**Отчёт для ЛР6.  
Бригада: Давыдов Александр**

*Давыдов – 7 букв*

*1 вариант (Алгоритм сортировки вставкой)*

Начиная со второго элемента, каждый элемент вставляется в отсортированную часть массива так, чтобы все элементы в отсортированной части оставались отсортированными.

Для вставки элемента находится подходящая позиция в отсортированной части путем сдвига всех больших элементов вправо.

Этот процесс повторяется для каждого элемента массива, пока весь массив не будет отсортирован.

*Структура №1. Массив (ArrayList)*

Числа от 1 до 100:

|  |  |
| --- | --- |
| Число элементов | Время, с |
| 100 | 0.0013788 |
| 1000 | 0.020963 |
| 10000 | 0.1726315 |
| 100000 | 11.4216224 |

Числа от 1 до 10:

|  |  |
| --- | --- |
| Число элементов | Время, с |
| 100 | 0.0013788 |
| 1000 | 0.020963 |
| 10000 | 0.1726315 |
| 100000 | 11.4216224 |

Код программы:  
  
import java.util.ArrayList;  
import java.util.Random;  
  
public class Main {  
 public static void main(String[] args) {  
 int count = 100;  
 ArrayList<Integer> numbers = *generateRandomNumbers*(count);  
 System.*out*.println("Неотсортированный массив: " + numbers);  
  
 long startTime = System.*nanoTime*();  
 *insertionSort*(numbers);  
 long endTime = System.*nanoTime*();  
  
 double elapsedTimeSeconds = (endTime - startTime) / 1e9;  
  
 System.*out*.println("Отсортированный массив: " + numbers);  
 System.*out*.println("Время выполнения сортировки: " + elapsedTimeSeconds + " секунд");  
 }  
  
 public static void insertionSort(ArrayList<Integer> arr) {  
 int n = arr.size();  
 for (int i = 1; i < n; ++i) {  
 int key = arr.get(i);  
 int j = i - 1;  
  
 while (j >= 0 && arr.get(j) > key) {  
 arr.set(j + 1, arr.get(j));  
 j = j - 1;  
 }  
 arr.set(j + 1, key);  
 }  
 }  
  
 public static ArrayList<Integer> generateRandomNumbers(int count) {  
 ArrayList<Integer> numbers = new ArrayList<>();  
 Random random = new Random();  
 for (int i = 0; i < count; i++) {  
 numbers.add(random.nextInt(100));  
 }  
 return numbers;  
 }  
}

*Структура №2. Связный список (LinkedList)*

|  |  |
| --- | --- |
| Число элементов | Время, с |
| 100 | 0.0022896 |
| 1000 | 0.2884324 |
| 10000 | 241.4197868 |

Код программы:  
import java.util.LinkedList;  
import java.util.Random;  
  
public class Main {  
 public static void insertionSort(LinkedList<Integer> list) {  
 int n = list.size();  
  
 for (int i = 1; i < n; ++i) {  
 int key = list.get(i);  
 int j = i - 1;  
  
  
 while (j >= 0 && list.get(j) > key) {  
 list.set(j + 1, list.get(j));  
 j = j - 1;  
 }  
 list.set(j + 1, key);  
 }  
 }  
  
 public static LinkedList<Integer> generateRandomNumbers(int count) {  
 LinkedList<Integer> numbers = new LinkedList<>();  
 Random random = new Random();  
 for (int i = 0; i < count; i++) {  
 numbers.add(random.nextInt(100));  
 }  
 return numbers;  
 }  
  
 public static void main(String[] args) {  
 int count = 10000;  
 LinkedList<Integer> list = *generateRandomNumbers*(count);  
  
 System.*out*.println("Исходный список: " + list);  
  
 long startTime = System.*nanoTime*();  
 *insertionSort*(list);  
 long endTime = System.*nanoTime*();  
  
 double elapsedTimeSeconds = (endTime - startTime) / 1e9;  
  
 System.*out*.println("Отсортированный список: " + list);  
 System.*out*.println("Время выполн ения сортировки: " + elapsedTimeSeconds + " секунд");  
 }  
}

* Сложность внешнего цикла: O(N)
* Сложность внутреннего цикла (в худшем случае): O(i) для каждого i от 1 до N-1
* Сложность операций внутри внутреннего цикла: O(1)

**Итоговая сложность в худшем случае:**

*O*(*N*2)