
	UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

INFORME DE LABORATORIO

(formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	ESTRUCTURA DE DATOS Y ALGORITMOS				
TÍTULO DE LA PRÁCTICA:	<i>Fundamentos de Programación</i>				
NÚMERO DE PRÁCTICA:	<i>01</i>	AÑO LECTIVO:	2025 – A	NRO. SEMESTRE:	Tercero III
FECHA DE PRESENTACIÓN	<i>09/05/2025</i>	HORA DE PRESENTACIÓN			
INTEGRANTE (s): Quispe Paucar, Josué Claudio				NOTA:	
DOCENTE(s): <ul style="list-style-type: none"> Mg. Ing. Rene Alonso Nieto Valencia. 					

SOLUCIÓN Y RESULTADOS
<p>I. SOLUCIÓN DE EJERCICIOS/PROBLEMAS</p> <p>Repositorio GitHub: https://github.com/JosueClaudioQP/EDA-Lab</p> <p>EJERCICIOS RESUELTOS POR EL DOCENTE:</p> <p>Enunciado 1: Escribe un programa en Java que permita ingresar las edades de un grupo de personas y determine la edad promedio, la mayor y la menor. El usuario debe poder especificar cuántas edades desea ingresar.</p> <p>Lenguaje natural:</p> <ol style="list-style-type: none"> El programa solicita al usuario que indique cuántas personas hay. Crea un arreglo para almacenar las edades de todas esas personas. Luego, el usuario ingresa una a una las edades. El programa recorre el arreglo de edades y: Suma todas las edades. Compara cada edad para encontrar cuál es la mayor y cuál es la menor. Calcula el promedio dividiendo la suma entre el número de personas. Finalmente, imprime el promedio, la edad más alta y la edad más baja. <p>Pseudocódigo:</p> <p>Inicio</p> <p style="padding-left: 20px;">Escribir "Ingrese el número de personas: "</p> <p style="padding-left: 20px;">Leer n</p> <p style="padding-left: 20px;">Crear arreglo edades de tamaño n</p> <p style="padding-left: 20px;">Escribir "Ingrese las edades:"</p>

Para i desde 0 hasta n-1 hacer

Leer edades[i]

Fin Para

suma ← 0

mayor ← edades[0]

menor ← edades[0]

Para cada edad en edades hacer

suma ← suma + edad

Si edad > mayor entonces

mayor ← edad

Fin Si

Si edad < menor entonces

menor ← edad

Fin Si

Fin Para

promedio ← suma / n

Escribir "Edad promedio: ", promedio

Escribir "Edad mayor: ", mayor

Escribir "Edad menor: ", menor

Fin

Enunciado 2: Escribe un programa en Java que permita calcular la suma de los primeros N números naturales usando un bucle while. El usuario debe ingresar el valor de N.

Lenguaje natural:

1. Pide cuántas personas hay en el grupo.
2. Pide las edades de todas las personas una por una.
3. Calcula la suma de todas las edades.
4. Compara las edades para encontrar la mayor y la menor.
5. Calcula el promedio dividiendo la suma entre la cantidad de personas.
6. Muestra los resultados por pantalla.

Pseudocódigo:

Inicio

Crear un lector para leer datos del usuario

Mostrar "Ingrese el número de personas:"

Leer n



Crear un arreglo de tamaño n para guardar las edades

Mostrar "Ingrese las edades:"

Para i desde 0 hasta n-1 hacer

Leer edad

Guardar edad en el arreglo en la posición i

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 3</p>

Fin Para

Inicializar suma con 0

Inicializar mayor y menor con el primer valor del arreglo

Para cada edad en el arreglo hacer

Sumar edad a la suma

Si edad > mayor entonces

mayor ← edad

Fin Si

Si edad < menor entonces

menor ← edad

Fin Si

Fin Para

promedio ← suma / n

Mostrar "Edad promedio: " + promedio

Mostrar "Edad mayor: " + mayor

Mostrar "Edad menor: " + menor

Fin

Enunciado 3: Implementa un algoritmo que determine si una lista de números ingresados por el usuario está ordenada de manera ascendente. Debes usar un concepto de invariante dentro del bucle para garantizar que la propiedad de orden se mantiene durante la ejecución.

Lenguaje natural:

1. Pide al usuario cuántos números va a ingresar.
2. Solicita al usuario que ingrese esos números.
3. Verifica, recorriendo la lista, si cada número es mayor o igual que el anterior.
4. Si encuentra algún número que sea menor que el anterior, concluye que la lista no está ordenada.
5. Finalmente, muestra en pantalla si la lista está ordenada o no.

Pseudocódigo:

Inicio

Crear un lector para leer datos del usuario

Mostrar "Ingrese el número de elementos:"

Leer n

Crear un arreglo de tamaño n para guardar los números

Mostrar "Ingrese los números:"

Para i desde 0 hasta n-1 hacer

Leer número

Guardar número en la posición i del arreglo

Fin Para


estaOrdenada ← verdadero

Para i desde 1 hasta n-1 hacer
 Si arreglo[i] < arreglo[i - 1] entonces
 estaOrdenada ← falso
 Salir del bucle
 Fin Si
Fin Para

Si estaOrdenada entonces
 Mostrar "¿Está ordenada la lista?: Sí"
Sino
 Mostrar "¿Está ordenada la lista?: No"
Fin Si
Fin

EJERCICIOS PROPUESTOS:

1. 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.

```
Laboratorio1 >  Calificaciones.java > {} Laboratorio1
1  package Laboratorio1;
2  import java.util.*;
3
4  class Calificaciones {
5
6      public static void main(String[] args){
7          Scanner sc = new Scanner(System.in);
8
9          System.out.print("Ingresar numero de alumnos:");
10         int alum = sc.nextInt();
11         int[] notas = new int[alum];
12
13         System.out.println("Ingrese sus notas:");
14         for(int i = 0; i < notas.length; i++){
15             notas[i] = sc.nextInt();
16         }
17         ordenarSort(notas);
18         System.out.println("La mediana es:" + mediana(notas));
19         System.out.println("La moda es:" + moda(notas));
20         System.out.print("La desviacion estandar es:" + DesEst(notas));
21
22         sc.close();
23     }
```

```
24 public static double mediana(int[] num){
25     double med;
26     if(num.length%2 == 1){
27         med = num[num.length/2];
28     } else {
29         med = (num[num.length/2] + num[num.length/2-1])/2;
30     }
31     return med;
32 }
```

```
34 public static int moda(int[] nums){
35     int max = 0;
36     int mayor = nums[0];
37     for(int i = 0; i < nums.length; i++){
38         int conteo = 0;
39         for(int j = 0; j < nums.length; j++){
40             if(nums[i] == nums[j])
41                 conteo++;
42         }
43         if(conteo > max){
44             max = conteo;
45             mayor = nums[i];
46         }
47     }
48     return mayor;
49 }
```

```
51 public static double DesEst(int[] nums){
52     int media = 0;
53     for(int i = 0; i < nums.length; i++){
54         media += nums[i];
55     }
56     media = media / nums.length;
57
58     double desv = 0;
59     for(int j = 0; j < nums.length; j++){
60         desv += Math.pow(nums[j]-media,2);
61     }
62     desv = desv / (nums.length-1);
63     desv = Math.sqrt(desv);
64
65     return desv;
66 }
67 }
```

```
68     public static int[] ordenarSort(int[] num){
69         for(int i = 0; i < num.length-1; i++){
70             for(int j = 0; j < num.length-1-i; j++){
71                 if(num[j]>num[j+1]){
72                     int temp = num[j];
73                     num[j] = num[j+1];
74                     num[j+1] = temp;
75                 }
76             }
77         }
78         return num;
79     }
80 }
```

Lenguaje natural:

1. Ordena las notas de menor a mayor.
2. Calcula la mediana (el valor central de los datos ordenados).
3. Calcula la moda (la nota que más se repite).
4. Calcula la desviación estándar (una medida de dispersión que indica qué tanto se alejan las notas de la media).

Pseudocódigo:**Inicio**

Crear lector de datos

Mostrar "Ingresar número de alumnos:"

Leer alum

Crear arreglo notas de tamaño alum

Mostrar "Ingrese sus notas:"

Para i desde 0 hasta alum - 1 hacer

Leer nota

Guardar nota en notas[i]

Fin Para

Llamar a ordenarSort(notas)



Mostrar "La mediana es: " + mediana(notas)

Mostrar "La moda es: " + moda(notas)

Mostrar "La desviación estándar es: " + DesEst(notas)

Fin**Función ordenarSort(notas)**

Para i desde 0 hasta longitud - 2 hacer

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 7</p>

```

    Para j desde 0 hasta longitud - 2 - i hacer
        Si notas[j] > notas[j+1] entonces
            Intercambiar notas[j] con notas[j+1]
        Fin Si
    Fin Para
Fin Para
Retornar notas ordenadas
Fin

Función mediana(notas)
    Si longitud es impar entonces
        Retornar elemento en la posición longitud/2
    Sino
        Retornar promedio de los dos elementos centrales
Fin

Función moda(notas)
    Inicializar max en 0
    Inicializar mayor en el primer número

    Para cada elemento i en notas hacer
        Inicializar conteo en 0
        Para cada elemento j en notas hacer
            Si notas[i] == notas[j] entonces
                Incrementar conteo
        Fin Para
        Si conteo > max entonces
            max ← conteo
            mayor ← notas[i]
        Fin Si
    Fin Para

    Retornar mayor
Fin

Función DesEst(notas)
    Calcular la media (promedio) de todas las notas
    Inicializar variable desv en 0

    Para cada nota hacer
        Calcular (nota - media)2 y sumarlo a desv
    Fin Para

    Dividir desv entre (n - 1)

```

Calcular la raíz cuadrada del resultado

Retornar desviación estándar



Fin

- 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.

```
Laboratorio1 > EjercicioPropuesto2.java > EjercicioPropuesto2 > main(String[])
1  package Laboratorio1;
2  import java.util.*;
3
4  public class EjercicioPropuesto2 {
5
6      public static void main(String[] args) {
7          Scanner sc = new Scanner(System.in);
8
9          System.out.print(s:"Ingresa el numero a analizar: ");
10         int limite = sc.nextInt();
11
12         if (limite < 2) {
13             System.out.println(x:"El número debe ser mayor o igual a 2.");
14         }
15
16         boolean[] esPrimo = new boolean[limite + 1];
17         for (int i = 2; i <= limite; i++) {
18             esPrimo[i] = true;
19         }
20
21         for (int i = 2; i * i <= limite; i++) {
22             if (esPrimo[i]) {
23                 for (int j = i * i; j <= limite; j += i) {
24                     esPrimo[j] = false;
25                 }
26             }
27         }
28
29         System.out.println("Números primos hasta " + limite + ":");
30         for (int i = 2; i <= limite; i++) {
31             if (esPrimo[i]) {
32                 System.out.print(i + " ");
33             }
34         }
35
36         sc.close();
37     }
38 }
```

Lenguaje natural:

- Solicita un número entero al usuario.
- Verifica si el número es menor que 2; si lo es, muestra un mensaje de error.
- Crea un arreglo booleano donde se marcará si cada número es primo o no.

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 9</p>

4. Inicialmente, considera que todos los números desde 2 hasta el límite son primos.
5. Utiliza el algoritmo de Criba de Eratóstenes:
 - Para cada número i desde 2 hasta la raíz cuadrada del límite:
 - Si i es primo, marca como no primos todos los múltiplos de i mayores que i^2 .
6. Finalmente, imprime todos los números que quedaron marcados como primos.

Pseudocódigo:

Inicio

Crear lector de datos

Mostrar "Ingresa el número a analizar:"

Leer limite

Si $\text{limite} < 2$ entonces

Mostrar "El número debe ser mayor o igual a 2."

Terminar programa

Fin Si

Crear arreglo booleano $\text{esPrimo}[0..\text{limite}]$, inicialmente falso

Para i desde 2 hasta limite hacer

$\text{esPrimo}[i] \leftarrow \text{verdadero}$

Fin Para

Para i desde 2 hasta raíz cuadrada de limite hacer

Si $\text{esPrimo}[i]$ entonces

Para j desde $i*i$ hasta limite con incremento de i hacer

$\text{esPrimo}[j] \leftarrow \text{falso}$

Fin Para

Fin Si

Fin Para

Mostrar "Números primos hasta limite:"

Para i desde 2 hasta limite hacer

Si $\text{esPrimo}[i]$ entonces

Mostrar i

Fin Si



Fin Para

Fin

3. 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).

```
Laboratorio1 > EjercicioPropuesto3.java > EjercicioPropuesto3 > main(String[])
1  package Laboratorio1;
2  import java.util.*;
3
4  public class EjercicioPropuesto3 {
5
6      public static void main(String[] args) {
7          Scanner sc = new Scanner(System.in);
8
9          System.out.print(s:"Ingresar cantidad de numeros a ordenar: ");
10         int cant = sc.nextInt();
11         int[] nums = new int[cant];
12         System.out.println(x:"Ingresar numeros a ordenar:");
13         for(int i = 0; i < nums.length; i++){
14             nums[i] = sc.nextInt();
15         }
16
17         insertionSort(nums);
18
19         System.out.print(s:"Arreglo ordenado: ");
20         for (int num : nums) {
21             System.out.print(num + " ");
22         }
23
24         sc.close();
25     }
26
27     public static void insertionSort(int[] nums) {
28         int n = nums.length;
29
30         for (int i = 1; i < n; i++) {
31             int actual = nums[i];
32             int j = i - 1;
33
34             while (j >= 0 && nums[j] > actual) {
35                 nums[j + 1] = nums[j];
36                 j--;
37             }
38             nums[j + 1] = actual;
39
40             System.out.print("Paso " + i + ": ");
41             for (int k = 0; k < n; k++) {
42                 System.out.print(nums[k] + " ");
43             }
44             System.out.println();
45         }
46     }
47 }
```

Lenguaje natural:

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 11</p>

1. Solicita al usuario cuántos números desea ordenar.
2. El usuario ingresa los números, que se almacenan en un arreglo.
3. Se ordenan los números utilizando el algoritmo Insertion Sort:
 - Se recorre el arreglo desde la segunda posición.
 - En cada iteración, el elemento actual se compara con los elementos anteriores.
 - Si el elemento actual es menor, se desplazan los mayores hacia la derecha hasta encontrar su posición correcta.
4. Después de cada paso del ordenamiento, se imprime el estado del arreglo.
5. Al finalizar, se imprime el arreglo ya ordenado.

Pseudocódigo:

Inicio

Crear lector de datos

Mostrar "Ingresar cantidad de números a ordenar:"

Leer cant

Crear arreglo nums de tamaño cant

Mostrar "Ingresar números a ordenar:"

Para i desde 0 hasta cant - 1 hacer

Leer número

nums[i] ← número

Fin Para

Llamar a insertionSort(nums)

Mostrar "Arreglo ordenado:"

Para cada número en nums hacer

Imprimir número

Fin Para

Fin

Función insertionSort(nums)

Para i desde 1 hasta longitud del arreglo - 1 hacer

actual ← nums[i]

j ← i - 1

Mientras j ≥ 0 y nums[j] > actual hacer

nums[j + 1] ← nums[j]

j ← j - 1

Fin Mientras

nums[j + 1] ← actual



Mostrar "Paso i: "

Para k desde 0 hasta longitud - 1 hacer

Imprimir nums[k]

Fin Para

Salto de línea

	<p style="text-align: center;">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p style="text-align: center;">Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 12</p>

Fin Para Fin
<p>II. SOLUCIÓN DEL CUESTIONARIO</p> <p><i>¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.</i></p> <p><i>Lo que más se me complicó fue el desarrollo de los diferentes algoritmos, el lenguaje Java ya lo hemos usado, sin embargo la lógica a usar al momento de programar y tratar de hacerlo de la mejor manera fue lo más complicado.</i></p>
<p>III. CONCLUSIONES</p>

RETROALIMENTACIÓN GENERAL

REFERENCIAS Y BIBLIOGRAFÍA