1. Цель работы

Целью работы является изучение методов и получение практических навыков анализа сложности алгоритмов.

2. Задание

Согласно варианту №16:

16	Все четные значения элементов уменьшить в два раза	O(n)
	Подсчитать количество элементов с нулевым значением	O(1)

3. Листинг программы

```
1. // Вариант 16
2. #define _CRTDBG_MAP_ALLOC
3. #ifdef _DEBUG
4. #ifndef DBG_NEW
5. #define DBG_NEW new ( _NORMAL_BLOCK , __FILE__ , __LINE__ )
6. #define newDBG_NEW
7. #endif
8. #endif
9.
10. #include <iostream>
11. #include <iomanip>
12.
13. using namespace std;
14.
15. int get_num_int() // Запрос и проверка числа на корректность
16. {
17.     int x;
18.
19.     cin >> x;
20.     while (cin.fail() || (cin.peek() != '\n')) // Проверка на корректность
21.     {
22.         cin.clear(); // Очищение флага ошибки
23.         cin.ignore(numeric_limits<streamsize>::max(), '\n'); // Очистка буфера
    запроса
24.         cout << "Повторите ввод: ";
25.         cin >> x;
26.     }
27.
28.     return x;
29. }
30.
31. unsigned int request_len()
32. {
33.     int n;
34.
35.     n = get_num_int();
36.
37.     while (n <= 0) // Проверка на корректный ввод
38.     {
39.         cout << "Повторите ввод: ";
40.         n = get_num_int();
41.     }
42.
43.     return n;
44. }
45.
46. void fill_arr(int* numbers, const unsigned int n)
47. {
```

```
48.     srand(time(NULL));
49.     int min = -1 * ((n / 2) - 1);
50.     int max = (n / 2);
51.
52.     for (unsigned int i = 0; i < n; i++) // Заполнение массива
53.     {
54.         numbers[i] = (min) + rand() % (max - min + 1);
55.     }
56. }
57.
58. void reduce_even(int* numbers, const unsigned int n)
59. {
60.     for (unsigned int i = 0; i < n; i++) // Поиск чётных и уменьшение на 2
61.     {
62.         if (numbers[i] % 2 == 0)
63.         {
64.             numbers[i] /= 2;
65.         }
66.     }
67. }
68.
69. void print_zero(int counts)
70. {
71.     cout << "Количество нулевых элементов: " << counts;
72. }
73.
74. void print_arr(int* numbers, const unsigned int n)
75. {
76.     for (unsigned int i = 0; i < n; i++) // Вывод массива
77.     {
78.         cout << numbers[i] << setw(5);
79.     }
80.     cout << endl;
81. }
82.
83. int main()
84. {
85.     setlocale(LC_ALL, "rus");
86.
87.     cout << "Введи размерность массива целым неотрицательным числом (!=0)!" <<
    endl;
88.     cout << "n = ";
89.
90.     unsigned int n = request_len();
91.
92.     int* numbers = new int[n];
93.     int counts = 0;
94.     int choose;
95.
96.     fill_arr(numbers, n);
97.
98.     cout << "Ваш массив заполненный случайными числами: " << endl;
99.     print_arr(numbers, n);
100.
101.         while (true)
102.         {
103.             cout << "Выберите дальнейшее действие: " << endl;
104.             cout << "1. Уменьшить все чётные значения в два раза" << endl;
105.             cout << "2. Подсчитать количество нулевых элементов" << endl;
106.             cin >> choose;
107.
108.             if (choose == 1)
109.             {
110.                 reduce_even(numbers, n);
111.                 cout << "Ваш массив преобразован: " << endl;
```

```
                print_arr(numbers, n);
                break;
            }
            else if (choose == 2)
            {
                for (unsigned int i = 0; i < n; i++)
                {
                    if (numbers[i] == 0)
                    {
                        counts += 1;
                    }
                }

                print_zero(counts);
                break;
            }
        }

        delete[] numbers;

        // Для обнаружения утечек памяти
        _CrtSetReportMode(_CRT_WARN, _CRTDBG_MODE_FILE);
        _CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDOUT);
        _CrtSetReportMode(_CRT_ERROR, _CRTDBG_MODE_FILE);
        _CrtSetReportFile(_CRT_ERROR, _CRTDBG_FILE_STDOUT);
        _CrtSetReportMode(_CRT_ASSERT, _CRTDBG_MODE_FILE);
        _CrtSetReportFile(_CRT_ASSERT, _CRTDBG_FILE_STDOUT);
        _CrtDumpMemoryLeaks();

}
```

4. Расчет пространственной и временной сложностей алгоритма

- Пространственная сложность алгоритма:

  Разработанный алгоритм использует следующие данные:

  - один массив размерностью n;
  - шесть переменных целого типа;
  - пять переменных целового типа без знака.

  $v = n*C_{int} + 6*C_{int} + 5*C_{uint}$

  где C_{int} – Константа, характеризующая память, выделяемая под переменную целого типа, C_{uint} - Константа, характеризующая память, выделяемая под переменную целого типа без знака.

- Временная сложность алгоритма:

  Временную сложность алгоритма определяем на основе анализа текста программы, реализующей этот алгоритм.

  $t_{Reduce} = n*K_{64}$

  $t_{Zero} = K_{71}$

  $t_{Alg} = K_{90} + K_{92} + K_{93} + K_{94} + n*K_{121} + K_{130} + t_{Reduce} + t_{Zero} + t_{Fill} + 2*t_{print}$

где K_i – константа, характеризующая время выполнения операций, помеченных i; t_{Reduce}, t_{Zero}, t_{Fill}, t_{print} и t_{Alg} – временные сложности функций reduce_even, print_zero, fill_arr, print_arr и всего алгоритма в целом, соответственно.

5. **Расчет теоретической пространственной и теоретической временной сложностей алгоритма**

   - Теоретическая пространственная сложность алгоритма:
     $V(n) = O(v) = O(\max(O(n*C_{int}), O(6 * C_{int}), O(5 * C_{uint}))) = O(\max(O(n), O(1), O(n))) = O(n)$
   - Теоретическая временная сложность:
     $T_{Reduce}(n) = O(t_{Reduce}) = O(\max(O(n*K_{64}))) = O(n)$
     $T_{Zero}(n) = O(t_{Reduce}) = O(\max(O(K_{71}))) = O(1)$

Согласно заданию на лабораторную работу, необходимо реализовать две функции, теоретические временные сложности которых не превышают заданных. По рассчитаным данным можно сделать вывод, что характеристики алгоритма соответсвуют заданию.

6. **Выводы**

В процессе лабораторной работы были получены практические навыки анализа сложности алгоритмов. Разработанный алгоритм по сложности соответсвует поставленному заданию.