         bit_array[((arr_ln - 1) - num / 8)] = bit_array[((arr_ln - 1) - num / 8)] | (bit_one << (num % 8));
58     }
59     A.close();
60
61     ofstream B;
62     B.open("B_output.txt");
63
64     bool is_written = false;
65
66     for (int i = 0; i < ln; ++i) {
67         if (bit_array[((arr_ln - 1) - i / 8)] & (bit_one << (i % 8))) {
68             if (is_written) {
69                 B << "\n";
70             }
71             is_written = true;
72             B << i;
73         }
74     }
75     B.close();
76
77     double size = sizeof(bit_array[0]) * arr_ln / 1024.0;
78     printf("Объём памяти, занимаемый битовым массивом: %f Кбайт", size);
79     delete[] bit_array;
80 }

```

Рисунок 14 — Исходный код 3.a

2.4 Результаты тестирования

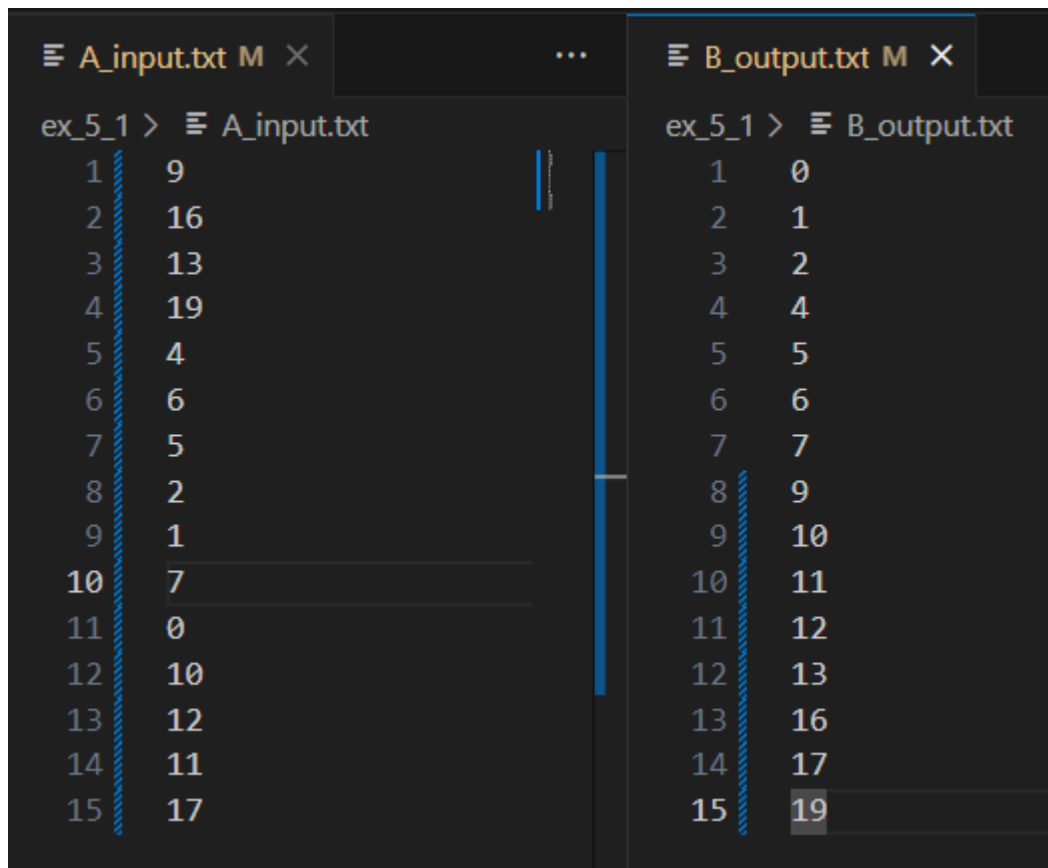
На рисунке 15 представлены результаты тестирования программы 3.a. На рисунке 16 представлены исходная и отсортированная последовательности длины 15.

```

Объём памяти, занимаемый битовым массивом: 1024.000000 Кбайт
Время выполнения: 2.000000 секунд

```

Рисунок 15 — Результат тестирования 3.a



A_input.txt		B_output.txt	
ex_5_1 >	A_input.txt	ex_5_1 >	B_output.txt
1	9	1	0
2	16	2	1
3	13	3	2
4	19	4	4
5	4	5	5
6	6	6	6
7	5	7	7
8	2	8	9
9	1	9	10
10	7	10	11
11	0	11	12
12	10	12	13
13	12	13	16
14	11	14	17
15	17	15	19

Рисунок 16 — Результат сортировки последовательности

3.1 Выводы

В ходе решения поставленных задач были освоены приёмы работы с битовым представлением беззнаковых целых чисел, а также реализован эффективный алгоритм внешней сортировки на основе битового массива.

Достоинствами битовой сортировки являются скорость и маленький объём занимаемой памяти. К недостаткам же можно отнести невозможность сортировать последовательности с повторяющимися элементами, а также неуниверсальность ввиду зависимости необходимой длины массива от максимального числа, а не их количества.