Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«ОРЕНБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Институт математики и информационных технологий

Кафедра программного обеспечения вычислительной техники и

автоматизированных систем

**ОТЧЕТ**

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

**Анализ временной сложности алгоритмов**

ОГУ 09.03.04.4023.321 О

|  |
| --- |
| Руководитель  старший преподаватель  С.А. Климачев  «\_\_\_»\_\_\_\_\_\_\_\_\_\_2023г.  Студент группы 22Пинж(б)РпиС-2  К.Р. Безруков  «\_\_\_»\_\_\_\_\_\_\_\_\_\_2023г. |

Оренбург 2023

**1 Постановка задачи**

Цель работы:формирование умений и навыков анализа временной сложности алгоритмов.

Задача: Известны данные о количестве осадков, выпавших в каждый день заданного промежутка времени. Определите общее количество осадков, выпавших в четные дни.

**2 Блок-схема алгоритма**

На рисунке 1 представлена блок-схема алгоритма.

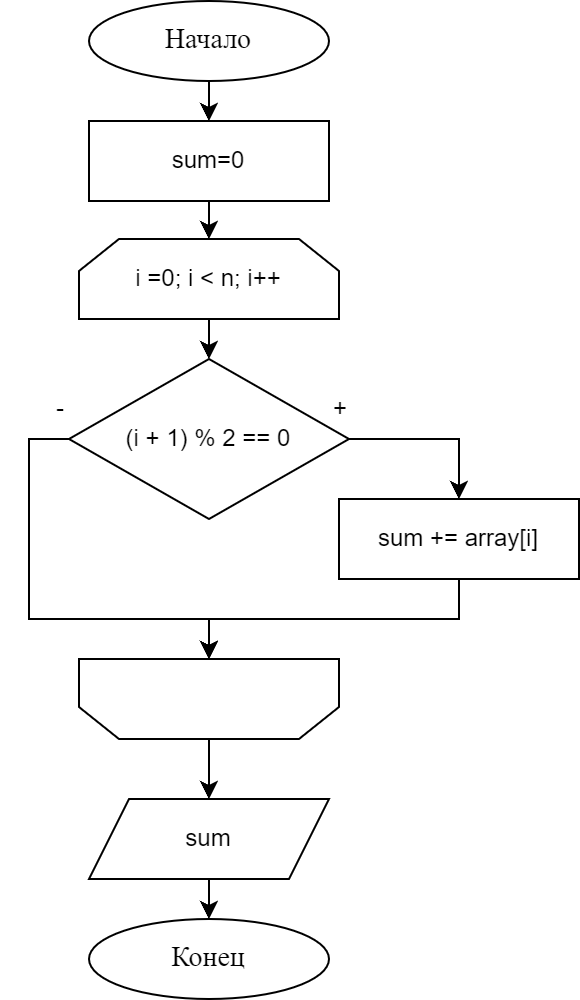
****

Рисунок 1 — Блок-схема алгоритма

**3 Анализ временной сложности алгоритма**

Выполним асимптотический анализ вычислительной сложности алгоритма вычисления того, сколько значений температур в каждом из трех режимов больше нижней границы.

long getSumEven(int\* array, int size) {

clock\_t start = clock();

long sum = 0;

for (int i = 0; i < size; i++)

if ((i + 1) % 2 == 0)

sum += array[i];

time = (double(clock() - start)) / CLOCKS\_PER\_SEC;

return sum;

}

1) Определим параметры, от которых зависит время выполнения алгоритма. В данном случае определяющим параметром будет количество дней n. Следовательно, функция времени будет иметь только один аргумент: *T*(*n*).

2) В качестве базовых операций, выполняемых алгоритмом, выберем операции (1) (i+1)%2 == 0 и (2) sum += array[i]. А в качестве всех операций добавим к базовым следующие операции: (3) i < size, (4) i++. Первая операция будет выполнена n раз, вторая n/2 раз, треьтя n+1 раз и четвёртая n раз. Таким образом, *Tб*(*n*) = *n* + *n* / 2= 3*n* / 2, *Tв*(*n*) = *n* + *n* / 2 + *n + n + 1* = 7*n* / 2 + 1.

3) Докажем, что 3*n* / 2 = *Ο*(*n*). Найдем такие *с*>0 и *n*0 **{**0, 1, 2, …}, при которых выполняется неравенство 0 ≤ 3*n* / 2 ≤ *cn*. Поскольку функции пересекаются в точке *n*0: 3*n*0 / 2 = *cn*0, то *c = 3 / 2*, и константа *n*0 = 1 обеспечивает выполнение неравенства. Докажем, что 7*n* / 2 + 1 = *Ο*(*n*). Найдем такие *с*>0 и *n*0 **{**0, 1, 2, …}, при которых выполняется неравенство 0 ≤ 7*n* / 2 + 1 ≤ *cn*. Поскольку функции пересекаются в точке *n*0: 7*n*0 / 2 + 1 = *cn*0, то *c = 7 / 2 + 1*, и константы *n*0 = 1, c = 9 / 2 обеспечат выполнение неравенства. Следовательно, алгоритм имеет линейную сложность.

Выполним алгоритм для разного объема входных данных (1000, 6000, 36000, 216000, 1296000, 7776000) и построим графики зависимостей.

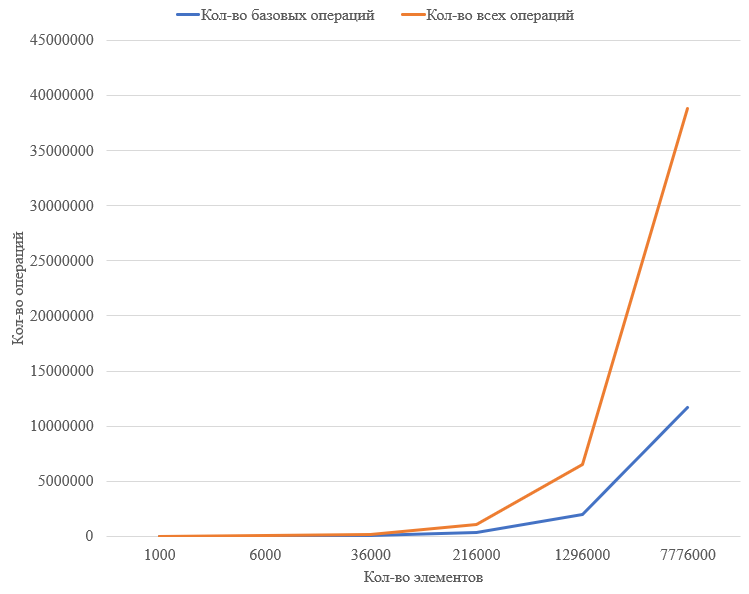


Рисунок 2 – График зависимости количества базовых/всех операций от объема входных данных

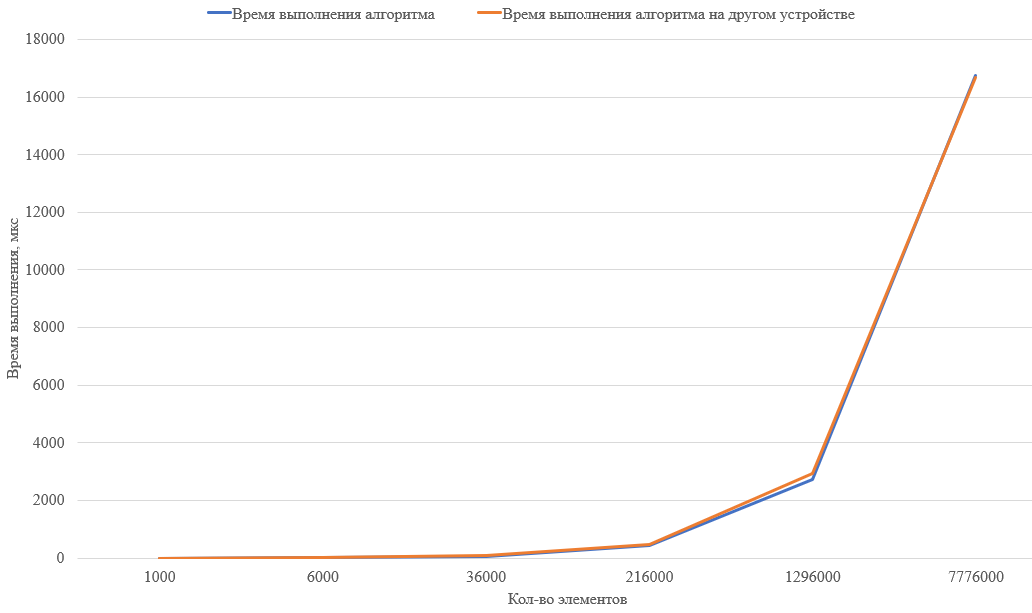


Рисунок 3 – График зависимости времени выполнения алгоритма(на разных устройствах) от объема входных данных

**4 Вывод**

В ходе лабораторной работы было выяснено, что с ростом входных данных в алгоритме увеличивается и количество операций, выполняемых алгоритмом. Было выяснено, что некоторые операции можно не учитывать при вычислении эффективности алгоритма; можно учитывать только базовые операции, такие как: операции сравнения, арифметические операции (аддитивные и мультипликативные). Эффективность алгоритма измеряется не во времени выполнения, а в количестве операций (или в объеме занимаемой памяти), т.к. одна и та же программа может быть выполнена с разной скоростью на разных устройствах, в зависимости от технических характеристик этих устройств.