

|  |  |
| --- | --- |
| МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ  федеральное государственное бюджетное образовательное учреждение высшего образования  **«МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ ГОСУДАРСТВЕННЫЙ**  **ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)»** |  |

**Кафедра «Высшая математика»**

**Лабораторная работа № 3**

по дисциплине

«Структуры и алгоритмы обработки данных»

на тему:

**«**Сортировки и определение характеристик одномерных массивов**»**

**Выполнил:**

Учебная группа: 1бПМ   
ФИО: Греча К. П.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Руководитель лабораторной работы:**

Должность: старший преподаватель

Звание: б/з

ФИО: Кутейников И. А.

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_г.

**Цель:**

Требуется определить массив целых чисел (например, размера 30),

заполнить его случайными числами (в диапазоне от A до B) или ввести его

элементы с клавиатуры и определить его характеристики в соответствии с

вариантом (Среди всех симметричных чисел найти наиболее длинное (по

количеству цифр).

1 Инициализация элементов массива числами, вводимыми с

клавиатуры;

2 Инициализация элементов массива случайными числами;

3 Вывод массива на экран;

4 Сортировка массива;

4.1 Сортировка в соответствии с вариантом;

4.2 Собственная реализация быстрой сортировки;

4.3 Сортировка встроенной функцией сортировки;

4.4 Сравнение времени работы сортировок;

5 Решение задачи уровня Б;

6 Решение задачи уровня С (опционально);

7 Выход из программы.

Реализовать заданный метод сортировки числового массива в

соответствии с индивидуальным заданием. Для всех вариантов добавить

реализацию быстрой сортировки (quicksort). Оценить время работы каждого

алгоритма сортировки и сравнить его со временем стандартной функции

сортировки, используемой в выбранном языке программирования.

**Алгоритм:**

import java.util.Arrays;  
import java.util.Scanner;  
import java.util.Random;  
public class Main {  
 static int SizeEnter() {  
 int size = 0;  
 Scanner in = new Scanner(System.*in*);  
 do {  
 System.*out*.print("\n Введите размер массива: ");  
 size = in.nextInt();  
 if (size < 1) {  
 System.*out*.println("Должен быть > 0");  
 }  
 } while (size < 1);  
 return size;  
 }  
// метод для ручного ввода:  
 static int[] fillByHand(int[] array) {  
 Scanner in = new Scanner(System.*in*);  
 int size = *SizeEnter*();  
 array = new int[size];  
 for (int i = 0; i < array.length; i++) {  
 System.*out*.print("Введите элемент № " + (i + 1) + ": ");  
 array[i] = in.nextInt();  
 }  
 return array;  
 }  
// Случайный выбор  
 static int[] fillByRandom(int[] array) {  
 int size = *SizeEnter*();  
 Scanner in = new Scanner(System.*in*);  
 Random rand = new Random();  
 int MaxElVal = 0;  
 int MinElVal = 0;  
 do {  
 System.*out*.print("\nВведите максимальное значение элемента: ");  
 MaxElVal = in.nextInt();  
 if (MaxElVal < 1) {  
 System.*out*.println("Максимальное значение элемента должно быть > 0");  
 }  
 }  
 while (MaxElVal < 1);  
  
 do {  
 System.*out*.print("\nВведите минимальное значение элемента: ");  
 MinElVal = in.nextInt();  
 if (MinElVal >= 0) {  
 System.*out*.println("Минимальное значение элемента должно быть < 0");  
 }  
 }  
 while (MinElVal >= 0);  
 array = new int[size];  
 for (int i = 0; i < array.length; i++) {  
 array[i] = rand.nextInt(MaxElVal - MinElVal + 1) + MinElVal;  
 }  
 return array;  
 }  
  
 //Case 4 :  
 static void printArray(int[] array) {  
 for (int num : array) {  
 System.*out*.print(num + " ");  
 }  
 System.*out*.println("Отсортированный массив: ");  
 }  
  
//Сортировка Шелла:  
 static int[] shellSort(int[] array) {  
 int n = array.length;  
 int d = n / 2;  
  
 while (d > 0) {  
 for (int i = d; i < n; i++) {  
 int temp = array[i];  
 int j = i;  
  
 while (j >= d && array[j - d] > temp) {  
 array[j] = array[j - d];  
 j -= d;  
 }  
  
 array[j] = temp;  
 }  
 d /= 2;  
 }  
 return array;  
 }  
//Быстрая сортировка:  
  
 public static int[] quickSort(int[] array) {  
 if (array.length <= 1) {  
 return array;  
 }  
 int key = array[array.length - 1];  
 int i = -1;  
  
 for (int j = 0; j < array.length - 1; j++) {  
 if (array[j] < key) {  
 i++;  
 int temp = array[i];  
 array[i] = array[j];  
 array[j] = temp;  
 }  
 }  
 int temp = array[i + 1];  
 array[i + 1] = array[array.length - 1];  
 array[array.length - 1] = temp;  
  
 int[] leftPart = *quickSort*(Arrays.*copyOfRange*(array, 0, i + 2 - 1 ));  
 int[] rightPart = *quickSort*(Arrays.*copyOfRange*(array, i + 2, array.length));  
//Элементы из лефтпарт копируются в массив начиная с 0:  
 for (int j = 0; j < leftPart.length; j++) {  
 array[j] = leftPart[j];  
 }  
//Опорный элемент перемещается на позицию лефтпарт:  
 array[leftPart.length] = key;  
//Элементы из райтпарт копируются в array начиная с лефтпарт.ленгхт + 1:  
 for (int j = 0; j < rightPart.length; j++) {  
 array[leftPart.length + 1 + j] = rightPart[j];  
 }  
  
 return array;  
 }  
// Сравнение времени сортировок:  
static void SortTimes(int[] array) {  
 int[] array1 = Arrays.*copyOf*(array, array.length);  
 int[] array2 = Arrays.*copyOf*(array, array.length);  
 int[] array3 = Arrays.*copyOf*(array, array.length);  
  
 long startTime, endTime, shellSortTime, quickSortTime, builtinSortTime;  
  
 startTime = System.*nanoTime*();  
 *shellSort*(array1);  
 endTime = System.*nanoTime*();  
 shellSortTime = endTime - startTime;  
  
 startTime = System.*nanoTime*();  
 *quickSort*(array2);  
 endTime = System.*nanoTime*();  
 quickSortTime = endTime - startTime;  
  
 startTime = System.*nanoTime*();  
 Arrays.*sort*(array3);  
 endTime = System.*nanoTime*();  
 builtinSortTime = endTime - startTime;  
  
 System.*out*.println("Время выполнения сортировки Шелла: " + shellSortTime + " нс");  
 System.*out*.println("Время выполнения быстрой сортировки: " + quickSortTime + " нс");  
 System.*out*.println("Время выполнения встроенной сортировки: " + builtinSortTime + " нс");  
 }  
  
  
 //Case 5:  
 // Нахождение симметрии в числах  
 static boolean Symmetric(int num) {  
 int newNum=num;  
 int reverseNum = 0;  
 while (num != 0) {  
 int remainder = num % 10;  
 reverseNum = reverseNum \* 10 + remainder;  
 num = num / 10;  
 }  
 return newNum == reverseNum;  
 }  
 static int LongSymmetricNumber(int[] array){  
 int longestSymmetricNum=0;  
 for(int num:array) {  
 if (*Symmetric*(num) && num > longestSymmetricNum) {  
 longestSymmetricNum = num;  
 }  
 }  
//После завершения цикла возвращается значение longestSymmetricNum:  
 return longestSymmetricNum;  
}  
  
  
  
  
  
  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 int ans, local\_ans;  
 int []array = new int[0];  
 int elem, x;  
 do  
 {  
 System.*out*.println("\nМеню");  
 System.*out*.println("1. Создайте массив");  
 System.*out*.println("2. Вывести массив на экран");  
 System.*out*.println("3. Изменить элемент массива");  
 System.*out*.println("4. Сортировка массива");  
 System.*out*.println("5. Выполнить задачу №10 ");  
 System.*out*.println("6. Выход");  
 System.*out*.print("7. Выбрать пункт меню: ");  
 ans = in.nextInt();  
 switch (ans)  
 {  
 case 1:  
 do  
 {  
 System.*out*.println("Пукнт меню 1: ");  
 System.*out*.println("1. Заполнить в ручную");  
 System.*out*.println("2. Заполнить случайно");  
 System.*out*.println("3. Вернуться назад");  
 System.*out*.print("Выбрать пункт меню: ");  
 local\_ans = in.nextInt();  
 switch (local\_ans)  
 {  
 case 1:  
 array = *fillByHand*(array);  
 break;  
 case 2:  
 array = *fillByRandom*(array);  
 break;  
 case 3:  
 System.*out*.println("Назад");  
 break;  
 default:  
 System.*out*.println("Неверный ввод");  
 break;  
 }  
 }  
 while (local\_ans!=3);  
 break;  
  
  
 case 2:  
 System.*out*.println("Меню пункт 2: ");  
 if (array.length == 0)  
 {  
 System.*out*.println("\n Массив не существует");  
 break;  
 }  
 else {  
 for (int i = 0; i < array.length; i++)  
 {  
 System.*out*.print("\nЭлемент № " + (i + 1) + " = " + array[i]);  
 }  
 }  
 break;  
  
  
 case 3:  
 System.*out*.println("\n Меню пункт 3: ");  
 if (array.length == 0)  
 {  
  
 System.*out*.println("\n Массив не существует");  
 break;  
 }  
 else  
 {  
 do  
 {  
 System.*out*.println("Элемента массива который хотите изменить: ");  
 elem = in.nextInt();  
 if ((elem > array.length) | (elem < 0))  
 {  
 System.*out*.println("Такого элемента не существует");  
 }  
 }  
 while ((elem - 1 < 0) | (elem - 1 > array.length));  
 System.*out*.print("Введено новое значение элемента: ");  
 array[elem - 1] = in.nextInt();  
 }  
 break;  
  
  
 case 4:  
 do {  
 System.*out*.println("Пункт меню 4: ");  
 System.*out*.println("1. Сортировка Шелла: ");  
 System.*out*.println("2. Быстрая сортировка: ");  
 System.*out*.println("3. Встроенная сортировка: ");  
 System.*out*.println("4. Сравнение времени сортировок: ");  
 System.*out*.println("5. Вернуться назад");  
 System.*out*.print("Выбрать пункт меню: ");  
 local\_ans = in.nextInt();  
 switch (local\_ans) {  
 case 1:  
 int[] shellSortedArray = *shellSort*(Arrays.*copyOf*(array, array.length));  
 System.*out*.print("Отсортированный массив с помощью сортировки Шелла: ");  
 *printArray*(shellSortedArray);  
 break;  
  
 case 2:  
 int[] quickSortedArray = *quickSort*(Arrays.*copyOf*(array, array.length));  
 System.*out*.println("Массив отсортирован с помощью быстрой сортировки");  
 *printArray*(quickSortedArray);  
 break;  
 case 3:  
 int[] builtinSortedArray = Arrays.*copyOf*(array, array.length);  
 Arrays.*sort*(builtinSortedArray);  
 System.*out*.println("Массив отсортирован с помощью встроенной сортировки");  
 *printArray*(builtinSortedArray);  
 break;  
 case 4:  
 *SortTimes*(array);  
 break;  
 case 5:  
 System.*out*.println("Назад ");  
 break;  
 default:  
 System.*out*.println("Неверный пункт меню");  
 }  
 }  
 while(local\_ans !=5);  
 break;  
 case 5:  
 int longestSymmetricNum = *LongSymmetricNumber*(array);  
 System.*out*.println("Наиболее длинное симметричное число: " + longestSymmetricNum);  
 break;  
  
 case 6:  
 System.*out*.println("Выход");  
 break;  
 default:  
 System.*out*.println("ОШИБКА ввода");  
 }  
  
 }  
 while(ans!=6);  
 }  
}

**Результат:**

Ввод:

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Вывод:

Выбрать пункт меню: 2

Введите размер массива: 10

Введите максимальное значение элемента: 100

Введите минимальное значение элемента: -100

Пункт меню 1:

1. Заполнить в ручную

2. Заполнить случайно

3. Вернуться назад

Выбрать пункт меню: 3

Назад

Меню

1. Создайте массив

2. Вывести массив на экран

3. Изменить элемент массива

4. Сортировка массива

5. Выполнить задачу №10

6. Выход

7. Выбрать пункт меню: 3

Меню пункт 2:

Элемент № 1 = 99

Элемент № 2 = 17

Элемент № 3 = -51

Элемент № 4 = 3

Элемент № 5 = -46

Элемент № 6 = -80

Элемент № 7 = -84

Элемент № 8 = 98

Элемент № 9 = -72

Элемент № 10 = 89

7. Выбрать пункт меню: 4

Пункт меню 4:

1. Сортировка Шелла:

2. Быстрая сортировка:

3. Встроенная сортировка:

4. Сравнение времени сортировок:

5. Вернуться назад

Выбрать пункт меню: 1

Отсортированный массив с помощью сортировки Шелла: -84 -80 -72 -51 -46 3 17 89 98 99

Выбрать пункт меню: 2

Массив отсортирован с помощью быстрой сортировки

-84 -80 -72 -51 -46 3 17 89 98 99

Выбрать пункт меню: 3

Массив отсортирован с помощью встроенной сортировки

-84 -80 -72 -51 -46 3 17 89 98 99

Выбрать пункт меню: 4

Время выполнения сортировки Шелла: 3400 нс

Время выполнения быстрой сортировки: 9400 нс

Время выполнения встроенной сортировки: 4200 нс

Меню

1. Создайте массив

2. Вывести массив на экран

3. Изменить элемент массива

4. Сортировка массива

5. Выполнить задачу №10

6. Выход

7. Выбрать пункт меню: 5

**Наиболее длинное симметричное число: 99**

**Заключение:**

В процессе выполнения лабораторной работы было освоено умение пользоваться switch. Так же было приобретено умение работать с массивами, сортировать их, были изучены различные методы сортировки массива. Также были улучшены умения применять операторы цикла, изменять элементы массива, выводить на экран массивы.