Федеральное агентство по образованию Российской Федерации
Государственное образовательное учреждение
высшего профессионального образования
Нижегородский государственный университет им. Н.И. Лобачевского
Институт ИИТММ

Отчет по лабораторной работе

“Сравнение сортировок массивов данных”

Выполнил:
студент группы 3823Б1ПР5
Большаков С.А

Проверил:
Лебедев И.Г

Нижний Новгород 2023 г.

Содержание

Постановка задачи

Отсортировать массив из текстового файла, созданного первой программой, используя пузырьковый, быструю сортировку и сортировку методом вставки.

Сравнить время выполнения программы при разных сортировках, сделать соответствующие выводы.

Руководство пользователя

Первая программа создает массив и записывает его в текстовый документ. Вы можете ввести данные массива вручную, из файла, либо задать случайным образом, указав количество элементов, минимальное и максимальное значение. Закончив создавать массив, программа сохранит его в текстовый документ под названием “numbers”.

Во второй программе сначала необходимо указать имя документа, из которого необходимо считать массив (полный путь до файла в проводнике). После этого вам будет доступен консольный интерфейс с вариантами действий над данным массивом. 3 вида сортировки, нахождение норм для массива и его нормировка (разумеется с возможностью вывести итог всех действий на экран). При необходимости, можно выбрать команду “Сброс”, чтобы вернуть массив к изначальному состоянию. По окончании всех операций, вы можете выйти из программы выполнив команду “Выход”.

Руководство программиста

```
#include <stdio.h>
#include <stdlib.h>
 //Генерации случайных чисел
void generateRandomNumbers(int n, int min_value, int max_value, int array[]) {
```

```
    for (int i = 0; i < n; ++i) {
        array[i] = rand() % (max_value - min_value + 1) + min_value;
    }
}
 //Ввод чисел с клавиатуры
void inputNumbersFromKeyboard(int n, int array[]) {
    printf("Введите %d чисел:\n", n);
    for (int i = 0; i < n; ++i) {
        scanf_s("%d", &array[i]);
    }
}
 //Чтение чисел из файла
int inputNumbersFromFile(const char* filename, int array[]) {
    FILE* file;
    if (fopen_s(&file, filename, "r") != 0) {  //Открытие файла
        printf("Ошибка: Не удалось открыть файл %s для чтения.\n", filename);
        return 0;
    }

    int n = 0;
    while (fscanf_s(file, "%d", &array[n]) == 1) {
        n++;
    }

    fclose(file); //Закрытие файла
    return n;
}
 //Запись чисел в файл
void writeNumbersToFile(const char* filename, int n, int array[]) {
    FILE* file;
    if (fopen_s(&file, filename, "w") != 0) { //Открытие файла
        printf("Ошибка: Не удалось открыть файл %s для записи.\n", filename);
        exit(1);
    }

    printf("Запись чисел в файл %s:\n", filename);
    for (int i = 0; i < n; ++i) { //Запись чисел в файл
        fprintf(file, "%d\n", array[i]);
    }
```

```
    fclose(file); //Закрытие файла
}
 //Вывод массива на экран
void printArray(int n, int array[]) {
    printf("Массив чисел:\n");
    for (int i = 0; i < n; ++i) {
        printf("%d ", array[i]);
    }
    printf("\n");
}

int main() {
    int n = 0;
    int min_value, max_value;

    int choice;
    printf("Выберите режим ввода:\n");
    printf("1. Случайные числа\n");
    printf("2. Ввод с клавиатуры\n");
    printf("3. Чтение из файла\n");
    scanf_s("%d", &choice); //Выбор режима ввода

    switch (choice) { //В зависимости от выбранного режима ввода будет выполнена
нужная часть программы
    case 1:

        printf("Введите количество чисел: ");
        scanf_s("%d", &n);

        printf("Введите минимальное значение: ");
        scanf_s("%d", &min_value);

        printf("Введите максимальное значение: ");
        scanf_s("%d", &max_value);



        int* array = (int*)malloc(n * sizeof(int));

        if (array == NULL) {
            printf("Ошибка: Не удалось выделить память для массива.\n");
```

```
            return 1;
        }

        generateRandomNumbers(n, min_value, max_value, array);

        printArray(n, array);


        writeNumbersToFile("numbers.txt", n, array);


        free(array);
        break;
    case 2:

        printf("Введите количество чисел: ");
        scanf_s("%d", &n);


        int* arrayKeyboard = (int*)malloc(n * sizeof(int));

        if (arrayKeyboard == NULL) {
            printf("Ошибка: Не удалось выделить память для массива.\n");
            return 1;
        }

        inputNumbersFromKeyboard(n, arrayKeyboard);
        printArray(n, arrayKeyboard);
        writeNumbersToFile("numbers.txt", n, arrayKeyboard)
        free(arrayKeyboard);
        break;
    case 3:
    {
        char filename[100];
        printf("Введите имя файла: ");
        scanf_s("%s", filename, (unsigned)_countof(filename))
        int arrayFromFile[1000];
        int elementsRead = inputNumbersFromFile(filename, arrayFromFile);
        if (elementsRead > 0)
            printArray(elementsRead, arrayFromFile);
```

```
            writeNumbersToFile("numbers.txt", elementsRead, arrayFromFile);
        }
        else {
            printf("Не удалось считать числа из файла.\n");
        }
        break;
    }
    default:
        printf("Ошибка: Неверный выбор режима ввода.\n");
        return 1;
    }
    return 0;
}
```

Вторая программа

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_SIZE 1000
```

```
// Функция для печати массива
void printArray(double arr[], int size) {
    printf("Текущий массив: ");
    for (int i = 0; i < size; i++) {
        printf("%.3lf ", arr[i]);
    }
    printf("\n");
}
void quickSort(double arr[], int low, int high) { //функция быстрой сортировки
    if (low < high) {

        int pivotIndex = partition(arr, low, high); //Разбиваем массив и получаем опорный элемент
        quickSort(arr, low, pivotIndex - 1); //Сортируем элементы перед и после опорного элемента
        quickSort(arr, pivotIndex + 1, high);
    }
}

// Вспомогательная функция для быстрой сортировки: разбиение массива
int partition(double arr[], int low, int high) {
    double pivot = arr[high];
    int i = low - 1;

    for (int j = low; j <= high - 1; j++) {
        if (arr[j] < pivot) {
            i++;
            // Обмен значениями
            double temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }


    double temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
```

```
}

// Функция для сортировки вставкой
void insertionSort(double arr[], int size) {
    int i, j;
    double key;
    for (i = 1; i < size; i++) {
        key = arr[i];
        j = i - 1;

        // Перемещение элементов массива, которые больше key, на одну позицию вперед
        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

// Функция для сортировки массива (пузырьковая сортировка)
void sortArray(double arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {
                // Обмен значениями
                double temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    printf("Массив отсортирован пузырьковым способом.\n");
}

// Функция для вычисления первой нормы массива
double calculateFirstNorm(double arr[], int size) {
    double norm = 0;
    for (int i = 0; i < size; i++) {
        norm += fabs(arr[i]);
    }
```

```
    return norm;
}

// Функция для вычисления второй нормы массива
double calculateSecondNorm(double arr[], int size) {
    double norm = 0.0;
    for (int i = 0; i < size; i++) {
        norm += pow(arr[i], 2);
    }
    return sqrt(norm);
}

// Функция для вычисления бесконечной нормы массива
double calculateInfinityNorm(double arr[], int size) {
    double norm = 0;
    for (int i = 0; i < size; i++) {
        if (fabs(arr[i]) > norm) {
            norm = fabs(arr[i]);
        }
    }
    return norm;
}

// Функция для нормировки массива
void normalizeArray(double arr[], int size) {
    double norm = calculateSecondNorm(arr, size);
    if (norm != 0) {
        for (int i = 0; i < size; i++) {
            arr[i] /= norm;
        }
        printf("Массив нормирован.\n");
    }
    else {
        printf("Норма вектора равна нулю, невозможно выполнить нормировку.\n");
    }
}

// Функция для сброса массива к исходному состоянию
void resetArray(double original[], double current[], int size) {
    for (int i = 0; i < size; i++) {
```

```
        current[i] = original[i];
    }
    printf("Массив сброшен к исходному состоянию.\n");
}

int main() {
    double originalNumbers[MAX_SIZE]; // Исходный массив
    double currentNumbers[MAX_SIZE];  // Текущий массив, над которым выполняются действия
    int size = 0;
    char filename[100];
    char line[100];

    // Запрос имени файла
    printf("Введите имя файла: ");
    scanf_s("%s", filename, (unsigned)_countof(filename));

    // Чтение чисел из файла
    FILE* file;
    if (fopen_s(&file, filename, "r") != 0) {
        printf("Ошибка открытия файла.\n");
        return 1;
    }

    while (fgets(line, sizeof(line), file) != NULL) {
        sscanf_s(line, "%lf", &originalNumbers[size]);
        currentNumbers[size] = originalNumbers[size];  // Сохраняем копию исходного массива
        size++;
    }

    fclose(file);

    // Главный цикл программы
    while (1) {
        int choice;

        // Вывод меню
        printf("\nВыберите действие:\n");
        printf("1. Печать\n");
        printf("2. Сортировка пузырьком\n");
```

```
        printf("3. Сброс\n");
        printf("4. Вычислить первую норму\n");
        printf("5. Вычислить вторую норму\n");
        printf("6. Вычислить бесконечную норму\n");
        printf("7. Нормировать вектор\n");
        printf("8. Быстрая сортировка\n");
        printf("9. Сортировка вставкой\n");
        printf("10. Выход\n");
        printf("Ваш выбор: ");
        scanf_s("%d", &choice); //Выбор необходимого действия

        switch (choice) {
        case 1:
            printArray(currentNumbers, size);
            break;
        case 2:
            sortArray(currentNumbers, size);
            break;
        case 3:
            resetArray(originalNumbers, currentNumbers, size); // Используем функцию сброса
            break;
        case 4:
            printf("Первая норма массива: %.3lf\n", calculateFirstNorm(currentNumbers, size));
            break;
        case 5:
            printf("Вторая норма массива: %.3lf\n", calculateSecondNorm(currentNumbers,
size));
            break;
        case 6:
            printf("Бесконечная норма массива: %.3lf\n",
calculateInfinityNorm(currentNumbers, size));
            break;
        case 7:
            normalizeArray(currentNumbers, size);
            break;
        case 8:
            quickSort(currentNumbers, 0, size - 1);
            printf("Массив отсортирован с использованием быстрой сортировки.\n");
            break;
        case 9:
            insertionSort(currentNumbers, size);
```

```
            printf("Массив отсортирован с использованием сортировки вставкой.\n");
            break;
        case 10:
            printf("Программа завершена.\n");
            return 0;
        default:
            printf("Некорректный выбор. Повторите попытку.\n");
        }
    }

    return 0;
}
```

Заключение.

Я пронаблюдал за выполнением сортировки в различных условиях, и могу сделать некоторые выводы: Пузырьковая сортировка - самая медленная, с увеличением размера массива время выполнения будет расти больше, чем для остальных видов. Сортировка вставками лучше справляется с поставленной задачей, а также при некоторых обстоятельствах (если

соседние элементы массива не слишком сильно друг от друга отличаются) выполняет работу быстрее всех остальных. Быстрая сортировка, как ни странно, является самой быстрой, потому что работает, не сравнивая поочередно 2 элемента. В большинстве программ, особенно сложных, лучше всего подойдет именно она, позволяя стабильно получать хороший результат (быстрое выполнение программы)

Литература

Липачѐв Е.К. Технология программирования. Базовые конструкции языка C/C++. – Казань: Казанский университет, 2012

Хэзфилд Р., Кирби Л. И др. Искусство программирования на C. Фундаментальные алгоритмы, структуры данных и примеры

приложений. Энциклопедия программиста. – К.: Издательство «ДиаСофт», 2001.

Фридман А., Кландер Л., Михаэлис М., Шильдт Х. C/C++. Архив программ

Павловская Т.а c/c++ программирование на языке высокого уровня

Приложения

Первая программа

```
#include <stdio.h>
#include <stdlib.h>


void generateRandomNumbers(int n, int min_value, int max_value, int array[]) {
   for (int i = 0; i < n; ++i) {
      array[i] = rand() % (max_value - min_value + 1) + min_value;
```

```
    }
}

void inputNumbersFromKeyboard(int n, int array[]) {
    printf("Введите %d чисел:\n", n);
    for (int i = 0; i < n; ++i) {
        scanf_s("%d", &array[i]);
    }
}

int inputNumbersFromFile(const char* filename, int array[]) {
    FILE* file;
    if (fopen_s(&file, filename, "r") != 0) {
        printf("Ошибка: Не удалось открыть файл %s для чтения.\n", filename);
        return 0;
    }

    int n = 0;
    while (fscanf_s(file, "%d", &array[n]) == 1) {
        n++;
    }

    fclose(file);
    return n;
}

void writeNumbersToFile(const char* filename, int n, int array[]) {
    FILE* file;
    if (fopen_s(&file, filename, "w") != 0) {
        printf("Ошибка: Не удалось открыть файл %s для записи.\n", filename);
        exit(1);
    }

    printf("Запись чисел в файл %s:\n", filename);
    for (int i = 0; i < n; ++i) {
        fprintf(file, "%d\n", array[i]);
    }

    fclose(file);
}

void printArray(int n, int array[]) {
    printf("Массив чисел:\n");
    for (int i = 0; i < n; ++i) {
        printf("%d ", array[i]);
    }
    printf("\n");
}

int main() {
    int n = 0;
    int min_value, max_value;

    int choice;
    printf("Выберите режим ввода:\n");
    printf("1. Случайные числа\n");
    printf("2. Ввод с клавиатуры\n");
    printf("3. Чтение из файла\n");
    scanf_s("%d", &choice);
```

```
switch (choice) {
case 1:

    printf("Введите количество чисел: ");
    scanf_s("%d", &n);

    printf("Введите минимальное значение: ");
    scanf_s("%d", &min_value);

    printf("Введите максимальное значение: ");
    scanf_s("%d", &max_value);



    int* array = (int*)malloc(n * sizeof(int));

    if (array == NULL) {
        printf("Ошибка: Не удалось выделить память для массива.\n");
        return 1;
    }

    generateRandomNumbers(n, min_value, max_value, array);

    printArray(n, array);



    writeNumbersToFile("numbers.txt", n, array);



    free(array);
    break;
case 2:

    printf("Введите количество чисел: ");
    scanf_s("%d", &n);



    int* arrayKeyboard = (int*)malloc(n * sizeof(int));

    if (arrayKeyboard == NULL) {
        printf("Ошибка: Не удалось выделить память для массива.\n");
        return 1;
    }

    inputNumbersFromKeyboard(n, arrayKeyboard);
    printArray(n, arrayKeyboard);



    writeNumbersToFile("numbers.txt", n, arrayKeyboard);



    free(arrayKeyboard);
    break;
case 3:
{
    char filename[100];
```

```
        printf("Введите имя файла: ");
        scanf_s("%s", filename, (unsigned)_countof(filename));


        int arrayFromFile[1000];

        int elementsRead = inputNumbersFromFile(filename, arrayFromFile);
        if (elementsRead > 0) {

            printArray(elementsRead, arrayFromFile);



            writeNumbersToFile("numbers.txt", elementsRead, arrayFromFile);
        }
        else {
            printf("Не удалось считать числа из файла.\n");
        }
        break;
    }
    default:
        printf("Ошибка: Неверный выбор режима ввода.\n");
        return 1;
    }

    return 0;
}
```

Вторая программа

```
 #include <stdio.h>
#include <stdlib.h>
#include <math.h>

#define MAX_SIZE 1000



void printArray(double arr[], int size) {
    printf("Текущий массив: ");
    for (int i = 0; i < size; i++) {
        printf("%.3lf ", arr[i]);
    }
    printf("\n");
}
void quickSort(double arr[], int low, int high) {
    if (low < high) {

        int pivotIndex = partition(arr, low, high);



        quickSort(arr, low, pivotIndex - 1);
        quickSort(arr, pivotIndex + 1, high);
    }
}



int partition(double arr[], int low, int high) {
    double pivot = arr[high];
    int i = low - 1;

    for (int j = low; j <= high - 1; j++) {
```

```
        if (arr[j] < pivot) {
            i++;

            double temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }

    double temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;

    return i + 1;
}

void insertionSort(double arr[], int size) {
    int i, j;
    double key;
    for (i = 1; i < size; i++) {
        key = arr[i];
        j = i - 1;

        while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
    }
}

void sortArray(double arr[], int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (arr[j] > arr[j + 1]) {

                double temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
    printf("Массив отсортирован пузырьковым способом.\n");
}

double calculateFirstNorm(double arr[], int size) {
    double norm = 0;
    for (int i = 0; i < size; i++) {
        norm += fabs(arr[i]);
    }
    return norm;
}

double calculateSecondNorm(double arr[], int size) {
    double norm = 0.0;
    for (int i = 0; i < size; i++) {
        norm += pow(arr[i], 2);
```

```
    }
    return sqrt(norm);
}



double calculateInfinityNorm(double arr[], int size) {
    double norm = 0;
    for (int i = 0; i < size; i++) {
        if (fabs(arr[i]) > norm) {
            norm = fabs(arr[i]);
        }
    }
    return norm;
}



void normalizeArray(double arr[], int size) {
    double norm = calculateSecondNorm(arr, size);
    if (norm != 0) {
        for (int i = 0; i < size; i++) {
            arr[i] /= norm;
        }
        printf("Массив нормирован.\n");
    }
    else {
        printf("Норма вектора равна нулю, невозможно выполнить нормировку.\n");
    }
}



void resetArray(double original[], double current[], int size) {
    for (int i = 0; i < size; i++) {
        current[i] = original[i];
    }
    printf("Массив сброшен к исходному состоянию.\n");
}

int main() {
    double originalNumbers[MAX_SIZE];
    double currentNumbers[MAX_SIZE];
    int size = 0;
    char filename[100];
    char line[100];



    printf("Введите имя файла: ");
    scanf_s("%s", filename, (unsigned)_countof(filename));



    FILE* file;
    if (fopen_s(&file, filename, "r") != 0) {
        printf("Ошибка открытия файла.\n");
        return 1;
    }

    while (fgets(line, sizeof(line), file) != NULL) {
        sscanf_s(line, "%lf", &originalNumbers[size]);
        currentNumbers[size] = originalNumbers[size];
        size++;
    }

    fclose(file);
```

```
    while (1) {
        int choice;



        printf("\nВыберите действие:\n");
        printf("1. Печать\n");
        printf("2. Сортировка пузырьком\n");
        printf("3. Сброс\n");
        printf("4. Вычислить первую норму\n");
        printf("5. Вычислить вторую норму\n");
        printf("6. Вычислить бесконечную норму\n");
        printf("7. Нормировать вектор\n");
        printf("8. Быстрая сортировка\n");
        printf("9. Сортировка вставкой\n");
        printf("10. Выход\n");
        printf("Ваш выбор: ");
        scanf_s("%d", &choice);

        switch (choice) {
        case 1:
            printArray(currentNumbers, size);
            break;
        case 2:
            sortArray(currentNumbers, size);
            break;
        case 3:
            resetArray(originalNumbers, currentNumbers, size);
            break;
        case 4:
            printf("Первая норма массива: %.3lf\n", calculateFirstNorm(currentNumbers, size));
            break;
        case 5:
            printf("Вторая норма массива: %.3lf\n", calculateSecondNorm(currentNumbers, size));
            break;
        case 6:
            printf("Бесконечная норма массива: %.3lf\n", calculateInfinityNorm(currentNumbers, size));
            break;
        case 7:
            normalizeArray(currentNumbers, size);
            break;
        case 8:
            quickSort(currentNumbers, 0, size - 1);
            printf("Массив отсортирован с использованием быстрой сортировки.\n");
            break;
        case 9:
            insertionSort(currentNumbers, size);
            printf("Массив отсортирован с использованием сортировки вставкой.\n");
            break;
        case 10:
            printf("Программа завершена.\n");
            return 0;
        default:
            printf("Некорректный выбор. Повторите попытку.\n");
        }
    }

    return 0;
}
```