МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Челябинский государственный университет»**

**(ФГБОУ ВО «ЧелГУ»)**

Институт информационных технологий

Кафедра информационных технологий и экономической информатики

ОТЧЕТ

по лабораторной работе № 1

Авторы отчета \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись инициалы, фамилия группа

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

подпись инициалы, фамилия группа

Отчет защищен \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

дата оценка

Челябинск 2024 г.

Содержание

[Глава 1 3](#_gjdgxs)

[Глава 2](#_30j0zll) 21

[Глава 3](#_1fob9te) 23

# Глава 1

Формулировка задания: Сгенерируйте n-мерный случайный вектор v = [v1, v2, . . . , vn] с неотрицательными элементами. Для полученного вектора v осуществите подсчет функций и реализацию алгоритмов:

1. (постоянная функция);
2. (Суммая элементов);
3. (произведение элементов);
4. полагая, что элементы – коэффициенты многочлена степени, вычислите значение путем прямого (наивного) вычисления 

для x = 1,5 (т.е. оценивая каждый член по одному) и методом Горнера представление полинома: 

1. алгоритм сортировки пузырьком (Bubble sort) элементов 
2. алгоритм быстрой сортировки (Quick sort) элементов
3. гибридный алгоритм сортировки Timsort элементов
4. Алгоритмы возведения в степень;

**Описание и решение каждого алгоритма:**

1. Постоянная функция: Алгоритм Постоянная функция в C# представляет собой метод, который всегда возвращает одно и то же значение, независимо от входных параметров [см.рисунок 1]. Это может быть полезно в различных ситуациях, например, для тестирования или в качестве заглушки.

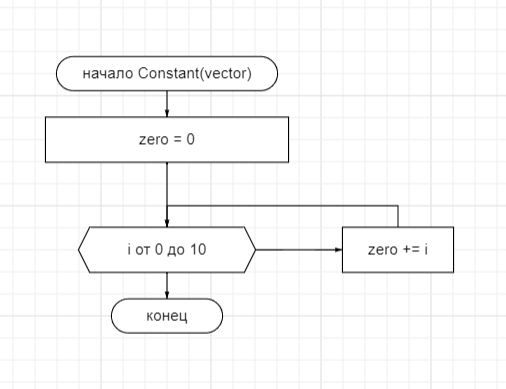


Рис.1 Блок-схема Constant(vector)

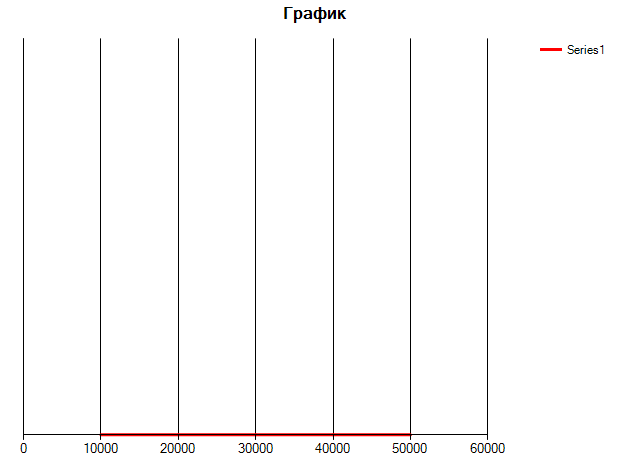


Рис. 2 График затраченного времени на итерацию

**Сумма элементов**

2. Сумма элементов: Алгоритм представляет собой сложение всех элементов[см.рисунок 3].

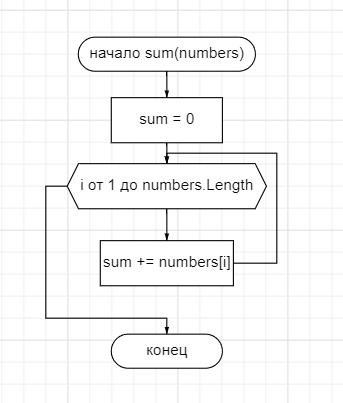


Рис. 3 Блок-схема sum(numbers)

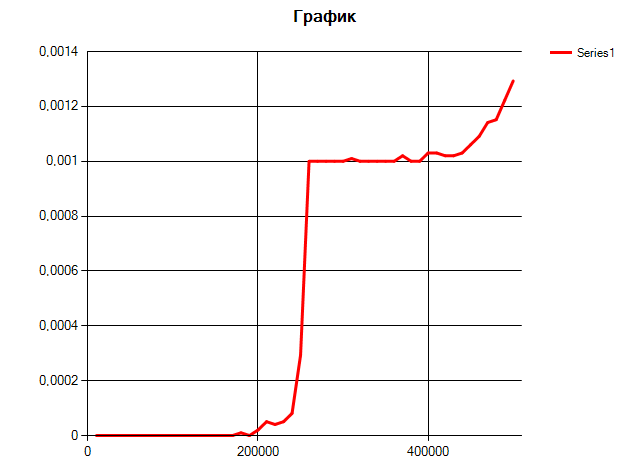


Рис. 4 График затраченного времени на сложение элементов

**Произведение элементов**

3. Произведение элементов: Алгоритм для вычисления произведения элементов массива в C#.

Алгоритм представляет собой перемножение всех элементов массива[см.рисунок 5].

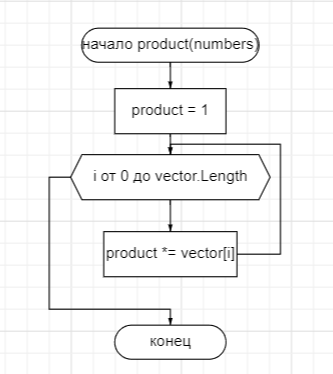


Рис. 5 Блок-схема произведения элементов

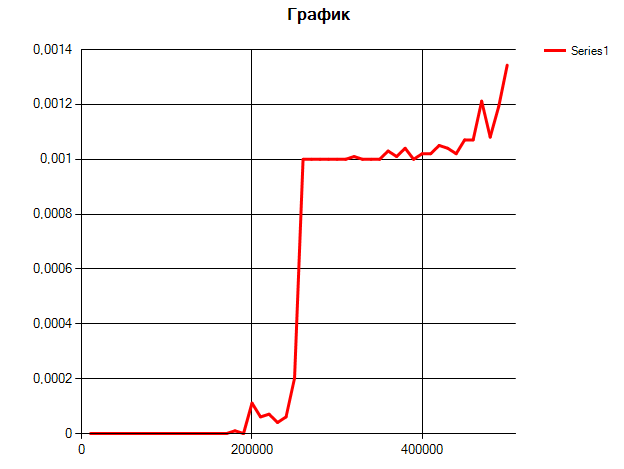


Рис. 6 График затраченного времени на умножение всех элементов

**Полином**

4. Полином: Алгоритм для вычисления полинома. Полином задается массивом коэффициентов, где каждый элемент массива соответствует коэффициенту при соответствующей степени переменной x[см.рисунок 7].

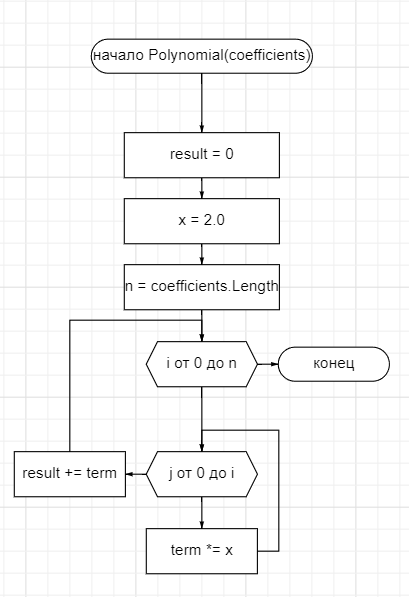


Рис. 7 Блок-схема вычисления полинома

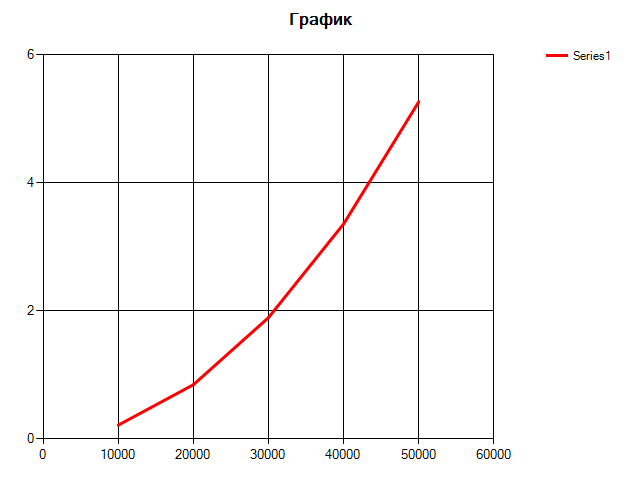


Рис. 8 График затраченного времени на вычисление полинома\

**Bubble sort**

5. Bubble sort: Алгоритм сортировки пузырьком (Bubble Sort) — это простой метод сортировки, который часто используется для обучения основам алгоритмов. Он работает путем многократного прохода по массиву и сравнения соседних элементов, меняя их местами, если они расположены в неправильном порядке[см.рисунок 10]. Этот процесс продолжается до тех пор, пока массив не будет отсортирован.

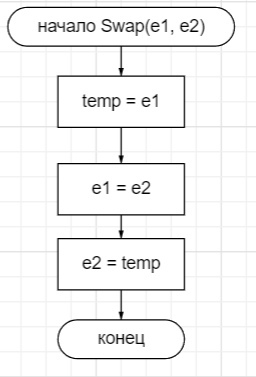


Рис. 9 Блок-схема Swap(e1,e2)

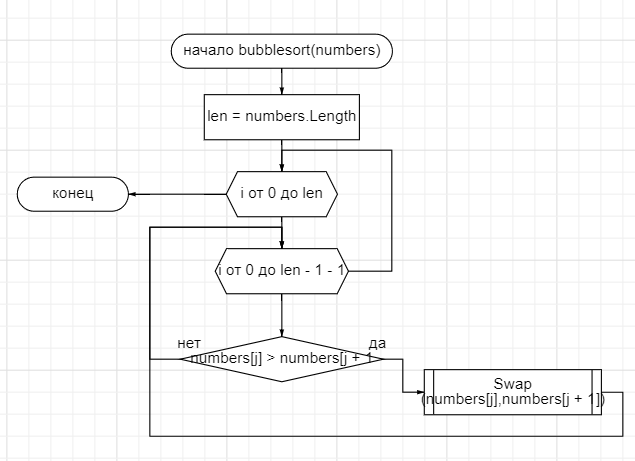


Рис. 10 Блок-схема сортировки Bubble sort

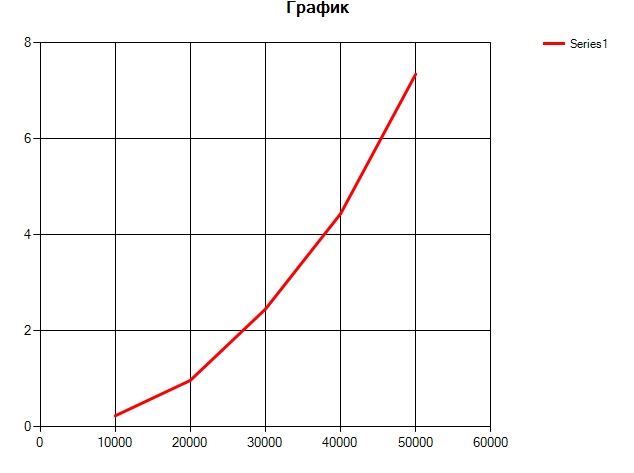


Рис. 11 График затраченного времени на сортировку всех элементов

**Quicksort**

6. Quicksort: это алгоритм сортировки, основанный на методе "разделяй и властвуй". Он работает следующим образом:

1. Выбирается опорный элемент (pivot) из массива. Обычно это первый, последний или средний элемент.
2. Массив разделяется на два подмассива: элементы, меньшие опорного элемента, помещаются перед ним;
3. Рекурсивно применяется quicksort к левому и правому подмассивам.
4. Рекурсия продолжается до тех пор, пока размер подмассива не станет меньше или равен 1.

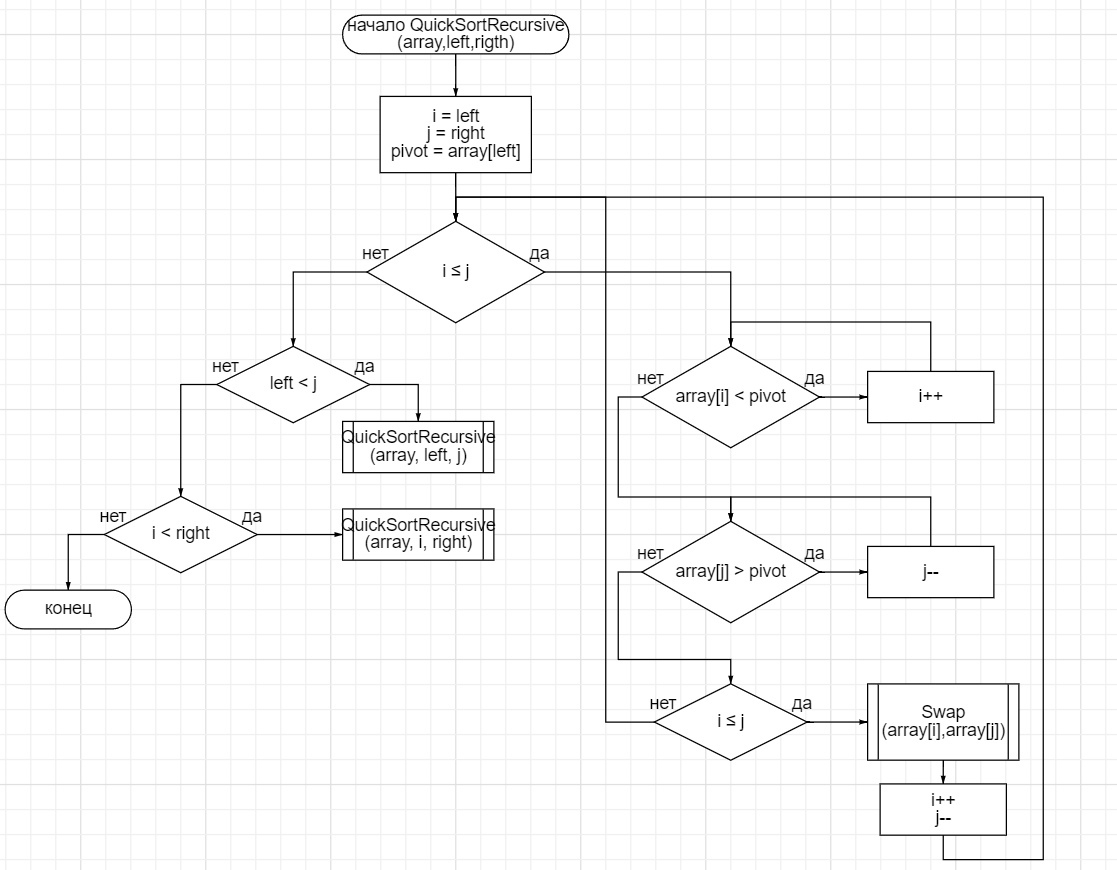


Рис. 12 Блок-схема QuickySort

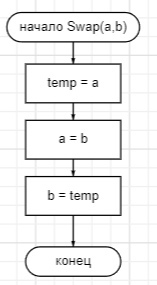


Рис. 13 Блок-схема Swap(a,b)

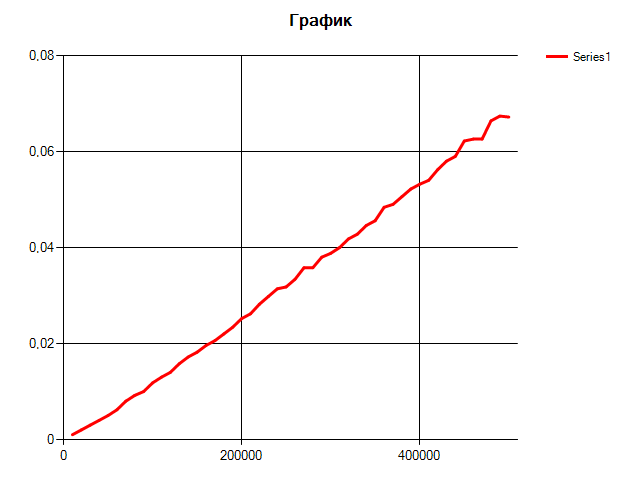


Рис. 14 График затраченного времени на сортировку массива

**Timsort**

7. Timsort: сочетает в себе элементы сортировки вставками и сортировки слиянием, что позволяет ему эффективно работать с реальными данными, которые часто бывают частично упорядочены. Timsort начинает с деления входного массива на небольшие подмассивы. Эти подмассивы должны быть достаточно короткими, чтобы их можно было эффективно отсортировать с помощью сортировки вставками. Обычно минимальный размер подмассива составляет от 32 до 64 элементов[см.рисунок 15]. Если массив меньше этого размера, он сортируется целиком с помощью сортировки вставками.

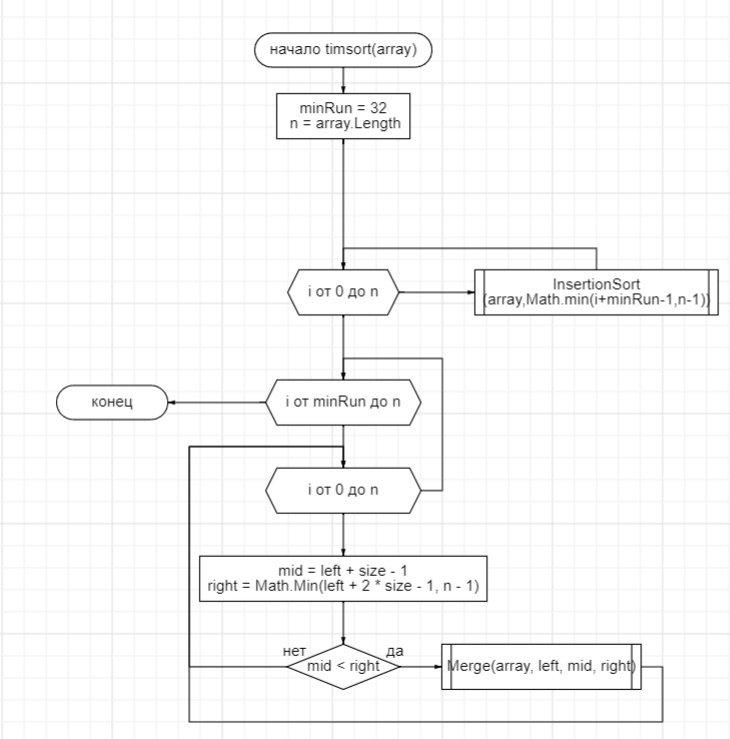


Рис. 15 Блок-схема Timsort сортировки

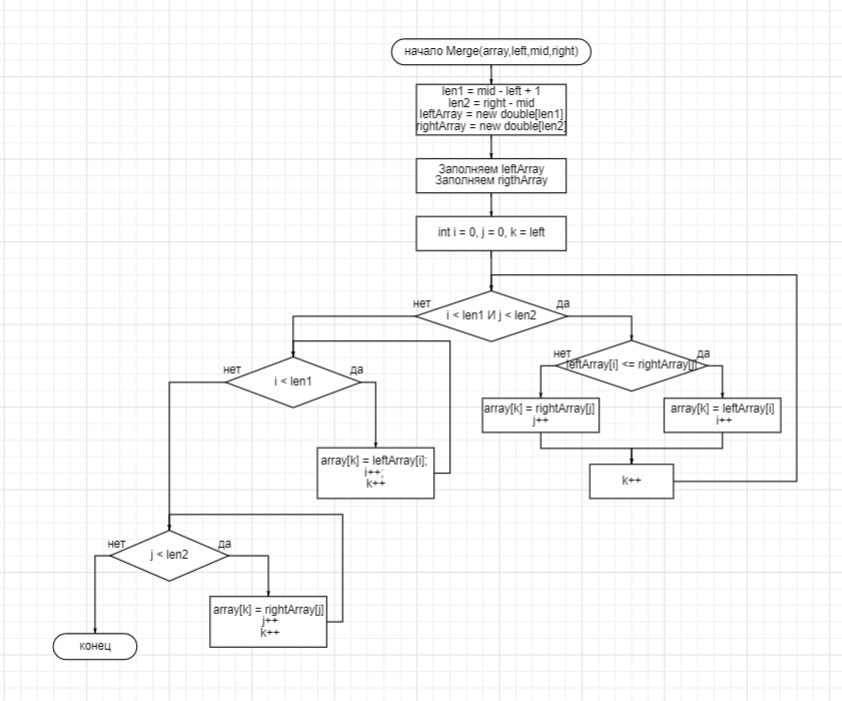


Рис. 16 Блок-схема Merge(array,left,mid,right)

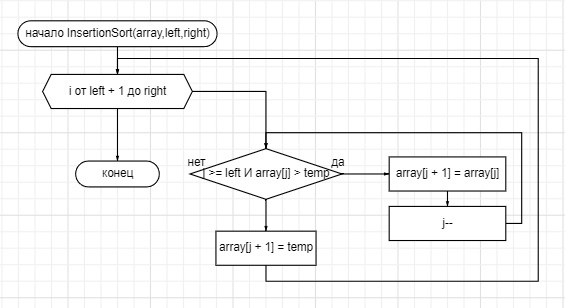


Рис. 17 Блок-схема insertionSort(array,left,right)

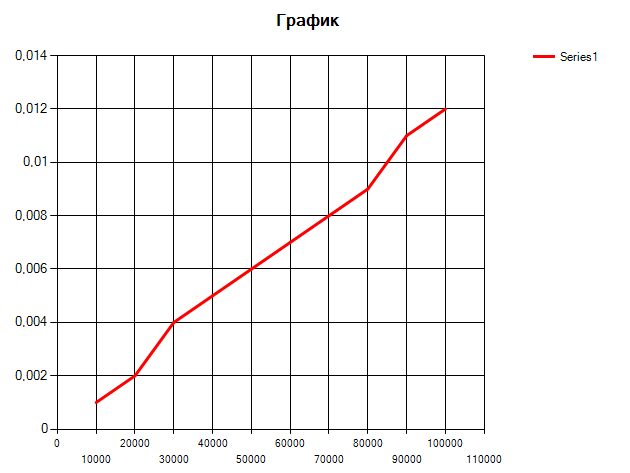


Рис. 18 График затраченного времени на сортировку

8. Алгоритмы возведения в степень: Алгоритмы возведения в степень позволяют эффективно вычислять значения видагде *x* — основание, а *n* — показатель степени. Существует несколько подходов к реализации этих алгоритмов, каждый из которых имеет свои преимущества и недостатки. Список реализованных алгоритмов возведения в степень:

1. PowerNative.
2. PowerRecursive
3. PowerQuickPow
4. PowerQuickPowClassic

**PowerNative**

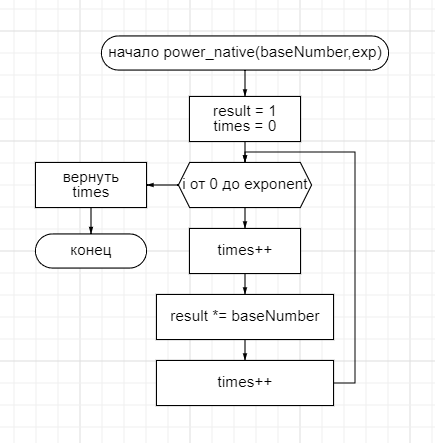


Рис. 19 Блок-схема power\_native(baseNumbers,exp)

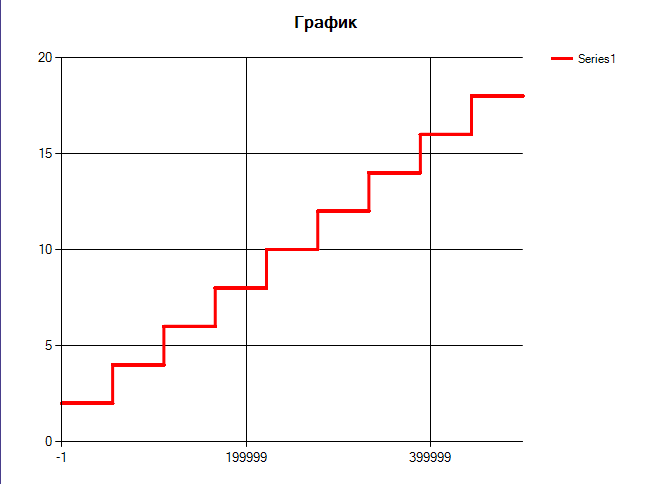


Рис. 20 График затраченных шагов на power\_native

**PowerRecursive**

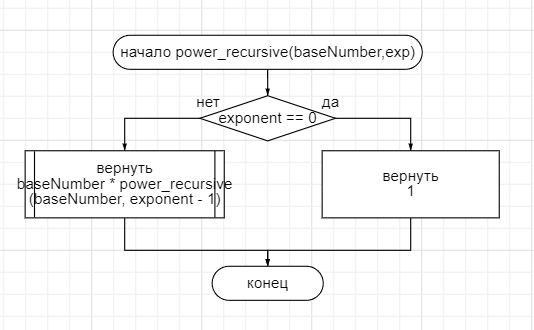


Рис. 21 Блок-схема power\_recursive

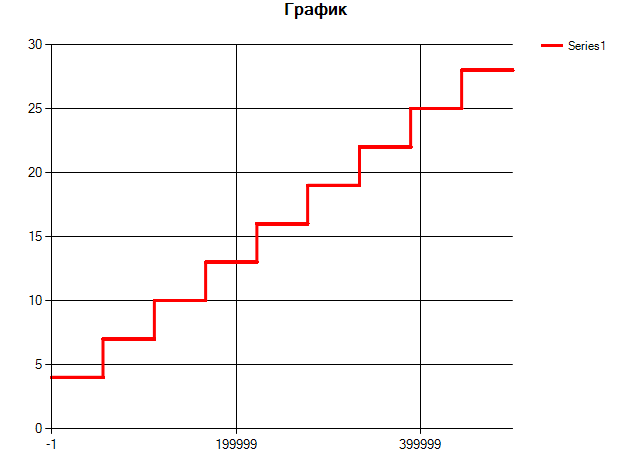


Рис. 22 График количества шагов на возведения в степень

**PowerQuickPow**

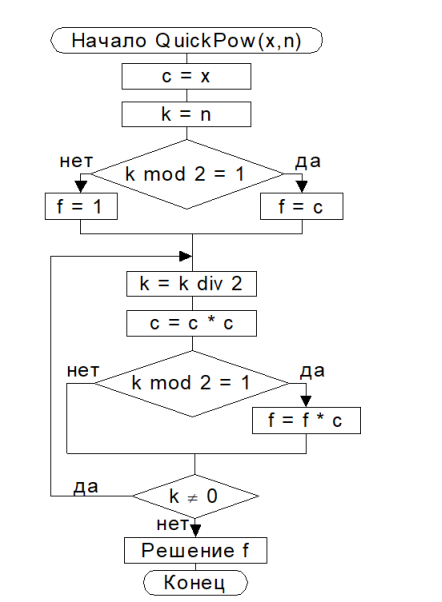
****

Рис. 23 Блок-схема QuickPow(x,n)

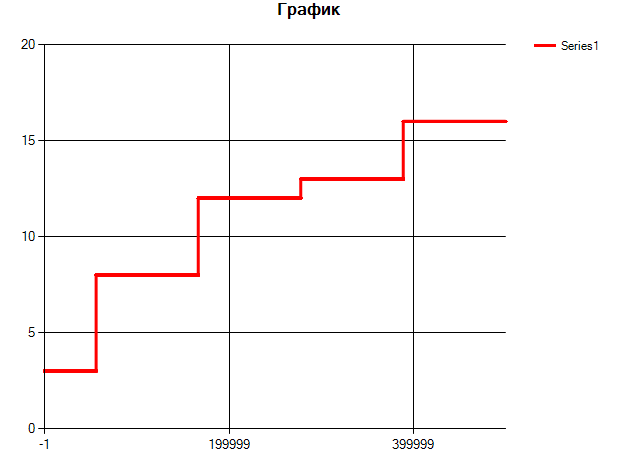


Рис. 24 График затраченных шагов на возведение в степень

**PowerPowClassic**

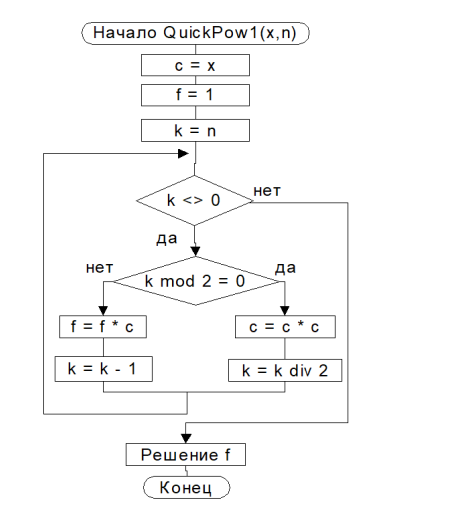
****

Рис. 25 Блок-схема QuickyPow1(x,n)

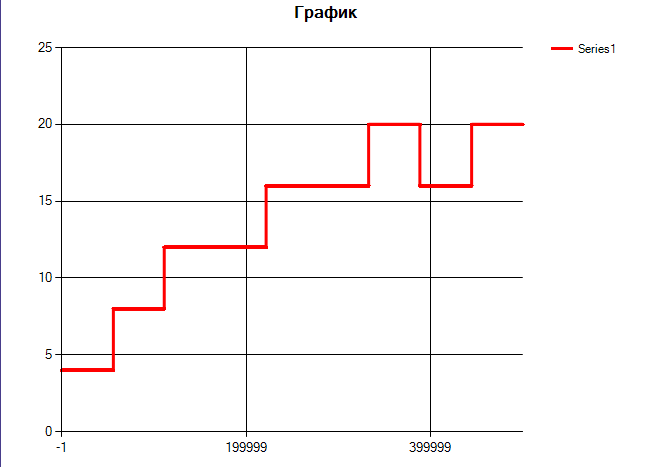


Рис. 26 График затраченных шагов на возведение в степень

# Глава 2

Формулировка задания: Сгенерируйте случайные матрицы A и B размером n×n с неотрицательными элементами. Найдите обычное матричное произведение матриц A и B.

Решение этой задачи состоит из нескольких шагов:

1. Определения размера матрицы.

2. Генерация случайных матриц

3. Вычисление матричного произведения

Блок-схемы наглядно показывают решение этой задачи[]

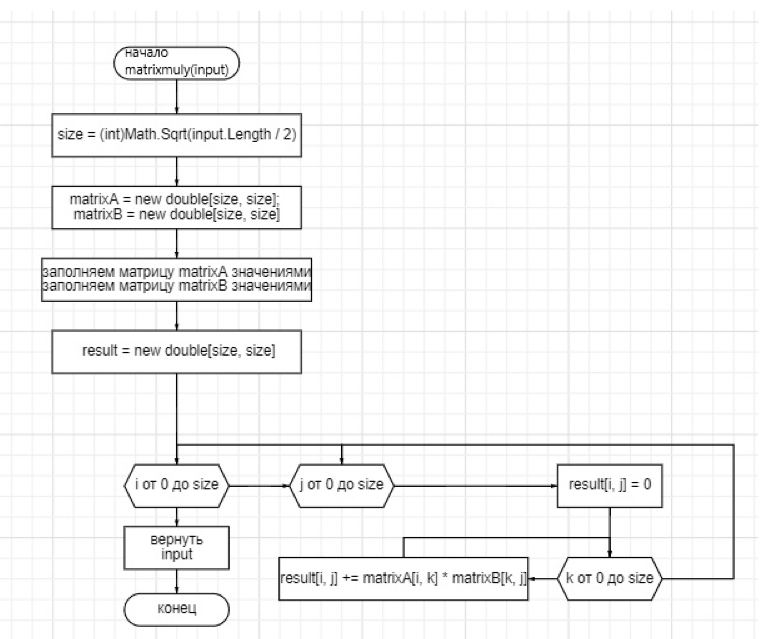


Рис. 27 Блок-схема алгоритма умножения матриц

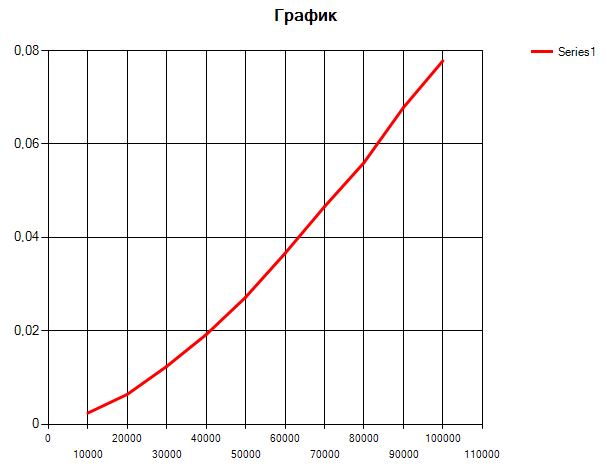


Рис.28 График затраченного времени умножение матриц

**Глава 3**

Формулировка задания: На каждого члена команды найти алгоритм не ниже линейного класса сложности и провести с ним эксперимент.

**Selection Sort**

Сортировка выбором (Selection Sort) — это алгоритм сортировки, который работает по принципу последовательного выбора минимального (или максимального) элемента из неотсортированной части массива и перемещения его в отсортированную часть.

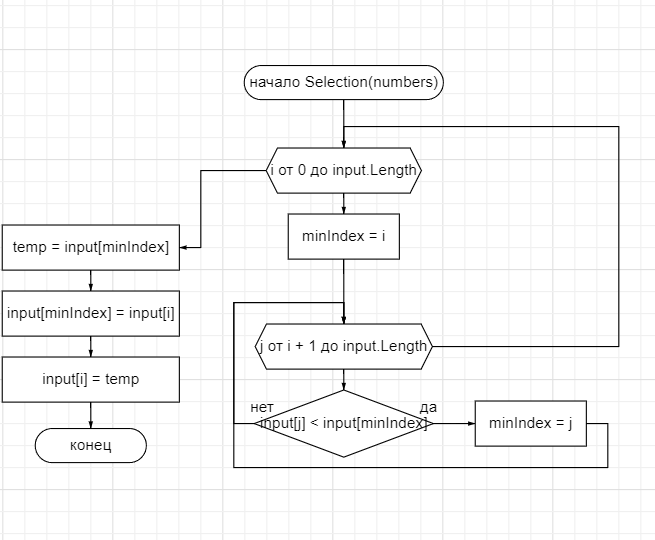


Рис. 29 Блок-схема сортировки выбором

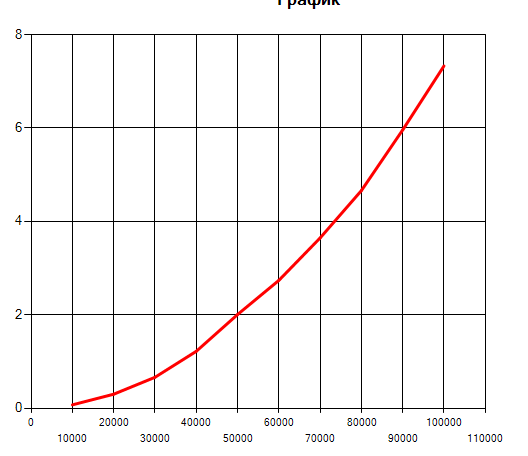


Рис. 30 График затраченного времени сортировку

Сортировка ведрами (Bucket Sort) — это алгоритм сортировки, который основан на предположении о равномерном распределении входных данных. Он разбивает элементы на несколько "ведер" (или корзин), а затем сортирует каждое ведро с помощью другого алгоритма сортировки[см.рисунок 31].

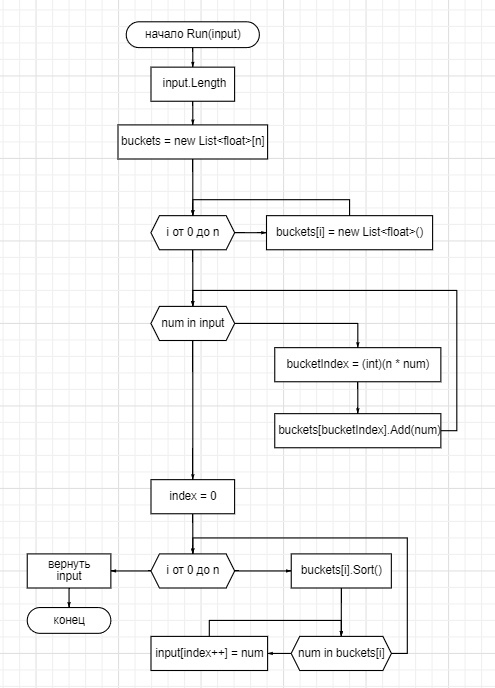


Рис. 31 Блок-схема сумма элементов

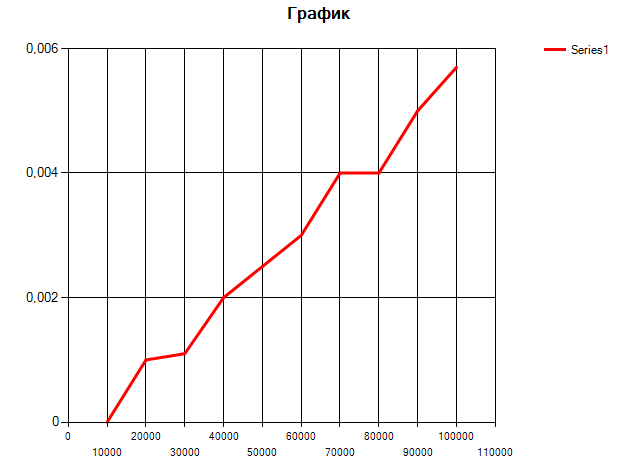


Рис. 32 График затраченного времени сортировку “ведрами”