
	<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:	TÉCNICAS Y DISEÑO DE ALGORITMOS				
NÚMERO DE PRÁCTICA:	02	AÑO LECTIVO:	2025 – A	NRO. SEMESTRE:	TERCERO III
TIPO DE PRÁCTICA:	INDIVIDUAL	X			
	GRUPAL	—	MÁXIMO DE ESTUDIANTES	00	
FECHA INICIO:	18/05/2025	FECHA FIN:	17/05/2025	DURACIÓN:	90 minutos.
RECURSOS A UTILIZAR: <ul style="list-style-type: none"> • Github. • Lenguaje de Programación Java. • 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 técnicas y diseño de algoritmos. • Aplicar conceptos elementales de programación (condicionales, bucles, arreglos, etc.) a resolver en recursividad en problemas de algoritmos. • Desarrollar pruebas. 	
TEMAS: <ul style="list-style-type: none"> • Introducción. • Recursividad. • Técnicas y Diseño de Algoritmos. 	
COMPETENCIAS	C.a
	C.b
	C.c
	C.d

CONTENIDO DE LA GUÍA

II. EJERCICIO/PROBLEMA RESUELTO POR EL DOCENTE

Repositorio GitHub: <https://github.com/JosueClaudioQP/EDA-Lab>

En un editor Java, realizar la integración de los siguientes ejercicios, revisar los resultados obtenidos y realizar una explicación del funcionamiento de forma concreta y clara.

1. Ejercicio 1: Implementación de un método recursivo.

```
public class Recursividad {  
  
    void repetir() {  
        repetir();  
    }  
  
    public static void main(String[] ar) {  
        Recursividad re = new Recursividad();  
        re.repetir();  
    }  
}
```

2. Ejercicio 2: Implementación de un método recursivo que reciba un parámetro de tipo entero y luego llame en forma recursiva con el valor del parámetro menos 1.

```
public class Recursividad {  
  
    void imprimir(int x) {  
        System.out.println(x);  
        imprimir(x - 1);  
    }  
  
    public static void main(String[] ar) {  
        Recursividad re = new Recursividad();  
        re.imprimir(5);  
    }  
}
```

3. Ejercicio 3: Implementar un método recursivo que imprima en forma descendente de 5 a 1 de uno en uno.

```
public class Recursividad {  
  
    void imprimir(int x) {  
        if (x > 0) {  
            System.out.println(x);  
            imprimir(x - 1);  
        }  
    }  
  
    public static void main(String[] ar) {  
        Recursividad re = new Recursividad();  
        re.imprimir(5);  
    }  
}
```

4. Ejercicio 4: Imprimir los números de 1 a 5 en pantalla utilizando recursividad.

```
public class Recursividad {  
  
    void imprimir(int x) {  
        if (x > 0) {  
            imprimir(x - 1);  
            System.out.println(x);  
        }  
    }  
  
    public static void main(String[] ar) {  
        Recursividad re = new Recursividad();  
        re.imprimir(5);  
    }  
}
```

5. Ejercicio 5: Obtener el factorial de un número. Recordar que el factorial de un número es el resultado que se obtiene de multiplicar dicho número por el anterior y así sucesivamente hasta llegar a uno.
Ej. el factorial de 4 es $4 * 3 * 2 * 1$ es decir 24.

```
public class Recursividad {  
  
    int factorial(int fact) {  
        if (fact > 0) {  
            int valor = fact * factorial(fact - 1);  
            return valor;  
        } else  
            return 1;  
    }  
  
    public static void main(String[] ar) {  
        Recursividad re = new Recursividad();  
        int f = re.factorial(4);  
    }  
}
```

```
        System.out.println("El factorial de 4 es " + f);  
    }  
}
```

6. Ejercicio 6. Implementar un método recursivo para ordenar los elementos de un vector.

```
class Recursividad {  
    static int[] vec = { 312, 614, 88, 22, 54 };  
  
    void ordenar(int[] v, int cant) {  
        if (cant > 1) {  
            for (int f = 0; f < cant - 1; f++)  
                if (v[f] > v[f + 1]) {  
                    int aux = v[f];  
                    v[f] = v[f + 1];  
                    v[f + 1] = aux;  
                }  
            ordenar(v, cant - 1);  
        }  
    }  
  
    void imprimir() {  
        for (int f = 0; f < vec.length; f++)  
            System.out.print(vec[f] + " ");  
        System.out.println("\n");  
    }  
  
    public static void main(String[] ar) {  
        Recursividad r = new Recursividad();  
        r.imprimir();  
        r.ordenar(vec, vec.length);  
        r.imprimir();  
    }  
}
```

III. EJERCICIOS/PROBLEMAS PROPUESTOS

De acuerdo a los ejercicios propuestos desarrollar los algoritmos y mostrar las siguientes indicaciones:

- Enunciado del ejercicio.
- Código en java desarrollado.
- Resultados obtenidos.
- Explicación breve y concreta del código implementado.

1. Invertir vector de enteros, permite ingresar tamaño y captura de valores del arreglo, el método **invertirArray** calcula y muestra el resultado.

N = 3

A = [1 2 3] -> Asalida=[3 2 1]

```
public int[] invertirArray(int[] A) {
```

```
//Ingresar el código aquí
return Asalida;
}
```

```
EjerciciosPropuestos > Propuesto1.java > Propuesto1 > invertirArray(int[])
1 package Laboratorio2.EjerciciosPropuestos;
2
3 public class Propuesto1 {
4     public int[] invertirArray(int[] a){
5         if(a.length > 0){
6             for(int i = 0; i < a.length-i-1; i++){
7                 int aux = a[i];
8                 a[i] = a[a.length-i-1];
9                 a[a.length-i-1] = aux;
10            }
11        }
12        return a;
13    }
14
15    void imprimir(int[] a) {
16        for (int f = 0; f < a.length; f++)
17            System.out.print(a[f] + " ");
18        System.out.println("\n");
19    }
20 }
```

Resultado en la consola:

```
Ejercicio1:
5 9 7 8 1

1 8 7 9 5
```

Explicación: La clase Propuesto1 contiene dos métodos que permiten invertir e imprimir un arreglo de enteros. El método `invertirArray` recorre el arreglo desde los extremos hacia el centro, intercambiando los elementos de forma que el primero pasa al final, el segundo al penúltimo, y así sucesivamente; sin embargo, la condición del bucle debería ser $i < a.length / 2$ para mayor claridad y eficiencia. Por otro lado, el método `imprimir` muestra los elementos del arreglo separados por espacios, seguido de una línea en blanco, facilitando la visualización del contenido del arreglo.

2. **Rotar a la Izquierda**, permite ingresar tamaño y captura de valores del arreglo, el método `rotarIzquierdaArray` calcula y muestra el resultado.

Si $d=2$

$A=[1\ 2\ 3\ 4\ 5] \rightarrow A_{invertido}=[3\ 4\ 5\ 1\ 2]$

```
public int[] rotarIzquierdaArray(int[] A) {
//Ingresar el código aquí
return Ainvertido;
}
```

```

4      public int[] rotarIzquierdaArray(int[] A){
5          int d = 2;
6          int n = A.length;
7          int[] Ainvertido = new int[n];
8
9          for (int i = 0; i < n; i++) {
10             Ainvertido[i] = A[(i + d) % n];
11         }
12
13         return Ainvertido;
14     }
15 }
16

```

Resultado en la consola:

```

Ejercicio2:
3 4 5 1 2

```

Explicación: La clase Propuesto2 contiene un método que rota un arreglo de enteros hacia la izquierda. El método `rotarIzquierdaArray` desplaza todos los elementos del arreglo `A` dos posiciones hacia la izquierda, y los elementos que "salen" por el inicio del arreglo se colocan al final. Para lograrlo, crea un nuevo arreglo `Ainvertido` del mismo tamaño y copia en él cada elemento del arreglo original en su nueva posición usando la fórmula $(i + d) \% n$, donde $d = 2$ es la cantidad de posiciones a rotar y n es la longitud del arreglo. Finalmente, retorna el nuevo arreglo rotado.

3. Triangulo recursivo 1. El método `trianguloRecursivo1` calcula y muestra el resultado.

- Si $b = 5$
- Salida:

```

*
**
***
****
*****

```

```

public void trianguloRecursivo1(int base) {
//Ingresar el código aquí
}

```

```
EjerciciosPropuestos > Propuesto3.java > {} Laboratorio2.EjerciciosProp
1  package Laboratorio2.EjerciciosPropuestos;
2
3  public class Propuesto3 {
4      public void trianguloRecursivo1(int base){
5          if(base > 0){
6              trianguloRecursivo1(base-1);
7              for(int i = 0; i < base; i++){
8                  System.out.print(s:"*");
9              }
10             System.out.println();
11         }
12     }
13 }
14
```

Resultado en consola:

```
Ejercicio3:
*
**
***
****
*****
```

Explicación del código: La clase Propuesto3 contiene un método recursivo llamado trianguloRecursivo1 que imprime un triángulo de asteriscos alineado a la izquierda. El método recibe un número entero base, que representa la altura y la base del triángulo. Utiliza recursión para primero llegar al caso base (cuando base es 0) y luego, en cada retorno de la llamada recursiva, imprime una línea con una cantidad creciente de asteriscos, desde 1 hasta el valor de base. Así, el resultado es un triángulo que se forma desde arriba hacia abajo, aumentando una estrella por línea.

4. Triángulo recursivo 2. El método **trianguloRecursivo2** calcula y muestra el resultado.

- Si b = 5
- Salida:

```

*
**
***
****
*****
```

```
public void trianguloRecursivo2(int base) {
//Ingresar el código aqui
}
```

```
EjerciciosPropuestos > Propuesto4.java > {} Laboratorio2.EjerciciosProp
1 package Laboratorio2.EjerciciosPropuestos;
2
3 public class Propuesto4 {
4     public void trianguloRecursivo2(int base){
5         if(base > 0){
6             int nivel = 5;
7             trianguloRecursivo2(base-1);
8             for(int i = 0; i < nivel-base; i++){
9                 System.out.print(s:" ");
10            }
11            for(int i = 0; i < base; i++){
12                System.out.print(s:"*");
13            }
14            System.out.println();
15        }
16    }
17 }
18
19
```

Resultado en consola:

```
Ejercicio4:
*
**
***
****
*****
```

Explicación del código: La clase Propuesto4 define un método llamado trianguloRecursivo2 que imprime un triángulo rectángulo alineado a la derecha utilizando recursión. El parámetro base representa la altura del triángulo y también controla cuántas líneas se imprimen. En cada llamada recursiva, se decrementa base hasta llegar a cero, y en el regreso de la recursión se imprime cada línea del triángulo. Para alinear las estrellas a la derecha, se imprimen primero espacios (calculados como nivel - base, donde nivel está fijo en 5) y luego los asteriscos correspondientes al valor de base. El resultado es un triángulo rectángulo invertido hacia la derecha.

5. **Triángulo recursivo 3.** El método **trianguloRecursivo3** calcula y muestra el resultado.

- Si b = 5
- Salida:
*
**


```
public void trianguloRecursivo3(int base) {  
    //Ingresar el código aqui  
}
```

```
EjerciciosPropuestos > Propuesto5.java > {} Laboratorio2.EjerciciosPropuestos  
1 package Laboratorio2.EjerciciosPropuestos; Propuesto5  
2  
3 public class Propuesto5 {  
4     public void trianguloRecursivo3(int base){  
5         if(base > 0){  
6             trianguloRecursivo3(base-1);  
7             int nivel = 5;  
8             for(int i = 0; i < nivel-base; i++){  
9                 System.out.print(s:" ");  
10            }  
11            for(int i = 0; i < (2*base-1); i++){  
12                System.out.print(s:"*");  
13            }  
14            System.out.println();  
15        }  
16    }  
17 }  
18
```

Resultado en consola:

```
Ejercicio5:  
*  
**  
***  
****  
*****  
*****  
*****
```

Explicación del código: La clase Propuesto5 contiene el método trianguloRecursivo3, que imprime un triángulo isósceles utilizando recursión. El parámetro base indica la cantidad de niveles del triángulo. En cada llamada recursiva, se reduce base hasta llegar a 0, y luego, al regresar, se imprime cada nivel con espacios y asteriscos. Se imprimen primero algunos espacios (nivel - base) para centrar los asteriscos, y luego una cantidad impar de asteriscos (2 * base - 1) para formar la forma simétrica del triángulo. El resultado es un triángulo centrado con forma de pirámide.

6. Cuadrado recursivo. El método cuadradoRecursivo calcula y muestra el resultado.

- Si b = 5
- Salida:

* *

* *

* *

```
public void cuadradoRecursoivo(int base) {  
//Ingresar el código aquí  
}
```

```
EjerciciosPropuestos > Propuesto6.java > Propuesto6 > dibujarCuadrado(int, int)  
1 package Laboratorio2.EjerciciosPropuestos; Propuesto6.java is a non-p  
2  
3 import java.util.Scanner;  
4  
5 public class Propuesto6 {  
6     public static void dibujarCuadrado(int lado, int fila) {  
7         if (fila > lado)  
8             return;  
9  
10        for (int col = 1; col <= lado; col++) {  
11            if (fila == 1 || fila == lado || col == 1 || col == lado) {  
12                System.out.print(s:"*");  
13            } else {  
14                System.out.print(s:" ");  
15            }  
16        }  
17  
18        System.out.println();  
19        dibujarCuadrado(lado, fila + 1);  
20    }  
21  
22    Run | Debug  
23    public static void main(String[] args) {  
24        Scanner sc = new Scanner(System.in); Resource leak: 'sc' is n  
25        System.out.print(s:"Ingrese el valor del lado del cuadrado: ");  
26        int lado = sc.nextInt();  
27        dibujarCuadrado(lado, fila:1);  
28    }  
29
```

Resultado en consola:

```
Ingrese el valor del lado del cuadrado: 5  
*****  
*     *  
*     *  
*     *  
*****
```



Explicación del código: La clase Propuesto6 permite dibujar un cuadrado hueco de asteriscos usando recursión. El método dibujarCuadrado recibe dos parámetros: lado, que indica el tamaño del cuadrado (número de filas y columnas), y fila, que representa la fila actual a imprimir. En cada llamada, el método imprime una fila del cuadrado, colocando asteriscos en los bordes (primera y última fila o columna) y espacios en el interior. Luego, llama recursivamente a sí mismo para imprimir la siguiente fila hasta completar el cuadrado. En el método main, se solicita al usuario el valor del lado y se inicia el dibujo desde la primera fila.



IV. CUESTIONARIO

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

Las principales dificultades fueron comprender a fondo el uso de la recursión y cómo controlar adecuadamente las condiciones de parada para evitar errores como bucles infinitos o desbordamientos de pila. Además, al principio hubo cierta confusión con el manejo de índices en los arreglos y en la alineación de espacios al imprimir figuras. También fue un reto encontrar ejemplos claros o documentación sencilla sobre algunos patrones recursivos, especialmente en ejercicios gráficos con asteriscos.

2. Diferencias entre algoritmos de secuencialidad, decisión e iteración.
Secuencialidad, condicionalidad e iteración.
3. Que son las clases y métodos genéricos.
Las clases y métodos genéricos en Java permiten escribir código flexible y reutilizable que puede trabajar con distintos tipos de datos sin necesidad de duplicar el código. Por ejemplo, una clase genérica puede manejar listas de enteros, cadenas o cualquier otro tipo de objeto utilizando un tipo parámetro (<T>). Esto mejora la seguridad del tipo en tiempo de compilación y evita errores de conversión. Un ejemplo común es la clase ArrayList<T> del paquete java.util.

	<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: 12</p>
Empty space for content		
<p>V. REFERENCIAS Y BIBLIOGRAFÍA RECOMENDADAS:</p> <ul style="list-style-type: none"> • 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. • Malik D., Data Structures Usign C++, 2003, Thomson Learning. 		

	<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: 13</p>

- Knuth D., The Art of Computer Programming, Vol. 1 y 3, Addison - Wesley.

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