
	<p align="center">UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación		
Aprobación: 2022/03/01	Código: GUIA-PRLD-001	Página: 1



GUÍA DE LABORATORIO

(formato docente)

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
TIPO DE PRÁCTICA:	INDIVIDUAL	X			
	GRUPAL	—	MÁXIMO DE ESTUDIANTES	00	
FECHA INICIO:	06/05/2025	FECHA FIN:	10/05/2025	DURACIÓN:	90 minutos.
RECURSOS A UTILIZAR: <ul style="list-style-type: none"> • Github. • Lenguaje Natural Pseudocódigo). • Ide Java Eclipse/Visual Studio Code. 					
DOCENTE(s): <ul style="list-style-type: none"> • Mg. Ing. Rene Alonso Nieto Valencia. 					

OBJETIVOS/TEMAS Y COMPETENCIAS	
OBJETIVOS: <ul style="list-style-type: none"> • Aprenda principios básicos sobre el diseño de algoritmos. • Aplicar conceptos elementales de programación (condicionales, bucles, arreglos, etc.) a problemas de algoritmos. • Desarrollar pruebas. 	
TEMAS: <ul style="list-style-type: none"> • Diseño de Algoritmos. • Estructuras de Datos. • Arreglos, Iteraciones e Invariantes. 	
COMPETENCIAS	C.a
	C.b
	C.c
	C.d

CONTENIDO DE LA GUÍA

	<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-PRLD-001</p>	<p>Página: 2</p>

I. MARCO CONCEPTUAL

INTRODUCCIÓN

Estas notas de clase cubren las ideas clave involucradas en el diseño de algoritmos. Veremos cómo dependen del diseño de estructuras de datos adecuadas, y cómo algunas estructuras y algoritmos son más eficientes que otros para la misma tarea.

Nos concentramos en algunas tareas básicas, como el almacenamiento, la clasificación y la búsqueda de datos, que subyacen en gran parte de la informática, pero las técnicas discutidas se aplicarán de manera mucho más general.

Algoritmos Un algoritmo para una tarea en particular se puede definir como “una secuencia finita de instrucciones, cada una de las cuales tiene un significado claro y se puede realizar con una cantidad finita de esfuerzo en un período de tiempo finito”. Como tal, un algoritmo debe ser preciso suficiente para ser entendido por los seres humanos. Sin embargo, para ser ejecutado por una computadora, generalmente necesitaremos un programa que esté escrito en un lenguaje formal riguroso; y dado que las computadoras son bastante inflexibles en comparación con la mente humana, los programas generalmente deben contener más detalles que los algoritmos. Aquí ignoraremos la mayoría de esos detalles de programación y nos concentraremos en el diseño de algoritmos en lugar de programas.

Obviamente, los algoritmos se pueden describir en un lenguaje sencillo, y algunas veces lo haremos. Sin embargo, para los informáticos suele ser más fácil y claro utilizar algo que se encuentre entre el inglés formateado y el código de un programa informático, pero que no se pueda ejecutar porque se omiten ciertos detalles. Esto se llama pseudocódigo, que viene en una variedad de formas. A menudo, estas notas presentarán segmentos de pseudocódigo que son muy similares a los lenguajes que nos interesan principalmente, a saber, la superposición de C y Java, con la ventaja de que pueden insertarse fácilmente en programas ejecutables.

ESTRUCTURAS DE DATOS Y TIPOS DE DATOS ABSTRACTOS



Para muchos problemas, la capacidad de formular un algoritmo eficiente depende de poder organizar los datos de manera adecuada. El término estructura de datos se utiliza para denotar una forma particular de organizar los datos para tipos particulares de operaciones.

A menudo queremos hablar sobre estructuras de datos sin tener que preocuparnos por todos los detalles de implementación asociados con lenguajes de programación particulares, o cómo se almacenan los datos en la memoria de la computadora. Podemos hacer esto formulando modelos matemáticos abstractos de clases particulares de estructuras de datos o tipos de datos que tienen características comunes. Estos se denominan tipos de datos abstractos y se definen sólo por las operaciones que se pueden realizar en ellos.

Por lo general, especificamos cómo se construyen a partir de tipos de datos más primitivos (por ejemplo, enteros o cadenas), cómo extraer esos datos de ellos y algunas comprobaciones básicas para controlar el flujo de procesamiento en los algoritmos. La idea de que los detalles de implementación están ocultos para el usuario y protegidos del acceso externo se conoce como encapsulación.

ARREGLOS, ITERACIONES E INVARIANTES

En última instancia, los datos se almacenan en las computadoras como patrones de bits, aunque en la actualidad la mayoría de los lenguajes de programación tratan con objetos de nivel superior, como caracteres, números enteros y números de coma flotante. Generalmente, necesitamos construir algoritmos que manipulan

	<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-PRLD-001</p>	<p>Página: 3</p>

colecciones de dichos objetos, por lo que necesitamos procedimientos para almacenarlos y procesarlos secuencialmente.

Arreglos

En informática, la forma obvia de almacenar una colección ordenada de elementos es como una matriz. Los elementos de la matriz generalmente se almacenan en una secuencia de ubicaciones de memoria de computadora, pero para discutirlos, necesitamos una forma conveniente de escribirlos en papel. Simplemente podemos escribir los elementos en orden, separados por comas y encerrados entre corchetes.

Por lo tanto, **[1, 4, 17, 3, 90, 79, 4, 6, 81]**

es un ejemplo de un arreglo de enteros. Si llamamos a este arreglo a, podemos escribirlo como:

a = [1, 4, 17, 3, 90, 79, 4, 6, 81]

Este arreglo "a" tiene 9 elementos y, por lo tanto, decimos que su tamaño es 9. En la vida cotidiana, generalmente comenzamos a contar desde 1. Sin embargo, cuando trabajamos con matrices en informática, con mayor frecuencia (aunque no siempre) comenzamos desde 0. Así, para nuestro arreglo a, sus posiciones son 0, 1, 2, . . . , 7, 8. El elemento en la octava posición es 81, y usamos la notación a[8] para denotar este elemento. Más generalmente, para cualquier entero i que denote una posición, escribimos a[i] para denotar el elemento en la i-ésima posición. Esta posición i se llama índice (y el plural es índices). Luego, en el ejemplo anterior, a[0] = 1, a[1] = 4, a[2] = 17, y así sucesivamente.

Bucles e iteraciones

El enfoque estándar en la mayoría de los lenguajes de programación para repetir un proceso una cierta cantidad de veces, como moverse secuencialmente a través de una matriz para realizar las mismas operaciones en cada elemento, implica un bucle. En pseudocódigo, esto normalmente tomaría la forma general

for i=1, ...,N,
do something

y en lenguajes de programación como C y Java esto se escribiría como **for-loop**:

for(i = 0 ; i < N ; i++) {
do something
}

en el que un contador "i" realiza un seguimiento de hacer "algo" N veces. Por ejemplo, podríamos calcular la suma de los 20 elementos en una matriz "a" usando:

for(i = 0 ; i < 20 ; i++) {
sum += a[i]
}

Decimos que hay iteración sobre el índice i. La estructura general del bucle for es

for(INICIALIZACIÓN; CONDICIÓN; ACTUALIZACIÓN) {
PROCESO REPETIDO
}

Invariantes

Un invariante, como sugiere su nombre, es una condición que no cambia durante la ejecución de un programa o algoritmo determinado. Puede ser una desigualdad simple, como "i < 20", o algo más abstracto, como "los elementos de la matriz están ordenados". Los invariantes son importantes para las estructuras de datos y los algoritmos porque permiten pruebas de corrección y verificación.

II. EJERCICIO/PROBLEMA RESUELTO POR EL DOCENTE

1. Arreglos Unidimensionales.

Enunciado: 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.

Datos de Entrada: Numero de edades/personas y Listado de Edades.

Datos de Salida: Promedio de edades, mayor edad, menor edad.

SOLUCIÓN:

```
import java.util.Scanner;

public class EdadesGrupo {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingrese el número de personas: ");
        int n = scanner.nextInt();
        int[] edades = new int[n];

        System.out.println("Ingrese las edades:");
        for (int i = 0; i < n; i++) {
            edades[i] = scanner.nextInt();
        }

        int suma = 0, mayor = edades[0], menor = edades[0];
        for (int edad : edades) {
            suma += edad;
            if (edad > mayor)
                mayor = edad;
            if (edad < menor)
                menor = edad;
        }

        double promedio = (double) suma / n;

        System.out.println("Edad promedio: " + promedio);
        System.out.println("Edad mayor: " + mayor);
        System.out.println("Edad menor: " + menor);

        scanner.close();
    }
}
```

2. Iteraciones:

Enunciado: 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.

Datos de Entrada: cantidad de Numero naturales

Datos de Salida: suma de números naturales

SOLUCIÓN:

```
import java.util.Scanner;

public class SumaNaturales {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingrese un número N: ");
        int n = scanner.nextInt();
        int suma = 0, contador = 1;

        while (contador <= n) {
            suma += contador;
            contador++;
        }

        System.out.println("La suma de los primeros " + n + " números naturales es: "
+ suma);

        scanner.close();
    }
}
```

3. Invariantes:

Enunciado: 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.

Datos de Entrada: lista de números con orden ascendente.

Datos de Salida: True o False.



SOLUCIÓN:

```
import java.util.Scanner;

public class ListaOrdenada {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Ingrese el número de elementos: ");
        int n = scanner.nextInt();
        int[] numeros = new int[n];

        System.out.println("Ingrese los números:");
        for (int i = 0; i < n; i++) {
            numeros[i] = scanner.nextInt();
        }
    }
}
```

	<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-PRLD-001</p>	<p>Página: 6</p>

```

        boolean estaOrdenada = true; // Invariante: Se supone que la lista está
ordenada
        for (int i = 1; i < n; i++) {
            if (numeros[i] < numeros[i - 1]) {
                estaOrdenada = false; // Se rompe la invariante si encontramos un
desorden
                break;
            }
        }

        System.out.println("¿Está ordenada la lista?: " + (estaOrdenada ? "Sí" :
"No"));

        scanner.close();
    }
}

```

III. EJERCICIOS/PROBLEMAS PROPUESTOS

De acuerdo a los ejercicios propuestos desarrollar los algoritmos y luego desarrollar utilizando java para validar el algoritmo la implementación del programa, la formulación del algoritmo utilizar lenguaje natural.

Indicando:


- Datos de entrada.
 - Datos de salida.
 - Proceso a realizar.
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
 2. 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
 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).

IV. CUESTIONARIO

1. ¿Cuáles fueron las dificultades que encontraste al desarrollar los ejercicios propuestos? por ejemplo, poca documentación, complejidad del lenguaje, etc.

V. REFERENCIAS Y BIBLIOGRAFÍA RECOMENDADAS:

- Weiss M., Data Structures & Problem Solving Using Java, 2010, Addison-Wesley.
- Weiss M., Data Structures and Algorithms Analysis in Java, 2012, Addison-Wesley.
- Cormen T., Leiserson C., Rivest R., Stein C., Introduction to Algorithms, 2022, The MIT Press
- The Java™ Tutorials - <https://docs.oracle.com/javase/tutorial/>
- Sedgewick, R., Algorithms in Java, Parts 1-4: Fundamentals, Data Structures, Sorting, Searching, Part 5:
- Graph Algorithms, Addison-Wesley.

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

- Malik D., Data Structures Usign C++, 2003, Thomson Learning.
- Knuth D., The Art of Computer Programming, Vol. 1 y 3, Addison - Wesley.

TÉCNICAS E INSTRUMENTOS DE EVALUACIÓN	
TÉCNICAS: <i>Actividades Resueltas</i> <i>Ejercicios Propuestos</i>	INSTRUMENTOS: <i>Rubricas</i>
CRITERIOS DE EVALUACIÓN Los criterios de evaluación se encuentran en el silabo DUFA ANEXO en la sección EVOLUCIÓN CONTINUA	