**INFORME DE LABORATORIO**

**(formato estudiante)**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **INFORMACIÓN BÁSICA** | | | | | |
| **ASIGNATURA:** | **ESTRUCTURA DE DATOS Y ALGORITMOS** | | | | |
| **TÍTULO DE LA PRÁCTICA:** | *FUNDAMENTOS DE PROGRAMACIÓN* | | | | |
| **NÚMERO DE PRÁCTICA:** | *01* | **AÑO LECTIVO:** | **2025 – A** | **NRO. SEMESTRE:** | **Tercero**  **III** |
| **FECHA DE PRESENTACIÓN** | *09/05/2025* | **HORA DE PRESENTACIÓN** | *23:30* | | |
| **INTEGRANTE (s):**  QUIÑONEZ DELGADO AARÓN FERNANDO | | | | **NOTA:** |  |
| **DOCENTE(s):**   * **Mg. Ing. Rene Alonso Nieto Valencia.** | | | | | |

|  |
| --- |
| **SOLUCIÓN Y RESULTADOS** |
| **Link de GitHub:** [**https://github.com/aaronQuinonez/Lab-EDA-E-Quinonez-Delgado-Aaron**](https://github.com/aaronQuinonez/Lab-EDA-E-Quinonez-Delgado-Aaron)   1. **SOLUCIÓN DE EJERCICIOS/PROBLEMAS**   *Desarrolla un programa en Java que implemente un sistema de gestión de calificaciones de estudiantes. El*  *programa debe permitir al usuario ingresar las calificaciones de N estudiantes y calcular la mediana, moda*  *y desviación estándar*     1. *Ejecución del código.*      **Inicio del programa**.   Se solicita al usuario que ingrese la cantidad de estudiantes.   Se crea un arreglo para almacenar las calificaciones, con una longitud igual a la cantidad de estudiantes ingresada.   Se repite un proceso para cada estudiante:   * Se pide al usuario que ingrese la calificación del estudiante. * La calificación se guarda en el arreglo correspondiente.    Se ordenan las calificaciones de menor a mayor utilizando el método de ordenamiento burbuja:   * Se comparan pares de elementos adyacentes. * Si el primer elemento es mayor que el segundo, se intercambian. * Este proceso se repite hasta que todo el arreglo esté ordenado.    Se imprime en pantalla la lista de calificaciones ordenadas.   Se calcula la **mediana**:   * Si el número de calificaciones es impar, la mediana es el valor que se encuentra justo en el centro del arreglo. * Si el número de calificaciones es par, la mediana es el promedio de los dos valores centrales.    Se calcula la **moda**:   * Se recorre el arreglo para contar cuántas veces aparece cada calificación. * Se determina cuál es la calificación que más veces se repite (frecuencia máxima). * Se asume que solo hay una única moda.    Se calcula la **desviación estándar**:   * Se halla la media (promedio) de todas las calificaciones. * Se calcula la suma de los cuadrados de la diferencia entre cada calificación y la media. * Se divide esta suma entre el número total de calificaciones. * Finalmente, se obtiene la raíz cuadrada del resultado anterior.    Se muestra por pantalla la mediana, la moda y la desviación estándar.   **Fin del programa**.  **Algoritmo GestionDeCalificaciones**  **Inicio**  Escribir "Ingrese la cantidad de estudiantes:"  Leer n  Crear un arreglo Notas de tamaño n  **Para** i desde 0 hasta n - 1 hacer  Escribir "Ingrese la nota del estudiante ", i + 1  Leer Notas[i]  **FinPara**  Llamar a OrdenarArray(Notas)  Llamar a ImprimirArray(Notas)  mediana ← CalcularMediana(Notas)  Escribir "La mediana es: ", mediana  moda ← CalcularModa(Notas)  Escribir "La moda es: ", moda  desviacion ← CalcularDesviacionEstandar(Notas)  Escribir "La desviación estándar es: ", desviacion  Fin  **Subalgoritmo** OrdenarArray(Notas)  **Para** i desde 0 hasta longitud(Notas) - 2 **hacer**  **Para** j desde 0 hasta longitud(Notas) - 2 - i **hacer**  **Si** Notas[j] > Notas[j+1] **entonces**  Intercambiar Notas[j] con Notas[j+1]  FinSi  FinPara  FinPara  **FinSubalgoritmo**  **Subalgoritmo** ImprimirArray(Notas)  Escribir "Notas: "  Para i desde 0 hasta longitud(Notas) - 1 hacer  Escribir Notas[i] con coma si no es el último  FinPara  **FinSubalgoritmo**  **Funcion** CalcularMediana(Notas) → Real  Si longitud(Notas) es par entonces  return (Notas[n/2] + Notas[(n/2) - 1]) / 2.0  Sino  return Notas[n/2]  FinSi  **FinFuncion**  **Funcion** CalcularModa(Notas) → Entero  moda ← Notas[0]  contadorMax ← 0  Para i desde 0 hasta longitud(Notas) - 1 hacer  contador ← 0  Para j desde 0 hasta longitud(Notas) - 1 hacer  Si Notas[j] = Notas[i] entonces  contador ← contador + 1  FinSi  FinPara  Si contador > contadorMax entonces  contadorMax ← contador  moda ← Notas[i]  FinSi  FinPara  return moda  **FinFuncion**  **Funcion** CalcularDesviacionEstandar(Notas) → Real  suma ← 0  Para cada nota en Notas hacer  suma ← suma + nota  FinPara  media ← suma / longitud(Notas)  sumaCuadrados ← 0  Para cada nota en Notas hacer  sumaCuadrados ← sumaCuadrados + (nota - media)^2  FinPara  return RaízCuadrada(sumaCuadrados / longitud(Notas))  **FinFuncion**  *Implementa un programa en Java que encuentre todos los números primos en un rango definido por el*  *usuario utilizando el algoritmo de la Criba de Eratóstenes*     1. *Ejecución*      Iniciar el programa.   Solicitar al usuario dos números: el rango inferior y el rango superior.   Verificar que el rango inferior sea mayor o igual que 2 y que sea menor que el rango superior. Si no se cumple, mostrar un mensaje de error y finalizar el programa.   Crear un arreglo booleano desde 0 hasta el rango superior, donde cada posición representa un número.   Inicializar los valores del arreglo en true a partir del número 2, porque asumimos inicialmente que todos son primos.   Aplicar la Criba de Eratóstenes:   * Para cada número i desde 2 hasta la raíz cuadrada del rango superior:   + Si el número i es primo (es decir, true en el arreglo):     - Marcar como false todos sus múltiplos mayores o iguales a i\*i, ya que no son primos.    Imprimir todos los números dentro del rango que quedaron marcados como true, es decir, los que sí son primos.  **Algoritmo Criba de Eratóstenes**  **Inicio**  Escribir "Ingrese el rango inferior"  Leer rangoInf  Escribir "Ingrese el rango superior"  Leer rangoSup  Si rangoInf < 2 O rangoInf >= rangoSup Entonces  Escribir "Ingrese un rango válido"  Terminar programa  FinSi  Crear arreglo booleano numPrimo[rangoSup + 1]    Para i desde 2 hasta rangoSup hacer  numPrimo[i] ← Verdadero  FinPara  Para i desde 2 hasta i\*i <= rangoSup hacer  Si numPrimo[i] = Verdadero Entonces  Para j desde i\*i hasta rangoSup con paso i hacer  numPrimo[j] ← Falso  FinPara  FinSi  FinPara  Escribir "Los números primos solicitados son:"  Para i desde rangoInf hasta rangoSup hacer  Si numPrimo[i] = Verdadero Entonces  Escribir i  FinSi  FinPara  Fin  *Desarrolla un algoritmo que implemente el Ordenamiento por Inserción, asegurando que en cada paso*  *del bucle el segmento procesado de la lista permanece ordenado (principio de invariante).*     1. *Ejecución*      Iniciar el programa.   Solicitar al usuario la longitud del arreglo a ordenar.   Crear un arreglo de esa longitud.   Solicitar al usuario que ingrese los valores del arreglo, uno por uno.   Mostrar el arreglo antes de ordenar.   Aplicar el algoritmo de **ordenamiento por inserción**:   * Comenzar desde el segundo elemento (posición 1). * Comparar ese elemento con los anteriores y mover los mayores una posición a la derecha. * Insertar el elemento en su posición correcta. * Repetir este proceso para cada elemento del arreglo, asegurando que el segmento ya procesado siempre quede ordenado.    Mostrar el arreglo después de haberlo ordenado.  **Algoritmo Ordenamiento por Inserción**  **Inicio**  Escribir "Ingrese la longitud de su array:"  Leer n  Crear arreglo arr de tamaño n  **Para** i desde 0 hasta n-1 hacer  Escribir "Ingrese el valor para la posición ", i  Leer arr[i]  **FinPara**  Escribir "Antes de ordenar:"  **Para** cada elemento num en arr **hacer**  Escribir num  **FinPara**  **Ordenamiento por inserción**  **Para** i desde 1 hasta n-1 **hacer**  num ← arr[i]  j ← i - 1  Mientras j >= 0 Y arr[j] > num hacer  arr[j + 1] ← arr[j]  j ← j - 1  FinMientras  arr[j + 1] ← num  **FinPara**  Escribir "Después de ordenar:"  **Para** cada elemento num en arr **hacer**  Escribir num  **FinPara**  **Fin** |
| 1. **SOLUCIÓN DEL CUESTIONARIO**   *1. ¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca*  *documentación, complejidad del lenguaje, etc.*  La dificultad fue al momento de decidir cómo diseñar el algoritmo para cada problema, por ejemplo, en el primer problema tenía que ver que algoritmo de ordenamiento podría usar, luego en el problema 2 para implementar un algoritmo en base a la Criba de Eratóstenes y en el problema 3 usando el ordenamiento por inserción. La elaboración de cada algoritmo fue compleja. |
| 1. **CONCLUSIONES**   Los tres ejercicios permitieron aplicar y reforzar conceptos clave de programación en Java. Se trabajó el manejo de arreglos, estructuras de control y cálculos estadísticos con el sistema de calificaciones. Luego, con la Criba de Eratóstenes se implementó una forma eficiente de hallar números primos. Finalmente, el algoritmo de inserción ayudó a comprender cómo mantener ordenado un arreglo paso a paso. |

|  |
| --- |
| **RETROALIMENTACIÓN GENERAL** |
|  |

|  |
| --- |
| **REFERENCIAS Y BIBLIOGRAFÍA** |
|  |