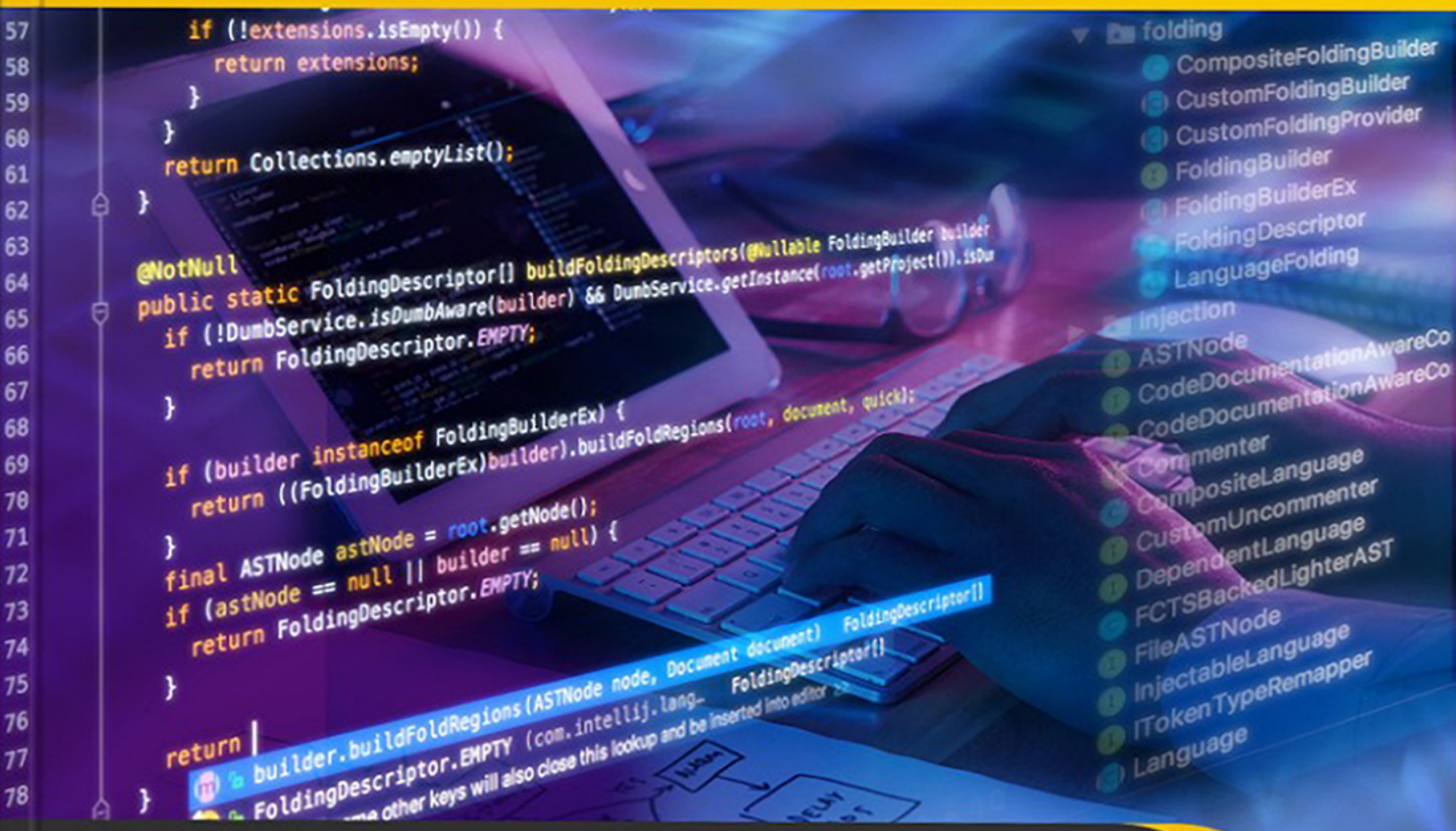
Pino Herrera | Martínez Moreno | Vergara Camacho | Contreras Vega

# Introducción a la Programación

## Problemario de algoritmos pasos a paso, pruebas de escritorio y codificación en JAVA




ISBN: 978-607-9011-99-4

HESS grupo editorial

Universidad Veracruzana

# Introducción a la Programación

*Problemario de algoritmos pasos a paso, pruebas de escritorio y codificación en JAVA*

**Introducción a la Programación.**
**Problemario de algoritmos pasos a paso,**
**pruebas de escritorio y codificación en JAVA**

Primera edición, Agosto 2020

Autores:

Javier Pino Herrera
Patricia Martínez Moreno
José Antonio Vergara Camacho
Gerardo Contreras Vega

**GRUPO EDITORIAL HESS, S.A. DE C.V.**

Manuel Gutiérrez Nájera No. 91, Col. Obrera
Alcaldía Cuauhtémoc, C.P. 06800, Ciudad de México.

**ISBN 978-607-9011-99-4**

El contenido, las opiniones y la originalidad de los artículos publicados en este libro son responsabilidad exclusiva de sus autores y no reflejan necesariamente el punto de vista de los coordinadores o editor.

Prohibida la reproducción total o parcial por cualquier medio sin autorización expresa de los autores, titular de los derechos patrimoniales y/o la Universidad Veracruzana.

Editado en México

# Introducción a la Programación

*Problemario de algoritmos pasos a paso, pruebas de escritorio y codificación en JAVA*


**Autores:**

Javier Pino Herrera

Patricia Martínez Moreno

José Antonio Vergara Camacho

Gerardo Contreras Vega

**Colaboradores:**

Ana Laura Arau Gracia

Ángeles Montserrat Larios Chacha

Carlos Daniel Valerio Vargas

Adriana Torres Ramírez

Alexis Concepción Lara

Jaime Antonio Hernández Cabrera



HESS grupo editorial

Universidad Veracruzana

# Directorio

Sara Ladrón de Guevara
**Rectora**

María Magdalena Hernández Alarcón
**Secretaria Académica**

Salvador F. Tapia Spinoso
**Secretario de Administración y Finanzas**

Arturo Bocardo Valle
**Directora del Área Económico-Administrativa**

Liliana I. Betancourt Trevedhan
**Directora de Desarrollo Académico e Innovación Educativa**

José Antonio Vergara Camacho
**Director de la Facultad de Contaduría y Administración Coatzacoalcos**

Mercedes Asunción Morán Urcelay
**Secretaria Académica de la Facultad de Contaduría y Administración Coatzacoalcos**

**Cuerpo Académico**
**Aplicación y enseñanza de la ingeniería de software**

Patricia Martínez Moreno
José Antonio Vergara Camacho
Gerardo Contreras Vega
Javier Pino Herrera
Irwing Alejandro Ibañez Castillo

# Prefacio

Mientras se iniciaban los trabajos de este libro, en el proceso de revisión de la literatura y el estado del arte, nos percatamos de la gran variedad de autores y referencias que existen sobre el tema. Sin embargo, la idea concebida para abordarlo era diferente, sin dejar de mencionar que las descripciones problemáticas de las instrucciones son transcripciones de diversos autores.

La idea inicial de este libro nace, a partir de un grupo de profesores que impartimos las asignaturas de introducción a la programación y programación en Java, quienes somos parte del Cuerpo Académico "Aplicación y enseñanza de la ingeniería de software" en la Licenciatura en Ingeniería de Software. Nos percatamos de la necesidad de crear y compilar una serie de ejercicios prácticos, básicamente un problemario y en este sentido, una inspiración para ello fue el autor reconocido Osvaldo Cairó con su Metodología de la Programación1 , así como diversos compendió de autores de la web.

Una vez concebida la idea significativa, la finalidad de este libro, es mostrar a todo aquel que desee incursionar en estos temas, desde edades tempranas y a los jóvenes que ingresan al programa educativo de la Licenciatura en Ingeniería de Software. De una manera sencilla y fácil, los primeros pasos hacia el área del desarrollo de software con un enfoque completamente práctico y poco teórico. Desde el análisis lógico matemático, pasando por la prueba de escritorio manual, y con la utilización del software libre PSeInt2, hasta llegar al desarrollo de un programa con el lenguaje de programación Java. Las bases para conceptualizar la lógica algorítmica la desarrollen bajo diversas estrategias y metodologías. Sin embargo, es necesario mencionar que se excluyen los Diagramas de Flujo de Datos (DFD)3; hemos comprobado, que sin ellos puede avanzarse en el área de la programación, también de manera efectiva.

---

1 Metodología de la programación, 3ra Edición - Osvaldo Cairo Battistutti.

2 Herramienta de pseudolenguaje en español para asistir a un estudiante en sus primeros pasos en programación. versión: 20200501. *http://pseint.sourceforge.net/*

3 Representación gráfica a través de un flujo de información

Los libros inevitablemente reflejan las experiencias, opiniones y puntos de vista de sus autores, aunque puede darse la situación por parte del lector en la que no la compartan, aún es sano, puesto que es reflejo de la diversidad de la temática y es parte importante para su evolución. No obstante, esperamos que aquellos interesados en la introducción a la programación, y los estudiantes de esta materia, puedan encontrar aquí un material de interés.

# Introducción

Este libro nace de la experiencia desde las aulas, desde el año 2017 se iniciaron los trabajos, cuando se lleva a cabo la creación o migración del programa educativo Sistemas Computacionales Administrativos al programa de la Licenciatura en Ingeniería de Software en la Universidad Veracruzana Campus Coatzacoalcos.

La meta principal es contribuir y facilitar a todos esos estudiantes y jóvenes que deseen incursionar en la programación de software. La estructura abordada consiste en mostrar la descripción lógica matemática a resolver, y entonces, resolverla a través de la aplicación del español estructurado o seudocódigo. Posteriormente, en cada uno de los contenidos de los tres capítulos, se buscó abordarlos de manera práctica en su totalidad.

La idea, es introducir a los estudiantes, primero con la metodología español estructurado con su prueba de escritorio para el razonamiento lógico matemático, donde se incluyen las estructuras algorítmicas, desde las básicas a las complejas. Posteriormente, se valida con la prueba de escritorio manual y con el software PseInt con la finalidad de simular la ejecución del funcionamiento del problema abordado paso a paso. Seguido, visualizar ese algoritmo ejecutado y funcionado correctamente, pero ahora en el lenguaje de programación Java, para reconocer las características propias y diferentes de un lenguaje de programación; básicamente, identificar poco a poco la sintaxis de Java.

De tal manera que el libro se encuentra estructurado en tres capítulos. Capítulo 1. Estructura lineal. Capítulo 2. Estructuras de selectivas. Capítulo 3. Estructuras de repetitivas y arreglos unidimensionales.

# Contenido

**Directorio** ..... 7

**Prefacio** ..... 8

**Introducción** ..... 10

**CAPÍTULO 1.**

**Estructura lineal** ..... 17

**ESTRUCTURA LINEAL**

Ejercicio 1 ..... 25
Ejercicio 2 ..... 29
Ejercicio 3 ..... 33
Ejercicio 4 ..... 38
Ejercicio 5 ..... 42
Ejercicio 6 ..... 47
Ejercicio 7 ..... 53
Ejercicio 8 ..... 58
Ejercicio 9 ..... 63
Ejercicio 10 ..... 67
Ejercicio 11 ..... 72

**CAPÍTULO 2.**

**Estructuras Selectivas** ..... 75

**ESTRUCTURA SELECTIVA SI (IF)**

Ejercicio 1 ..... 80
Ejercicio 2 ..... 84
Ejercicio 3 ..... 87

**ESTRUCTURA SELECTIVA SI / SI NO (IF / ELSE)**

Ejercicio 1 ..... 91
Ejercicio 2 ..... 97
Ejercicio 3 ..... 104
Ejercicio 4 ..... 108
Ejercicio 5 ..... 113
Ejercicio 6 ..... 120
Ejercicio 7 ..... 125
Ejercicio 8 ..... 130
Ejercicio 9 ..... 136
Ejercicio 10 ..... 140
Ejercicio 11 ..... 145

**ESTRUCTURA SELECTIVA SI ANIDADO**

Ejercicio 1 ..... 151
Ejercicio 2 ..... 164
Ejercicio 3 ..... 169
Ejercicio 4 ..... 178
Ejercicio 5 ..... 185
Ejercicio 7 ..... 195

**ESTRUCTURA SELECTIVA SEGÚN (SWITCH CASE)**

Ejercicio 1 ..... 202
Ejercicio 2 ..... 207
Ejercicio 3 ..... 212
Ejercicio 4 ..... 219
Ejercicio 5 ..... 224

**CAPÍTULO 3.**

## Estructuras repetitivas y arreglos unidimensionales ..... 231

**ESTRUCTURA MIENTRAS (WHILE)**

Ejercicio 1 ..... 237

Ejercicio 2 ..... 241
Ejercicio 3 ..... 246
Ejercicio 4 ..... 250
Ejercicio 5 ..... 256
Ejercicio 6 ..... 260
Ejercicio 7 ..... 263
Ejercicio 8 ..... 267

**CICLOS REPETITIVOS: ESTRUCTURA PARA (FOR)**

Ejercicio 1 ..... 272
Ejercicio 2 ..... 276
Ejercicio 3 ..... 279
Ejercicio 4 ..... 283
Ejercicio 5 ..... 286

**ESTRUCTURAS COMBINADAS (PARA + SI)**

Ejercicio 1 ..... 291

**ARREGLOS**

Ejercicio 1 ..... 296
Ejercicio 2 ..... 306
Ejercicio 3 ..... 316
Ejercicio 4 ..... 321
Ejercicio 5 ..... 327
Ejercicio 6 ..... 332
Ejercicio 7 ..... 337

**Referencias bibliográficas** .....341

# 1.

# Estructura lineal

## Introducción

El objetivo de este capítulo consiste en introducir al lector a los primeros pasos y estructura que conforma a lo que se llama algoritmo, a través de una serie de ejercicios prácticos. Al estudiar este capítulo:

- Conocerá a través de diversos ejercicios prácticos, los pasos para resolver un algoritmo, utilizando solo el español estructurado.
- Comprenderá la estructura secuencial y lineal de un algoritmo.
- Entenderá algunos conflictos básicos al resolver un algoritmo
- Conocerá cómo realizar una prueba de escritorio, símbolos, nombres, que se usan como ejemplos a lo largo del libro.
- Entenderá cómo manejar y ejecutar PseInt con algoritmos básicos.
- Identificará el mismo algoritmo resuelto en español estructurado y pseudocódigo, pero ahora con Java.

# CAPÍTULO 1.
# Estructura lineal

### ¿Qué es un algoritmo?

*"El término resolución de un problema se refiere al proceso completo que abarca desde la descripción inicial del problema hasta el desarrollo de un programa de computadora que lo resuelva. La resolución de un problema exige el diseño de un algoritmo que resuelva el problema propuesto."* **Luis Joyanes (2007).**

*"Indica que un algoritmo puede entenderse como una secuencia definida de reglas (operaciones) que especifica cómo producir un resultado (output) desde un input dado en un número finito de pasos."* **Shanker (1987).**

Entonces, se concluye que un algorítmo es:

*"Es llevar a cabo la solución de algo paso a paso, con orden y aplicar buenas prácticas para un mejor entendimiento y lectura y así resolver lo planteado (problema, necesidad, situación)."*

Cabe señalar, que el desarrollo de algoritmos es la base principal de un programa en cierto lenguaje de programación. Es entender, comprender en un lenguaje común y coloquial lo que se requiere, ello es lo principal y lo menos importante (por decirlo así) es la codificación. Lo anterior, en analogía con el proceso de desarrollo de software, en su etapa inicial de requerimientos, siendo esta, la etapa más delicada porque es necesario entender, comprender y plasmar todo lo que hará el software, así en este caso sucede lo mismo al realizar un algorítmo; cuando se comprende muy bien lo que se desea resolver en un algorítmo será más sencillo codificarlo en el lenguaje de programación de su preferencia.

De acuerdo con Joyanes: *"en la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o las computadoras. Un lenguaje de programación es sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo".*

En un algoritmo primero es tener claro lo que se pide y que se va a realizar, a partir de una instrucción o definición del problema en un texto. Y entonces iniciar con el análisis analizar, para ello se recomienda tomar papel y lápiz y es escribir en una hoja lo que se va entiendo hasta llegar a clarificarlo para luego ejecutarlo en una serie de pasos e instrucciones lo que significa en un algorítmo.

### Errores comunes

- Se ha observado que cuando se inicia el aprendizaje de algoritmos se pasa por alto el análisis, el aplicar la habilidad de razonamiento para enseguida encontrar la solución e iniciar el desarrollo de este.
- En otros casos se resuelven algoritmos sin errores de sintaxis, indentación o de buenas prácticas, en este sentido muy limpios y claros, sin embargo el algorítmo no resuelve lo planteado o solicitado.
- Caso contrario, nuevos aprendices se centran en resolver lo solicitado y planteado, sin aplicar las buenas prácticas, situación que dice "después lo aplico" y eso tiempo nunca llega, lo cual a la larga deja a un hueco como profesionista en el área de la programación.

La sugerencia es detenerse y clarificar lo que se pide, de tal manera:

1. Leer y tratar de comprender lo que pide la instrucción o el problema, se sugiere escribir en una hoja en blanco: valores, cifras, variables, incógnitas. Y además, cuestionarse: ¿Qué tenemos? ¿Qué necesitamos?
2. Identificar la estructura algorítmica a implementar para resolver el problema en cuestión.
3. Aplicar reglas de indentación (mejora la legibilidad del código), o instrucciones de buenas prácticas.
4. Escribir, diseñar, dibujar el algoritmo paso a paso, como se muestra en cada ejemplo de este libro.

Cabe mencionar que algunas personas cuando se inician en el aprendizaje de algoritmos y programas de computadora, les toma más tiempo que a

otros, esto no debe ser una preocupación sino una motivación, cada persona aprende a su propio ritmo. Así que no desesperes y sigue intentándolo una y otra vez, esto es cuestión de práctica, tiempo y constancia. Recuerda el dicho "Roma no se construyó en un día".

Por otro lado, Mark Zuckerberg, mencionó: "*Todo el mundo debe saber programar*". De tal manera, sea la carrera que estudies: filosofía, veterinaria, diseño, administración, otras, ello suma a la capacidad de resolver situaciones reales de día a día y a adquirir una visión "pragmática de la vida misma".

Con base en lo anterior, se observa que en el arte de la programación se desarrollan habilidades importantes en el ser humano, habilidades y valores cuando se trabaja con algoritmos, por mencionar: creatividad, perseverancia, tenacidad, paciencia, harás amigos en línea a través de contactarte en grupos y foros de discusión, encontrarás pasión y gusto, entre otros.

A partir de ahora, se explica cada concepto a entender para realizar los primeros algoritmos, conceptos como: buenas prácticas, variable, constantes, tipo de datos, prueba de escritorio, otros.

## Buenas prácticas en un algorítmo

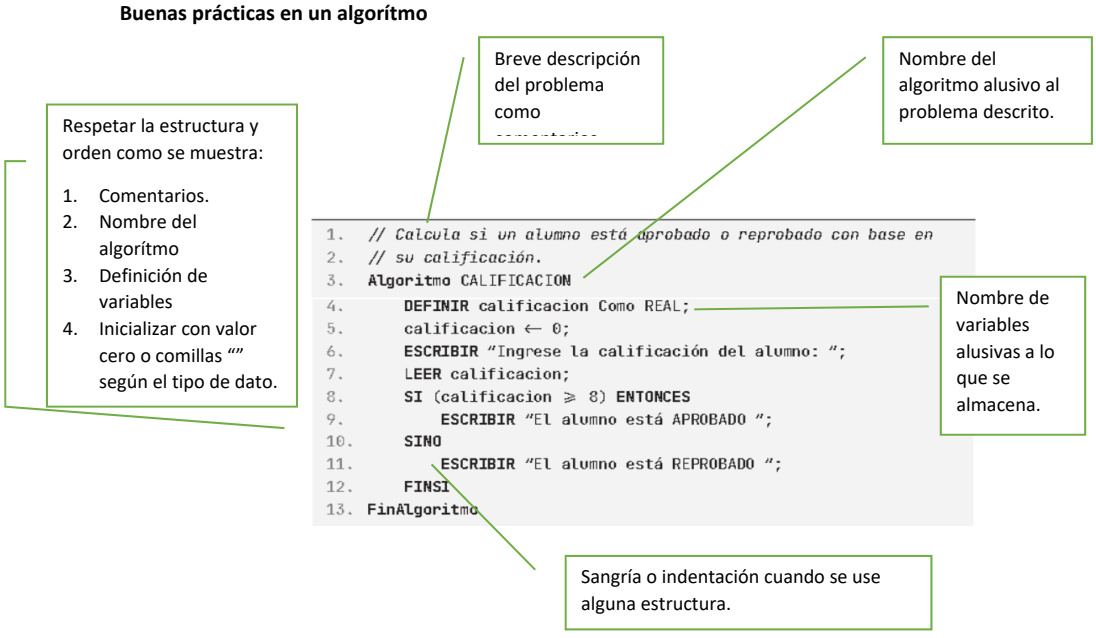



**Figura 1. Elementos de buenas prácticas en un algorítmo.**

### Variable

Hemos mencionado la palabra "variable" en donde su definición indica que es algo cambiante, que no es fijo. Y en efecto esa es la analogía de una variable y estas se definen con un nombre corto y un nombre alusivo de lo que almacenará, estas van a almacenar valores (numéricos o de texto), y esos valores pueden cambiar pero siempre respetando el mismo tipo de dato. Como en la figura 1, donde se definió la variable: *CALIFICACION* de tipo real, la cual almacenará valores de calificaciones con decimales. Los valores pueden cambiar durante el recorrido del algoritmo lo que significa que inicialmente entra con un valor de 8.3 de calificación y luego alterarlo o cambiar ese valor durante el mismo proceso del algoritmo y entonces tomaría otro valor diferente.

### Constante

Por ejemplo son aquellas, que no cambian durante el proceso del algoritmo, por ejemplo: el valor de PI: 3.1416. Entonces, la constante de nombre *PI* de tipo real, almacenará siempre su mismo valor y de tipo de dato: real (por manejar punto decimal).

### Tipo de datos

En repetidas ocasiones se ha mencionado la palabra "tipo de dato" de una variable o constante, y ¿qué son estos? Son aquellos que dependiendo de la información que se almacenará, se clasifican en:

| Tipo de dato | Ejemplo |
|---|---|
| Entero | -102<br>5<br>2225 |
| Real | -33.3<br>5.1<br>8963.4 |

| Tipo de dato | Ejemplo |
|---|---|
| Cadena de caracteres | “Hola mundo”<br>“Pedro Ruíz”<br>“uva”<br>“Sí” |
| Booleno<br>(2 valores) | True/Falso<br>Verdadero/Falso |
| Fecha | 01/01/2020<br>01 enero del 2020 |

**Tabla 1. Tipos de datos**

### Prueba de escritorio

En los ejercicios mostrados de este libro se incluyó la prueba de escritorio. Y ¿qué es la prueba de escritorio?, es el funcionamiento simulado de un algoritmo el cual se lleva a cabo de manera manual, ayuda a determinar la validez de este. Al realizar la prueba de escritorio se manipulan variables las cuales se siguen paso a paso, lo que significa que de manera manual ejecuto (línea por línea) el algoritmo de inicio a fin y los valores que se generan de esa ejecución o corrida de algoritmo (ejecutar es sinónimo de correr un algoritmo o programa de computadora) en otras palabras, consiste en generar una tabla con tantas columnas como variables tenga el algoritmo y seguir las instrucciones asignando valores solicitados.

Cabe señalar, como ya se mencionó, la prueba de escritorio sirve para identificar si se ejecuta correctamente el algoritmo por ello se recomienda realizar más de una ejecución o corrida con valores correctos solicitados y con valores incorrectos, y esto asegura aún más la veracidad y validez de este.

### Español estructurado. ¿Qué significa?

Son instrucciones en nuestro idioma nativo, el español, donde se introducen las diversas estructuras algorítmicas que nos llevan paso a paso bajo cierto orden para que sea muy fácil de entender y se desarrolla la solución de un problema de manera manual, dando paso al pseudocódigo y posteriormente a codificarlo. Es el primer acercamiento para desarrollar un algoritmo.

### ¿Qué es el pseudocódigo y porque usar PSeInt?

El pseudocódigo (o falso lenguaje) se aplica a partir de ciertos lenguajes de programación aunque no son propiamente lenguajes de programación, por ello "pseudo" en donde se edita y plasma el algoritmo bajo la estructura y sintaxis propia del lenguaje de programación. La sintaxis son reglas técnicas que se aplican al plasmar y editar algoritmos y al no aplicarse es imposible que el algoritmo o programa se ejecute de manera correcta marcando error de ejecución, cuando se realicen los algoritmos en PSeInt esto se podrá confirmar.

PSeInt es una herramienta desarrollada para el aprendizaje de algoritmos, que ayuda principalmente a estudiantes en aprender los fundamentos de la programación y al desarrollo de la lógica matemática, es la antesala al mundo de la programación.

En PseInt, puedes usar todas las estructuras algorítmicas explicadas en este libro, y también "funciones" que se utilizan de manera similar que en otros lenguajes. Las funciones son scripts (líneas de código) almacenadas en un archivo que lleva un nombre (ver tabla 2) y resuelve cierta acción matemática o lógica sólo con el hecho de colocar el nombre de la función seguido de la variable que desees aplicar dicha acción de la función. Éstas no ayudan a evitar ciertos cálculos que tendríamos que plasmar en nuestro algoritmo o programa.

| Función | Significado |
|---|---|
| MOD | Módulo (resto de la división entera) |
| RC o RAIZ | Raíz cuadrada |
| ABS | Valor absoluto |
| EXP | Exponente |
| TRUNC | Parte entera |
| REDON | Entero más cercano |
| AZAR | Entero aleatorio entre 0 y -1 |

**Tabla 2. Funciones de PSeInt**

Existen más funciones que una vez adentrado en este mundo lo descubrirás poco a poco. Una vez comprendido lo anterior, te encuentras listo para resolver una serie de problemas básicos, se inicia con problemas graduales, por ahora sólo resolverás algoritmos simples sin ninguna estructura a los que llamamos algoritmos lineales. El consejo es que observes la estructura,

su secuencia con base en la figura 1 y leas detenidamente lo que se pide, antes de cualquier otra acción.

Cada algoritmo está conformado por su leyenda o descripción a resolver, el algoritmo en español estructurado, su prueba de escritorio, el mismo algoritmo desarrollado en pseudocódigo en PSeInt y posteriormente codificado en Java. Lo que demuestra un panorama general y el proceso completo de llegar a codificar un programa, podrás comparar identificar y ver las diferencias de un resultado con otro. ¡Bienvenido!

# ESTRUCTURA LINEAL

## Ejercicio 1

Una persona desea invertir su capital en un banco y desea saber cuánto dinero ganará después de un mes si el banco le pagará intereses del 2% mensual, cree un algoritmo para solucionarlo.

```
1.  GANANCIA_INVERSION
2.  interes: Real
3.  dias, capital, ganancia: Entero
4.  Inicio
5.  interes←0.02;
6.  dias←0;
7.  capital←0;
8.  ganancia←0;
9.  Escribir “teclea capital”;
10. Leer capital;
11. Escribir “teclea dias_mes”;
12. Leer dias;
13. ganancia←capital * dias * interes;
14. Escribir “la ganancia es:”, ganancia;
15. Fin
```

### PRUEBA DE ESCRITORIO

| GANANCIA_INVERSION | | | |
|---|---|---|---|
| interes | dias | capital | ganancia |
| 0 | 0 | 0 | 0 |
| 0.02 | 30 | 4200 | 2520 |

**ALGORITMO EN PseInt**
**PRUEBA DE ESCRITORIO EN PseInt**

```
1.  // Una persona desea invertir su capital en un banco y desea
    saber
2.  // cuánto dinero ganará después, de un mes si el banco le pagará
3.  // intereses del 2% mensual, cree un algoritmo para
    solucionarlo.
4.  Algoritmo Ganancia_inversion
5.      DEFINIR interes Como Real;
6.      DEFINIR dias, capital, ganancia Como Entero;
7.      ESCRIBIR “Ingrese el monto a invertir: “;
8.      LEER capital;
9.      ESCRIBIR “Ingrese el número total de días del mes a
       considerar: “;
10.    LEER dias;
11.     interes ← 0.02;
12.     ganancia ← (capital * dias) * interes;
13.     ESCRIBIR “La ganancia por cobrar después del mes es de: “;
14.     ESCRIBIR ganancia;
15. FinAlgoritmo
```

## Ejecutar Paso a Paso

Inicia la ejecución del algoritmo, inicializando las variables definidas y solicitando la introducción del valor del monto que se desea invertir.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el monto a invertir:
3.  > 4200
```

Una vez ingresado el valor del capital, nos solicita ingresar el número de días del mes a considerar. Ingresamos su valor.

```
1.  Ingrese el número total de días del mes a considerar:
2.  > 30
```

El programa realiza los cálculos y se despliega en pantalla el resultado. Finaliza la ejecución del programa.

```
La ganancia por cobrar después del mes es de:
2520
*** Ejecución Finalizada. **
```

**Ejecutar**

```
*** Ejecución Iniciada. ***
Ingrese el monto a invertir:
> 4200
Ingrese el número total de días del mes a considerar:
> 31
La ganancia por cobrar después del mes es de:
2976
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
/* Una persona desea invertir su capital en un banco y desea saber cuánto dinero
 * ganará después de un mes si el banco le pagará intereses del 2% mensual.
 */
package EstructuraSecuencial;
import java.util.Scanner;
public class Ganancia_Inversion {
  public static void main(String[] args) {
    Scanner entrada = new Scanner(System.in);
      int dias;
      double capital, ganancia, interes;

      System.out.println("Teclea Capital: ");
      capital = entrada.nextDouble();
      System.out.println("Teclea Días Mes: ");
      dias = entrada.nextInt()
```

```
16.       interes = 0.02;
17.       ganancia = capital * dias * interes;
18.       System.out.println(“La ganancia es: “ + ganancia);
19.     }
20. }
```

## Resultado Java

```
1.  run:
2.  Teclea Capital:
3.  5000
4.  Teclea Días Mes:
5.  1
6.  La ganancia es: 100.0
7.  BUILD SUCCESSFUL (total time: 6 seconds)
```

# Ejercicio 2

Un constructor sabe que necesita 0.5 metros cúbicos de arena por metro cuadrado de revoque a realizar. Escriba un algoritmo que le permita obtener la cantidad de arena necesaria para revocar una pared cualquiera, según sus medidas (largo y alto) expresadas en metros.

```
1.  CANT_ARENA
2.  metros←0.5;
3.  largo←0;
4.  ancho←0;
5.  Escribir “teclea largo de la pared”;
6.  Leer largo;
7.  Escribir “teclea ancho de la pared”;
8.  Leer ancho;
9.  arena←largo*ancho*metros;
10. Escribir “la cantidad de arena es: “, arena;
11. Fin
```

## PRUEBA DE ESCRITORIO

| CANT_ARENA | | | |
|---|---|---|---|
| metros | largo | ancho | arena |
| 0.5 | 0 | 0 | 0 |
| 0.5 | 10 | 5 | 25 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Escriba un algoritmo que le permita obtener la cantidad de
    arena
2.  // necesaria para revocar una pared cualquiera, según sus
    medidas (largo
3.  // y alto) expresadas en metros.
```

```
4.  Algoritmo CANT_ARENA
5.      DEFINIR metros, arena, largo, ancho Como Real;
6.      metros ← 0.5;
7.      ESCRIBIR “Ingrese el largo de la pared en metros: “;
8.      LEER largo;
9.      ESCRIBIR “Ingrese el ancho de la pared en metros: “;
        LEER ancho;
10.     arena ← (largo * ancho) * metros;
11.     ESCRIBIR “La cantidad de arena necesaria es de: “;
12.     ESCRIBIR arena;
13. FinAlgoritmo
```

## Ejecutar Paso a Paso

Al iniciar la ejecución del programa, se inicializan las variables definidas y se solicita la introducción del valor del largo de la pared en metros.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el largo de la pared en metros:
3.  > 10
```

Posteriormente, el valor del ancho de la pared en metros.

```
1.  Ingrese el ancho de la pared en metros:
2.  > 5
```

Se realizan los cálculos especificados y se imprime en pantalla el resultado. Finaliza la ejecución del programa.

```
1.  La cantidad de arena necesaria es de:
2.  25
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el largo de la pared en metros:
> 20
Ingrese el ancho de la pared en metros:
> 18
La cantidad de arena necesaria es de:
180
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Un constructor sabe que necesita 0,5 metros cúbicos de arena por metro cuadrado
 * de revoque a realizar. Escribir un algoritmo que le permita obtener la cantidad
 * de arena necesaria para revocar una pared cualquiera según sus medidas (largo y alto)
 * expresadas en metros.
 */
package EstructuraSecuencial;

import java.util.Scanner;

 public class Cant_Arena {

    public static void main(String[] args) {

        Scanner entrada = new Scanner(System.in);
        double metros, largo, ancho, arena;

        System.out.println("Ingrese el largo de la pared en metros: ");
        largo = entrada.nextDouble();
        System.out.println("Ingrese el ancho de la pared en metros: ");
        ancho = entrada.nextDouble();

```

```
22.         metros = 0.5;
23.         arena = largo * ancho * metros;
24.
25.         System.out.println(“La cantidad de arena necesaria es de: “ + arena);
26.
27.     }
28.  }
```

## Resultado Java

```
1.  run:
2.  Ingrese el largo de la pared en metros:
3.  10
4.  Ingrese el ancho de la pared en metros:
5.  5
6.  La cantidad de arena necesaria es de: 25.0
7.  BUILD SUCCESSFUL (total time: 5 seconds)
```

# Ejercicio 3

Construya el algoritmo tal que, dado el radio, la generatriz y la altura de un cono, calcule e imprima el área de la base, el área lateral, el área total y su volumen.

**Consideraciones:**

- El área de la base se calcula aplicando la siguiente fórmula: $AB = \pi * RADIO^2$
- El área lateral se calcula: $AL = \pi * RADIO * GENE$ RATRIZ
- El área total se calcula como: $AT = AB + A$
- El volumen se calcula de la siguiente forma: $VOL = \frac{1}{3} * AB * ALTURA$

```
1.  AREA_CONO
2.  radio, generatriz, altura, aBase, aLado, aTotal, volumen:  Real
3.  Inicio
4.  radio ←0
5.  generatriz ←0
6.  altura ← 0
7.  aBase ← 0
8.  aLado ← 0
9.  aTotal ← 0
10. volumen ← 0
11. Escribir “Ingrese el valor del radio:”
12. Leer radio
13. Escribir “Ingrese el valor de la generatriz:”
14. Leer generatriz
15. Escribir “Ingrese la altura del cono:”
16. Leer altura
17. aBase ← PI * (radio * radio)
18. aLado ← PI * radio * generatriz
19. aTotal ← aBase + aLado
20. volumen ← (1 / 3) * aBase * Altura
21. Escribir “El área de la base es:”, aBase
22. Escribir “El área lateral es:”, aLado
23. Escribir “El área total es:”, aTotal
24. Escribir “El volumen es:”, volumen
25. Fin
```

## PRUEBA DE ESCRITORIO

| radio | generatriz | altura | aBase | aLado | aTotal | volumen |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 5 | 6 | 314.15 | 157.07 | 471.23 | 628.31 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Construya el algoritmo tal que, dado el radio, la generatriz
    y la
2.  // altura de un cono, calcule e imprima el área de la base, el
    área
3.  // lateral, el área total y su volumen.
4.  Algoritmo AREA_CONO
5.      DEFINIR radio, generatriz, altura, aBase, aLado, aTotal,
    volumen,
6.        Como REAL;
7.      radio ← 0;
8.      generatriz ← 0;
9.      altura ← 0;
10.     aBase ← 0;
11.     aLado ← 0;
12.     volumen ← 0;
13.     ESCRIBIR “Ingrese el radio del cono: “;
14.     LEER radio;
15.     ESCRIBIR “Ingrese la generatriz del cono: “;
        LEER generatriz;
16.     ESCRIBIR “Ingrese la altura del cono: “;
17.     LEER altura;
18.     aBase ← PI * (radio * radio);
19.     aLado ← PI * radio * generatriz;
20.     aTotal ← ab * al; //AT= PI * r * (gen + r)
21.     volumen ← (1 / 3) * aBase * altura;
22.     ESCRIBIR “El área de la base es: “;
23.     ESCRIBIR aBase;
24.     ESCRIBIR “El área lateral es: “;
```

```
25.     ESCRIBIR aLado;
26.     ESCRIBIR “El área total es: “;
27.     ESCRIBIR aTotal;
28.     ESCRIBIR “El volumen es: “;
29.     ESCRIBIR volumen;
30. FinAlgoritmo
```

## Ejecutar Paso a Paso

Inicia la ejecución del programa, nos pide ingresar el valor del radio del cono.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el radio del cono:
3.  > 10
```

Nos solicita ingresar el valor de la generatriz del cono

```
1.  Ingrese la generatriz del cono:
2.  > 5
```

Nos solicita ingresar el valor de la altura del cono

```
1.  Ingrese la altura del cono:
2.  > 6
```

Realiza los cálculos según las fórmulas y datos dados. Nos imprime en consola el valor de cada uno de los resultados.

```
1.  El área de la base es:
2.  314.159265359
3.  El área lateral es:
4.  157.0796326795
5.  El área total es:
6.  471.2388980385
7.  El volumen es:
```

```
628.318530718
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el radio del cono:
> 10
Ingrese la generatriz del cono:
> 5
Ingrese la altura del cono:
> 6
El área de la base es:
314.159265359
El área lateral es:
157.0796326795
El área total es:
471.2388980385
El volumen es:
628.318530718
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Construya el algoritmo tal que, dado el radio, la generatriz y   la altura de un cono,
* calcule e imprima el área de la base, el área lateral, el área total y su volumen.
*/
package EstructuraSecuencial;

import java.util.Scanner;

public class Area_Cono {

     public static void main(String[] args) {
         Scanner entrada = new Scanner(System.in);
         double radio, generatriz, altura;
         double aBase, aLado, aTotal, volumen;
         final double PI = Math.PI;
```

```
15.
16.          System.out.println(“Ingresa el radio del cono: “);
17.          radio = entrada.nextDouble();
18.          System.out.println(“Ingresa la Generatriz del cono: “);
19.          generatriz = entrada.nextDouble();
20.          System.out.println(“Ingresa la altura del cono: “);
21.          altura = entrada.nextDouble();
22.
23.          aBase = PI * (radio * radio);
24.          aLado = PI * radio * generatriz;
25.          aTotal = aBase + aLado;
26.          volumen = (aBase * altura) / 3;
27.
28.          System.out.println(“El área de la base es: “ + aBase);
29.          System.out.println(“El área lateral es: “ + aLado);
30.          System.out.println(“El área Total es: “ + aTotal);
31.          System.out.println(“El Volumen total es: “ + volumen);
32.
33.      }
34.  }
```

## Resultado Java

```
1.  run:
2.  Ingresa el radio del cono:
3.  10
4.  Ingresa la Generatriz del cono:
5.  5
6.  Ingresa la altura del cono:
7.  6
8.  El área de la base es: 314.1592653589793
9.  El área lateral es: 157.07963267948966
10. El área Total es: 471.23889803846896
11. El Volumen total es: 628.3185307179587
12. BUILD SUCCESSFUL (total time: 5 seconds)
```

## Ejercicio 4

Construya el algoritmo tal que, dado el radio de una esfera, calcule e imprima el área y su volumen.

Consideraciones:

- El área de una esfera la calculamos de esta forma: $Área = 4 * \pi * (radio)^2$
- El volumen como: $Volumen = \frac{4}{3} * \pi * radio^3$

```
1.  ESFERA
2.  area, radio, volumen: Real
3.  Inicio
4.  area←0
5.  volumen←0
6.  radio←0
7.  Escribir “Ingrese el valor del radio: “
8.  Leer radio
9.  area ← (4 * PI) * (radio * radio)
10. volumen ← ((4 * PI) * ((radio * radio) * radio)) /3
11. Escribir “El valor del área de la esfera es:”
12. Escribir area
13. Escribir “El valor del volumen de la esfera es:”
14. Escribir volumen
15. Fin
```

### PRUEBA DE ESCRITORIO

| ESFERA | | |
|---|---|---|
| radio | area | volumen |
| 0 | 0 | 0 |
| 5 | 314.159265359 | 523.5987755983 |

**ALGORITMO EN PseInt**
**PRUEBA DE ESCRITORIO EN PseInt**

```
1.  // Construya el algoritmo tal que, dado el radio de una esfera,
    calcule
2.  // e imprima el área y su volumen.
3.  Algoritmo ESFERA
4.      DEFINIR area, radio, volumen prueba Como REAL;
5.      ESCRIBIR “Ingrese el valor del radio de la esfera: “;
6.      LEER radio;
7.      area ← (4 * PI) * (radio * radio);
8.      volumen ← (4 * PI) * ((radio * radio) * radio) / 3;
9.      ESCRIBIR “ El resultado del área de la esfera es: “;
10.     ESCRIBIR area;
11.     ESCRIBIR “ El resultado del Volumen de la esfera es:”;
12.     ESCRIBIR volumen;
13. FinAlgoritmo
```

## Ejecutar Paso a Paso

Inicia la ejecución del algoritmo. Ingresamos el valor del radio de la esfera para los cálculos posteriores.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el valor del radio de la esfera:
3.  > 5
```

Se llevan a cabo los cálculos considerando las fórmulas y datos ingresados, se imprimen en pantalla los resultados.

```
1.  El resultado del área de la esfera es:
2.  314.159265359
3.  El resultado del Volumen de la esfera es:
4.  523.5987755983
```

Finaliza la ejecución del algoritmo.

```
5.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el valor del radio de la esfera:
3.  > 5
4.  El resultado del área de la esfera es:
5.  314.159265359
6.  El resultado del Volumen de la esfera es:
7.  523.5987755983
8.  *** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
1.  /* Construya el algoritmo tal que, dado el radio de una esfera,
2.  * calcule e imprima el área y su volumen.
3.  */
4.  package EstructuraSecuencial;
5.
6.  import java.util.Scanner;
7.
8.  public class Esfera {
9.      public static void main(String[] args) {
10.      Scanner entrada = new Scanner(System.in);
11.      double area, volumen, radio;
12.      final double PI = Math.PI;
13.
14.      System.out.println("Introduzca el valor del radio: ");
15.      radio = entrada.nextDouble();
16.
17.      area = (4 * PI) * (radio * radio);
18.      volumen = ((4 * PI) * ((radio * radio) * radio)) / 3;
19.
```

```
20.      System.out.println(“El resultado del área de la esfera es: “ + area);
21.      System.out.println(“El resultado del Volumen de la esfera es: “ + volumen);
22.      }
23.  }
```

## Resultado Java

```
1.  run:
2.  Introduzca el valor del radio:
3.  5
4.  El resultado del área de la esfera es: 314.1592653589793
5.  El resultado del Volumen de la esfera es: 523.5987755982989
6.  BUILD SUCCESSFUL (total time: 3 seconds)
```

## Ejercicio 5

Construya el algoritmo tal que, dado como dato el lado de un hexaedro o cubo, calcule el área de la base, el área lateral, el área total y el volumen.

**Consideraciones:**

- Para calcular el área de la base aplicamos la siguiente fórmula: $AB = L^2$
- Para calcular el área lateral hacemos: $AL = 4 * L^2$
- Para calcular el área total hacemos: $AT = 6 * L^2$
- Para calcular el volumen hacemos: $V = L^3$

```
1.  HEXA_CUBO
2.  lado, aBase, aLado, aTotal, volumen: Real
3.  Inicio
4.  lado←0
5.  aBase←0
6.  aLado←0
7.  aTotal←0
8.  volumen←0
9.  Escribir “Ingrese el valor del lado del hexaedro o cubo:”
10. Leer lado
11. aBase ← lado^2
12. aLado ← 4 * (lado*lado)
13. aTotal ← 6 * (lado*lado)
14. volumen ← lado*lado*lado
15. Escribir “El valor del área de la base:”
16. Escribir aBase
17. Escribir “El valor del área lateral de la base:”
18. Escribir aLado
19. Escribir “El valor del área total de la base:”
20. Escribir aTotal
21. Escribir “El volumen de la esfera es:”
22. Escribir volumen
23. Fin
```

## PRUEBA DE ESCRITORIO

| HEXA_CUBO | | | | |
|---|---|---|---|---|
| lado | aBase | aLado | aTotal | volumen |
| 0 | 0 | 0 | 0 | 0 |
| 10 | 100 | 400 | 600 | 1000 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Construya el algoritmo tal que, dado como dato el lado de un
    hexaedro
2.  // o cubo, calcule el área de la base, el área lateral, el área
    total // y el volumen.
3.  Algoritmo HEXA_CUBO
4.      DEFINIR lado, aBase, aLado, aTotal, volumen Como REAL;
5.      ESCRIBIR “Ingrese el valor del lado del Hexaedro o Cubo: “;
6.      LEER lado;
7.      aBase ← lado * lado;
8.      aLado ← 4 * (lado * lado);
9.      aTotal ← 6 * (lado * lado);
10.     volumen ← lado * lado * lado;
11.     ESCRIBIR “El área de la base es: “;
12.     ESCRIBIR aBase;
13.     ESCRIBIR “El área lateral de la base es: “;
14.     ESCRIBIR aLado;
15.     ESCRIBIR “El área total de la base es: “;
16.     ESCRIBIR aTotal;
17.     ESCRIBIR “El volumen es: “;
18.     ESCRIBIR volumen;
19. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Ejecución iniciada

Se le solicita al usuario el valor del lado de Hexaedro o cubo, el dato ingresado se almacena en la variable l.

```
*** Ejecución Iniciada. ***
Ingrese el valor del lado del Hexaedro o Cubo:
> 10
```

Se realizan los cálculos mediante las fórmulas previamente registradas y el valor ingresado por el usuario. Imprime en pantalla los resultados de las operaciones y finaliza la ejecución del algoritmo.

```
El área de la base es:
100
El área lateral de la base es:
400
El área total de la base es:
600
El volumen es:
1000
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el valor del lado del Hexaedro o Cubo:
> 10
El área de la base es:
100
El área lateral de la base es:
400
El área total de la base es:
600
El volumen es:
1000
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Construya el algoritmo tal que dado como dato el lado de un hexaedro o cubo,
* calcule el área de la base, al área lateral, el área total y el volumen.
*/
 package EstructuraSecuencial;

 import java.util.Scanner;

 public class Hexa_Cubo {

     public static void main(String[] args) {
         Scanner entrada = new Scanner(System.in);
         double lado, aBase ,aLado, aTotal, volumen;

         System.out.println("Ingrese el valor del lado del Hexaedro o Cubo: ");
         lado = entrada.nextDouble();

         aBase = lado * lado;
         aLado = 4 * (lado * lado);
         aTotal = 6 * lado * lado;
         volumen = lado * lado * lado;

         System.out.println("El Área de la Base es: " + aBase);
         System.out.println("El Área Lateral es: " + aLado);
         System.out.println("El Área Total es: " + aTotal);
         System.out.println("El Volumen es: " + volumen);

     }

 }
```

## Resultado Java

```
run:
Ingrese el valor del lado del Hexaedro o Cubo:
```

```
3.  10
4.  El Área de la Base es: 100.0
5.  El Área Lateral es: 400.0
6.  El Área Total es: 600.0
7.  El Volumen es: 1000.0
8.  BUILD SUCCESSFUL (total time: 3 seconds)
```

# Ejercicio 6

Construya el algoritmo tal que, dadas las coordenadas de los puntos P1, P2 y P3 que corresponden a los vértices de un triángulo, calcule su perímetro.

Donde:

- X1 y Y1 representan las coordenadas en el eje de las X y las Y, del punto P1.
- X2 y Y2 expresan las coordenadas en el eje de las X y las Y, del punto P2.
- X3 y Y3 representan las coordenadas en el eje de las X y las Y, del punto P3.

**Consideraciones:**

Para calcular la distancia entre dos puntos P1 y P2 hacemos:

$$D = \sqrt{(X1 - X2)^2 + (Y1 - Y2)^2}$$

```
TRIANGULO
x1, x2, x3, y1, y2, y3, distancia1, distancia2, distancia3,
perimetro: REAL
Inicio
ESCRIBIR “Introduce el valor de x1: “
LEER x1
ESCRIBIR “Introduce el valor de x2: “
LEER x2
ESCRIBIR “Introduce el valor de x3: “
LEER x3
ESCRIBIR “Introduce el valor de y1: “
LEER y1
ESCRIBIR “Introduce el valor de y2: “
LEER y2
ESCRIBIR “Introduce el valor de y3: “
LEER y3
distancia1← RAIZ ((x1-x2) ^2 + (y1-y2) ^2)
distancia2← RAIZ ((x2-x3) ^2 + (y2-y3) ^2)
distancia3← RAIZ ((x3-x1) ^2 + (y3-y1) ^2)
perimetro← distancia1 + distancia2 + distancia3
```

```
28. ESCRIBIR “El perímetro del triángulo es: “
29. ESCRIBIR perimetro
30. Fin
```

## PRUEBA DE ESCRITORIO

| TRIANGULO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| x1 | y1 | x2 | y2 | x3 | y3 | distancia1 | distancia2 | distancia3 | perímetro |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 10 | 23 | 15 | 11 | 23 | 32 | 13 | 22 | 16 | 51.2835 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Construya el algoritmo tal que, dadas las coordenadas de los
    puntos
2.  // P1, P2 y P3 que corresponden a los vértices de un triángulo,
    calcule
3.  // su perímetro.
4.  Algoritmo TRIANGULO
5.      DEFINIR x1, x2, x3, y1, y2, y3, distancia 1, distancia2,
    perimetro
6.       Como REAL;
7.      ESCRIBIR “Introduce el valor de x1: “;
8.      LEER x1;
9.      ESCRIBIR “Introduce el valor de x2: “;
10.     LEER x2;
11.     ESCRIBIR “Introduce el valor de x3: “;
12.     LEER x3;
13.     ESCRIBIR “Introduce el valor de y1: “;
14.     LEER y1;
15.     ESCRIBIR “Introduce el valor de y2: “;
16.     LEER y2;
```

```
17.     ESCRIBIR “Introduce el valor de y3: “;
18.     LEER y3;
19.     distancia1 ← RAIZ ((x1 - x2) ^ 2 + (y1 - y2) ^ 2);
20.     distancia2 ← RAIZ ((x2 - x3) ^ 2 + (y2 - y3) ^ 2);
21.     distancia3 ← RAIZ ((x3 - x1) ^ 2 + (y3 - y1) ^ 2);
22.     perimetro ← distancia1 + distancia2 + distancia3;
23.     ESCRIBIR “El perímetro del triángulo es: “;
24.     ESCRIBIR perimetro;
25. FinAlgoritmo
```

## Ejecutar Paso a Paso

Inicia la ejecución del programa.

```
26. *** Ejecución Iniciada. ***
```

Se inicializan las variables y nos solicita en pantalla la introducción de los valores de cada coordenada.

```
1.  Ingrese el valor de x1:
2.  > 22
3.  Ingrese el valor de x2:
4.  > 28
5.  Ingrese el valor de x3:
6.  > 17
7.  Ingrese el valor de y1:
8.  > 13
9.  Ingrese el valor de y2:
10. > 23
11. Ingrese el valor de y3:
12. > 15
```

Una vez ingresados todos los valores solicitados, el algoritmo realiza los cálculos e imprime en pantalla el resultado.

```
El perímetro del triángulo es:
30.6485391056
*** Ejecución Finalizada. ***
```

## Ejecutar

```
 *** Ejecución Iniciada. ***
Ingrese el valor de x1:
> 22
Ingrese el valor de x2:
> 28
Ingrese el valor de x3:
> 17
Ingrese el valor de y1:
> 13
Ingrese el valor de y2:
> 23
Ingrese el valor de y3:
> 15
El perímetro del triángulo es:
30.6485391056
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
 package EstructuraSecuencial;

 import java.util.Scanner;

 public class Triangulo {

   public static void main(String[] args) {
     Scanner entrada = new Scanner(System.in);
     double x1, x2, x3, y1, y2, y3, distancia1, distancia2,
distancia3, perimetro;

```

```
11.      System.out.println(“Ingresa la coordenada de x1: “);
12.      x1 = entrada.nextDouble();
13.      System.out.println(“Ingresa la coordenada de y1: “);
14.      y1 = entrada.nextDouble();
15.      System.out.println(“Ingresa la coordenada de x2: “);
16.      x2 = entrada.nextDouble();
17.      System.out.println(“Ingresa la coordenada de y2: “);
18.      y2 = entrada.nextDouble();
19.      System.out.println(“Ingresa la coordenada de x3: “);
20.      x3 = entrada.nextDouble();
21.      System.out.println(“Ingresa la coordenada de y3: “);
22.      y3 = entrada.nextDouble();
23.
24.      distancia1 = Math.sqrt( Math.pow( (x1 - x2), 2) + Math.pow(
    (y1 - y2), 2) );
25.      distancia2 = Math.sqrt( Math.pow( (x2 - x3), 2) + Math.pow(
    (y2 - y3), 2) );
26.      distancia3 = Math.sqrt( Math.pow( (x3 - x1), 2) + Math.pow(
    (y3 - y1), 2) );
27.      perimetro = distancia1 + distancia2 + distancia3;
28.
29.      System.out.println(“El perímetro del Triangulo es: “ +
    perimetro);
30.      }
31.  }
```

## Resultado Java

```
1.  run:
2.  Ingresa la coordenada de x1:
3.  22
4.  Ingresa la coordenada de y1:
5.  13
6.  Ingresa la coordenada de x2:
7.  28
8.  Ingresa la coordenada de y2:
9.  23
10. Ingresa la coordenada de x3:
11. 17
```

```
12. Ingresa la coordenada de y3:
13. 15
14. El perímetro del Triangulo es: 30.64853910556055
15. BUILD SUCCESSFUL (total time: 41 seconds)
```

# Ejercicio 7

Construya el algoritmo tal que, dadas las coordenadas de los puntos P1, P2 y P3 que corresponden a los vértices de un triángulo, calcule su superficie.

Donde:

- X1 y Y1 representan las coordenadas en el eje de las X y las Y, del punto P1.
- X2 y Y2 expresan las coordenadas en el eje de las X y las Y, del punto P2.
- X3 y Y3 representan las coordenadas en el eje de las X y las Y, del punto P3.

**Consideraciones:**

- Para calcular el área de un triángulo dadas de las coordenadas de los vértices que la componen, debemos aplicar la siguiente formula.

$$\acute{A}REA = \frac{1}{2} * \left| X1 * (Y2 - Y3) + X2 * (Y3 - Y1) + X3 * (Y1 - Y2) \right|$$

O bien, esta otra:

$$\acute{A}REA = \frac{1}{2} * \left| (X2 - X1) * (Y3 - Y1) - (X3 - X1) * (Y2 - Y1) \right|$$

```
1.  TRIANGULO_SUPERFICIE
2.  x1, x2, x3, y1, y2, y3, area: Real
3.  ESCRIBIR “Introduce el valor de x1: “
4.  LEER x1
5.  ESCRIBIR “Introduce el valor de x2: “
6.  LEER x2
7.  ESCRIBIR “Introduce el valor de x3: “
8.  LEER x3
9.  ESCRIBIR “Introduce el valor de y1: “
10. LEER y1
11. ESCRIBIR “Introduce el valor de y2: “
12. LEER y2
13. ESCRIBIR “Introduce el valor de y3: “
14. LEER y3
```

```
15. area←(((x1 * y2) +(x2 * y3) + (x3 * y1)) - ((x1 * y3) + (x3 *
    y2) + (x2 * y1))) / 2
16. ESCRIBIR “El área del triángulo es: “
17. ESCRIBIR area
18. Fin
```

## PRUEBA DE ESCRITORIO

| TRIANGULO_SUPERFICIE | | | | | | |
|---|---|---|---|---|---|---|
| x1 | x2 | x3 | y1 | y2 | y3 | área |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 5 | 7 | 12 | 15 | 11 | 27 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Construya el algoritmo tal que, dadas las coordenadas de los
    puntos
2.  // P1, P2 y P3 que corresponden a los vértices de un triángulo,
    calcule // su superficie.
3.  Algoritmo TRIANGULO_SUPERFICIE
4.      DEFINIR x1, x2, x3, y1, y2, y3, area Como REAL;
5.      ESCRIBIR “Introduce el valor de x1: “;
6.      LEER x1;
7.      ESCRIBIR “Introduce el valor de x2: “;
8.      LEER x2;
9.      ESCRIBIR “Introduce el valor de x3: “;
10.     LEER x3;
11.     ESCRIBIR “Introduce el valor de y1: “;
12.     LEER y1;
13.     ESCRIBIR “Introduce el valor de y2: “;
14.     LEER y2;
15.     ESCRIBIR “Introduce el valor de y3: “;
16.     LEER y3;
17.     area ← (((x1 * y2) + (x2 * y3) + (x3 * y1)) - ((x1 * y3)
```

```
18.       + (x3 * y2) + (x2 * y1))) / 2;
19.     ESCRIBIR “El área del triángulo es: “;
20.     ESCRIBIR area;
21. FinAlgoritmo
```

## Ejecutar Paso a Paso

Inicia la ejecución del programa.

```
22. *** Ejecución Iniciada. ***
```

Se inicializan las variables y nos solicita en pantalla la introducción de los valores de cada coordenada.

```
1.  Ingrese el valor de x1:
2.  > 20
3.  Ingrese el valor de x2:
4.  > 5
5.  Ingrese el valor de x3:
6.  > 7
7.  Ingrese el valor de y1:
8.  > 12
9.  Ingrese el valor de y2:
10. > 15
11. Ingrese el valor de y3:
12. > 11
```

Al terminar de asignar los valores, se realizan las operaciones para obtener la superficie del triángulo.

Se imprime en pantalla el resultado. Finaliza la ejecución del algoritmo.

```
1.  El área del triángulo es:
2.  27
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el valor de x1:
> 20
Ingrese el valor de x2:
> 5
Ingrese el valor de x3:
> 7
Ingrese el valor de y1:
> 12
Ingrese el valor de y2:
> 15
Ingrese el valor de y3:
> 11
El área del triángulo es:
27
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
 /* Construya el algoritmo tal que dadas las coordenadas de los puntos P1, P2 y P3
  * que corresponden a los vértices de un triángulo, calcule su superficie.
  */
 package EstructuraSecuencial;

 import java.util.Scanner;

 public class Triangulo_Superficie {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        double x1, x2, x3, y1, y2, y3, area;

        System.out.println("Ingresa la cordenada de x1: ");
        x1 = entrada.nextDouble();
        System.out.println("Ingresa la cordenada de y1: ");
        y1 = entrada.nextDouble();
        System.out.println("Ingresa la cordenada de x2: ");
```

```
19.         x2 = entrada.nextDouble();
20.         System.out.println("Ingresa la cordenada de y2: ");
21.         y2 = entrada.nextDouble();
22.         System.out.println("Ingresa la cordenada de x3: ");
23.         x3 = entrada.nextDouble();
24.         System.out.println("Ingresa la cordenada de y3: ");
25.         y3 = entrada.nextDouble();
26.
27.         area = (((x1 * y2) + (x2 * y3) + (x3 * y1)) - ((x1 * y3)
28. + (x3 * y2) + (x2 * y1))) / 2;
29.
30.         System.out.println("El Área del Triangulo con respecto a
    sus Vertices es: "
31. + area);
32.
33.      }
34.
35.  }
```

## Resultado Java

```
1.  run:
2.  Ingresa la cordenada de x1:
3.  20
4.  Ingresa la cordenada de y1:
5.  12
6.  Ingresa la cordenada de x2:
7.  5
8.  Ingresa la cordenada de y2:
9.  15
10. Ingresa la cordenada de x3:
11. 7
12. Ingresa la cordenada de y3:
13. 11
14. El Área del Triangulo con respecto a sus Vertices es: 27.0
15. BUILD SUCCESSFUL (total time: 17 seconds)
```

## Ejercicio 8

Construya el algoritmo tal que, dado el perímetro de la base, la apotema y la altura de un prisma pentagonal; calcule el área de la base, el área lateral, el área total y el volumen.

**Consideraciones:**

- Para calcular el área de la base, hacemos: $AB = \frac{PER * APO}{2}$
- Para calcular el área lateral, aplicamos la siguiente fórmula: $AL = PER * ALT$
- Para calcular el área total hacemos: $AT = 2 * AB + AL$
- Para calcular el volumen hacemos: $VOL = AB * ALT$

```
1.  PRISMA_PENTAGONAL
2.  Definir aBase, perimetro, apotema, aLado, altura, aTotal,
    volumen como entero;
3.  Escribir “Ingrese el perímetro de la base:”
4.  Leer perimetro
5.  Escribir “Ingrese la apotema:”
6.  Leer apotema
7.  Escribir “Ingrese el valor de la altura:”
8.  Leer altura
9.  aBase ← (perimetro * apotema) / 2
10. aLado ← perimetro * altura
11. aTotal ← (2 * aBase) + aLado
12. volumen ← aBase * altura
13. Escribir “El área de la base es:”
14. Escribir aBase
15. Escribir “El área lateral es:”
16. Escribir aLado
17. Escribir “El área total es:”
18. Escribir aTotal
19. Escribir “El volumen del prisma pentagonal es:”
20. Escribir volumen
21. Fin
```

## PRUEBA DE ESCRITORIO

| PRISMA_PENTAGONAL | | | | | | |
|---|---|---|---|---|---|---|
| aBase | perimetro | apotema | altura | aTotal | aLado | volumen |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 138 | 23 | 12 | 14 | 598 | 322 | 1932 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Construya el algoritmo tal que, dado el perímetro de la base,
    la
2.  // apotema y la altura de un prisma pentagonal; calcule el área
    de la
3.  // base, el área lateral, el área total y el volumen.
4.  Algoritmo PRISMA_PENTAGONAL
5.      DEFINIR aBase, perimetro, apotema, aLado, altura, aTotal,
    volumen
6.        Como Real;
7.      ESCRIBIR “Ingrese el perímetro de la base: “;
8.      LEER perimetro;
9.      ESCRIBIR “Ingrese el valor de la apotema: “;
10.     LEER apotema;
11.     ESCRIBIR “Ingrese el valor de la altura: “;
12.     LEER altura;
13.     aBase ← (perimetro * apotema) / 2;
14.     aLado ← perimetro * altura;
15.     aTotal ← (2 * aBase) + aLado;
16.     volumen ← aBase * altura;
17.     ESCRIBIR “El área de la base es: “;
18.     ESCRIBIR aBase;
19.     ESCRIBIR “El área lateral es: “;
20.     ESCRIBIR aLado;
21.     ESCRIBIR “El área total es: “;
22.     ESCRIBIR aTotal;
23.     ESCRIBIR “El volumen del prisma pentagonal es: “;
```

```
    ESCRIBIR volumen;
FinAlgoritmo
```

## Ejecutar Paso a Paso

Inicia la ejecución del algoritmo. Se solicita el perímetro de la base.

```
*** Ejecución Iniciada. ***
Ingrese el perímetro de la base:
> 23
```

Luego nos solicita el valor de la apotema.

```
Ingrese el valor de la apotema:
> 12
```

Por último, nos pide el valor de la altura del prisma pentagonal.

```
Ingrese el valor de la altura:
> 14
```

El programa imprime en pantalla el área base, área lateral, área total y volumen. Finaliza ejecución.

```
El área de la base es:
138
El área lateral es:
322
El área total es:
598
El volumen del prisma pentagonal es:
1932
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el perímetro de la base:
> 23
Ingrese el valor de la apotema:
> 12
Ingrese el valor de la altura:
> 14
El área de la base es:
138
El área lateral es:
322
El área total es:
598
El volumen del prisma pentagonal es:
1932
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
/* Calcula el área total, área lateral y el área de la base de un prisma pentagonal.
* perimetro, apotema. altura, aTotal, aLateral, aBase : Real*/
package EstructuraSecuencial;

import java.util.Scanner;

public class Prisma_Pentagonal {

    public static void main(String[] args) {
      Scanner entrada = new Scanner(System.in);
      double perimetro, apotema, altura, aTotal, aLado, aBase, volumen;

      System.out.println("Ingresa el perimetro de la Base: ");
      perimetro = entrada.nextDouble();
      System.out.println("Ingresa la apotema: ");
```

```
16.       apotema = entrada.nextDouble();
17.       System.out.println(“Ingresa la altura: “);
18.       altura = entrada.nextDouble();
19.
20.       aBase = (perimetro * apotema) / 2;
21.       aLado = perimetro * altura;
22.       aTotal = (2 * aBase) + aLado;
23.       volumen = aBase * altura;
24.
25.       System.out.println(“El área de la base es: “ + aBase);
26.       System.out.println(“El área lateral es: “ + aLado);
27.       System.out.println(“El área total es: “ + aTotal);
28.       System.out.println(“El volumen del prisma pentagonal es: “
    + volumen);
29.    }
30. }
```

## Resultado Java

```
1.  run:
2.  Ingresa el perimetro de la Base:
3.  23
4.  Ingresa la apotema:
5.  12
6.  Ingresa la altura:
7.  14
8.  El área de la base es: 138.0
9.  El área lateral es: 322.0
10. El área total es: 598.0
11. El volumen del prisma pentagonal es: 1932.0
12. BUILD SUCCESSFUL (total time: 6 seconds)
```

# Ejercicio 9

Construya un algoritmo que calcule el monto total de una capital según sea el capital ingresado inicialmente y la tasa de interés vigente.

```
1.  MONTO_CAPITAL
2.  capital, interés, monto: Real
3.  Inicio
4.  capital←0
5.  interés←0
6.  monto←0
7.  Escribir “Ingrese el capital: ”
8.  Leer capital
9.  Escribir “Ingrese la tasa de interés: ”
10. Leer interés
11. monto ← capital*[1 + (interés / 100)]
12. Escribir “El monto es: ”
13. Escribir monto
14. Fin
```

### PRUEBA DE ESCRITORIO

| MONTO_CAPITAL | | |
|---|---|---|
| capital | interes | monto |
| 0 | 0 | 0 |
| 100 | 15 | 115 |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Construya un algoritmo que calcule el monto total de una
    capital
2.  // según sea el capital ingresado inicialmente y la tasa de
    interés
```

```
// vigente.
Algoritmo MONTO_CAPITAL
    DEFINIR capital, interes, monto Como REAL;
    capital ← 0;
    interes ← 0;
    monto ← 0;
    ESCRIBIR “Ingrese el capital: “;
    LEER capital;
    ESCRIBIR “Ingrese la tasa de interés: “;
    LEER interes;
    monto ← capital * (1 + (interes / 100));
    ESCRIBIR “El monto es: “;
    ESCRIBIR monto;
   FinAlgoritmo
```

### Ejecutar Paso a Paso

Ingreso de la variable capital(Total del Capital)

```
*** Ejecución Iniciada. ***
Ingrese el capital:
> 100
```

Ingreso de la variable interes (Total del Interés)

```
Ingrese la tasa de interés:
> 15
```

Cálculo de la variable monto (Monto)

```
El monto es:
115
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el capital:
> 100
Ingrese la tasa de interés:
> 15
El monto es:
115
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
 // Construya un algoritmo que calcule el monto total de una capital
 // según sea el capital ingresado inicialmente y la tasa de interés
 // vigente.
 package EstructuraSecuencial;

 import java.util.Scanner;

 public class Monto_Capital {

     public static void main(String[] args) {

         Scanner entrada = new Scanner(System.in);
         double capital, interes, monto;
         System.out.println("Ingrese el capital: ");
         capital = entrada.nextDouble();
         System.out.println("Ingrese la tasa de interés: ");
         interes = entrada.nextDouble();

         monto = capital * (1 + (interes / 100));

         System.out.println("El monto es: " + monto);
     }
 }
```

### Resultado Java

```
run:
Ingrese el capital:
100
Ingrese la tasa de interés:
15
El monto es: 114.99999999999999
BUILD SUCCESSFUL (total time: 6 seconds)
```

## Ejercicio 10

Escribe un algoritmo que calcule el área total, lateral y el área de la base de un prisma pentagonal, solicitando para esto el perímetro de la base, apotema y altura del prisma.

```
1.  AREAS_PRISMA
2.  perimetro, apotema, altura, aTotal, aLateral, aBase: Real
3.  Inicio
4.  perimetro←0
5.  apotema←0
6.  altura←0
7.  aTotal←0
8.  aLateral←0
9.  aBase←0
10. Escribir “Ingresa el perímetro de la base: ”
11. Leer perímetro
12. Escribir “Ingresa la Apotema: ”
13. Leer apotema
14. Escribir “Ingresa la Altura: ”
15. Leer altura
16. aBase ← (perimetro * apotema) / 2
17. aLateral ← (perimetro * altura) / 2
18. aTotal ← (2 * aBase) + aLateral
19. Escribir “El área de la base es: ”
20. Escribir aBase
21. Escribir “El área lateral es: ”
22. Escribir aLateral
23. Escribir “El área total es: ”
24. Escribir aTotal
25. Fin
```

## PRUEBA DE ESCRITORIO

| AREAS_PRISMA | | | | | |
|---|---|---|---|---|---|
| perimetro | apotema | altura | aBase | aLateral | aTotal |
| 0 | 0 | 0 | 0 | 0 | 0 |
| 5 | 3 | 2 | 7.5 | 5 | 20 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Escribe un algoritmo que calcule el área total, lateral y el
    área
2.  // de la base de un prisma pentagonal, solicitando para esto el
3.  // perímetro de la base, apotema y altura del prisma.
4.  Algoritmo AREAS PRISMA
5.      DEFINIR perimetro, apotema, altura, aTotal, aLateral, aBase
    Como REAL;
6.      perimetro ← 0;
7.      apotema ← 0;
8.      altura ← 0;
9.      aTotal ← 0;
10.     aLateral← 0;
11.     aBase ← 0;
12.     ESCRIBIR “Ingrese el perímetro de la base: “;
13.     LEER perimetro;
14.     ESCRIBIR “Ingrese la Apotema: “;
15.     LEER apotema;
16.     ESCRIBIR “Ingrese la Altura: “;
17.     LEER altura;
18.     aBase ← (perimetro * apotema) / 2;
19.     aLateral ← (perimetro * altura) / 2;
20.     aTotal ← (2 * aBase) + aLateral;
21.     ESCRIBIR “El área de la base es: “;
22.     ESCRIBIR aBase;
23.     ESCRIBIR “El área lateral es: “;
24.     ESCRIBIR aLateral;
```

```
25.     ESCRIBIR “El área total es: “;
26.     ESCRIBIR aTotal;
27. FinAlgoritmo
```

## Ejecutar Paso a Paso

Ingreso de la variable **perimetro** (Perímetro de la Base)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el perímetro de la base:
3.  > 5
```

Ingreso de la variable **apotema** (Apotema)

```
1.  Ingrese la apotema:
2.  > 3
```

Ingreso de la variable **altura** (Altura)

```
1.  Ingrese la altura:
2.  > 2
```

Cálculo de la variable **aBase** (Área de la Base), **aLateral** (Área Lateral) y **aTotal** (Área Total)

```
1.  El área de la base es:
2.  7.5
3.  El área lateral es:
4.  5
5.  El área total es:
6.  20
7.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el perímetro de la base:
> 5
Ingrese la apotema:
> 3
Ingrese la altura:
> 2
El área de la base es:
7.5
El área lateral es:
5
El área total es:
20
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
/* Calcula el área total, área lateral y el área de la base de un prisma pentagonal.
    perimetro, apotema. altura, aTotal, aLateral, aBase : Real*/

 package EstructuraSecuencial;

 import java.util.Scanner;

 public class Areas_Prisma {

   public static void main(String[] args) {
     Scanner entrada = new Scanner(System.in);
     double perimetro, apotema, altura, aTotal, aLateral, aBase;

     System.out.println("Ingresa el perimetro de la Base: ");
     perimetro = entrada.nextDouble();
     System.out.println("Ingresa la apotema: ");
     apotema = entrada.nextDouble();
     System.out.println("Ingresa la altura: ");
```

```
19.       altura = entrada.nextDouble();
20.
21.       aBase = (perimetro * apotema) / 2;
22.       aLateral = (perimetro * altura) / 2;
23.       aTotal = (2 * aBase) + aLateral;
24.
25.       System.out.println("El área de la base es: " + aBase);
26.       System.out.println("El área lateral es: " + aLateral);
27.       System.out.println("El área total es: " + aTotal);
28.     }
29.
30.  }
```

## Resultado Java

```
1.  run:
2.  Ingresa el perimetro de la Base:
3.  5
4.  Ingresa la apotema:
5.  3
6.  Ingresa la altura:
7.  2
8.  El área de la base es: 7.5
9.  El área lateral es: 5.0
10. El área total es: 20.0
11. BUILD SUCCESSFUL (total time: 6 seconds)
```

# Ejercicio 11

Escribe un algoritmo que calcule e imprima los segundos que existen en el número de días ingresados por el usuario.

```
1.  SEGUNDOS_POR_DIA
2.  dias, segundos: Real
3.  Inicio
4.  dias←0
5.  segundos←0
6.  Escribir “Ingrese los días”
7.  Leer dias
8.  segundos ← dias * 24 * 60 * 60
9.  Escribir “Los días:”
10. Escribir dias
11. Escribir “Son equivalentes a estos segundos:”
12. Escribir segundos
13. Fin
```

### PRUEBA DE ESCRITORIO

| SEGUNDOS_POR_DIA | |
|---|---|
| dias | segundos |
| 0 | 0 |
| 2 | 172,800 |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Escribe un algoritmo que calcule e imprima los segundos que
    existen
2.  // en el número de días ingresados por el usuario.
3.  Algoritmo SEGUNDOS_POR_DIA
4.      DEFINIR dias, segundos Como REAL;
```

```
5.      ESCRIBIR “Ingrese los días: “;
6.      LEER dias;
7.      segundos ← dias * 24 * 60 * 60;
8.      ESCRIBIR “Los días: “;
9.      ESCRIBIR dias;
10.     ESCRIBIR “Son equivalentes a estos segundos: “;
11.     ESCRIBIR segundos;
12. FinAlgoritmo
```

## Ejecutar Paso a Paso

Ingreso de la variable **dias** (Días a convertir)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese los días:
3.  > 2
```

Cálculo de la variable **segundos** (Total de Segundos equivalentes)

```
1.  Los días:
2.  2
3.  Son equivalentes a estos segundos:
4.  172800
5.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese los días:
3.  > 2
4.  Los días:
5.  2
6.  Son equivalentes a estos segundos:
```

```
172800
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
 // Escribe un algoritmo que calcule e imprima los segundos que existen
 // en el número de días ingresados por el usuario.
 package EstructuraSecuencial;

 import java.util.Scanner;

 public class Segundos_Por_Dia {

     public static void main(String[] args) {
         Scanner entrada = new Scanner(System.in);
         double dias, segundos;
         System.out.println("Ingrese los días:");
         dias = entrada.nextDouble();

         segundos = dias * 24 * 60 * 60;

         System.out.println("Los días " + dias);
         System.out.println("Son equivalentes a " + segundos + " segundos.");
     }
 }
```

### Resultado Java

```
run:
Ingrese los días:
2
Los días 2.0
Son equivalentes a 172800.0 segundos.
BUILD SUCCESSFUL (total time: 5 seconds)
```

# 2.

# Estructuras Selectivas

## Introducción

El objetivo de este capítulo consiste en introducir al lector a un siguiente nivel en el desarrollo de algoritmos con las estructuras de selectivas a través de una serie de ejercicios prácticos. Al estudiar este capítulo:

- Conocerá la estructura selectiva simpe (SI) (IF)
- Comprenderá la estructura selectiva doble (SINO) (ELSE) de un algoritmo.
- Entenderá la estructura algorítmica SI ANIDADO (SI MÚLTILPE)
- Conocerá a través de diversos ejercicios prácticos, los pasos para resolver un algoritmo, utilizando las diversas estructuras selectivas
- Comprenderá la estructura secuencial y lineal de un algoritmo.
- Entenderá algunos conflictos básicos al resolver un algoritmo con las estructuras selectivas.
- Conocerá cómo realizar una prueba de escritorio, símbolos, nombres, que se usan como ejemplos a lo largo del libro.
- Entenderá cómo manejar y ejecutar PseInt con algoritmos con SI, SINO y SI ANIDADO.
- Identificará el mismo algoritmo resuelto en español estructurado y pseudocódigo, pero ahora con Java, haciendo uso de la estructuras selectivas SI, SINO o SI ANIDADO

# CAPÍTULO 2.
# Estructuras selectivas

El implementar las estructuras algorítmicas selectivas en un algoritmo dan la oportunidad de seleccionar algún camino u opción que se presentan con el uso de estas. Existen específicamente tres tipos de estás y a partir de ellas puedes combinarlas para resolver un algoritmo.

- SI sencillo o simple.
- SINO o SI doble
- SI múltiple o SI anidado

Al manipular las estructuras algorítmicas selectivas, sin duda alguna harás uso operadores de comparación y lógicos. Veamos la utilidad de ello en a la tabla 3 y tabla 4.

### Operadores de comparación

Son aquellos que se colocan en la condición o instrucción de entrada a evaluar como verdadera, aquellos como su nombre lo indica sirven para comparar valores. (Tabla 3).

| Operador | Significado |
|---|---|
| = | Igual |
| > | Mayor que |
| < | Menor que |
| ⩾ | Mayor o igual que |
| ⩽ | Menor o igual que |
| <> | Distinto/diferente |
| ≠ | Distinto de. |
| ^= | Distinto de. |

**Tabla 3. Operadores de comparación**

### Operadores lógicos

Se utilizan para seleccionar o excluir alguna condición o instrucción, según lo que se desee resolver en el algorítmo. (Tabla 4). Estás se utilizan cuando se tienen dos condiciones a evaluar.

| Operador | Significado |
|---|---|
| Y o & | Ambas condiciones |
| O ó \| | Sólo alguna condición |
| NO o ~ | Eliminar o ignorar la condición |

**Tabla 4. Operadores lógicos**

### SI sencillo o simple.

Es aquella estructura como se muestra en la Figura 2, da paso a un camino, opción por donde transita el algoritmo paso a paso. Si la instrucción o condición se cumple como verdadera en este ejemplo es: (sueldo < 1000), si esto es verdad, entonces entrará a realizar la serie de instrucciones y operaciones dentro del SI sencillo o simple.

```
1.  SUELDO_TRABAJADOR
2.  sueldo, nuevoSueldo: Real
3.  Inicio
4.  sueldo←0
5.  nuevoSueldo←0
6.  Escribir "El sueldo del trabajador es:"
7.  Leer sueldo
8.  Si (sueldo<1000) entonces
9.      nuevoSueldo←sueldo*1.15
10.     Escribir "El nuevo sueldo del trabajador es:"
11.     Escribir nuevoSueldo
12. FinSi
13. Fin Proceso
```

Condición o instrucción por evaluar sólo si se cumple como verdadera, entonces se ejecutará lo que está dentro del SI y en caso contrario finaliza el algorítmo.

**Figura 2. Uso del SI sencillo o simple**

### SINO o SI doble

Es la estructura selectiva que tiene dos caminos a elegir (figura 3), lo que significa que el algoritmo puede elegir o seleccionar uno pero el otro no, sólo uno. Y será a partir de que se cumpla como verdadera la condición o instrucción a ejecutarse.

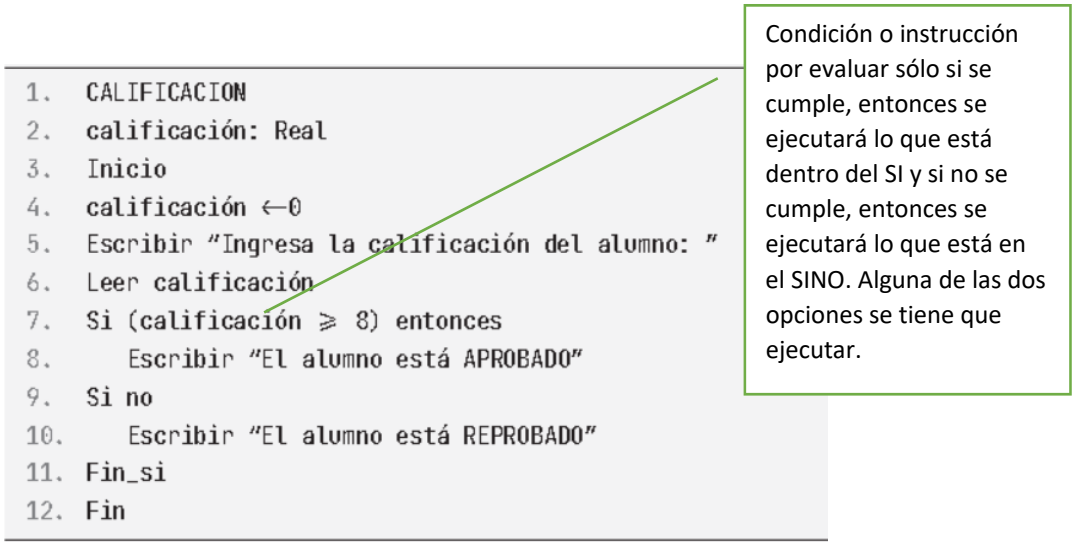



**Figura 3. Uso del SINO o SI doble**

### SI múltiple o SI anidado

Esta estructura sigue la misma lógica del SINO o SI doble con más de dos opciones de alternativas o caminos a seguir (figura 4). Dentro de esos caminos u opciones, el algoritmo sólo toma uno de los muchos que se muestran, ese camino transitado será porque es el que cumple como verdadera la condición o instrucción.

```
1.  // Determina si un número ingresado es negativo, nulo o
    positivo.
2.  Algoritmo NUMEROS
3.      DEFINIR numero Como ENTERO;
4.      numero ← 0;
5.      ESCRIBIR "Ingrese el número entero: ";
6.      LEER numero;
7.      SI (numero = 0) ENTONCES
8.          ESCRIBIR "El número es NULO ";
9.      SINO
10.         SI (numero ⩾ 1) ENTONCES
11.             ESCRIBIR "El número es POSITIVO";
12.         SINO
13.             SI (numero < 0) ENTONCES
14.                 ESCRIBIR "El número es NEGATIVO";
15.             FINSI
16.     FINSI
17.     FINSI
18. FinAlgoritmo
```

Se tienen tres SI múltiple o anidados. Dos de ellos con SINO o SI doble y un SI sencillo. Pero se encuentran anidados.

Observe como se colocan escalonados y muy importante respetar las buenas prácticas de sangría o indentación.

**Figura 4. Uso del SI múltiple o SI anidado**

Veamos los problemas que hacen uso de estas estructuras con SI.

# ESTRUCTURA SELECTIVA SI (IF)

## Ejercicio 1

Escribir un algoritmo que aplique un aumento del 15% al sueldo de un trabajador, si éste es menor a $1,000.00.

```
1.  SUELDO_TRABAJADOR
2.  sueldo, nuevoSueldo: Real
3.  Inicio
4.  sueldo←0
5.  nuevoSueldo←0
6.  Escribir “El sueldo del trabajador es:”
7.  Leer sueldo
8.  Si (sueldo<1000) entonces
9.      nuevoSueldo←sueldo*1.15
10.     Escribir “El nuevo sueldo del trabajador es:”
11.     Escribir nuevoSueldo
12. FinSi
13. Fin Proceso
```

### PRUEBA DE ESCRITORIO

| SUELDO_TRABAJADOR | |
|---|---|
| **sueldo** | **nSueldo** |
| 0 | 0 |
| 1000 | No se genera un nuevo Sueldo |
| 900 | 1035 |
| 800 | 920 |
| 1500 | No se genera un nuevo Sueldo |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Escribir un algoritmo que aplique un aumento del 15% al
    sueldo de
2.  // un trabajador, si éste es menor a $1,000.00.
3.  Algoritmo SUELDO_TRABAJADOR
4.      DEFINIR sueldo, nSueldo Como REAL;
5.      sueldo ← 0;
6.      nSueldo ← 0;
7.      ESCRIBIR "El sueldo del trabajador es: ";
8.      LEER sueldo;
9.      SI (sueldo < 1000) ENTONCES
10.     .   nSueldo ← (sueldo * 1.15);
11.     .   ESCRIBIR "El nuevo sueldo del trabajador es: ";
12.     .   ESCRIBIR nSueldo;
13.     FINSI
14. FinAlgoritmo
```

### Ejecutar Paso a Paso

Ingreso de la variable **sueldo** (Sueldo)

```
1.  *** Ejecución Iniciada. ***
2.  El sueldo del trabajador es:
3.  > 900
```

Comparación de la variable **sueldo** (Sueldo) en la estructura **Si** y cálculo de **nSueldo** (Nuevo Sueldo)

```
1.  El nuevo sueldo del trabajador es:
2.  1035
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
El sueldo del trabajador es:
> 900
El nuevo sueldo del trabajador es:
1035
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Aplica un aumento del 15% al sueldo de un trabajador
   si este es menor a $1,000.00 */
package EstructuraSelectiva;

import java.util.Scanner;

public class Sueldo_Trabajador {

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

        double sueldo, nuevoSueldo;

        System.out.println("El sueldo del trabajado es: ");
        sueldo = entrada.nextDouble();

        if (sueldo < 1000) {
            nuevoSueldo = sueldo * 1.15;
     System.out.println("El nuevo Sueldo del Trabajador es: " +
nuevoSueldo);
        }

    }
}
```

## Resultado Java

```
run:
El sueldo del trabajado es:
900
El nuevo Sueldo del Trabajador es: 1035.0
BUILD SUCCESSFUL (total time: 2 seconds)
```

## Ejercicio 2

Escribe un algoritmo que calcule la temperatura de acuerdo con el número de sonidos emitidos por un grillo.

```
1.  TEMPERATURA_SONIDOS_GRILLO
2.  número, temperatura: Real
3.  Inicio
4.  numero←0
5.  temperatura←0
6.  Escribir “Ingrese el número de sonidos emitidos por el grillo: ”
7.  Leer numero
8.  Si (numero>0) entonces
9.     temperatura ←(numero/4) +40
10.    Escribir “La temperatura es:”
11.    Escribir temperatura
12. FinSi
13. Fin
```

### PRUEBA DE ESCRITORIO

| TEMPERATURA_SONIDOS_GRILLO | |
|---|---|
| **numero** | **temperatura** |
| ~~0~~ | ~~0~~ |
| 5 | 41.25 |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Escribe un algoritmo que calcule la temperatura de acuerdo
    con el
2.  // número de sonidos emitidos por un grillo.
3.  Algoritmo TEMPERATURA_SONIDOS_GRILLO
4.      DEFINIR numero, temperatura Como REAL;
5.      numero ← 0;
6.      temperatura ← 0;
```

```
7.      ESCRIBIR "Ingrese el número de sonidos emitidos por el
    grillo: ";
8.      LEER numero;
9.      SI (numero > 0) ENTONCES
10.     .   temperatura ← (numero / 4) + 40;
11.     .   ESCRIBIR "La temperatura es: ";
12.     .   ESCRIBIR temperatura;
13.     FINSI
14. FinAlgoritmo
```

## Ejecutar Paso a Paso

Ingreso de la variable **número** (Número de Sonidos)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el número de sonidos emitidos por el grillo:
3.  > 5
```

Comparación de la variable **número** (Número de Sonidos) en la estructura **Si** y cálculo de **temperatura** (Temperatura)

```
1.  La temperatura es:
2.  41.25
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el número de sonidos emitidos por el grillo:
3.  > 5
4.  La temperatura es:
5.  41.25
6.  *** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
1.  package EstructuraSelectiva;
2.
3.  import java.util.Scanner;
4.
5.  public class Temperatura_Sonidos_Grillo {
6.
7.     public static void main(String[] args) {
8.        Scanner entrada = new Scanner(System.in);
9.        double numero, temperatura;
10.       System.out.println(“Ingrese el número de sonidos emitidos
    por el grillo: “);
11.       numero = entrada.nextDouble();
12.
13.       if (numero > 0) {
14.           temperatura = (numero / 4) + 40;
15.           System.out.println(“La temperatura es: “ +
    temperatura);
16.       }
17.
18.    }
19.
20.  }
```

## Resultado Java

```
1.  run:
2.  Ingrese el número de sonidos emitidos por el grillo:
3.  5
4.  La temperatura es: 41.25
5.  BUILD SUCCESSFUL (total time: 3 seconds)
```

# Ejercicio 3

Dada la estabilidad económica que existe en un determinado país de américa latina, las agencias automotrices comienzan a ofrecer distintos planes de financiamiento para la comercialización de sus vehículos. La empresa XGW ofrece el siguiente plan de financiación: dado el monto total de vehículo, el cliente debe pagar el 35% de enganche y el resto en 18 mensualidades sin intereses. Construye un algoritmo que permita obtener cuál es el importe del enganche y las mensualidades que debe pagar el cliente.

```
1.  ENGANCHE_MENSUALIDADES
2.  costo, enganche, mensualidad: Real.
3.  Inicio
4.  costo←0
5.  enganche←0
6.  mensualidad←0
7.  Escribir “Ingrese el Costo total del vehículo:”
8.  Leer costo
9.  enganche ← costo * 0.35
10. Si enganche>0 entonces
11.    mensualidad ←(costo - enganche) / 18
12.    Escribir “El enganche fue de:”
13.    Escribir enganche
14.    Escribir “Las mensualidades son de:”
15.    Escribir mensualidad
16. FinSi
17. Fin
```

## PRUEBA DE ESCRITORIO

| ENGANCHE_MENSUALIDADES | | |
|---|---|---|
| **costo** | **enganche** | **mensualidad** |
| ~~0~~ | ~~0~~ | ~~0~~ |
| 20,000 | 7,000 | 722.22 |

### ALGORITMO EN PSeInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula el importe del enganche y las mensualidades para
2.  // comprar un vehículo.
3.  Algoritmo ENGANCHE_MENSUALIDADES
4.      DEFINIR costo, enganche, mensualidad Como REAL;
5.      costo ← 0;
6.      enganche ← 0;
7.      mensualidad ← 0;
8.      ESCRIBIR "Ingrese el Costo total del vehículo: ";
9.      LEER costo;
10.     enganche ← costo * 0.35;
11.     SI (enganche > 0) ENTONCES
12.     .   mensualidad ← (costo - enganche) / 18;
13.     .   ESCRIBIR "El enganche fue de: ";
14.     .   ESCRIBIR enganche;
15.     .   ESCRIBIR "Las mensualidades son de: ";
16.     .   ESCRIBIR mensualidad;
17.     FINSI
18. FinAlgoritmo
```

### Ejecutar Paso a Paso

Ingreso de la variable **costo** (Costo del vehículo)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el costo total del vehículo:
3.  > 20000
```

Cálculo de la variable **enganche** (Enganche) y su Comparación en la estructura **Si**, y cálculo de **mensualidad** (Mensualidad)

```
1.  El enganche fue de:
2.  7000
3.  Las mensualidades son de:
```

```
722.2222222222
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el costo total del vehículo:
> 20000
El enganche fue de:
7000
Las mensualidades son de:
722.2222222222
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Calcule el importe del enganche y las mensualidades para comprar un vehiculo.
   costo , enganche , mensualidad */
package EstructuraSelectiva;

import java.util.Scanner;

public class Enganche_Mensualidades {

  public static void main(String[] args) {
     Scanner entrada = new Scanner(System.in);
     double costo, enganche, mensualidad;

     System.out.println("Ingresa el costo total del vehículo: ");
     costo = entrada.nextDouble();

     enganche = costo * 0.35;

     if (enganche > 0) {
```

```
19.           mensualidad = (costo - enganche) / 18;
20.           System.out.println(“El enganche fue de: “ + enganche);
21.           System.out.println(“Las mensualidades son de: “ +
    mensualidad);
22.       }
23.
24.   }
25.
26. }
```

## Resultado Java

```
1.  run:
2.  Ingresa el costo total del vehículo:
3.  20000
4.  El enganche fue de: 7000.0
5.  Las mensualidades son de: 722.2222222222222
6.  BUILD SUCCESSFUL (total time: 4 seconds)
```

# ESTRUCTURA SELECTIVA SI / SI NO (IF / ELSE)

## Ejercicio 1

Escribir un algoritmo para lavar los platos de la comida.

```
1.  LAVAR_PLATOS
2.  traste←” “;
3.  traste←sucio;
4.  Escribir “Teclea como están los trastes”;
5.  Leer como_estan;
6.  Si como_estan = “sucios” Entonces
7.    Escribir “lavar trastes”;
8.      Escribir “revolver jabón con agua”;
9.      Escribir “meter esponja al agua”;
10.     Escribir “tallar trastes”;
11.     Escribir “enjuagar trastes”;
12.     Escribir “secar trastes”;
13.     Escribir “guardar trastes”;
14. Sino
15.     Escribir “No_lavarlos”;
16. FinSi
17. FIN
```

## PRUEBA DE ESCRITORIO

| LAVAR_PLATOS | | |
|---|---|---|
| **TRASTE** | **Si estado = sucios** | **Si estado = limpios** |
| sucios | Lavar trastes<br>Mezclar jabón con agua<br>Meter esponja en la mezcla<br>Fregar trastes con la esponja<br>Enjuagar trastes<br>Secar trastes<br>Guardar trastes | 0 |
| limpios | 0 | No lavarlos |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Escribir un algoritmo para lavar los platos de la comida.
2.  Algoritmo LAVAR_PLATOS
3.      DEFINIR traste, estado Como CARACTER;
4.      ESCRIBIR "¿Cómo se encuentran los trastes?";
5.      ESCRIBIR "¿Limpios o sucios?";
6.      LEER estado;
7.      SI estado = "sucios" O estado = "Sucios" ENTONCES
8.      .   ESCRIBIR "Lavar trastes";
9.      .   ESCRIBIR " Mezclar jabón con agua";
10.     .   ESCRIBIR " Meter esponja en la mezcla";
11.     .   ESCRIBIR " Fregar trastes con la esponja";
12.     .   ESCRIBIR " Enjuagar trastes";
13.     .   ESCRIBIR " Secar trastes";
14.     .   ESCRIBIR " Guardar trastes";
15.     FINSI
16.     SI estado = "limpios" O estado = "Limpios" ENTONCES
17.     .   ESCRIBIR "No lavarlos";
18.     FINSI
19. FinAlgoritmo
```

## Ejecutar Paso a Paso

Inicia la ejecución del programa, nos pide ingresar el estado en que se encuentran los trastes. De la respuesta del usuario va a depender el camino que seguirá el programa, se opta por la primera opción, en el caso de que los trastes estén limpios.

**Si los trastes están limpios.**

```
*** Ejecución Iniciada. ***
¿Cómo se encuentran los trastes?
¿Limpios o sucios?
> limpios
```

Al presionar *Enter*, escribe en pantalla la instrucción para no lavar los trastes, ya que estos ya están limpios.

```
No lavarlos
*** Ejecución Finalizada. ***
```

**Si los trastes no están limpios.**

Al iniciar el programa, pregunta el estado en que se encuentran los trastes, ingresamos que están sucios.

```
*** Ejecución Iniciada. ***
¿Cómo se encuentran los trastes?
¿Limpios o sucios?
> sucios
```

El programa responde a nuestra respuesta con una serie de instrucciones a realizar en el caso de que los trastes no estén limpios. Finaliza el programa.

```
Lavar trastes
 Mezclar jabón con agua
 Meter esponja en la mezcla
```

```
4.   Fregar los trastes con la esponja
5.   Enjuagar los trastes
6.   Secar los trastes
7.   Guardar los trastes
8.  *** Ejecución Finalizada. ***
```

## Ejecutar

### Caso Trastes limpios.

```
1.  *** Ejecución Iniciada. ***
2.  ¿Cómo se encuentran los trastes?
3.  ¿Limpios o sucios?
4.  > limpios
5.  No lavarlos
6.  *** Ejecución Finalizada. ***
```

### Caso Trastes sucios.

```
1.  *** Ejecución Iniciada. ***
2.  ¿Cómo se encuentran los trastes?
3.  ¿Limpios o sucios?
4.  > sucios
5.  Lavar trastes
6.   Mezclar jabón con agua
7.   Meter esponja en la mezcla
8.   Fregar los trastes con la esponja
9.   Enjuagar los trastes
10.  Secar los trastes
11.  Guardar los trastes
12. *** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
1.  // Escribir un algoritmo para lavar los platos de la comida.
2.  package EstructuraSelectiva;
3.
4.  import java.util.Scanner;
5.
6.  public class Lavar_Platos {
7.
8.      public static void main(String[] args) {
9.          Scanner entrada = new Scanner(System.in);
10.        String traste;
11.
12.         System.out.println(“Teclea como están los trastes: “);
13.         traste = entrada.next();
14.
15.         if (traste.equalsIgnoreCase(“sucios”)) {
16.             System.out.println(“Lavar los Trastes”);
17.             System.out.println(“ Mezclar jabón con agua.”);
18.             System.out.println(“ Meter esponja en la mezcla.”);
19.             System.out.println(“ Fregar los trastes con la
    esponja.”);
20.             System.out.println(“ Enjuagar los trastes.”);
21.             System.out.println(“ Secar los trastes.”);
22.             System.out.println(“ Guardar los trastes.”);
23.         } else {
24.             System.out.println(“No Lavarlos”);
25.         }
26.     }
27. }
```

## Resultado Java

## Caso Trastes limpios.

```
1.  run:
2.  Teclea como están los trastes:
3.  limpios
4.  No Lavarlos
5.  BUILD SUCCESSFUL (total time: 4 seconds)
```

### Caso Trastes sucios.

```
run:
Teclea como están los trastes:
Sucios
Lavar los Trastes
 Mezclar jabón con agua.
 Meter esponja en la mezcla.
 Fregar los trastes con la esponja.
 Enjuagar los trastes
 Secar los trastes
 Guardar los trastes
BUILD SUCCESSFUL (total time: 2 seconds)
```

## Ejercicio 2

Escribir un algoritmo para reparar un pinchazo de bicicleta.

```
1.  PINCHAZO
2.  Escribir “Teclea cómo están las llantas de la bicicleta”;
3.  Leer como_estan;
4.  Si como_estan = “ponchada” Entonces
5.     Escribir “Desmontar_rueda”;
6.     Escribir “Quitar_cubierta”;
7.     Escribir “Sacar_camara”;
8.     Escribir “Inflar_camara”;
9.     Escribir “Meter_parte_encubo_agua”;
10.    Escribir “Checar_si_salen_burbujas”;
11.    Escribir “Marcar_pinchazo”;
12.    Escribir “Echarle_pegamento”;
13.    Escribir “Esperar_a_que_seque”;
14.    Escribir “Parchar”;
15.    Escribir “Apretar_paraque_parche_quedebien”;
16.    Escribir “Montar_camara”;
17.    Escribir “Montar_cubierta”;
18.    Escribir “Montar llanta”;
19.    Escribir “Inflar_llanta”;
20.  Sino
21.    Escribir “No parcharla”;
22. FinSi
23. FIN
```

## PRUEBA DE ESCRITORIO

| PINCHAZO | | |
|---|---|---|
| **Llanta** | **Si estado = ponchada** | **Si no** |
| Ponchada | Desmontar rueda<br>Quitar cubierta<br>Sacar cámara<br>Inflar cámara<br>Meter pieza en cubo de agua<br>Checar de dónde salen las burbujas<br>Marcar pinchazo<br>Aplicar pegamento en el área dañada<br>Esperar a que seque el pegamento<br>Parchar<br>Aplicar presión en el parche<br>Montar cámara<br>Montar cubierta<br>Montar llanta<br>Inflar llanta | 0 |
| No ponchada | 0 | No parcharla |

## ALGORITMO EN PSeInt
## PRUEBA DE ESCRITORIO EN PSeInt

```
1.  // Escribir un algoritmo para reparar un pinchazo de bicicleta.
2.  Algoritmo PINCHAZO
3.      DEFINIR estado Como ENTERO;
4.      ESCRIBIR "¿Las llantas de la bicicleta se encuentran
5.        ponchadas?";
6.      ESCRIBIR "1.Si       2.No";
7.      LEER estado;
8.      SI (estado == 1) ENTONCES
9.      .   ESCRIBIR "Desmontar la rueda";
10.     .   ESCRIBIR "Quitar la cubierta";
11.     .   ESCRIBIR "Sacar la cámara";
12.     .   ESCRIBIR "Inflar la cámara";
13.     .   ESCRIBIR "Meter la pieza en un cubo de agua";
14.     .   ESCRIBIR "Checar de dónde salen las burbujas";
15.     .   ESCRIBIR "Marcar el pinchazo";
```

```
16.      .   ESCRIBIR “Aplicar pegamento en el área dañada”;
17.      .   ESCRIBIR “Esperar que seque el pegamento”;
18.      .   ESCRIBIR “Parchar el pinchazo”;
19.      .   ESCRIBIR “Aplicar presión en el pinchazo para que
20.      .    quede bien el parche”;
21.      .   ESCRIBIR “Montar la cámara”;
22.      .   ESCRIBIR “Montar la cubierta”;
23.      .   ESCRIBIR “Montar la llanta”;
24.      .   ESCRIBIR “Inflar la llanta”;
25.      SINO
26.      .   ESCRIBIR “No hay que parchar la llanta”;
27.      FINSI
28.  FinAlgoritmo
```

## Ejecutar Paso a Paso

### Llanta ponchada.

Inicia la ejecución del programa, inicializa las variables a utilizar. Imprime en pantalla la solicitud para ingresar el estado de las llantas. Seleccionamos la primera opción.

```
1.  *** Ejecución Iniciada. ***
2.  ¿Las llantas de la bicicleta se encuentran ponchadas?
3.  1. Si            2. No
4.  > 1
```

El programa imprime en pantalla los pasos a seguir si la(s) llanta(s) se encuentra ponchada(s). Finaliza la ejecución del programa.

```
1.  Desmontar la rueda
2.  Quitar la cubierta
3.  Sacar la cámara
4.  Inflar la cámara
5.  Meter la pieza en un cubo de agua
6.  Checar de dónde salen las burbujas
```

```
Marcar el pinchazo
Aplicar pegamento en el área dañada
Esperar que seque el pegamento
Parchar el pinchazo
Aplicar presión en el pinchazo para que quede bien el parche
Montar la cámara
Montar la cubierta
Montar la llanta
Inflar la llanta
*** Ejecución Finalizada. ***
```

**Llanta no ponchada.**

Inicia la ejecución del programa y nos solicitan el estado de las llantas de la bicicleta. Seleccionamos la segunda opción, en donde las llantas de la bicicleta no se encuentran ponchadas.

```
*** Ejecución Iniciada. ***
¿Las llantas de la bicicleta se encuentran ponchadas?
1. Si           2. No
> 2
```

El programa imprime en pantalla un mensaje, en el cual nos indica que no es necesario parchar. Finaliza la ejecución del programa.

```
No hay que parchar la llanta
*** Ejecución Finalizada. ***
```

## Ejecutar

**Llanta ponchada.**

```
*** Ejecución Iniciada. ***
¿Las llantas de la bicicleta se encuentran ponchadas?
1. Si           2. No
> 1
```

```
Desmontar la rueda
Quitar la cubierta
Sacar la cámara
Inflar la cámara
Meter la pieza en un cubo de agua
Checar de dónde salen las burbujas
Marcar el pinchazo
Aplicar pegamento en el área dañada
Esperar que seque el pegamento
Parchar el pinchazo
Aplicar presión en el pinchazo para que quede bien el parche
Montar la cámara
Montar la cubierta
Montar la llanta
Inflar la llanta
*** Ejecución Finalizada. ***
```

**Llanta no ponchada.**

```
*** Ejecución Iniciada. ***
¿Las llantas de la bicicleta se encuentran ponchadas?
1. Si            2. No
> 2
No hay que parchar la llanta
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```java
// Escribir un algoritmo para reparar un pinchazo de bicicleta.
package EstructuraSelectiva;

import java.util.Scanner;

public class Pinchazo {

   public static void main(String[] args) {
      Scanner entrada = new Scanner(System.in);
      String llanta;
```

```
11.
12.       System.out.println("¿Cómo están las llantas de la
    bicicleta?");
13.       llanta = entrada.next();
14.
15.       if (llanta.equalsIgnoreCase("ponchada")) {
16.          System.out.println("Desmontar la rueda");
17.          System.out.println("Quitar la cubierta");
18.          System.out.println("Sacar la cámara");
19.          System.out.println("Inflar la cámara");
20.          System.out.println("Meter la pieza en cubo agua");
21.          System.out.println("Checar de dónde salen las
    burbujas");
22.          System.out.println("Marcar el pinchazo");
23.          System.out.println("Aplicar pegamento en el área
    dañada");
24.          System.out.println("Esperar a que seque el pegamento");
25.          System.out.println("Parchar el pinchazo");
26.          System.out.println("Aplicar presión en el pinchazo para
    que quede bien el parche");
27.          System.out.println("Montar la cámara");
28.          System.out.println("Montar la cubierta");
29.          System.out.println("Montar la llanta");
30.          System.out.println("Inflar la llanta");
31.       } else {
32.          System.out.println("No hay que parchar la llanta");
33.       }
34.    }
35.
36. }
```

## Resultado Java

## Llanta ponchada.

```
1.  run:
2.  ¿Cómo están las llantas de la bicicleta?
3.  ponchadas
4.   Desmontar la rueda
```

```
   Quitar la cubierta
   Sacar la cámara
   Inflar la cámara
   Meter la pieza en cubo agua
   Checar de dónde salen las burbujas
   Marcar el pinchazo
   Aplicar pegamento en el área dañada
   Esperar a que seque el pegamento
   Parchar el pinchazo
   Aplicar presión en el pinchazo para que quede bien el parche
   Montar la cámara
   Montar la cubierta
   Montar la llanta
   Inflar la llanta
BUILD SUCCESSFUL (total time: 4 seconds)
```

**Llanta no ponchada.**

```
run:
¿Cómo están las llantas de la bicicleta?
bien
No hay que parchar la llanta
BUILD SUCCESSFUL (total time: 3 seconds)
```

## Ejercicio 3

Construya un algoritmo dado como dato la calificación de un alumno en un examen, escriba "APROBADO" si su calificación es mayor o igual que 8 y "REPROBADO" en caso contrario.

```
1.  CALIFICACION
2.  calificación: Real
3.  Inicio
4.  calificación ←0
5.  Escribir "Ingresa la calificación del alumno: "
6.  Leer calificación
7.  Si (calificación ⩾ 8) entonces
8.     Escribir "El alumno está APROBADO"
9.  Si no
10.    Escribir "El alumno está REPROBADO"
11. Fin_si
12. Fin
```

### PRUEBA DE ESCRITORIO

| CALIFICACIÓN | | | |
|---|---|---|---|
| **calificacion** | Si calificacion >8 | Si no | |
| 0 | El alumno está APROBADO | El alumno está REPROBADO | 0 |
| ~~5~~ | ~~Falso~~ | ~~Verdadero~~ | ~~El alumno está REPROBADO~~ |
| 10 | Verdadero | Falso | El alumno está APROBADO |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula si un alumno está aprobado o reprobado con base en
2.  // su calificación.
3.  Algoritmo CALIFICACION
```

```
4.      DEFINIR calificacion Como REAL;
5.      calificacion ← 0;
6.      ESCRIBIR “Ingrese la calificación del alumno: “;
7.      LEER calificacion;
8.      SI (calificacion ⩾ 8) ENTONCES
9.      .   ESCRIBIR “El alumno está APROBADO “;
10.     SINO
11.     .   ESCRIBIR “El alumno está REPROBADO “;
12.     FINSI
13. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Alumno aprobado.

Ingreso de la variable **calificacion**(Calificación)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese la calificación del alumno:
3.  > 10
```

Se compara la variable **calificacion** (Calificación) con la Condición y se imprime el Resultado

```
1.  El alumno está APROBADO
2.  *** Ejecución Finalizada. ***
```

### Alumno reprobado.

Ingreso de la variable **calificacion**(Calificación)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese la calificación del alumno:
3.  > 5
```

Se compara la variable **calificacion** (Calificación) con la Condición y se imprime el Resultado

```
El alumno está REPROBADO
*** Ejecución Finalizada. ***
```

## Ejecutar

### Alumno aprobado.

```
*** Ejecución Iniciada. ***
Ingrese la calificación del alumno:
> 10
El alumno está APROBADO
*** Ejecución Finalizada. ***
```

### Alumno reprobado.

```
*** Ejecución Iniciada. ***
Ingrese la calificación del alumno:
> 5
El alumno está REPROBADO
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
/* Construye un algoritmo dado como dato la calificación de un alumno en un
 * examen, escriba “APROBADO” si su calificacion es mayoro igual que  8 y
 * “REPROBADO” en caso contrario.
 */
package EstructuraSelectiva;

import java.util.Scanner;
```

```
8.
9.  public class Calificacion {
10.
11.     public static void main(String[] args) {
12.         Scanner entrada = new Scanner(System.in);
13.         double calificacion;
14.
15.         System.out.println("Ingrese la calificación del alumno:
    ");
16.         calificacion = entrada.nextDouble();
17.
18.         if (calificacion ⩾ 8) {
19.             System.out.println("El alumno está APROBADO");
20.         } else {
21.             System.out.println("El alumno está REPROBADO");
22.         }
23.     }
24.
25. }
```

## Resultado Java

## Alumno aprobado.

```
1.  run:
2.  Ingrese la calificación del alumno:
3.  10
4.  El alumno está APROBADO
5.  BUILD SUCCESSFUL (total time: 1 second)
```

## Alumno reprobado.

```
1.  run:
2.  Ingrese la calificación del alumno:
3.  5
4.  El alumno está REPROBADO
5.  BUILD SUCCESSFUL (total time: 1 second)
```

## Ejercicio 4

Construya un algoritmo dado como dato el sueldo de un trabajador, aplique un aumento del 15% si su sueldo es inferior a $1,000.00 y el 12% en caso contrario. Imprima el nuevo sueldo del trabajador.

```
1.  AUMENTO_SUELDO
2.  sueldo, nSueldo: Real
3.  Inicio
4.  sueldo←0
5.  nSueldo←0
6.  Escribir “Ingrese el sueldo del trabajador:”
7.  Leer sueldo
8.  Si (sueldo < 1000) entonces
9.     nSueldo ← sueldo * 1.15
10.    Escribir “El nuevo sueldo del trabajador será de:”
11.    Escribir nSueldo
12. SiNo
13.    nSueldo ← sueldo * 1.12
14.    Escribir “El nuevo sueldo del trabajador será de:”
15.    Escribir nSueldo
16. FinSi
17. Fin
```

### PRUEBA DE ESCRITORIO

| AUMENTO_SUELDO | | | |
|---|---|---|---|
| **sueldo** | **Si sueldo < 1000** | **Si no** | **nSueldo** |
| ~~0~~ | | | ~~0~~ |
| ~~500~~ | Verdadero | Falso | ~~575~~ |
| 3,000 | Falso | Verdadero | 3,360 |

**ALGORITMO EN PSeInt**
**PRUEBA DE ESCRITORIO EN PseInt**

```
1.  // Calcula e imprime el salario nuevo de un trabajador.
2.  Algoritmo AUMENTO_SUELDO
3.      DEFINIR sueldo, nSueldo Como REAL;
4.      sueldo ← 0;
5.      nSueldo ← 0;
6.      ESCRIBIR “Ingrese el sueldo del trabajador: “;
7.      LEER sueldo;
8.      SI (sueldo < 1000) ENTONCES
9.      .   nSueldo ← sueldo * 1.5;
10.     .   ESCRIBIR “El nuevo sueldo del trabajador es: “;
11.     .   ESCRIBIR nSueldo;
12.     SINO
13.     .   nSueldo ← sueldo * 1.2;
14.     .   ESCRIBIR “El nuevo sueldo del trabajador es: “;
15.     .   ESCRIBIR nSueldo;
16.     FINSI
17. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Sueldo menor a 1000.

Ingreso de la variable **sueldo** (Sueldo actual)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el sueldo del trabajador:
3.  > 500
```

Se compara la variable **sueldo** (Sueldo actual) con la Condición y se imprime el Resultado

```
El nuevo sueldo del trabajador es:
575
*** Ejecución Finalizada. ***
```

### Sueldo mayor a 1000.

Ingreso de la variable **sueldo** (Sueldo actual)

```
*** Ejecución Iniciada. ***
Ingrese el sueldo del trabajador:
> 1200
```

Se compara la variable **sueldo** (Sueldo actual) con la Condición y se imprime el Resultado

```
El nuevo sueldo del trabajador es:
1344
*** Ejecución Finalizada. ***
```

## Ejecutar

### Sueldo menor a 1000

```
*** Ejecución Iniciada. ***
Ingrese el sueldo del trabajador:
> 500
El nuevo sueldo del trabajador es:
575
*** Ejecución Finalizada. ***
Sueldo mayor a 1000.
*** Ejecución Iniciada. ***
Ingrese el sueldo del trabajador:
> 1200
El nuevo sueldo del trabajador es:
1344
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```java
/* Aplica un aumento del 15% al sueldo de un trabajador
* si este es menor a $1,000.00, en caso contrario un
* aumento del 12%
*/
package EstructuraSelectiva;

import java.util.Scanner;

public class Aumento_Sueldo {

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

     double sueldo, nuevoSueldo;

     System.out.println("Ingrese el sueldo del trabajador: ");
     sueldo = entrada.nextDouble();

     if (sueldo < 1000) {
        nuevoSueldo = sueldo * 1.15;
        System.out.println("El nuevo Sueldo del Trabajador es: \n" + nuevoSueldo);
     } else {
        nuevoSueldo = sueldo * 1.12;
        System.out.println("El nuevo Sueldo del Trabajador es: \n" + nuevoSueldo);
     }
  }
}
```

## Resultado Java

## Sueldo menor a 1000

```
run:
El sueldo del trabajado es:
500
El nuevo Sueldo del Trabajador es:
575
BUILD SUCCESSFUL (total time: 1 second)
```

### Sueldo mayor a 1000.

```
run:
El sueldo del trabajado es:
1200
El nuevo Sueldo del Trabajador es:
1344.0000000000002
BUILD SUCCESSFUL (total time: 1 second)
```

## Ejercicio 5

Construye un algoritmo dado como datos la matrícula y 5 calificaciones de un alumno; imprima la matrícula, el promedio y la palabra "Aprobado" si el alumno tiene un promedio mayor o igual que 6, y la palabra "No aprobado", en caso contrario.

```
1.  PROMEDIO
2.  nota1, nota2, nota3, nota4, nota5, promedio: Real
3.  matricula: Alfanumérico
4.  Inicio
5.  nota1←0
6.  nota2←0
7.  nota3←0
8.  nota4←0
9.  nota5←0
10. promedio←0
11. matricula←""
12. Escribir "Ingresa la Matrícula:";
13. Leer matricula
14. Escribir "Ingresa las calificaciones:";
15. Leer nota1, nota2, nota3, nota4, nota5
16. promedio ← (nota1 + nota2 + nota3 + nota4 + nota5) / 5
17. Escribir matrícula
18. Escribir promedio
19. Si (promedio ≥ 6) entonces
20.     Escribir "APROBADO"
21.  Si no
22.     Escribir "NO APROBADO"
23. Fin_si
24. Fin
```

## PRUEBA DE ESCRITORIO

| PROMEDIO | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **matricula** | **nota1** | **nota2** | **nota3** | **nota4** | **nota5** | **promedio** | Si **promedio ≥6** | Si no | Salida |
| ~~“”~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ | ~~APROBADO~~ | ~~NO APROBADO~~ | ~~0~~ |
| ~~SU123~~ | ~~8~~ | ~~6~~ | ~~7~~ | ~~8~~ | ~~9~~ | ~~7.6~~ | ~~Verdadero~~ | ~~Falso~~ | ~~APROBADO~~ |
| AB210 | 5 | 6 | 7 | 6 | 5 | 5.8 | Falso | Verdadero | NO APROBADO |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.   // Calcula el promedio de un alumno a partir de cinco
     calificaciones
2.    // ingresadas y muestra si aprueba o no.
3.   Algoritmo PROMEDIO
4.       DEFINIR nota1, nota2, nota3, nota4, nota5, promedio Como
     REAL;
5.       DEFINIR matricula Como ALFANUMERICO;
6.       nota1 ← 0;
7.       nota2 ← 0;
8.       nota3 ← 0;
9.       nota4 ← 0;
10.      nota5 ← 0;
11.      promedio ← 0;
12.      matricula ← “”;
13.      ESCRIBIR “Ingrese la matricula: “;
14.      LEER matricula;
15.      ESCRIBIR “Ingrese las calificaciones: “;
16.      LEER nota1, nota2, nota3, nota4, nota5;
17.      promedio ← (nota1 + nota2 + nota3 + nota4 + nota5) / 5
18.      ESCRIBIR “Matricula: “;
19.      ESCRIBIR matricula;
20.      ESCRIBIR “Promedio: “;
21.      ESCRIBIR promedio;
22.      SI (promedio ⩾ 6) ENTONCES
23.      .   ESCRIBIR “El alumno está APROBADO “;
```

```
24.     SINO
25.     .   ESCRIBIR “El alumno está REPROBADO “;
26.     FINSI
27. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Estudiante Aprobado

Ingreso de la variable **matricula**(Matrícula)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese la matricula:
3.  > SU123
```

Ingreso de las variables **C1, C2, C3, C4, C5** (Calificaciones)

```
1.  Ingrese las calificaciones:
2.  > 8
3.  > 6
4.  > 7
5.  > 8
6.  > 9
```

Cálculo de la variable **promedio** (Promedio), impresión de la variable **matricula** (Matrícula) y **promedio** (Promedio), comparación de **promedio** (Promedio) con la condición e impresión del resultado.

```
1.  Matricula:
2.  SU123
3.  Promedio:
4.  7.6
5.  El alumno está APROBADO
6.  *** Ejecución Finalizada. ***
```

### Estudiante Reprobado

Ingreso de la variable **matricula** (Matrícula)

```
*** Ejecución Iniciada. ***
Ingrese la matricula:
> SU123
```

Ingreso de las variables **C1, C2, C3, C4, C5** (Calificaciones)

```
Ingrese las calificaciones:
> 5
> 6
> 5
> 6
> 5
```

Cálculo de la variable **promedio** (Promedio), impresión de la variable **matricula** (Matrícula) y **promedio** (Promedio), comparación de **promedio** (Promedio) con la condición e impresión del resultado.

```
Matricula:
SU123
Promedio:
5.4
El alumno está REPROBADO
*** Ejecución Finalizada. ***
```

## Ejecutar

### Estudiante Aprobado

```
*** Ejecución Iniciada. ***
Ingrese la matricula:
> SU123
Ingrese las calificaciones:
```

```
> 8
> 6
> 7
> 8
> 9
Matricula:
SU123
Promedio:
7.6
El alumno está APROBADO
*** Ejecución Finalizada. ***
```

## Estudiante Reprobado

```
*** Ejecución Iniciada. ***
Ingrese la matricula:
> SU123
Ingrese las calificaciones:
> 5
> 6
> 5
> 6
> 5
Matricula:
SU123
Promedio:
5.4
El alumno está REPROBADO
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Calcula el promedio de un alumno a partir de cinco
calificaciones
// ingresadas y muestra si aprueba o no.
package EstructuraSelectiva;
```

```
4.
5.  import java.util.Scanner;
6.
7.  public class Promedio {
8.
9.      public static void main(String[] args) {
10.         Scanner entrada = new Scanner(System.in);
11.         double nota1, nota2, nota3, nota4, nota5, promedio;
12.         String matricula;
13.
14.         System.out.println(“Ingrese la matricula: “);
15.         matricula = entrada.nextLine();
16.
17.         System.out.println(“Ingresa la calificación 1: “);
18.         nota1 = entrada.nextDouble();
19.         System.out.println(“Ingresa la calificación 2: “);
20.         nota2 = entrada.nextDouble();
21.         System.out.println(“Ingresa la calificación 3: “);
22.         nota3 = entrada.nextDouble();
23.         System.out.println(“Ingresa la calificación 4: “);
24.         nota4 = entrada.nextDouble();
25.         System.out.println(“Ingresa la calificación 5: “);
26.         nota5 = entrada.nextDouble();
27.
28.         promedio = (nota1 + nota2 + nota3 + nota4 + nota5) / 5;
29.
30.         System.out.println(“Matricula: “ + matricula);
31.         System.out.println(“Promedio: “ + promedio);
32.
33.         if (promedio ⩾ 6) {
34.             System.out.println(“El alumno está APROBADO”);
35.         } else {
36.             System.out.println(“El alumno está REPROBADO”);
37.         }
38.     }
39. }
```

## Resultado Java

## Estudiante Aprobado

```
run:
Ingrese la matricula:
SU123
Ingresa la calificación 1:
8
Ingresa la calificación 2:
6
Ingresa la calificación 3:
7
Ingresa la calificación 4:
8
Ingresa la calificación 5:
9
Matricula: SU123
Promedio: 7.6
El alumno está APROBADO
BUILD SUCCESSFUL (total time: 11 seconds)
```

## Estudiante Reprobado

```
run:
Ingrese la matricula:
SU123
Ingresa la calificación 1:
5
Ingresa la calificación 2:
6
Ingresa la calificación 3:
5
Ingresa la calificación 4:
6
Ingresa la calificación 5:
5
Matricula: SU123
Promedio: 5.4
El alumno está REPROBADO
BUILD SUCCESSFUL (total time: 11 seconds)
```

## Ejercicio 6

Construye un algoritmo que, dados el nombre del grupo y el número de alumnos preinscritos, exprese el nombre del grupo, el número de alumnos inscritos y si el grupo será abierto o cerrado, puesto que, para abrir un grupo, se necesitan mínimo 30 alumnos.

```
1.  GRUPO
2.  nombre←Caracteres
3.  alumnos←entero;
4.  Inicio
5.  nombre←""
6.  alumnos←0
7.  Escribir "Ingresa el nombre del Grupo: "
8.  Leer nombre
9.  Escribir "Ingresa el total de alumnos preinscritos: "
10. Leer alumnos
11. Si (alumnos ⩾ 30) Entonces
12.    Escribir "GRUPO ABIERTO"
13. Sino
14.    Escribir "GRUPO CERRADO"
15. FinSi
16. Fin
```

### PRUEBA DE ESCRITORIO

| GRUPO | | | | |
|---|---|---|---|---|
| **nombre** | **alumnos** | Si (alumnos ⩾ 30) | Sino | Salida |
| "" | ~~0~~ | GRUPO ABIERTO | GRUPO CERRADO | ~~""~~ |
| ~~6" B"~~ | ~~25~~ | Falso | Verdadero | ~~GRUPO CERRADO~~ |
| 5" A" | 36 | Verdadero | Falso | GRUPO ABIERTO |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula si un grupo es abierto o cerrado a partir de
2.  // los lineamientos para apertura de grupos.
3.  Algoritmo GRUPO
4.      DEFINIR alumnos Como ENTERO;
5.      DEFINIR nombre Como CARACTER;
6.      alumnos ← 0;
7.      nombre ← 0;
8.      ESCRIBIR “Ingrese el nombre del grupo: “;
9.      LEER nombre;
10.     ESCRIBIR “Ingrese el número de alumnos preinscritos:  “;
11.     LEER alumnos;
12.     ESCRIBIR “Grupo: “;
13.     ESCRIBIR nombre;
14.     SI (alumnos ⩾ 30) ENTONCES
15.     .   ESCRIBIR “Grupo ABIERTO”;
16.     SINO
17.     .   ESCRIBIR “Grupo CERRADO”;
18.     FINSI
19. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Grupo Abierto

Ingreso de la variable **nombre** (Nombre del grupo)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el nombre del grupo:
3.  > 6 “B”
```

Ingreso de la variable **alumnos** (Alumnos preinscritos)

```
1.  Ingrese el total de alumnos preinscritos:
2.  > 40
```

Impresión del nombre del grupo y comparación de la variable **alumnos** (Alumnos preinscritos) con la condición e impresión del resultado.

```
Grupo:
6 “B”
GRUPO ABIERTO
*** Ejecución Finalizada. ***
```

### Grupo Cerrado

Ingreso de la variable **nombre** (Nombre del grupo)

```
*** Ejecución Iniciada. ***
Ingrese el nombre del grupo:
> 6 “B”
```

Ingreso de la variable **alumnos** (Alumnos preinscritos)

```
Ingrese el total de alumnos preinscritos:
> 25
```

Impresión del nombre del grupo y comparación de la variable **alumnos** (Alumnos preinscritos) con la condición e impresión del resultado.

```
Grupo:
6 “B”
GRUPO CERRADO
*** Ejecución Finalizada. ***
```

## Ejecutar

### Grupo Abierto

```
*** Ejecución Iniciada. ***
Ingrese el nombre del grupo:
> 6 “B”
```

```
Ingrese el total de alumnos preinscritos:
> 40
Grupo:
6 “B”
GRUPO ABIERTO
*** Ejecución Finalizada. ***
```

### Grupo Cerrado

```
*** Ejecución Iniciada. ***
Ingrese el nombre del grupo:
> 6 “B”

Ingrese el total de alumnos preinscritos:
> 25
Grupo:
6 “B”
GRUPO CERRADO
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Calcula si un grupo es abierto o cerrado a partir de
// los lineamientos para apertura de grupos.
package EstructuraSelectiva;

import java.util.Scanner;

public class Grupo {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int alumnos;
        String nombre;

        System.out.println(“Ingrese el nombre del grupo: “);
        nombre = entrada.nextLine();
        System.out.println(“Ingrese el total de alumnos
```

```
    preinscritos: “);
17.         alumnos = entrada.nextInt();
18.
19.         System.out.println(“El grupo: “ + nombre);
20.
21.         if (alumnos ⩾ 30) {
22.             System.out.println(“Grupo ABIERTO”);
23.         } else {
24.             System.out.println(“Grupo CERRADO”);
25.         }
26.     }
27. }
```

## Resultado Java

## Grupo Abierto

```
1.  run:
2.  Ingrese el nombre del grupo:
3.  Lis 601
4.  Ingrese el total de alumnos preinscritos:
5.  49
6.  El grupo: Lis 601
7.  Grupo ABIERTO
8.  BUILD SUCCESSFUL (total time: 4 seconds)
```

## Grupo Cerrado

```
9.  run:
10. Ingrese el nombre del grupo:
11. Lis 601
12. Ingrese el total de alumnos preinscritos:
13. 20
14. El grupo: Lis 601
15. Grupo CERRADO
16. BUILD SUCCESSFUL (total time: 14 seconds)
```

## Ejercicio 7

Escribe un algoritmo que calcule el descuento considerando las siguientes especificaciones:

Si el monto es mayor a $100, se aplica un descuento del 10% sobre la compra.

Si el monto es menor a $100, se aplica un descuento del 2% sobre el monto total de la compra.

```
1.  DESCUENTO
2.  monto, descuento, montoDesc: Real
3.  Inicio
4.  monto←0
5.  descuento←0
6.  montoDesc←0
7.  Escribir “Ingresa el monto: ”
8.  Leer monto
9.  Escribir “El monto era de: ”
10. Escribir monto
11. Si (monto ⩾ 100) entonces
12.   descuento ← monto * 0.1
13.   montoDesc← monto - descuento
14. Si no
15.   descuento ← monto * 0.02
16.   montoDesc ← monto - descuento
17. Fin_si
18. Escribir “El descuento es de: ”
19. Escribir descuento
20. Escribir “El monto con descuento es: ”
21. Escribir montoDesc
22. Fin
```

## PRUEBA DE ESCRITORIO

| DESCUENTO | | | | |
|---|---|---|---|---|
| **monto** | **Si (monto≥ 100)** | **Si no** | **descuento** | **montoDesc** |
| ~~0~~ | | | ~~0~~ | ~~0~~ |
| ~~80~~ | ~~Falso~~ | ~~Verdadero~~ | ~~1.60~~ | ~~78.4~~ |
| 100 | Verdadero | Falso | 10 | 90 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula el descuento considerando, si es más de $100
2.  // aplica el 10%, si es menor de $100, aplica el 2%.
3.  Algoritmo DESCUENTO
4.      DEFINIR monto, descuento, montoDesc Como REAL;
5.      monto ← 0;
6.      descuento ← 0;
7.      montoDesc ← 0;
8.      ESCRIBIR “Ingrese el monto: “;
9.      LEER monto;
10.     SI (monto ⩾ 100) ENTONCES
11.     .   descuento ← monto * 0.1;
12.     .   montoDesc ← monto - descuento;
13.     SINO
14.     .   descuento ← monto * 0.02;
15.     .   montoDesc ← monto - descuento;
16.     FINSI
17.     ESCRIBIR “El descuento es de: “;
18.     LEER descuento;
19.     ESCRIBIR “El monto con descuento es: “;
20.     LEER montoDesc;
21. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Descuento del 10%

Ingreso de la variable **monto** (Monto total)

```
*** Ejecución Iniciada. ***
Ingrese el monto:
> 120
```

Comparación de la variable **monto** (Monto total) con la condición, cálculo de la variable **descuento** (Descuento) y **montoDesc** (Monto con descuento) e impresión del **descuento** (Descuento) y **montoDesc** (Monto con descuento)

```
El descuento es de:
12
El monto con descuento es:
108
*** Ejecución Finalizada. ***
```

### Descuento del 2%

Ingreso de la variable **monto** (Monto total)

```
*** Ejecución Iniciada. ***
Ingrese el monto:
> 80
```

Comparación de la variable **monto** (Monto total) con la condición, cálculo de la variable **descuento** (Descuento) y **montoDesc** (Monto con descuento) e impresión del **descuento** (Descuento) y **montoDesc** (Monto con descuento)

```
El descuento es de:
1.6
El monto con descuento es:
78.4
*** Ejecución Finalizada. ***
```

## Ejecutar

### Descuento del 10%

```
*** Ejecución Iniciada. ***
Ingrese el monto:
> 120
El descuento es de:
12
El monto con descuento es:
108
*** Ejecución Finalizada. ***
```

### Descuento del 2%

```
*** Ejecución Iniciada. ***
Ingrese el monto:
> 80
El descuento es de:
1.6
El monto con descuento es:
78.4
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
/* Calcule el descuento considerando, si es más de o igual a
$100 aplica el 10%
   si es menor de $100 aplica el   2%*/
package EstructuraSelectiva;

import java.util.Scanner;

public class Descuento {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        double monto, descuento, montoDesc;
```

```
12.         System.out.println(“Ingresa el monto: “);
13.         monto = entrada.nextDouble();
14.
15.         if (monto ⩾ 100) {
16.             descuento = (monto * 10) / 100;
17.             montoDesc = monto - descuento;
18.         } else {
19.             descuento = (monto * 2) / 100;
20.             montoDesc = monto - descuento;
21.         }
22.
23.         System.out.println(“El descuento es de: “ + descuento);
24.         System.out.println(“El monto con descuento es: “ +
    montoDesc);
25.     }
26.
27. }
```

## Resultado Java

## Descuento del 10%

```
1.  run:
2.  Ingresa el monto:
3.  120
4.  El descuento es de: 12.0
5.  El monto con descuento es: 108.0
6.  BUILD SUCCESSFUL (total time: 1 second)
```

## Descuento del 2%

```
1.  run:
2.  Ingresa el monto:
3.  80
4.  El descuento es de: 1.6
5.  El monto con descuento es: 78.4
6.  BUILD SUCCESSFUL (total time: 1 second)
```

## Ejercicio 8

Escribe un algoritmo que solicite el nombre del jugador, número de goles; y que al jugador que tenga 6 goles o más, imprima en pantalla “GOLEADOR”, en caso contrario, deberá imprimirse “NO GOLEADOR.

```
1.  GOLES_ANOTADOS
2.  nombre: Cadena de caracteres
3.  goles: Real
4.  Inicio
5.  nombre←“”
6.  goles←0
7.  Escribe “Ingresa el nombre del jugador:”;
8.  Lee nombre
9.  Escribe “Ingresa el número de goles:”;
10. Lee goles
11. Escribe “Nombre del jugador”;
12. Escribe nombre
13. Si (goles⩾6) entonces
14.    Escribe “GOLEADOR”
15. SiNo
16.    Escribe “NO GOLEADOR”
17. FinSi
18. Fin
```

### PRUEBA DE ESCRITORIO

| GOLES_ANOTADOS | | | | |
|---|---|---|---|---|
| **nombre** | **goles** | **Si (goles ⩾ 6)** | **Si no** | Salida |
| ~~“”~~ | ~~0~~ | | | |
| ~~Antonio~~ | ~~5~~ | ~~Falso~~ | ~~Verdadero~~ | ~~NO GOLEADOR~~ |
| Luis | 8 | Verdadero | Falso | GOLEADOR |

### ALGORITMO EN PSeInt
### PRUEBA DE ESCRITORIO EN PSeInt

```
1.  // Calcula el tipo de jugador a partir de los goles anotados.
2.  // Si tiene 6 o más goles es "Goleador", en caso contrario
3.  // "No Goleador"
4.  Algoritmo GOLES_ANOTADOS
5.      DEFINIR goles Como Entero;
6.      DEFINIR nombre Como Carácter;
7.      goles ← 0;
8.      nombre ← "";
9.      ESCRIBIR "Ingrese el nombre del jugador: ";
10.     LEER nombre;
11.     ESCRIBIR "Ingrese la cantidad de goles anotados: ";
12.     LEER goles;
13.     ESCRIBIR "El jugador: ";
14.     ESCRIBIR nombre;
15.     SI (goles ⩾ 6) ENTONCES
16.     .   ESCRIBIR "Es GOLEADOR";
17.     SINO
18.     .   ESCRIBIR "Es NO GOLEADOR";
19.     FINSI
20. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Goleador

Ingreso de la variable **nombre** (Nombre del Jugador)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el nombre del jugador:
3.  > Antonio
```

Ingreso de la variable **goles** (Número de Goles)

```
1.  Ingrese el número de goles:
2.  > 8
```

Impresión de la variable nombre (Nombre del Jugador) y goles (Número de Goles).

Comparación de la variable goles (Número de Goles) con la condición e impresión de resultado.

```
El jugador:
Antonio
Es GOLEADOR
*** Ejecución Finalizada. ***
```

### No Goleador

Ingreso de la variable **nombre** (Nombre del Jugador)

```
*** Ejecución Iniciada. ***
Ingrese el nombre del jugador:
> Antonio
```

Ingreso de la variable **goles** (Número de Goles)

```
Ingrese el número de goles:
> 4
```

Impresión de la variable nombre (Nombre del Jugador) y goles (Número de Goles).

Comparación de la variable goles (Número de Goles) con la condición e impresión de resultado.

```
El jugador:
Antonio
Es NO GOLEADOR
*** Ejecución Finalizada. ***
```

## Ejecutar

### Goleador

```
*** Ejecución Iniciada. ***
Ingrese el nombre del jugador:
> Antonio
Ingrese el número de goles:
> 8
El jugador:
Antonio
Es GOLEADOR
*** Ejecución Finalizada. ***
```

### No Goleador

```
*** Ejecución Iniciada. ***
Ingrese el nombre del jugador:
> Antonio
Ingrese el número de goles:
> 4
El jugador:
Antonio
Es NO GOLEADOR
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
// Calcula el tipo de jugador a partir de los goles anotados.
// Si tiene 6 o más goles es “Goleador”, en caso contrario
// “No Goleador”
package EstructuraSelectiva;

import java.util.Scanner;

```

```
8.  public class Goles_Anotados {
9.
10.     public static void main(String[] args) {
11.         Scanner entrada = new Scanner(System.in);
12.         int goles;
13.         String nombre;
14.
15.         System.out.println(“Ingrese el nombre del jugador: “);
16.         nombre = entrada.nextLine();
17.         System.out.println(“Ingrese la cantidad de goles
    anotados: “);
18.         goles = entrada.nextInt();
19.
20.         System.out.println(“El jugador “ + nombre);
21.
22.         if (goles ⩾ 6) {
23.             System.out.println(“Es GOLEADOR”);
24.         } else {
25.             System.out.println(“Es NO GOLEADOR”);
26.         }
27.     }
28. }
```

## Resultado Java

## Goleador

```
1.  run:
2.  Ingrese el nombre del jugador:
3.  Antonio
4.  Ingrese la cantidad de goles anotados:
5.  8
6.  El jugador Antonio
7.  Es GOLEADOR
8.  BUILD SUCCESSFUL (total time: 8 seconds)
```

## No Goleador

```
run:
Ingrese el nombre del jugador:
Antonio
Ingrese la cantidad de goles anotados:
4
El jugador Antonio
Es NO GOLEADOR
BUILD SUCCESSFUL (total time: 8 seconds)
```

## Ejercicio 9

Escribe un algoritmo que identifique si el número ingresado es par o impar.

```
1.  PAR_IMPAR
2.  numero: Real
3.  Inicio
4.  numero←0
5.  Escribe “Ingrese el número:”;
6.  Lee numero
7.  Si (número MOD 2 = 0) entonces
8.  Escribe “EL NÚMERO ES PAR”
9.  Sino
10. Escribe “EL NÚMERO ES IMPAR”
11. Fin_si
12. Fin
```

### PRUEBA DE ESCRITORIO

| PAR_IMPAR | | | |
|---|---|---|---|
| **numero** | **Si (numero MOD 2 =0)** | **Sino** | Salida |
| ~~0~~ | | | ~~0~~ |
| ~~3~~ | ~~Falso~~ | ~~Verdadero~~ | ~~EL NÚMERO ES IMPAR~~ |
| 20 | Verdadero | Falso | EL NÚMERO ES PAR |

### ALGORITMO EN PSeInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Definir si el número ingresado es par o impar.
2.  Algoritmo PAR_IMPAR
3.      DEFINIR numero Como REAL;
4.      numero ← 0;
5.      ESCRIBIR “Ingrese el número: “;
6.      LEER numero;
7.      SI (numero MOD 2 = 0) ENTONCES
```

```
8.      .   ESCRIBIR “El número es PAR “;
9.      SINO
10.     .   ESCRIBIR “El número es IMPAR”;
11.     FINSI
12. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Número PAR

Ingreso de la variable **numero**

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el número:
3.  > 2
```

Comparación de la variable **numero** con la condición e impresión del resultado

```
1.  El número es PAR
2.  *** Ejecución Finalizada. ***
```

### Número IMPAR

Ingreso de la variable **numero**

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el número:
3.  > 3
```

Comparación de la variable **numero** con la condición e impresión del resultado

```
1.  El número es IMPAR
2.  *** Ejecución Finalizada. ***
```

## Ejecutar

### Número PAR

```
*** Ejecución Iniciada. ***
Ingrese el número:
> 2
El número es PAR
*** Ejecución Finalizada. ***
```

### Número IMPAR

```
*** Ejecución Iniciada. ***
Ingrese el número:
> 3
El número es IMPAR
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```java
// Define si el número ingresado es par o impar
package EstructuraSelectiva;

import java.util.Scanner;

public class Par_Impar {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        double numero;

        System.out.println(“Ingresa el número: “);
        numero = entrada.nextDouble();

        if (numero % 2 == 0) {
            System.out.println(“EL NÚMERO ES PAR”);
```

```
17.         } else {
18.             System.out.println(“EL NÚMERO ES IMPAR”);
19.         }
20.     }
21. }
```

## Resultado Java

## Número PAR

```
1.  run:
2.  Ingresa el número:
3.  2
4.  EL NÚMERO ES PAR
5.  BUILD SUCCESSFUL (total time: 2 seconds)
```

## Número IMPAR

```
1.  run:
2.  Ingresa el número:
3.  3
4.  EL NÚMERO ES PAR
5.  BUILD SUCCESSFUL (total time: 2 seconds)
```

## Ejercicio 10

Escribe un algoritmo que identifique si un producto va a ser pagado en efectivo o mediante tarjeta, los aspectos a considerar son:

Si el costo del producto a pagar es mayor a $200, se paga en efectivo.

Si el costo del producto es menor a $200, se paga con tarjeta.

```
1.  PRODUCTO
2.  nombre: Caracteres
3.  precio: Real
4.  Inicio
5.  nombre← ""
6.  precio←0
7.  Escribe "Ingresa el nombre del producto:"
8.  Lee nombre
9.  Escribe "Ingresa su precio:"
10. Lee precio
11. Si (precio ⩾ 200) entonces
12.   Escribe "El producto se paga en efectivo"
13. Sino
14.   Escribe "El producto se paga con tarjeta"
15. Fin_si
16. Fin
```

### PRUEBA DE ESCRITORIO

| PRODUCTO | | | | |
|---|---|---|---|---|
| **nombre** | **precio** | **Si (precio ⩾ 200)** | **Sino** | Salida |
| ~~""~~ | ~~0~~ | | | ~~""~~ |
| ~~Producto 1~~ | ~~180~~ | ~~Falso~~ | ~~Verdadero~~ | ~~El producto se paga con tarjeta~~ |
| Prducto 2 | 200 | Verdadero | Falso | El producto se paga en efectivo |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.   // Calcula si un producto se pagará con tarjeta o efectivo.
2.  // Si su precio es menor a 200 se paga con tarjeta, en caso
3.  // contrario con efectivo.
4.  Algoritmo PRODUCTO
5.      DEFINIR precio Como REAL;
6.      DEFINIR nombre Como CARACTER;
7.      precio ← 0;
8.      nombre ← “”;
9.      ESCRIBIR “Ingrese el nombre del producto: “;
10.     LEER nombre;
11.     ESCRIBIR “Ingrese el precio del producto: “;
12.     LEER precio;
13.     SI (precio ⩾ 200) ENTONCES
14.     .   ESCRIBIR “El producto se paga en efectivo”;
15.     SINO
16.     .   ESCRIBIR “El producto se paga con tarjeta”;
17.     FINSI
18. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Pago en efectivo

Ingreso de la variable **nombre** (nombre del producto)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el nombre del producto:
3.  > Producto 1
```

Ingreso de la variable **precio** (precio del producto)

```
1.  Ingrese el precio del producto:
2.  > 220
```

Comparación de la variable **precio** con la condición **Si** e impresión del resultado.

```
El producto se paga en efectivo
*** Ejecución Finalizada. ***
```

### Pago con tarjeta

Ingreso de la variable **nombre** (nombre del producto)

```
*** Ejecución Iniciada. ***
Ingrese el nombre del producto:
> Producto 1
```

Ingreso de la variable **precio** (precio del producto)

```
Ingrese el precio del producto:
> 180
```

Comparación de la variable **precio** con la condición **Si** e impresión del resultado.

```
El producto se paga con tarjeta
*** Ejecución Finalizada. ***
```

## Ejecutar

### Pago en efectivo

```
*** Ejecución Iniciada. ***
Ingrese el nombre del producto:
> Producto 1
Ingrese el precio del producto:
> 220
```

```
El producto se paga en efectivo
*** Ejecución Finalizada. ***
```

### Pago con tarjeta

```
*** Ejecución Iniciada. ***
Ingrese el nombre del producto:
> Producto 1
Ingrese el precio del producto:
> 180
El producto se paga con tarjeta
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```java
// Identifica si un producto es mayor a $200 se paga en efectivo
// y si es menos se paga con tarjeta.
package EstructuraSelectiva;

import java.util.Scanner;

public class Producto {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        String nombre;
        double precio;

        System.out.println("Ingresa el nombre del producto: ");
        nombre = entrada.nextLine();

        System.out.println("Ingresa su precio: ");
        precio = entrada.nextDouble();

        if (precio ⩾ 200) {
            System.out.println("El producto se paga en
```

```
    efectivo");
22.         } else {
23.             System.out.println("El producto se paga con
    tarjeta");
24.         }
25.
26.     }
27.
28. }
```

## Resultado Java

## Pago en efectivo

```
1.  run:
2.  Ingresa el nombre del producto:
3.  Producto 1
4.  Ingresa su precio:
5.  220
6.  El producto se paga en efectivo
7.  BUILD SUCCESSFUL (total time: 5 seconds)
```

## Pago con tarjeta

```
1.  run:
2.  Ingresa el nombre del producto:
3.  Producto 1
4.  Ingresa su precio:
5.  180
6.  El producto se paga con tarjeta
7.  BUILD SUCCESSFUL (total time: 5 seconds)
```

# Ejercicio 11

Construye un algoritmo, dado como datos de entrada tres números enteros. Determine si los mismos están en orden creciente.

```
1.  NUMEROS_CRECIENTES
2.  numero1, numero2, numero3: Entero.
3.  Inicio
4.  numero1←0
5.  numero2←0
6.  numero3←0
7.  Escribe “Ingresa el valor del primer número:”
8.  Lee numero1
9.  Escribe “Ingresa el valor del segundo número:”
10. Lee numero2
11. Escribe “Ingresa el valor del tercer número:”
12. Lee numero3
13. Escribe “Los números ingresados son: ”
14. Escribe numero1,”, “, numero2,”, “, numero3,”.”
15. Si (numero2 > numero1) y (numero3 > numero2) entonces
16.   Escribe “Los números están en orden creciente”
17. Si no
18.   Escribe “Los números no están en orden creciente”
19. Fin_si
20. Fin
```

## PRUEBA DE ESCRITORIO

| NUMEROS_CRECIENTES | | | | | | |
|---|---|---|---|---|---|---|
| Num1 | Num2 | Num3 | Num2 > num1 | Num3 > num2 | (Num2 > num1) y (num3 > num2) | SALIDA |
| ~~0~~ | ~~0~~ | ~~0~~ | | | | |
| ~~1~~ | ~~5~~ | ~~3~~ | ~~Verdadero~~ | ~~Falso~~ | ~~Falso~~ | ~~“Los números no están en orden creciente~~ |

| NUMEROS_CRECIENTES | | | | | | |
|---|---|---|---|---|---|---|
| Num1 | Num2 | Num3 | Num2 > num1 | Num3 > num2 | (Num2 > num1) y (num3 > num2) | SALIDA |
| ~~2~~ | ~~1~~ | ~~3~~ | ~~Falso~~ | ~~Verdadero~~ | ~~Falso~~ | ~~"Los números no están en orden creciente~~ |
| 4 | 6 | 10 | Verdadero | Verdadero | Verdadero | "Los números están en orden creciente" |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PSeInt

```
1.  // Determina si 3 números enteros están introducidos en
2.  // orden creciente.
3.  Algoritmo NUMEROS_CRECIENTES
4.      DEFINIR numero1, numero2, numero3 Como REAL;
5.      numero1 ← 0;
6.      numero2 ← 0;
7.      numero3 ← 0;
8.      ESCRIBIR "Ingrese el valor del primer número: ";
9.      LEER numero1;
10.     ESCRIBIR "Ingrese el valor del segundo número: ";
11.     LEER numero2;
12.     ESCRIBIR "Ingrese el valor del tercer número: ";
13.     LEER numero3;
14.     ESCRIBIR "Los números ingresados son: ";
15.     ESCRIBIR numero1,", ", numero2, ", ", numero3,".";
16.     SI (numero2 > numero1) Y (numero3 > numero3) ENTONCES
17.     .   ESCRIBIR "Los números están en orden creciente";
18.     SINO
19.     .  ESCRIBIR "Los números no están en orden creciente";
20.     FINSI
21. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Orden Creciente

Ingreso de la variable numero1 (número entero)

```
*** Ejecución Iniciada. ***
Ingrese el valor del primer número:
> 1
```

Ingreso de la variable numero2 (número entero)

```
Ingrese el valor del segundo número:
> 2
```

Ingreso de la variable numero3 (número entero)

```
Ingrese el valor del tercer número:
> 3
```

Impresión de la comparación de las variables con las condiciones **Si** e impresión de resultados.

```
Los números ingresados son:
1, 2, 3
Los números están en orden creciente.
*** Ejecución Finalizada. ***
```

### Orden No Creciente

Ingreso de la variable numero1 (número entero)

```
*** Ejecución Iniciada. ***
Ingrese el valor del primer número:
> 1
```

Ingreso de la variable numero2 (número entero)

```
Ingrese el valor del segundo número:
> 5
```

Ingreso de la variable numero3 (número entero)

```
Ingrese el valor del tercer número:
> 3
```

Impresión de la comparación de las variables con las condiciones **Si** e impresión de resultados.

```
Los números ingresados son:
1, 5, 3
Los números no están en orden creciente.
*** Ejecución Finalizada. ***
```

## Ejecutar

### Orden Creciente

```
*** Ejecución Iniciada. ***
Ingrese el valor del primer número:
> 1
Ingrese el valor del segundo número:
> 2
Ingrese el valor del tercer número:
> 3
Los números ingresados son:
1, 2, 3
Los números están en orden creciente.
*** Ejecución Finalizada. ***
```

### Orden No Creciente

```
*** Ejecución Iniciada. ***
Ingrese el valor del primer número:
> 1
Ingrese el valor del segundo número:
> 5
Ingrese el valor del tercer número:
> 3
Los números ingresados son:
1, 5, 3
Los números no están en orden creciente.
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Determina si 3 números enteros están introducidos en orden
creciente.
package EstructuraSelectiva;

import java.util.Scanner;

public class NumerosCreDecre {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int numero1, numero2, numero3;

        System.out.println(“Ingresa el primer número: “);
        numero1 = entrada.nextInt();
        System.out.println(“Ingresa el segundo número: “);
        numero2 = entrada.nextInt();
        System.out.println(“Ingresa el tercer número: “);
        numero3 = entrada.nextInt();

        System.out.println(“Los números ingresados son “ +
numero1 + “ “ + numero2 + “ “ + numero3);

```

```
22.         if (numero2 > numero1 && numero3 > numero2) {
23.             System.out.println(“Los números están en orden
    creciente”);
24.         } else {
25.             System.out.println(“Los números no están en orden
    creciente”);
26.         }
27.     }
28. }
```

## Resultado Java

## Orden Creciente

```
1.  run:
2.  Ingresa el primer número:
3.  1
4.  Ingresa el segundo número:
5.  2
6.  Ingresa el tercer número:
7.  3
8.  Los números ingresados son 1 2 3
9.  Los números están en orden creciente
10. BUILD SUCCESSFUL (total time: 3 seconds)
```

## Orden No Creciente

```
1.  run:
2.  Ingresa el primer número:
3.  1
4.  Ingresa el segundo número:
5.  5
6.  Ingresa el tercer número:
7.  3
8.  Los números ingresados son 1 5 3
9.  Los números no están en orden creciente
10. BUILD SUCCESSFUL (total time: 3 seconds)
```

# ESTRUCTURA SELECTIVA SI ANIDADO

## Ejercicio 1

Escribe un algoritmo que imprima el nombre y marcador. con el cual es ganador un equipo en cierto partido de futbol, se debe solicitar el nombre de los equipos del partido, y la cantidad de goles anotado por cada uno, se deben considerar también el número de goles anotados en los penaltis del partido.

```
1.  PARTIDO_FINAL
2.  equipoA, equipoB: Caracteres
3.  golesA, golesB, penalesA, penalesB: Enteros
4.  Inicio
5.  equipoA←“”
6.  equipoB←“”
7.  golesA←0
8.  golesB←0
9.  penalesA←0
10. penalesB←0
11. Escribe “Ingrese el nombre del equipo A: “;
12. Lee equipoA
13. Escribe “Ingrese la cantidad de goles del equipo A: “;
14. Lee golesA
15. Escribe “Ingrese el nombre del equipo B: “;
16. Lee equipoB
17. Escribe “Ingrese la cantidad de goles del equipo B: “;
18. Lee golesB
19. Si (golesA = golesB) entonces
20.      Escribe “Ingrese los goles anotados por el equipo A en los
    penaltis: “;
21.      Lee penalesA
```

```
22.      Escribe “Ingrese los goles anotados por el equipo B en los
    penaltis: “;
23.      Lee penalesB
24.     Si (penaltisA > penaltiB) entonces
25.              Escribe “GANÓ EL EQUIPO”, equipoA, “EN PENALES”
26.     Sino
27.              Escribe “GANÓ EL EQUIPO”, equipoB, “EN PENALES”
28.     Fin_si
29. Fin_si
30. Si (golA>golB) entonces
31.               Escribe “GANÓ EL EQUIPO ”
32.               Escribe equipoA
33. Sino
34.       Escribe “GANÓ EL EQUIPO ”
35.       Escribe equipoB
36. Fin_si
37. Fin
```

## PRUEBA DE ESCRITORIO

| PARTIDO_FINAL | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **equipo A** | **golesA** | **equipoB** | **golesB** | **penalesA** | **penalesB** | **Si SI(penaltisA > penaltisB)** | Si no Sino | **Si (golesA > golesB)** | **Si (golesA < golesB)** | **Salida** |
| “” | ~~0~~ | “” | ~~0~~ | ~~0~~ | ~~0~~ | | | | | |
| ~~Cachorros~~ | ~~7~~ | ~~Jaguares~~ | ~~4~~ | ~~Falso~~ | | | | ~~Verdadero~~ | ~~Falso~~ | ~~Ganó el equipo Cachorros~~ |
| ~~Sauce~~ | ~~0~~ | ~~La Perla~~ | ~~3~~ | ~~Falso~~ | | | | ~~Falso~~ | ~~Verdadero~~ | ~~Ganó el equipo La Perla~~ |

| PARTIDO_FINAL | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| ~~Júpiter~~ | ~~5~~ | ~~Army~~ | ~~5~~ | ~~3~~ | ~~2~~ | ~~Verdadero~~ | ~~Falso~~ | ~~Falso~~ | ~~Falso~~ | ~~Ganó el equipo Júpiter~~ |
| Chispa | 2 | Oeste | 2 | 0 | 1 | Falso | Verdadero | Falso | Falso | Ganó el equipó Oeste |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula el equipo ganador a partir de la información
    proporcionada
2.  // por el usuario.
3.  Algoritmo PARTIDO_FINAL
4.      DEFINIR golesA, golesB, penalesA, penalesB Como ENTERO;
5.      DEFINIR equipoA, equipoB Como CARACTER;
6.      golesA ← 0;
7.      golesB ← 0;
8.      penalesA ← 0;
9.      penalesB ← 0;
10.     equipoA ← “”;
11.     equipoB ← “”;
12.     ESCRIBIR “Ingrese el nombre del equipo A: “;
13.     LEER equipoA;
14.     ESCRIBIR “Ingrese los goles del equipo A: “;
15.     LEER golesA;
16.     ESCRIBIR “Ingrese el nombre del equipo B: “;
17.     LEER equipoB;
18.     ESCRIBIR “Ingrese los goles del equipo B: “;
19.     LEER golesB;
20.     SI (golesA = golesB) ENTONCES
21.     .   ESCRIBIR “Ingrese los penales anotados del equipo A: “;
22.     .   LEER penalesA;
23.     .   ESCRIBIR “Ingrese los penales anotados del equipo B: “;
24.     .   LEER penalesB;
25.     .   SI (penalesA > penalesB) ENTONCES
26.     .   .   ESCRIBIR “Ganó el equipo “;
27.     .   .   ESCRIBIR equipoA;
```

```
28.      .   SINO
29.      .   .   ESCRIBIR “Ganó el equipo “;
30.      .   .   ESCRIBIR equipoB;
31.      .   FINSI
32.      FINSI
33.      SI (golesA > golesB) ENTONCES
34.      .   ESCRIBIR “Ganó el equipo “;
35.      .   ESCRIBIR equipoA;
36.      SINO
37.      .   ESCRIBIR “Ganó el equipo “;
38.      .   ESCRIBIR equipoB;
39.      FINSI
40.  FinAlgoritmo
```

## Ejecutar Paso a Paso

### Gana Equipo A

Ingreso de la variable **equipoA** (nombre del equipo A)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el nombre del equipo A:
3.  > Cachorros
```

Ingreso de la variable **golesA** (goles anotados por el equipo A)

```
1.  Ingrese los goles del equipo A:
2.  > 7
```

Ingreso de la variable **equipoB** (nombre del equipo B)

```
1.  Ingrese el nombre del equipo B:
2.  > Jaguares
```

Ingreso de la variable **golesB** (goles anotados por el equipo B)

```
Ingrese los goles del equipo B:
> 4
```

Comparación con la condición e impresión del resultado

```
Ganó el equipo
Cachorros
*** Ejecución Finalizada. ***
```

### Gana Equipo B

Ingreso de la variable **equipoA** (nombre del equipo A)

```
*** Ejecución Iniciada. ***
Ingrese el nombre del equipo A:
> Cachorros
```

Ingreso de la variable **golesA** (goles anotados por el equipo A)

```
Ingrese los goles del equipo A:
> 4
```

Ingreso de la variable **equipoB** (nombre del equipo B)

```
Ingrese el nombre del equipo B:
> Jaguares
```

Ingreso de la variable **golesB** (goles anotados por el equipo B)

```
Ingrese los goles del equipo B:
> 7
```

Comparación con la condición e impresión del resultado

```
Ganó el equipo
Jaguares
*** Ejecución Finalizada. ***
```

**Gana Equipo A en penales.**

Ingreso de la variable **equipoA** (nombre del equipo A)

```
*** Ejecución Iniciada. ***
Ingrese el nombre del equipo A:
> Cachorros
```

Ingreso de la variable **golesA** (goles anotados por el equipo A)

```
Ingrese los goles del equipo A:
> 4
```

Ingreso de la variable **equipoB** (nombre del equipo B)

```
Ingrese el nombre del equipo B:
> Jaguares
```

Ingreso de la variable **golesB** (goles anotados por el equipo B)

```
Ingrese los goles del equipo B:
> 4
```

Comparación con la condición e ingreso de la variable **penalesA**(penales anotados por el equipo A)

```
Ingrese los penales anotados del equipo A:
> 4
```

Ingreso de la variable **penalesB**(penales anotados por el equipo B)

```
Ingrese los penales anotados del equipo B:
> 2
```

Comparación con la condición e impresión del resultado

```
Ganó el equipo
Cachorros
*** Ejecución Finalizada. ***
```

**Gana Equipo B en penales.**

Ingreso de la variable **equipoA** (nombre del equipo A)

```
*** Ejecución Iniciada. ***
Ingrese el nombre del equipo A:
> Cachorros
```

Ingreso de la variable **golesA** (goles anotados por el equipo A)

```
Ingrese los goles del equipo A:
> 4
```

Ingreso de la variable **equipoB** (nombre del equipo B)

```
Ingrese el nombre del equipo B:
> Jaguares
```

Ingreso de la variable **golesB** (goles anotados por el equipo B)

```
1.  Ingrese los goles del equipo B:
2.  > 4
```

Comparación con la condición e ingreso de la variable **penalesA**(penales anotados por el equipo A)

```
1.  Ingrese los penales anotados del equipo A:
2.  > 2
```

Ingreso de la variable **penalesB**(penales anotados por el equipo B)

```
1.  Ingrese los penales anotados del equipo B:
2.  > 4
```

Comparación con la condición e impresión del resultado

```
1.  Ganó el equipo
2.  Jaguares
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

### Gana Equipo A

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el nombre del equipo A:
3.  > Cachorros
4.  Ingrese los goles del equipo A:
5.  > 7
6.  Ingrese el nombre del equipo B:
7.  > Jaguares
8.  Ingrese los goles del equipo B:
```

```
> 4
Ganó el equipo
Cachorros
*** Ejecución Finalizada. ***
```

### Gana Equipo B

```
*** Ejecución Iniciada. ***
Ingrese el nombre del equipo A:
> Cachorros
Ingrese los goles del equipo A:
> 4
Ingrese el nombre del equipo B:
> Jaguares
Ingrese los goles del equipo B:
> 7
Ganó el equipo
Jaguares
*** Ejecución Finalizada. ***
```

### Gana Equipo A en penales.

```
*** Ejecución Iniciada. ***
Ingrese el nombre del equipo A:
> Cachorros
Ingrese los goles del equipo A:
> 4
Ingrese el nombre del equipo B:
> Jaguares
Ingrese los goles del equipo B:
> 4
Ingrese los penales anotados del equipo A:
> 4
Ingrese los penales anotados del equipo B:
> 2
Ganó el equipo
```

```
Cachorros
*** Ejecución Finalizada. ***
```

## Gana Equipo B en penales.

```
*** Ejecución Iniciada. ***
Ingrese el nombre del equipo A:
> Cachorros
Ingrese los goles del equipo A:
> 4
Ingrese el nombre del equipo B:
> Jaguares
Ingrese los goles del equipo B:
> 4
Ingrese los penales anotados del equipo A:
> 2
Ingrese los penales anotados del equipo B:
> 4
Ganó el equipo
Jaguares
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Calcula el equipo ganador a partir de la información
proporcionada
// por el usuario.
package EstructuraSelectiva;

import java.util.Scanner;

public class Partido_Final {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        String equipoA, equipoB;
```

```
12.         int golesA, golesB, penalesA, penalesB;
13.
14.         System.out.println(“Ingresa el nombre del equipo A:”);
15.         equipoA = entrada.next();
16.         System.out.println(“Ingresa los goles del equipo A:”);
17.         golesA = entrada.nextInt();
18.         System.out.println(“Ingresa el nombre del equipo B:”);
19.         equipoB = entrada.next();
20.         System.out.println(“Ingresa los goles del equipo B:”);
21.         golesB = entrada.nextInt();
22.
23.         if (golesA == golesB) {
24.             System.out.println(“Ingresa los penales anotados del
    equipo A:”);
25.             penalesA = entrada.nextInt();
26.             System.out.println(“Ingresa los penales anotados del
    equipo B:”);
27.             penalesB = entrada.nextInt();
28.             if (penalesA > penalesB) {
29.                 System.out.println(“Ganó el equipo “ + equipoA);
30.             } else {
31.                 System.out.println(“Ganó el equipo “ + equipoB);
32.             }
33.         } else {
34.             if (golesA > golesB) {
35.                 System.out.println(“Ganó el equipo “ + equipoA);
36.             } else {
37.                 System.out.println(“Ganó el equipo “ + equipoB);
38.             }
39.         }
40.     }
41. }
```

## Resultado en Java

### Gana Equipo A

```
run:
Ingresa el nombre del equipo A:
Cachorros
Ingresa los goles del equipo A:
4
Ingresa el nombre del equipo B:
Jaguares
Ingresa los goles del equipo B:
2
Ganó el equipo Cachorros
BUILD SUCCESSFUL (total time: 12 seconds)
```

### Gana Equipo B

```
run:
Ingresa el nombre del equipo A:
Cachorros
Ingresa los goles del equipo A:
2
Ingresa el nombre del equipo B:
Jaguares
Ingresa los goles del equipo B:
4
Ganó el equipo Jaguares
BUILD SUCCESSFUL (total time: 12 seconds)
```

### Gana Equipo A en penales

```
run:
Ingresa el nombre del equipo A:
Cachorros
Ingresa los goles del equipo A:
```

```
4
Ingresa el nombre del equipo B:
Jaguares
Ingresa los goles del equipo B:
4
Ingresa los penales anotados del equipo A:
4
Ingresa los penales anotados del equipo B:
2
Ganó el equipo Cachorros
BUILD SUCCESSFUL (total time: 12 seconds)
```

## Gana Equipo B en penales

```
run:
Ingresa el nombre del equipo A:
Cachorros
Ingresa los goles del equipo A:
4
Ingresa el nombre del equipo B:
Jaguares
Ingresa los goles del equipo B:
4
Ingresa los penales anotados del equipo A:
2
Ingresa los penales anotados del equipo B:
4
Ganó el equipo Jaguares
BUILD SUCCESSFUL (total time: 12 seconds)
```

## Ejercicio 2

Construye un algoritmo dado como dato un número entero. Determine e imprima si el mismo es positivo, negativo o nulo.

```
1.  NUMEROS
2.  numero: Entero
3.  Inicio
4.  numero←0
5.  Escribe “Ingresa el número entero:”
6.  Lee numero
7.  Si (numero = 0) entonces
8.     Escribe “El número es nulo”
9.  Sino
10.    Si (numero ⩾ 1) entonces
11.      Escribe “El número es positivo”
12.    Sino
13.      Si (numero < 0) entonces
14.         Escribe “El número es negativo”
15.      Fin_si
16.    Fin_si
17. Fin_si
18. Fin
```

### PRUEBA DE ESCRITORIO

| NÚMEROS | | | | |
|---|---|---|---|---|
| **numero** | Si (**num**=0) | Si (**num** ⩾ 1) | Si (**num** <0) | Resultado |
| ~~0~~ | | | | |
| ~~−12~~ | ~~Falso~~ | ~~Falso~~ | ~~Verdadero~~ | ~~Es negativo~~ |
| ~~0~~ | ~~Verdadero~~ | ~~Falso~~ | ~~Falso~~ | ~~Es nulo~~ |
| 30 | Falso | Verdadero | Falso | Es positivo |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PSeInt

```
1.  // Determina si un número ingresado es negativo, nulo o
    positivo.
2.  Algoritmo NUMEROS
3.      DEFINIR numero Como ENTERO;
4.      numero ← 0;
5.      ESCRIBIR “Ingrese el número entero: “;
6.      LEER numero;
7.      SI (numero = 0) ENTONCES
8.      .   ESCRIBIR “El número es NULO “;
9.      SINO
10.     .    SI (numero ⩾ 1) ENTONCES
11.     .    .   ESCRIBIR “El número es POSITIVO”;
12.     .    SINO
13.     .    .    SI (numero < 0) ENTONCES
14.     .    .    .   ESCRIBIR “El número es NEGATIVO”;
15.     .    .    FINSI
16. .    FINSI
17.     FINSI
18. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Número Nulo

Ingreso de la variable **numero** (número entero)

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el número entero:
3.  > 0
```

Impresión de la variable **numero** (número entero), comparación de la variable con las condiciones Si e impresión de resultados.

```
1.  El número es NULO
2.  *** Ejecución Finalizada. ***
```

### Número Positivo

Ingreso de la variable **numero** (número entero)

```
*** Ejecución Iniciada. ***
Ingrese el número entero:
> 10
```

Impresión de la variable **numero** (número entero), comparación de la variable con las condiciones Si e impresión de resultados.

```
El número es POSITIVO
*** Ejecución Finalizada. ***
```

### Número Negativo

Ingreso de la variable **numero** (número entero)

```
*** Ejecución Iniciada. ***
Ingrese el número entero:
> -10
```

Impresión de la variable **numero** (número entero), comparación de la variable con las condiciones Si e impresión de resultados.

```
El número es NEGATIVO
*** Ejecución Finalizada. ***
```

## Ejecutar

### Número Nulo

```
*** Ejecución Iniciada. ***
Ingrese el número entero:
> 0
El número es NULO
*** Ejecución Finalizada. ***
```

### Número Positivo

```
*** Ejecución Iniciada. ***
Ingrese el número entero:
> 10
El número es POSITIVO
*** Ejecución Finalizada. ***
```

### Número Negativo

```
*** Ejecución Iniciada. ***
Ingrese el número entero:
> -10
El número es NEGATIVO
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Determina si el número ingresado es positivo, negativo o nulo.
package EstructuraSelectiva;

import java.util.Scanner;

public class Numeros {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int numero;

        System.out.println("Ingresa el número entero:");
        numero = entrada.nextInt();

        if (numero == 0) {
            System.out.println("El número es NULO");
        } else {
            if (numero >= 1) {
                System.out.println("El número es POSITIVO");
            } else {
```

```
                if (numero < 0) {
                    System.out.println("El número es NEGATIVO");
                }
            }
        }
    }
}
```

## Resultado Java

## Número Nulo

```
run:
Ingresa el número entero:
0
El número es NULO
BUILD SUCCESSFUL (total time: 1 second)
```

## Número Positivo

```
run:
Ingresa el número entero:
10
El número es POSITIVO
BUILD SUCCESSFUL (total time: 1 second)
```

## Número Negativo

```
run:
Ingresa el número entero:
-10
El número es NEGATIVO
BUILD SUCCESSFUL (total time: 1 second)
```

## Ejercicio 3

Construya un algoritmo dado el monto de la compra de un cliente, que determine lo que se debe pagar.

En una tienda efectúan un descuento a los clientes dependiendo del monto de la compra. El descuento se efectúa con base en el siguiente criterio:

Si el monto es menor que $500.00, no hay descuento.

Si el monto está comprendido entre $500 y $1000.00, tiene un 5% de descuento.

Si el monto está comprendido entre $1000 y $7000.00, tiene un descuento del 11%.

Si el monto está comprendido entre $7,000 y $15,000.00, tiene un 18% de descuento.

Si el monto es mayor a $15,000.00, tiene un 25% de descuento.

```
1.  DESCUENTO_MONTO
2.  monto, descuento: Real
3.  Inicio.
4.  monto←0
5.  descuento←0
6.  Escribir “Ingresa el monto de la compra:”;
7.  Leer monto;
8.  Si monto < 500 entonces
9.    descuento ← monto
10.   Escribir “El cliente no tiene descuento.”
11.   Escribir “El cliente debe pagar”, descuento
12. Si no
13.   Si monto > 500 y monto ⩽ 1000 entonces
14.     descuento ← monto -(monto * 0.05)
15.     Escribir “El cliente debe pagar:”, descuento
16.   Si no
17.     Si monto > 1000 y monto ⩽ 7000 entonces
18.       descuento ← monto - (monto * 0.11)
19.       Escribir “El cliente debe pagar:”, descuento
```

```
20.      Si no
21.        Si monto > 7000 y monto ⩽ 15000 entonces
22.          descuento ← monto - (monto * 0.18)
23.          Escribir “El cliente debe pagar:”, descuento
24.        Si no
25.          Si monto > 15000 entonces
26.            descuento ← monto - (monto * 0.25)
27.            Escribir “El cliente debe pagar:”, descuento
28.          FinSi
29.        FinSi
30.      FinSi
31.    FinSi
32.  FinSi
33.  Fin
```

## PRUEBA DE ESCRITORIO

| **Descuento_Monto** | | | | | |
|---|---|---|---|---|---|
| **monto** | 300 | 550 | 1350 | 8000 | 16500 |
| Si **monto**<500 | VERDADERO | FALSO | FALSO | FALSO | FALSO |
| Si **monto**>500 y **monto** ⩽ 1000 | | VERDADERO | FALSO | FALSO | FALSO |
| Si **monto**>1000 y **monto** ⩽ 7000 | | | VERDADERO | FALSO | FALSO |
| Si **monto**>7000 y **monto** ⩽ 15000 | | | | VERDADERO | FALSO |
| Si **monto**>15000 | | | | | VERDADERO |
| Operación | **descuento**← **monto** | **descuento**← **monto**-(**monto***0.05) | **descuento**← **monto**-(**monto***0.11) | **descuento**← **monto**-(**monto***0.18) | **descuento**← **monto**-(**monto***0.25) |
| Resultado | 300 | 522.50 | 1201.50 | 6560 | 12375 |

## ALGORITMO EN Pseint
## PRUEBA DE ESCRITORIO EN PSeInt

```
1.  // Determina el descuento a partir del monto ingresado.
2.  Algoritmo DESCUENTO_MONTO
3.      DEFINIR monto, descuento Como REAL;
4.      monto ← 0;
5.      descuento ← 0;
6.      ESCRIBIR “Ingrese el monto de la compra: “;
7.      LEER monto;
8.      SI (monto ⩽ 500) ENTONCES
9.      .   ESCRIBIR “El monto no tiene descuento”;
10.     .   ESCRIBIR “El cliente debe pagar”;
11.     .   ESCRIBIR monto;
12.     SINO
13.     .   SI (monto > 500) Y (monto ⩽ 1000) ENTONCES
14.     .   .   descuento ← monto - (monto* 0.05);
15.     .   .   ESCRIBIR “El cliente debe pagar”;
16.     .   .   ESCRIBIR descuento;
17.     .   SINO
18.     .   .    SI (monto > 1000) Y (monto ⩽ 7000) ENTONCES
19.     .   .    .   descuento ← monto - (monto * 0.11);
20.     .   .    .   ESCRIBIR “El cliente debe pagar”;
21.     .   .    .   ESCRIBIR descuento;
22.     .   .    .    SI (monto > 7000) Y (monto ⩽ 15000) ENTONCES
23.     .    .    .     descuento ← monto - (monto * 0.18);
24.     .    .   .     ESCRIBIR “El cliente debe pagar”;
25.     .    .   .     ESCRIBIR descuento;
26.     .    .    .     SI (monto > 15000) ENTONCES
27.     .    .    .     .   descuento ← monto - (monto * 0.25);
28.     .    .   .     .   ESCRIBIR “El cliente debe pagar”;
29.     .    .   .     .   ESCRIBIR descuento;
30.     .    .   .     FINSI
31.     .    .    FINSI
32. .    FINSI
33.     FINSI
34. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Monto sin descuento

Ingreso del monto de la compra

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el monto de la compra:
3.  > 490
```

Impresión del monto total de la compra

```
1.  El monto no tiene descuento
2.  El cliente debe pagar
3.  490
4.  *** Ejecución Finalizada. ***
```

### Monto con el 5% de descuento

Ingreso del monto de la compra

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el monto de la compra:
3.  > 620
```

Impresión del monto total de la compra

```
1.  El cliente debe pagar
2.  589
3.  *** Ejecución Finalizada. ***
```

### Monto con el 11% de descuento

Ingreso del monto de la compra

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el monto de la compra:
3.  > 1920
```

Impresión del monto total de la compra

```
El cliente debe pagar
1708.8
*** Ejecución Finalizada. ***
```

### Monto con el 18% de descuento

Ingreso del monto de la compra

```
*** Ejecución Iniciada. ***
Ingrese el monto de la compra:
> 8000
```

Impresión del monto total de la compra

```
El cliente debe pagar
6560
*** Ejecución Finalizada. ***
```

### Monto con el 25% de descuento

Ingreso del monto de la compra

```
*** Ejecución Iniciada. ***
Ingrese el monto de la compra:
> 18000
```

Impresión del monto total de la compra

```
El cliente debe pagar
13500
*** Ejecución Finalizada. ***
```

## Ejecutar

### Monto sin descuento

```
*** Ejecución Iniciada. ***
Ingrese el monto de la compra:
> 490
El monto no tiene descuento
El cliente debe pagar
490
*** Ejecución Finalizada. ***
```

### Monto con el 5% de descuento

```
*** Ejecución Iniciada. ***
Ingrese el monto de la compra:
> 620
El cliente debe pagar
589
*** Ejecución Finalizada. ***
```

### Monto con el 11% de descuento

```
*** Ejecución Iniciada. ***
Ingrese el monto de la compra:
> 1920
El cliente debe pagar
1708.8
*** Ejecución Finalizada. ***
```

### Monto con el 18% de descuento

```
*** Ejecución Iniciada. ***
Ingrese el monto de la compra:
> 8000
El cliente debe pagar
6560
*** Ejecución Finalizada. ***
```

### Monto con el 25% de descuento

```
*** Ejecución Iniciada. ***
Ingrese el monto de la compra:
> 18000
El cliente debe pagar
13500
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Determina el descuento a partir del monto ingresado.
package EstructuraSelectiva;
import java.util.Scanner;

public class Descuento_Monto {

  public static void main(String[] args) {
     Scanner entrada = new Scanner(System.in);
     double monto, descuento;

     System.out.println("Ingresa el monto de la compra: ");
     monto = entrada.nextDouble();

     if (monto ⩽ 500) {
        descuento = monto;
        System.out.println("El monto no tiene descuento");
        System.out.println("El cliente pagará $" + descuento);
     } else {
        if ((monto > 500) && (monto ⩽ 1000)) {
           descuento = monto - (monto * 0.05);
           System.out.println("El cliente pagará $" +
descuento);
        } else {
           if ((monto > 1000) && (monto ⩽ 7000)) {
              descuento = monto - (monto * 0.11);
              System.out.println("El cliente pagará $" +
descuento);
```

```
            } else {
               if ((monto > 7000) && (monto ⩽ 15000)) {
                  descuento = monto - (monto * 0.18);
                  System.out.println("El cliente pagará $" +
descuento);
                  } else {
                     if (monto > 15000) {
                        descuento = monto - (monto * 0.25);
                        System.out.println("El cliente pagará $"
+
descuento);
                     }
                  }
               }
            }

      }

}
```

## Resultado Java

## Monto sin descuento

```
run:
Ingresa el monto de la compra:
499
El monto no tiene descuento
El cliente pagará $499.0
BUILD SUCCESSFUL (total time: 5 seconds)
```

## Monto con el 5% de descuento

```
run:
Ingresa el monto de la compra:
```

```
3.  620
4.  El cliente pagará $589.0
5.  BUILD SUCCESSFUL (total time: 5 seconds)
```

### Monto con el 11% de descuento

```
1.  run:
2.  Ingresa el monto de la compra:
3.  1920
4.  El cliente pagará $1708.8
5.  BUILD SUCCESSFUL (total time: 5 seconds)
```

### Monto con el 18% de descuento

```
1.  run:
2.  Ingresa el monto de la compra:
3.  8000
4.  El cliente pagará $6560.0
5.  BUILD SUCCESSFUL (total time: 5 seconds)
```

### Monto con el 25% de descuento

```
1.  run:
2.  Ingresa el monto de la compra:
3.  18000
4.  El cliente pagará $13500.0
5.  BUILD SUCCESSFUL (total time: 5 seconds)
```

## Ejercicio 4

Diseñe un algoritmo que lea el costo básico de un artículo y calcule su precio total (precio total igual a precio básico más impuesto).

```
1.  IMPUESTO
2.  precio,impuesto:Real
3.  Inicio
4.  precio←0
5.  impuesto←0
6.  Escribir “Dame el precio de los productos”; Leer M
7.  Si precio < 20 entonces
8.    Escribir “No paga impuesto”;
9.  Si no
10.   Si precio ⩾ 20 y precio ⩽ 40 entonces
11.     impuesto ← precio * 1.30
12.   Sino
13.     Si precio > 40 y precio < 500 entonces
14.       impuesto ← precio * 1.40
15.     Sino
16.       impuesto ← precio * 1.50
17.     FinSi
18.   FinSi
19. FinSi
20. Escribir “El precio con impuestos es:”, MI
21. Fin
```

## PRUEBA DE ESCRITORIO

| IMPUESTO | | | | | |
|---|---|---|---|---|---|
| precio | Si precio < 20 (No paga impuesto) | Si precio ⩾ 20 Y precio ⩽ 40 (impuesto = precio * 1.30) | Si precio > 40 y precio < 500 (impuesto = precio * 1.40) | Sino precio ⩾ 500 (impuesto = precio * 1.50) | Salida |
| 15 | VERDADERO | FALSO | FALSO | FALSO | 15 |
| 30 | FALSO | VERDADERO | FALSO | FALSO | 39 |
| 200 | FALSO | FALSO | VERDADERO | FALSO | 280 |
| 1000 | FALSO | FALSO | FALSO | VERDADERO | 1500 |

## ALGORITMO EN PSeint
## PRUEBA DE ESCRITORIO EN PSeInt

```
1.  // Calcula el impuesto que se debe de pagar por los artículos
2.  // adquiridos.
3.  Algoritmo IMPUESTO
4.      DEFINIR precio, impuesto Como REAL;
5.      precio ← 0;
6.      impuesto ← 0;
7.      ESCRIBIR “Ingrese el precio del producto: “;
8.      LEER precio;
9.      SI (precio < 20)ENTONCES
10.     .  ESCRIBIR “El producto no paga impuestos”;
11.     .  impuesto ← precio;
12.     SINO
13.     .  SI (precio ⩾ 20) Y (precio ⩽ 40) ENTONCES
14.     .  .   impuesto ← precio * 1.30;
15.     .  SINO
16.     .  .    SI (precio > 40) Y (precio ⩽ 500) ENTONCES
17.     .  .    .   impuesto ← precio * 1.40;
18.     .  .    SINO
19.     .  .    .   impuesto ← precio * 1.50;
20. .  .    FINSI
21.     .  FINSI
22.     FINSI
23.     ESCRIBIR “El precio del producto es:”;
24.     ESCRIBIR impuesto;
25. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Sin impuesto

Ingreso del precio del articulo

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 5
```

Impresión del monto total con impuesto

```
El producto no paga impuestos
El precio del producto es:
5
*** Ejecución Finalizada. ***
```

### Impuesto del 30%

Ingreso del precio del articulo

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 30
```

Impresión del monto total con impuesto

```
El precio del producto es:
39
*** Ejecución Finalizada. ***
```

### Impuesto del 40%

Ingreso del precio del articulo

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 400
```

Impresión del monto total con impuesto

```
El precio del producto es:
560
*** Ejecución Finalizada. ***
```

### Impuesto del 50%

Ingreso del precio del articulo

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 600
```

Impresión del monto total con impuesto

```
El precio del producto es:
900
*** Ejecución Finalizada. ***
```

## Ejecutar

### Sin impuesto

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 5
El producto no paga impuestos
El precio del producto es:
5
*** Ejecución Finalizada. ***
```

### Impuesto del 30%

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 30
El precio del producto es:
39
*** Ejecución Finalizada. ***
```

### Impuesto del 40%

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 400
El precio del producto es:
560
*** Ejecución Finalizada. ***
```

### Impuesto del 50%

```
*** Ejecución Iniciada. ***
Ingrese el precio del producto:
> 600
El precio del producto es:
900
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/*Diseñe un algoritmo que lea el costo básico de un artículo y calcule
  su precio total (precio total igual a precio básico más impuesto).*/
package EstructuraSelectiva;
```

```
4.
5.  import java.util.Scanner;
6.
7.  public class Impuesto {
8.
9.      public static void main(String[] args) {
10.         Scanner entrada = new Scanner(System.in);
11.         double precio = 0;
12.         double impuesto = 0;
13.
14.         System.out.println(“Ingresa el precio del producto:”);
15.         precio = entrada.nextDouble();
16.
17.         if (precio < 20) {
18.             System.out.println(“El producto no paga impuestos”);
19.             impuesto = precio;
20.         } else {
21.             if (precio ⩾ 20 && precio ⩽ 40) {
22.                 impuesto = precio * 1.30;
23.             } else {
24.                 if (precio > 40 && precio < 500) {
25.                     impuesto = precio * 1.40;
26.                 } else {
27.                     impuesto = precio * 1.50;
28.                 }
29.             }
30.         }
31.         System.out.println(“El precio del producto es: “ +
    impuesto);
32.     }
33.  }
```

## Resultado Java

## Sin impuesto

```
1.  run:
2.  Ingresa el precio del producto:
```

```
3.  5
4.  El producto no paga impuestos
5.  El precio del producto es: 5.0
6.  BUILD SUCCESSFUL (total time: 0 seconds)
```

### Impuesto del 30%

```
1.  run:
2.  Ingresa el precio del producto:
3.  30
4.  El precio del producto es: 39.0
5.  BUILD SUCCESSFUL (total time: 0 seconds)
```

### Impuesto del 40%

```
1.  run:
2.  Ingresa el precio del producto:
3.  400
4.  El precio del producto es: 560.0
5.  BUILD SUCCESSFUL (total time: 0 seconds)
```

### Impuesto del 50%

```
1.  run:
2.  Ingresa el precio del producto:
3.  600
4.  El precio del producto es: 900.0
5.  BUILD SUCCESSFUL (total time: 0 seconds)
```

## Ejercicio 5

Construya un algoritmo que, dados la matrícula de un alumno, la carrera en la que está inscrito, su semestre y su promedio; determine si el mismo es apto para pertenecer a algunas de las facultades menores que tiene la universidad.

```
ALUMNO_ACEPTADO
matricula, carrera: Cadena de Caracteres
cambio, promedio, semestre: Real
Escribir “¿En qué carrera está inscrito?”
Leer carrera
Escribir “Elige alguna de las carreras”
Escribir “Economía (1)”
Escribir “Computación (2)”
Escribir “Administración (3)”
Escribir “Contabilidad (4)”
Escribir “¿A qué carrera te quieres cambiar?”
Leer cambio
Escribir “Ingresa la matrícula del alumno:”
Leer matricula
Escribir “¿En qué semestre está inscrito?:”
Leer Semestre
Escribir “¿Cuál es su promedio?”
Leer promedio
Si cambio = 1 entonces
  Si semestre ⩾ 6 y promedio ⩾ 8.8 entonces
    Escribir “APROBADO”
  Sino
    Escribir “NO APROBADO”
  Fin_si
Fin_si
Si cambio = 2 entonces
  Si semestre > 6 y promedio > 8.5 entonces
    Escribir “APROBADO”
  Sino
    Escribir “NO APROBADO”
```

```
  Fin_si
Fin_si
Si cambio = 3 entonces
  Si semestre > 5 y promedio > 8.5 entonces
    Escribir “APROBADO”
  Sino
    Escribir “NO APROBADO”
  Fin_si
Fin_si
Si cambio = 4 entonces
  Si semestre > 5 y PROM > 8.5 entonces
    Escribir “APROBADO”
  Sino
    Escribir “NO APROBADO”
  Fin_si
Fin_si
Fin
```

## PRUEBA DE ESCRITORIO

| ALUMNO_ACEPTADO | | | | | |
|---|---|---|---|---|---|
| **carrera** | **cambio** | **matricula** | **semestre** | **promedio** | **salida** |
| Computación | 1 | S12045255 | 6 | 9 | “APROBADO” |
| Contabilidad | 2 | S12034525 | 6 | 8 | “NO APROBADO” |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
// Dada la matrícula de un alumno, la carrera en la que está inscrito,
// su semestre y su promedio; determine si el mismo es apto para
// pertenecer a algunas de las facultades menores que tiene la
// universidad.
Algoritmo ALUMNO_ACEPTADO
    DEFINIR cambio, semestre, promedio Como REAL;
```

```
7.      DEFINIR carrera, matricula Como CARACTER;
8.      ESCRIBIR “¿En qué carrera está inscrito?”;
9.      LEER carrera;
10.     ESCRIBIR “Elige alguna de las carreras:”;
11.     ESCRIBIR “ Economía (1)”;
12.     ESCRIBIR “ Computación (2)”;
13.     ESCRIBIR “ Administración (3)”;
14.     ESCRIBIR “ Contabilidad (4)”;
15.     ESCRIBIR “¿A qué carrera te quieres cambiar?”;
16.     LEER cambio;
17.     ESCRIBIR “Ingrese la matricula del alumno”;
18.     LEER matricula;
19.     ESCRIBIR “¿En qué semestre está inscrito?”;
20.     LEER semestre;
21.     ESCRIBIR “¿Cuál es su promedio?”;
22.     LEER promedio;
23.     SI (cambio = 1) ENTONCES
24.     .   SI (semestre ⩾ 6) Y (promedio ⩾ 8.8) ENTONCES
25.     .   .  ESCRIBIR “APROBADO”;
26.     .   SINO
27.     .   .  ESCRIBIR “NO APROBADO”;
28. .   FINSI
29.     FINSI
30.     SI (cambio = 2) ENTONCES
31.     .   SI (semestre ⩾ 6) Y (promedio ⩾ 8.5) ENTONCES
32.     .   .  ESCRIBIR “APROBADO”;
33.     .   SINO
34.     .   .  ESCRIBIR “NO APROBADO”;
35. .   FINSI
36.     FINSI
37.     SI (cambio = 3) ENTONCES
38.     .   SI (semestre ⩾ 5) Y (promedio ⩾ 8.5) ENTONCES
39.     .   .  ESCRIBIR “APROBADO”;
40.     .   SINO
41.     .   .  ESCRIBIR “NO APROBADO”;
42. .   FINSI
43.     FINSI
44.     SI (cambio = 4) ENTONCES
45.     .   SI (semestre ⩾ 5) Y (promedio ⩾ 8.5) ENTONCES
```

```
46.      .   .  ESCRIBIR “APROBADO”;
47.      .   SINO
48.      .   .  ESCRIBIR “NO APROBADO”;
49.  .   FINSI
50.      FINSI
51.  FinAlgoritmo
```

## Ejecutar Paso a Paso

### Cambio Aprobado

Ingreso de la carrera en la que está cursando

```
1.  *** Ejecución Iniciada. ***
2.  ¿En qué carrera está inscrito?
3.  > Ingeniería de Software
```

Ingreso de elección de carrera a cambiarse

```
1.  Elige alguna de las carreras:
2.   Economía  (1)
3.   Computación  (2)
4.   Administración  (3)
5.   Contabilidad  (4)
6.  ¿A qué carrera te quieres cambiar?
7.  > 1
```

Ingreso de los datos del estudiante

```
1.  Ingrese la matrícula del alumno
2.  > Ingeniería de Software
3.  ¿En qué semestre está inscrito?
4.  > 7
5.  ¿Cuál es su promedio?
6.  > 9
```

Impresión en pantalla de la respuesta.

```
APROBADO
*** Ejecución Finalizada. ***
```

### Cambio Aprobado

Ingreso de la carrera en la que está cursando

```
*** Ejecución Iniciada. ***
¿En qué carrera está inscrito?
> Ingeniería de Software
```

Ingreso de elección de carrera a cambiarse

```
Elige alguna de las carreras:
 Economía  (1)
 Computación  (2)
 Administración  (3)
 Contabilidad  (4)
¿A qué carrera te quieres cambiar?
> 1
```

Ingreso de los datos del estudiante

```
Ingrese la matrícula del alumno
> Ingeniería de Software
¿En qué semestre está inscrito?
> 4
¿Cuál es su promedio?
> 9
```

Impresión en pantalla de la respuesta.

```
NO APROBADO
*** Ejecución Finalizada. ***
```

## Ejecutar

### Cambio Aprobado

```
*** Ejecución Iniciada. ***
¿En qué carrera está inscrito?
> Ingeniería de Software
Elige alguna de las carreras:
 Economía  (1)
 Computación  (2)
 Administración  (3)
 Contabilidad  (4)
¿A qué carrera te quieres cambiar?
> 1
Ingrese la matrícula del alumno
> Ingeniería de Software
¿En qué semestre está inscrito?
> 7
¿Cuál es su promedio?
> 9
APROBADO
*** Ejecución Finalizada. ***
```

### Cambio No Aprobado

```
*** Ejecución Iniciada. ***
¿En qué carrera está inscrito?
> Ingeniería de Software
Elige alguna de las carreras:
 Economía  (1)
 Computación  (2)
 Administración  (3)
 Contabilidad  (4)
¿A qué carrera te quieres cambiar?
> 1
Ingrese la matrícula del alumno
> Ingeniería de Software
¿En qué semestre está inscrito?
```

```
> 4
¿Cuál es su promedio?
> 9
NO APROBADO
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/*Construya un algoritmo que dados la matrícula de un alumno, la carrera en la que
  está inscrito, su semestre y su promedio; determine si el mismo es apto para pertenecer
  a algunas de las facultades menores que tiene la universidad.
 */
package EstructuraSelectiva;

import java.util.Scanner;

public class Alumno_Aceptado {

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

        String matricula, carrera;
        double cambio, promedio, semestre;

        System.out.println("¿En qué carrera está inscrito?");
        carrera = entrada.nextLine();

        System.out.println("Elige alguna de las carreras: ");
        System.out.println("Economía (1)");
        System.out.println("Computación (2)");
        System.out.println("Administración(3)");
        System.out.println("Contabilidad (4)");

        System.out.println("¿A qué carrera te quieres cambiar? ");
        cambio = entrada.nextDouble();
```

```
28.         System.out.println(“Ingresa la matricula del alumno:”);
29.         matricula = entrada.next();
30.         System.out.println(“¿En qué semestre está inscrito?”);
31.         semestre = entrada.nextDouble();
32.         System.out.println(“¿Cuál es su promedio?:”);
33.         promedio = entrada.nextDouble();
34.
35.         if (cambio == 1) {
36.             if (semestre ⩾ 6 && promedio ⩾ 8.8) {
37.                 System.out.println(“APROBADO”);
38.             } else {
39.                 System.out.println(“NO APROBADO”);
40.             }
41.         }
42.         if (cambio == 2) {
43.             if (semestre > 6 && promedio > 8.5) {
44.                 System.out.println(“APROBADO”);
45.             } else {
46.                 System.out.println(“NO APROBADO”);
47.             }
48.         }
49.         if (cambio == 3) {
50.             if (semestre > 5 && promedio > 8.5) {
51.                 System.out.println(“APROBADO”);
52.             } else {
53.                 System.out.println(“NO APROBADO”);
54.             }
55.         }
56.         if (cambio == 4) {
57.             if (semestre > 5 && promedio > 8.5) {
58.                 System.out.println(“APROBADO”);
59.             } else {
60.                 System.out.println(“NO APROBADO”);
61.             }
62.         }
63.     }
64. }
```

### Resultado Java

```
1. Cambio Aprobado
2. run:
```

¿En qué carrera está inscrito?

```
1. Ingeniería de Software
2. Elige alguna de las carreras:
3.  Economía (1)
4.  Computación (2)
5.  Administración (3)
6.  Contabilidad (4)
7. ¿A qué carrera te quieres cambiar?
8. 2
9. Ingresa la matricula del alumno:
10. S17016281
```

¿En qué semestre está inscrito?

```
11. 7
```

¿Cuál es su promedio?:

```
1. 9.3
2. APROBADO
3. BUILD SUCCESSFUL (total time: 25 seconds)
```

### Cambio No Aprobado

```
4. run:
```

¿En qué carrera está inscrito?

```
1. Ingeniería de Software
2. Elige alguna de las carreras:
```

```
3.   Economía (1)
4.   Computación (2)
5.   Administración (3)
6.   Contabilidad (4)
```

¿A qué carrera te quieres cambiar?

```
1.  2
2.  Ingresa la matricula del alumno:
3.  S17016281
```

¿En qué semestre está inscrito?

```
4.  4
```

¿Cuál es su promedio?:

```
1.  7.3
2.  NO APROBADO
3.  BUILD SUCCESSFUL (total time: 25 seconds)
```

## Ejercicio 7

Calcule el aumento de sueldo para un grupo de empleados de una empresa, teniendo en cuenta el siguiente criterio:

Si el sueldo es inferior a $1,000.00, aumento del 15%

Si el sueldo es mayor o igual a $1,000.00, aumento del 12 %

Imprima el sueldo nuevo del trabajador y el total de nómina de la empresa. considerando este nuevo aumento.

```
1.  Proceso AUMENTO_SUELDO_GRUPO
2.  trabajadores, nomina, sueldo, sueldoNuevo: entero
3.  Inicio
```

Escribir "¿Cuántos trabajadores tiene el grupo?";

```
4.  Leer trabajadores
```

Escribir "¿Cuál es el sueldo actual?";

```
1.  Leer sueldo
2.  Si sueldo < 1000 entonces
3.    sueldoNuevo ← sueldo * 1.15
4.  sino
5.    Si sueldo ⩾ 1000 entonces
6.      sueldoNuevo ← sueldo * 1.12
7.    Fin_si
8.  Fin_si
9.  Escribir "El nuevo sueldo del grupo de trabajadores será de: "
10. Escribir sueldoNuevo
11. nomina ← (sueldoNuevo * trabajadores)
12. Escribir "El total de nómina es:"
13. Escribir nomina
14. Fin
```

## PRUEBA DE ESCRITORIO

| AUMENTO_SUELDO_GRUPO | | | |
|---|---|---|---|
| **trabajadores** | **sueldo** | **sueldoNuevo** | **nomina** |
| ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ |
| 10 | 1500 | 1680 | 16800 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcule el aumento de sueldo para un grupo de empleados de una
2.  // empresa teniendo en cuenta el siguiente criterio, si sueldo es
3.  // inferior a 1,000 aumento del 15%, si sueldo es mayor o igual
4.  // a 1000 aumento del 12%.
5.  Algoritmo AUMENTO_SUELDO_GRUPO
6.      DEFINIR trabajadores, sueldo, sueldoNuevo, nomina Como REAL;
7.      trabajadores ← 0;
8.      sueldo ← 0;
9.      sueldoNuevo ← 0;
10.     nomina ← 0;
11.     ESCRIBIR "¿Cuántos trabajadores tiene el grupo?";
12.     LEER trabajadores;
13.     ESCRIBIR "¿Cuál es el sueldo actual?";
14.     LEER sueldo;
15.     SI (sueldo < 1000) ENTONCES
16.     .   sueldoNuevo ← sueldo * 1.15;
17.     SINO
18.     .    SI (sueldo ⩾ 1000) ENTONCES
19.     .    .   sueldoNuevo ← sueldo * 1.12;
20. .    FINSI
21.     FINSI
22.     nomina ← sueldoNuevo * trabajadores;
23.     ESCRIBIR "El nuevo sueldo del grupo de trabajadores será de:
    ";
24.     ESCRIBIR sueldoNuevo;
25.     ESCRIBIR " El total de nómina es: ";
26.     ESCRIBIR nomina;
27. FinAlgoritmo
```

## Ejecutar Paso a Paso

### Aumento del 15%

Ingreso de cantidad de trabajadores

```
*** Ejecución Iniciada. ***
¿Cuántos trabajadores tiene el grupo?
> 10
```

Ingreso del saldo actual de los trabajadores

```
¿Cuál es el salario actual?
> 500
```

### Impresión de resultados

```
El nuevo sueldo del grupo de trabajadores será de:
575
El total de nómina es:
5750
*** Ejecución Finalizada. ***
```

### Aumento del 12%

Ingreso de cantidad de trabajadores

```
*** Ejecución Iniciada. ***
¿Cuántos trabajadores tiene el grupo?
> 10
```

Ingreso del saldo actual de los trabajadores

```
¿Cuál es el salario actual?
> 1200
```

Impresión de resultados

```
El nuevo sueldo del grupo de trabajadores será de:
1344
El total de nómina es:
13440
*** Ejecución Finalizada. ***
```

## Ejecutar

### Aumento del 15%

```
*** Ejecución Iniciada. ***
¿Cuántos trabajadores tiene el grupo?
> 10
¿Cuál es el salario actual?
> 500
El nuevo sueldo del grupo de trabajadores será de:
575
El total de nómina es:
5750
*** Ejecución Finalizada. ***
```

### Aumento del 12%

```
*** Ejecución Iniciada. ***
¿Cuántos trabajadores tiene el grupo?
> 10
¿Cuál es el salario actual?
> 1200
El nuevo sueldo del grupo de trabajadores será de:
1344
```

```
El total de nómina es:
13440
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Calcule el aumento de sueldo para un grupo de empleados de una empresa teniendo
 * en cuenta el siguiente criterio, si sueldo es inferior a 1,000 aumento del 15%,
 * si sueldo es mayor o igual a 1000 aumento del 12%.
 */
package EstructuraSelectiva;

import java.util.Scanner;

public class Aumento_Sueldo_Grupo {

   public static void main(String[] args) {
      Scanner entrada = new Scanner(System.in);
      double trabajadores, sueldo, sueldoNuevo, nomina;

      System.out.println("¿Cuántos trabajadores tiene el grupo?");
      trabajadores = entrada.nextDouble();
      System.out.println("¿Cuál es el sueldo actual?");
      sueldo = entrada.nextDouble();
      if (sueldo < 1000) {
         sueldoNuevo = sueldo * 1.15;
         nomina = sueldoNuevo * trabajadores;
         System.out.println("El nuevo sueldo del grupo de trabajadores es: $"
+ sueldoNuevo);
         System.out.println("El total de la nómina es: " + nomina);
      } else {
         if (sueldo ⩾ 1000) {
            sueldoNuevo = sueldo * 1.12;
            nomina = sueldoNuevo * trabajadores;
```

```
            System.out.println("El nuevo sueldo del grupo de
trabajadores es: $"
+ sueldoNuevo);
            System.out.println("El total de la nómina es: $" +
nomina);
          }
       }
    }
}
```

### Resultado Java

### Aumento del 15%

```
run:
```

¿Cuántos trabajadores tiene el grupo?

```
10
```

¿Cuál es el sueldo actual?

```
500
El nuevo sueldo del grupo de trabajadores es: $575.0
El total de la nómina es: $5750.0
BUILD SUCCESSFUL (total time: 12 seconds)
```

#### Aumento del 12%

```
run:
```

¿Cuántos trabajadores tiene el grupo?

```
10
```

¿Cuál es el sueldo actual?

```
1. 1200
2. El nuevo sueldo del grupo de trabajadores es:
   $1344.0000000000002
3. El total de la nómina es: $13440.000000000004
4. BUILD SUCCESSFUL (total time: 12 seconds)
```

# ESTRUCTURA SELECTIVA SEGÚN (SWITCH CASE)

## Ejercicio 1

Escribir un algoritmo dado como datos: **Numero**, **Valor** y **Resultado** obteniendo los resultados según las siguientes funciones:

| Opción | Resultado |
|---|---|
| 1 | 100 * Valor |
| 2 | 100 ^ Valor |
| 3 | 100 / Valor |

```
1.  FUNCION
2.  opcion, valor: Entero
3.  resultado: Real
4.  opcion←0
5.  valor←0
6.  resultado←0
7.  Escribir “Ingrese la opción a ejecutar: (1) Multiplicar el
    número *100, (2) Elevar como potencia el número a 100, (3)
    Dividir el valor /100”;
8.  Leer opcion;
9.  Escribir “Ingrese el valor a trabajar:”;
10. Leer valor;
11. Según opcion igual
12.  1: resultado ← 100 * valor
13.  2: resultado ← 100 ^ valor
14.  3: resultado ← 100 / valor
15. De otra forma: resultado ← 0
16. FinSegún
17. Escribir “El valor de la función es:”
18. Escribir resultado
19. Fin
```

## PRUEBA DE ESCRITORIO

| FUNCION | | |
|---|---|---|
| **opcion** | **valor** | **resultado** |
| 0 | 0 | 0 |
| 1 | 3 | 300 |
| 2 | 6 | 1000000000000 |
| 3 | 5 | 20 |
| 8 | 8 | 0 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Determina la operación a realizar a partir de lo ingresado
    por el
2.  // usuario.
3.  Algoritmo FUNCION
4.      DEFINIR opcion, valor Como ENTERO;
5.      DEFINIR resultado Como REAL;
6.      opcion ← 0;
7.      valor ← 0;
8.      resultado ← 0;
9.      ESCRIBIR “Ingrese la opción a ejecutar: “;
10.     ESCRIBIR “(1) Multiplicar el número * 100,”;
11.     ESCRIBIR “(2) Elevar como potencia el número a 100, “;
12.     ESCRIBIR “(3) Dividir el valor / 100.”;
13.     LEER opcion;
14.     ESCRIBIR “Ingrese el valor a trabajar”;
15.     LEER valor;
16.     SEGUN (opcion) HACER
17.     .   1:
18.     .   .  resultado ← 100 * valor;
19.     .   2:
20.     .   .  resultado ← 100 ^ valor;
21.     .   3:
22.     .   .  resultado ← 100 / valor;
23.     .   DE OTRO MODO:
24.     .   .  resultado ← 0;
```

```
25.     FINSEGUN
26.     ESCRIBIR “El resultado de la función es:”;
27.     ESCRIBIR resultado;
28. FinAlgoritmo
```

## Ejecutar Paso a Paso

Solicitud e Ingreso de los datos

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese la opción a ejecutar:
3.  (1) Multiplicar el número * 100,
4.  (2) Elevar como potencia el número a 100,
5.  (3) Dividir el valor / 100.
6.  > 1
7.  Ingrese el valor a trabajar:
8.  > 3
9.
10. Impresión del valor de la operación
11. El resultado de la función es:
12. 300
13. *** Ejecución Finalizada. ***
```

## Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese la opción a ejecutar:
3.  (1) Multiplicar el número * 100,
4.  (2) Elevar como potencia el número a 100,
5.  (3) Dividir el valor / 100.
6.  > 1
7.  Ingrese el valor a trabajar:
8.  > 3
9.  El resultado de la función es:
10. 300
11. *** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
1.  // Determina la operación a realizar a partir de lo ingresado
    por el
2.  // usuario.
3.  package EstructuraSelectiva;
4.
5.  import java.util.Scanner;
6.
7.  public class Funcion {
8.
9.      public static void main(String[] args) {
10.         Scanner entrada = new Scanner(System.in);
11.         int opcion = 0;
12.         int valor = 0;
13.         double resultado = 0;
14.         System.out.println(“Ingrese la opción a ejecutar:”);
15.         System.out.println(“ (1) Multiplicar el número * 100,”);
16.         System.out.println(“ (2) Elevar como potencia el número
    a 100,”);
17.         System.out.println(“ (3) Dividir el valor / 100.”);
18.         opcion = entrada.nextInt();
19.         System.out.println(“Ingrese el valor a trabajar: “);
20.         valor = entrada.nextInt();
21.
22.         switch (opcion) {
23.             case (1):
24.                 resultado = 100 * valor;
25.                 break;
26.             case (2):
27.                 resultado = Math.pow(100, valor);
28.                 break;
29.             case (3):
30.                 resultado = 100 / valor;
31.                 break;
32.             default:
33.                 resultado = 0;
34.                 break;
35.         }
36.
```

```
37.         System.out.println(“El resultado de la funcion es: “ +
    resultado);
38.     }
39.
40. }
```

## Resultado Java

```
1.  run:
2.  Ingrese la opción a ejecutar:
3.   (1) Multiplicar el número * 100,
4.   (2) Elevar como potencia el número a 100,
5.   (3) Dividir el valor / 100.
6.  1
7.  Ingrese el valor a trabajar:
8.  3
9.  El resultado de la funcion es: 300.0
10. BUILD SUCCESSFUL (total time: 3 seconds)
```

## Ejercicio 2

Construye un algoritmo dado como datos la categoría y el sueldo de un trabajador, calcule el aumento correspondiente, teniendo en cuenta la siguiente tabla.

| Categoría | Porcentaje |
|---|---|
| 1 | 15% |
| 2 | 10% |
| 3 | 8% |
| 4 | 7% |

Imprima la categoría del trabajador y su nuevo sueldo.

```
1.  SUELDO_CATEGORIA
2.  categoria, sueldo, nuevoSueldo: Real
3.  Inicio
4.  categoria←0
5.  sueldo←0
6.  nuevoSueldo←0
7.  Escribir “Ingrese la categoría del trabajador:”
8.  Leer categoria
9.  Escribir “Ingrese el sueldo del trabajador”
10. Leer sueldo
11. Según categoria igual
12.  1: nuevoSueldo ← sueldo * 1.15
13.  2: nuevoSueldo ← sueldo * 1.10
14.  3: nuevoSueldo ← sueldo * 1.08
15.  4: nuevoSueldo ← sueldo * 1.07
16. Fin Según
17. Escribir “La categoría del trabajador es:”
18. Escribir categoria
19. Escribir “El nuevo sueldo es:”
20. Escribir nuevoSueldo
```

## PRUEBA DE ESCRITORIO

| SUELDO_CATEGORIA | | |
|---|---|---|
| categoria | sueldo | nuevoSueldo |
| 1 | 3000 | 3450 |
| 2 | 5000 | 5500 |
| 3 | 1800 | 1944 |
| 4 | 4000 | 4280 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula el aumento de un trabajador de acuerdo con su
    categoría.
2.  Algoritmo SUELDO_CATEGORIA
3.      DEFINIR categoria Como ENTERO;
4.      DEFINIR sueldo, nuevoSueldo Como REAL;
5.      categoria ← 0;
6.      sueldo ← 0;
7.      nuevoSueldo ← 0;
8.      ESCRIBIR “Ingrese la categoría del trabajador: “;
9.      LEER categoria;
10.     ESCRIBIR “Ingrese el sueldo del trabajador:”;
11.     LEER sueldo;
12.     SEGUN (categoria) HACER
13.     .   1:
14.     .   .  nuevoSueldo ← sueldo * 1.15;
15.     .   2:
16.     .   .  nuevoSueldo ← sueldo * 1.10;
17.     .   3:
18.     .   .  nuevoSueldo ← sueldo * 1.08;
19.     .   4:
20.     .   .  nuevoSueldo ← sueldo * 1.07;
21.     .   DE OTRO MODO:
22.     .   .  nuevoSueldo ← sueldo;
23.     FINSEGUN
24.     ESCRIBIR “La categoría del trabajador es:”;
25.     ESCRIBIR categoria;
26.     ESCRIBIR “El nuevo sueldo del trabajador es:”;
27.     ESCRIBIR nuevoSueldo;
28. FinAlgoritmo
```

## Ejecutar Paso a Paso

Ingreso de la categoría del trabajador y sueldo

```
*** Ejecución Iniciada. ***
Ingrese la categoría del trabajador:
> 1
Ingrese el sueldo del trabajador:
> 5000
```

Impresión de los datos del trabajador y nuevo sueldo

```
La categoría del trabajador es:
1
El nuevo sueldo del trabajador es:
5750
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese la categoría del trabajador:
> 1
Ingrese el sueldo del trabajador:
> 5000
La categoría del trabajador es:
1
El nuevo sueldo del trabajador es:
5750
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
// Calcula el aumento de un trabajador de acuerdo con su categoría.
package EstructuraSelectiva;

import java.util.Scanner;
```

```
5.  public class Sueldo_Categoria {
6.
7.      public static void main(String[] args) {
8.          Scanner entrada = new Scanner(System.in);
9.          int categoria;
10.         double sueldo, nuevoSueldo;
11.
12.         System.out.println(“Ingrese la categoría del trabajador:
    “);
13.         categoria = entrada.nextInt();
14.         System.out.println(“Ingrese el sueldo del trabajador:
    “);
15.         sueldo = entrada.nextDouble();
16.         switch (categoria) {
17.             case 1:
18.                 nuevoSueldo = sueldo * 1.15;
19.                 break;
20.             case 2:
21.                 nuevoSueldo = sueldo * 1.10;
22.                 break;
23.             case 3:
24.                 nuevoSueldo = sueldo * 1.08;
25.                 break;
26.             case 4:
27.                 nuevoSueldo = sueldo * 1.07;
28.                 break;
29.             default:
30.                 nuevoSueldo = sueldo;
31.                 break;
32.         }
33.         System.out.println(“La categoría del trabajador es: “ +
    categoria);
34.         System.out.println(“El nuevo Sueldo del trabajador es: “
    +
35. nuevoSueldo);
36.
37.     }
38. }
```

## Resultado Java

```
run:
Ingrese la categoría del trabajador:
1
Ingrese el sueldo del trabajador:
5000
La categoría del trabajador es: 1
El nuevo Sueldo del trabajador es: 5750.0
BUILD SUCCESSFUL (total time: 12 seconds)
```

## Ejercicio 3

El costo de las llamadas telefónicas internacionales depende de la zona geográfica en la que se encuentra el país de destino y el número de minutos hablados, en la siguiente tabla se presenta el costo de un minuto por zona; a cada zona se le asignado una clave. Construye un algoritmo que permita calcular e imprimir el costo total de una llamada.

| Clave | Zona | Precio |
|---|---|---|
| 12 | América Norte | 2 |
| 15 | América Central | 2.2 |
| 18 | América Sur | 4.5 |
| 19 | Europa | 3.5 |
| 23 | Asia | 6 |
| 25 | África | 6 |
| 29 | Oceanía | 5 |

```
1.  COSTO_LLAMADA
2.  costo, zona, minutos: Real
3.  Inicio
4.  minutos ← 0
5.  costo ← 0
6.  zona ← 0
7.    ESCRIBIR “Claves de las zonas geográficas:”;
8.    ESCRIBIR “(12) América Norte “;
9.    ESCRIBIR “(15) América Central “;
10. ESCRIBIR “(18) América Sur”;
11. ESCRIBIR “(19) Europa”;
12. ESCRIBIR “(23) Asia”;
13. ESCRIBIR “(25) África”;
14. ESCRIBIR “(29) Oceanía”;
15. Escribir “Ingresa la clave de la zona”
16. Leer zona
17. Escribir” Ingresa los minutos”
18. Leer minutos
19. Según zona igual
20.   12: costo ← minutos * 2
```

```
    15: costo ← minutos * 2.2
    18: costo ← minutos * 4.5
    19: costo ← minutos * 3.5
    23: costo ← minutos * 6
    25: costo ← minutos * 6
    29: costo ← minutos * 5
Fin según
Escribir “El costo total de la llamada fue de:”
Escribir costo
Fin
```

## PRUEBA DE ESCRITORIO

| COSTO_LLAMADA | | |
|---|---|---|
| **zona** | **minutos** | **costo** |
| 0 | 0 | 0 |
| 18 | 20 | 90 |
| 29 | 30 | 150 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
// Determina si un número ingresado es negativo, nulo o
positivo.
Algoritmo COSTO_LLAMADA
    DEFINIR costo, zona, minutos Como REAL;
    costo ← 0;
    zona ← 0;
    minutos ← 0;
    ESCRIBIR “Claves de las zonas geográficas:”;
    ESCRIBIR “(12) América Norte “;
    ESCRIBIR “(15) América Central “;
    ESCRIBIR “(18) América Sur”;
    ESCRIBIR “(19) Europa”;
    ESCRIBIR “(23) Asia”;
    ESCRIBIR “(25) África”;
    ESCRIBIR “(29) Oceanía”;
```

```
15.     ESCRIBIR “Ingrese la clave de la zona: “;
16.     LEER zona;
17.     ESCRIBIR “Ingrese los minutos: “;
18.     LEER minutos;
19.     SEGUN (zona) HACER
20.     .   12:
21.     .   .  costo ← minutos * 2;
22.     .   15:
23.     .   .  costo ← minutos * 2.2;
24.     .   18:
25.     .   .  costo ← minutos * 4.5;
26.     .   19:
27.     .   .  costo ← minutos * 3.5;
28.     .   23:
29.     .   .  costo ← minutos * 6;
30.     .   25:
31.     .   .  costo ← minutos * 6;
32.     .   29:
33.     .   .  costo ← minutos * 5;
34.     .   DE OTRO MODO:
35.     .   .  ESCRIBIR “No es una clave válida”;
36.     .   .  costo ← minutos;
37.     FINSEGUN
38.     ESCRIBIR “El costo total de la llamada es de: “;
39.     ESCRIBIR costo;
40.     FINSI
41. FinAlgoritmo
```

## Ejecutar Paso a Paso

Ejecución del programa (despliegue del menú)

```
1.  *** Ejecución Iniciada. ***
2.  Claves de las zonas geográficas:
3.     (12) América Norte
4.  (15) América Central
5.  (18) América Sur
```

```
6.  (19) Europa
7.  (23) Asia
8.  (25) África
9.  (29) Oceanía
```

Ingreso de la clave de la zona geográfica

```
1.  Ingrese la clave de la zona:
2.  > 25
```

Ingreso de los minutos hablados

```
1.  Ingrese los minutos:
2.  > 25
```

Impresión de la cantidad total por pagar

```
1.  El costo total de la llamada es de:
2.  150
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  Claves de las zonas geográficas:
3.     (12) América Norte
4.  (15) América Central
5.  (18) América Sur
6.  (19) Europa
7.  (23) Asia
8.  (25) África
9.  (29) Oceanía
10. Ingrese la clave de la zona:
11. > 25
```

```
Ingrese los minutos:
> 25
El costo total de la llamada es de:
150
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Calcula e imprime el costo de una llamada de acuerdo con su zona
// geográfica
package EstructuraSelectiva;

import java.util.Scanner;

public class Costo_Llamada {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        double costo = 0;
        double minutos = 0;
        int zona;

        System.out.println("Claves de las zonas geográficas:");
        System.out.println(" (12)  América Norte");
        System.out.println(" (15)  América Central");
        System.out.println(" (18)  América Sur");
        System.out.println(" (19)  Europa");
        System.out.println(" (23)  Asia");
        System.out.println(" (25)  África");
        System.out.println(" (29)  Oceanía");

        System.out.println("Ingrese la clave de la zona: ");
        zona = entrada.nextInt();
        System.out.println("Ingrese los minutos: ");
        minutos = entrada.nextDouble();
```

```
28.          switch (zona) {
29.              case (12):
30.                  costo = minutos * 2;
31.                  break;
32.              case (15):
33.                  costo = minutos * 2.2;
34.                  break;
35.              case (18):
36.                  costo = minutos * 4.5;
37.                  break;
38.              case (19):
39.                  costo = minutos * 3.5;
40.                  break;
41.              case (23):
42.                  costo = minutos * 6;
43.                  break;
44.              case (25):
45.                  costo = minutos * 6;
46.                  break;
47.              case (29):
48.                  costo = minutos * 5;
49.                  break;
50.              default:
51.                  System.out.println(“No es una clave válida”);
52.                  costo = minutos;
53.          }
54.          System.out.println(“El costo total de la llamada fue de: 
     “ + costo);
55.      }
56. }
```

## Resultado Java

```
1.  run:
2.  Claves de las zonas geográficas:
3.   (12)  América Norte
4.   (15)  América Central
5.   (18)  América Sur
```

```
6.   (19)  Europa
7.   (23)  Asia
8.   (25)  África
9.   (29)  Oceanía
10. Ingrese la clave de la zona:
11. 25
12. Ingrese los minutos:
13. 25
14. El costo total de la llamada fue de: 150.0
15. BUILD SUCCESSFUL (total time: 5 seconds)
```

## Ejercicio 4

Escriba el algoritmo que permita calcular lo que hay que pagarle a un trabajador teniendo en cuenta su sueldo y las horas extras trabajadas. Para el pago de horas extras se toma en cuenta la categoría del trabajador.

| Categoría | Precio Hora Extra |
|---|---|
| 1 | 30 |
| 2 | 38 |
| 3 | 50 |
| 4 | 70 |

```
1.  SUELDO_HORAS_EXTRAS
2.  sueldo, horasExtra, nuevoSueldo: Real
3.  categoria: Entero
4.  Inicio
5.  sueldo←0
6.  horasExtra←0
7.  nuevoSueldo←0
8.  categoria←0
9.  Escribir “Ingresa la categoría del trabajador:”
10. Leer categoria
11. Escribir “Ingresa el sueldo del trabajador:”
12. Leer sueldo
13. Escribir “Ingresa las horas extras del trabajador:”
14. Leer horasExtra
15. Según categoria igual
16.   1: nuevoSueldo ← sueldo + (horasExtra * 30)
17.   2: nuevoSueldo ← sueldo + (horasExtra * 38)
18.   3: nuevoSueldo ← sueldo + (horasExtra * 50)
19.   4: nuevoSueldo ← sueldo + (horasExtra * 70)
20. Fin según
21. Escribir “El sueldo del trabajador con las horas extras es:”
22. Escribir nuevoSueldo
23. Fin
```

## PRUEBA DE ESCRITORIO

| SUELDO_HORAS_EXTRA | | | |
|---|---|---|---|
| **categoria** | **sueldo** | **horasExtra** | **nuevoSueldo** |
| 1 | 2000 | 3 | 2090 |
| 2 | 3000 | 5 | 3190 |
| 3 | 1520 | 2 | 1620 |
| 4 | 6000 | 1 | 6070 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula el salario de las horas extras de un trabajador a
    partir
2.  // de su categoría.
3.  Algoritmo SUELDO_HORAS_EXTRA
4.      DEFINIR sueldo, horasExtra, nuevoSueldo Como REAL;
5.      DEFINIR categoria Como ENTERO;
6.      categoria ← 0;
7.      sueldo ← 0;
8.      horasExtra ← 0;
9.      nuevoSueldo ← 0;
10.     ESCRIBIR “Categorías”;
11.     ESCRIBIR “(1) (2) (3) (4)”;
12.     ESCRIBIR “Ingrese la categoría del trabajador: “;
13.     LEER categoria;
14.     ESCRIBIR “Ingrese el sueldo del trabajador”;
15.     LEER sueldo;
16.     ESCRIBIR “Ingrese las horas extras del trabajador”;
17.     LEER sueldo;
18.     SEGUN (categoria) HACER
19.     .   1:
20.     .   .  nuevoSueldo ← sueldo + (horasExtra * 30);
21.     .   2:
22.     .   .  nuevoSueldo ← sueldo + (horasExtra * 38);
23.     .   3:
24.     .   .  nuevoSueldo ← sueldo + (horasExtra * 50);
25.     .   4:
26.     .   .  nuevoSueldo ← sueldo + (horasExtra * 70);
```

```
27.     .   DE OTRO MODO:
28.     .   .  ESCRIBIR “Categoría no válida”;
29.     .   .  nuevoSueldo ← sueldo;
30.     FINSEGUN
31.     ESCRIBIR “El sueldo del trabajador con las horas extras
    es:”;
32.     ESCRIBIR nuevoSueldo;
33. FinAlgoritmo
```

## Ejecutar Paso a Paso

Solicitud e ingreso de la categoría del trabajador

```
1.  *** Ejecución Iniciada. ***
2.  Categorías
3.  (1) (2) (3) (4)
4.  Ingrese la categoría del trabajador:
5.  > 3
```

Ingreso del sueldo del trabajador

```
1.  Ingrese el sueldo del trabajador:
2.  > 5000
```

Ingreso de las horas extras trabajadas

```
1.  Ingrese las horas extras del trabajador:
2.  > 5
```

Impresión del nuevo sueldo con horas extras

```
1.  El sueldo del trabajador con las horas extras es:
2.  5250
3.  *** Ejecución Finalizada. ***
```

### Ejecutar

```
*** Ejecución Iniciada. ***
Categorías
(1) (2) (3) (4)
Ingrese la categoría del trabajador:
> 3
Ingrese el sueldo del trabajador:
> 5000
Ingrese las horas extras del trabajador:
> 5
El sueldo del trabajador con las horas extras es:
5250
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Calcula el aumento de un trabajador de acuerdo con su categoría.
package EstructuraSelectiva;

import java.util.Scanner;

public class Sueldo_Horas_Extra {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int categoria;
        double sueldo, nuevoSueldo, horasExtra;
        System.out.println("Categorías");
        System.out.println("(1) (2) (3) (4)");
        System.out.println("Ingrese la categoría del trabajador: ");
        categoria = entrada.nextInt();
        System.out.println("Ingrese el sueldo del trabajador: ");
        sueldo = entrada.nextDouble();
        System.out.println("Ingrese las horas extras del trabajador: ");
        horasExtra = entrada.nextDouble();
```

```
20.         switch (categoria) {
21.             case 1:
22.                 nuevoSueldo = sueldo + (horasExtra * 30);
23.                 break;
24.             case 2:
25.                 nuevoSueldo = sueldo + (horasExtra * 38);
26.                 break;
27.             case 3:
28.                 nuevoSueldo = sueldo + (horasExtra * 50);
29.                 break;
30.             case 4:
31.                 nuevoSueldo = sueldo + (horasExtra * 70);
32.                 break;
33.             default:
34.                 System.out.println(“Categoría no válida”);
35.                 nuevoSueldo = sueldo;
36.                 break;
37.         }
38.         System.out.println(“El nuevo Sueldo del trabajador es: “
39. + nuevoSueldo);
40.
41.     }
42. }
```

## Resultado Java

```
1.  run:
2.  Categorías
3.  (1) (2) (3) (4)
4.  Ingrese la categoría del trabajador:
5.  3
6.  Ingrese el sueldo del trabajador:
7.  5000
8.  Ingrese las horas extras del trabajador:
9.  5
10. El nuevo Sueldo del trabajador es: 5250.0
11. BUILD SUCCESSFUL (total time: 6 seconds)
```

## Ejercicio 5

La siguiente tabla expresa los costos diarios según el tipo de enfermedad, construye un algoritmo que calcule e imprima el costo total que representa un paciente, considerando que, si el paciente tiene de 14 a 22 años, se le agrega al costo total a pagar un 10% sobre el costo.

| Enfermedad | Costo paciente día |
|---|---|
| 1 | 25 |
| 2 | 16 |
| 3 | 20 |
| 4 | 30 |

```
1.  COSTO_ENFERMEDAD
2.  nombre: Cadena de caracteres
3.  costo, dias, costoTotal : Real
4.  edad, enfermedad: Entero
5.  Escribir “Las enfermedades pueden ser 1, 2, 3 y 4”
6.  Escribir “Ingrese la enfermedad:”
7.  Leer efermedad
8.  Escribir “Ingrese el nombre del paciente”
9.  Leer nombre
10. Escribir “Ingrese la edad del paciente:”
11. Leer edad
12. Escribir “Ingrese los días internados del paciente:”
13. Leer dias
14. Según enfermedad igual
15. 1 costo ← dias * 25
16. 2: costo ← dias * 16
17. 3: costo ← dias * 20
18. 4: costo ← dias * 30
19. Fin según
20. Si edad ⩾ 14 y edad ⩽ 22 Entonces
21.    costoTotal ← costo * 1.10
22.    Escribir “El costo total por los días en el hospital es:”
23.    Escribir costoTotal
24. Sino
```

```
    Escribir “El costo total por los días en el hospital es:”
    Escribir costo
Fin si
Fin
```

## PRUEBA DE ESCRITORIO

| COSTO_ENFERMEDAD | | | | | |
|---|---|---|---|---|---|
| **nombre** | **enfermedad** | **edad** | **dias** | **costo** | **costoTotal** |
| **Luis** | **3** | **13** | **3** | **60** | **0** |
| **Sofía** | **2** | **15** | **4** | **64** | **70.4** |
| **Paco** | **1** | **23** | **7** | **175** | **0** |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
// Determina el costo a pagar por los días ingresados en un hospital
Algoritmo COSTO_ENFERMEDAD
    DEFINIR nombre Como CARACTER;
    DEFINIR costo, dias, costoTotal Como REAL;
    DEFINIR edad, enfermedad Como ENTERO;
    ESCRIBIR “Las enfermedades pueden ser 1, 2, 3 y 4”;
    ESCRIBIR “Ingrese la enfermedad:”;
    LEER enfermedad;
    ESCRIBIR “Ingrese el nombre del paciente:”;
    LEER nombre;
    ESCRIBIR “Ingrese la edad del paciente:”;
    LEER edad;
    ESCRIBIR “Ingrese los días internados del paciente:”;
    LEER dias;
    SEGUN (enfermedad) HACER
    .   1:
    .   .  costo ← dias * 25;
    .   2:
    .   .  costo ← dias * 16;
```

```
20.     .   3:
21.     .   .  costo ← dias * 20;
22.     .   4:
23.     .   .  costo ← dias * 30;
24.     .   DE OTRO MODO:
25.     .   .  ESCRIBIR "Opción no válida";
26.     .   .  costo ← dias;
27.     FINSEGUN
28.     SI (edad ⩾ 14) Y (edad ⩽ 22) ENTONCES
29.     .   costoTotal ← costo * 1.10;
30.     .   ESCRIBIR "El costo total por los días en el hospital
    es:";
31.     .   ESCRIBIR costoTotal;
32.     SINO
33.     .   ESCRIBIR "El costo total por los días en el hospital
    es:";
34.     .   ESCRIBIR costo;
35.     FINSI
36. FinAlgoritmo
```

## Ejecutar Paso a Paso

Despliegue del menú de las enfermedades, a la vez, la elección de la enfermedad

```
1.  *** Ejecución Iniciada. ***
2.  Las enfermedades pueden ser 1, 2, 3, 4
3.  Ingrese la enfermedad:
4.  > 2
```

Ingresar el nombre del paciente

```
1.  Ingrese el nombre del paciente:
2.  > Pedro
```

Solicitud e ingreso de la edad del paciente

```
Ingrese la edad del paciente:
> 15
```

Solicitud e ingreso de los días internado

```
Ingrese los días internados del paciente:
> 3
```

Impresión del costo total por los días en el hospital

```
El costo total por los días en el hospital es:
52.8
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Las enfermedades pueden ser 1, 2, 3, 4
Ingrese la enfermedad:
> 2
Ingrese el nombre del paciente:
> Pedro
Ingrese la edad del paciente:
> 15
Ingrese los días internados del paciente:
> 3
El costo total por los días en el hospital es:
52.8
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
1.  /*Construye un algoritmo que calcule e imprima el costo total
    que representa un paciente.
2.   */
3.  package EstructuraSelectiva;
4.
5.  import java.util.Scanner;
6.
7.  public class Costo_Enfermedad {
8.
9.      public static void main(String[] args) {
10.         Scanner entrada = new Scanner(System.in);
11.         String nombre;
12.         double costo = 0;
13.         double costoTotal;
14.         int enfermedad, edad, dias;
15.
16.         System.out.println("Las enfermedades son del 1,2,3 y
    4");
17.         System.out.println("Ingrese la enfermedad:");
18.         enfermedad = entrada.nextInt();
19.         System.out.println("Ingrese el nombre del paciente: ");
20.         nombre = entrada.next();
21.         System.out.println("Ingrese la edad del paciente: ");
22.         edad = entrada.nextInt();
23.         System.out.println("Ingrese los días internados del
    paciente: ");
24.         dias = entrada.nextInt();
25.
26.         switch (enfermedad) {
27.             case (1):
28.                 costo = dias * 25;
29.                 break;
30.             case (2):
31.                 costo = dias * 16;
32.                 break;
33.             case (3):
34.                 costo = dias * 20;
35.                 break;
```

```
36.             case (4):
37.                 costo = dias * 30;
38.                 break;
39.             default:
40.                 System.out.println(“Opción no válida”);
41.                 costo = dias;
42.         }
43.         if (edad ⩾ 14 && edad ⩽ 22) {
44.             costoTotal = costo * 1.10;
45.             System.out.println(“El costo total por los días en
    el hospital es: “ +
46. costoTotal);
47.         } else {
48.             System.out.println(“El costo total por los dias en
    el hospital es: “ +
49. costo);
50.         }
51.     }
52.
53. }
```

## Resultado Java

```
1.  run:
2.  Las enfermedades son del 1,2,3 y 4
3.  Ingrese la enfermedad:
4.  2
5.  Ingrese el nombre del paciente:
6.  Pedro
7.  Ingrese la edad del paciente:
8.  15
9.  Ingrese los días internados del paciente:
10. 3
11. El costo total por los días en el hospital es:
    52.800000000000004
12. BUILD SUCCESSFUL (total time: 16 seconds)
```

# 3.

# Estructuras repetitivas y arreglos unidimensionales

## Introducción

El objetivo de este capítulo consiste en introducir al lector a un nivel más en el desarrollo de algoritmos con las estructuras de repetitivas, y de manera introductoria, a los arreglos unidimensionales a través de una serie de ejercicios prácticos. Al estudiar este capítulo:

- Conocerá, a través de diversos ejercicios prácticos, los pasos para resolver un algoritmo utilizando las estructuras repetitivas: PARA (FOR) y MIENTRAS (WHILE).
- Comprenderá la estructura de los arreglos unidimensionales.
- Entenderá algunos conflictos básicos al resolver un algoritmo utilizando PARA y MIENTRAS.
- Conocerá cómo realizar una prueba de escritorio, símbolos, nombres,  que se usan como ejemplos a lo largo del libro.
- Entenderá cómo manejar y ejecutar PseInt con algoritmos con estructuras repetitivas y arreglos unidimensionales.
- Identificará el mismo algoritmo resuelto en español estructurado y pseudocódigo, pero ahora con Java, siempre utilizando las estructuras repetitivas y arreglos unidimensionales.

# CAPÍTULO 3.
# Estructuras repetitivas y arreglos unidimensionales

## Estructuras repetitivas

Como su nombre lo indica, son estructuras algorítmicas las cuales se repiten una y otra vez, de manera cíclica. Las que existen son: MIENTRAS (While), REPETIR (Do While) Y PARA (For). Para efectos de este libro, se estudian las estructuras MIENTRAS Y PARA.

## MIENTRAS (While)

Al utilizar MIENTRAS (figura 5), de igual manera como las estructuras de selectivas, es necesario validar que la condición o instrucción se cumpla como verdadera, y entonces ejecutar una y otra vez de manera cíclica dicha instrucción hasta detenerse a partir de una indicación o instrucción. Por lo que, existen dos puntos importantes a destacar en esta estructura:

1. Identificar la condición o instrucción que hará ejecutar N veces la misma condición de entrada al ciclo (MIENTRAS).
2. Plasmar la operación o instrucción que será la VARIABLE que incremente y ejecute el ciclo las veces que se indique en el algoritmo y será medular

para llegar a detener la condición del punto 1 antes mencionado, de lo contrario el algoritmo se ciclará y marcará error.

Así bien, se observa que la variable CONTADOR debe ser menor que 50, toda vez dicha variable tiene valor de 0, lo que significa que entrará al ciclo MIENTRAS. Y por otro lado, se observa que la variable CONTADOR se incrementa en uno cuando entra al ciclo MIENTRAS.

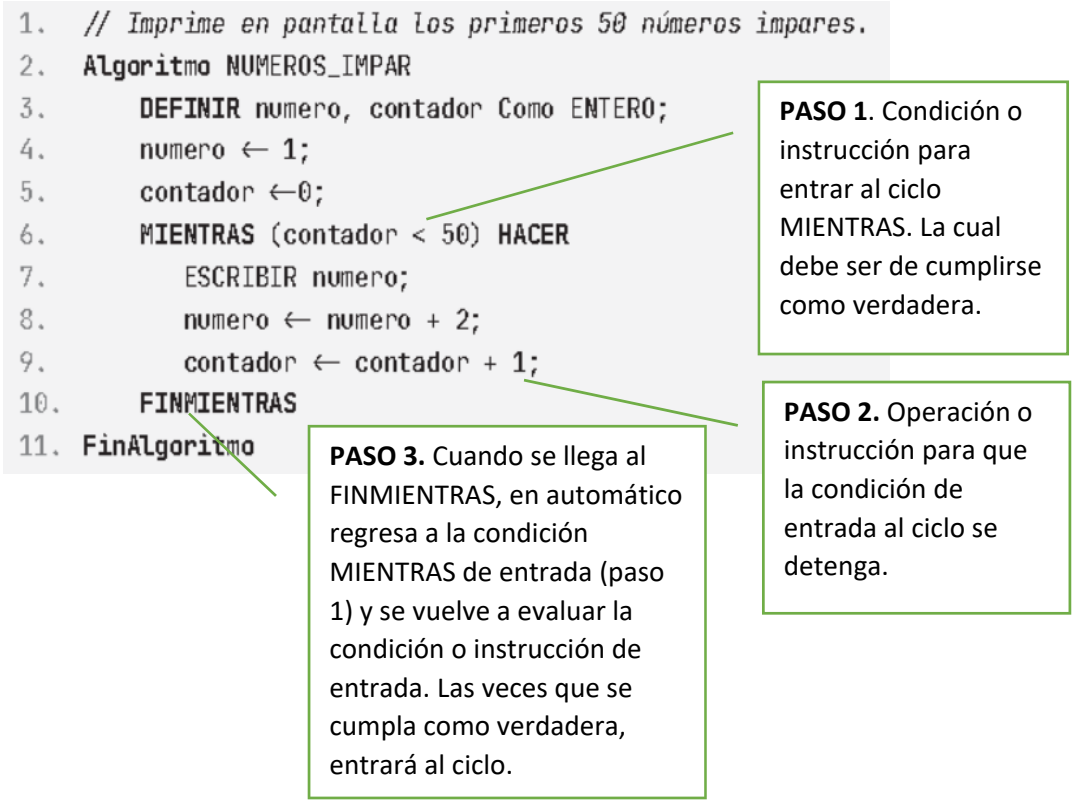



**Figura 5. Uso del MIENTRAS**

### PARA (For)

En la figura 6, se requiere generar los números del 1 hasta llegar al 100, incrementando de uno en uno en el ciclo PARA, de tal manera que la instrucción CON PASO 1 HACER, indica incremento de 1. En caso de que fuera incremento de 2 o 3, sería CON PASO 2 HACER o en su defecto CON PASO 3 HACER. Cabe señalar, lo importante de denotar en esta estructura: (Figura 6).

Identificar el valor (VARIABLE) de inicio que permitirá ejecutar N veces la misma condición de entrada al ciclo (PARA).

Identificar el valor (VARIABLE) de fin del ciclo para llegar a detener la VARIABLE del punto 1 antes mencionado, de lo contrario el algoritmo se ciclará y marcará error.

El incremento o VARIABLE del incremento que ejecuta el ciclo hasta las N veces que indica el valor final (del punto 2 antes mencionado). Está instrucción o incremento se representa CON PASO 1 HACER o en su defecto en español estructurado HACER VARIABLE + 1 (valor del incremento, según sea el caso).

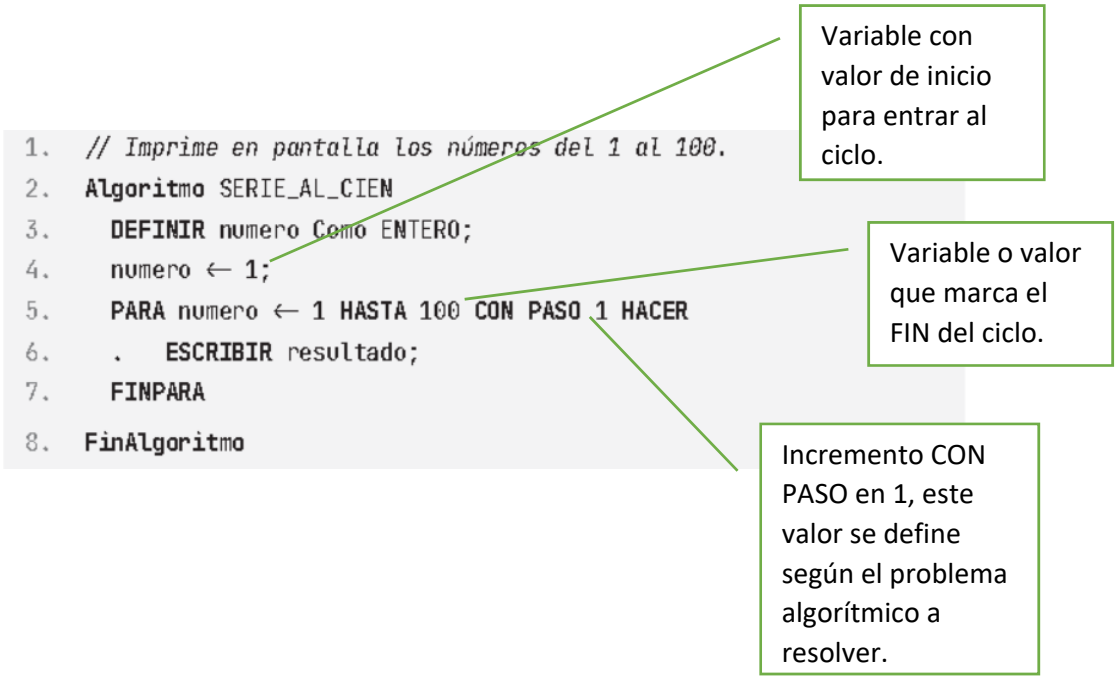



**Figura 6. Uso del PARA**

## ARREGLO o VECTOR UNIDIMNESIONAL

También nombrados o conocidos como vectores. Lo cuales tienen una peculiar característica a diferencia de las estructuras anteriores. En las estructuras anteriores la información se almacenaba en variables, en los arreglos o vectores se almacena en casillas de la memoria de la computadora.

Es importante identificar que un arreglo o vector unidimensional, está formado por tres partes:

1. El índice, el cual recorrerá a cada una de las casillas utilizando la estructura PARA.

2. La información o contenido que se guarda en cada casilla del arreglo o vector.
3. El nombre del arreglo mismo con un tamaño definido.

### Declaración de un arreglo

**ARREGLO: SUR [6]** (en español estructurado) o **DIMENSIÓN: Sur [6]** (en PseInt)

Donde:

- ARREGLO o DIMENSIÓN: palabra reservada propia que indica que es un arreglo y se crea en automático en la memoria de la computadora las celdas. (Tabla 5).
- SUR: es el nombre del arreglo definido por el programador.
- [6]: es el tamaño del arreglo. (que va de 0 a 5 de acuerdo con el índice)

Lo que indica, que el arreglo recorrerá seis casillas iniciando el índice con valor 0 hasta 5 y en cada casilla sólo se almacenan valores numéricos enteros.

| Información del arreglo SUR | 28 | 110 | 55 | 1581 | 36 | 87845 |
|---|---|---|---|---|---|---|
| Índice [i] | 0 | 1 | 2 | 3 | 4 | 5 |

**Tabla 5. Uso de ARREGLO o VECTOR UNIDIMENSIONAL**

### Lectura de un arreglo

### LEER SUR[i]

Donde:

LEER es palabra reservada, no se omite, siempre se coloca para leer un arreglo.

SUR: nombre del arreglo definido por el programador.

i: índice o casillas donde almacenaré información o contenido al arreglo en este caso se leyeron los valores: 28, 110, 55, 1581,36, 87845.

**LEER SUR [3]**

Donde:

LEER es palabra reservada, no se omite siempre se coloca para leer información o contenido que el usuario almacenará en un arreglo con posición en casilla especifica.

SUR: nombre del arreglo definido por el programador.

i: índice o posición de la casilla 3 del arreglo donde se almacena contenido o información, en este caso con valor 55.

### Escritura de un arreglo

**ESCRIBIR SUR[i]**

Donde:

ESCRIBIR es palabra reservada, no se omite siempre se coloca para mostrar en pantalla la información o contenido almacenado en un arreglo.

SUR: nombre del arreglo definido por el programador.

i: índice o posición de la casilla del arreglo y despliega (escribir) la información o contenido en pantalla, en este caso serían los valores: 28, 110, 55, 1581,36, 87845.

**ESCRIBIR SUR [3]**

Donde:

ESCRIBIR es palabra reservada, no se omite siempre se coloca para mostrar en pantalla la información o contenido almacenado en un arreglo con posición en casilla específica.

SUR: nombre del arreglo definido por el programador.

i: índice o posición de la casilla 3 del arreglo, donde se almacenó contenido o información y que será desplegado con valor 55.

Vayamos directo a los problemas usando PARA y arreglos unidimensionales.

# ESTRUCTURA MIENTRAS (WHILE)

## Ejercicio 1

Escribir un algoritmo para imprimir los primeros 50 números impares.

```
1.  NUMEROS_IMPAR
2.  numero, contador: Entero
3.  numero←1
4.  contador←0
5.  Mientras contador < 50 Hacer
6.  Escribir numero;
7.  numero ← numero + 2;
8.   contador ← contador + 1;
9.  FinMientras
10. Fin
```

### PRUEBA DE ESCRITORIO

| NUMEROS_IMPAR | | |
|---|---|---|
| **numero** | **contador** | **Mientras contador ⩽ 50** |
| 1 | 1 | verdadero |
| 3 | 2 | verdadero |
| 5 | 3 | verdadero |
| 7 | 4 | verdadero |
| ... | … | … |
| 97 | 49 | verdadero |
| 99 | 50 | verdadero |
| 101 | 51 | falso |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Imprime en pantalla los primeros 50 números impares.
2.  Algoritmo NUMEROS_IMPAR
3.      DEFINIR numero, contador Como ENTERO;
4.      numero ← 1;
5.      contador ←0;
6.      MIENTRAS (contador < 50) HACER
7.      .  ESCRIBIR numero;
8.      .  numero ← numero + 2;
9.      .  contador ← contador + 1;
10.     FINMIENTRAS
11. FinAlgoritmo
```

### Ejecutar Paso a Paso

Inicia la ejecución del programa y procede a inicializar las variables ya definidas. Debido a que no existe una solicitud por parte del programa hacia el usuario, procede a escribir el número que contiene la variable numero, siguiendo la instrucción del ciclo de repetición **Mientras**.

```
1.  *** Ejecución Iniciada. ***
2.  1
3.  3
4.  5
5.  7
6.  9
7.  11
8.  ...
```

Continúa escribiendo números impares, siguiendo con la estructura selectiva, hasta llegar al número 50 de repeticiones, entonces el programa finaliza.

```
...
89
91
93
95
97
99
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
1
3
5
7
9
11
...
89
91
93
95
97
99
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
// Imprime los 50 primeros números impares.
package EstructuraRepeticion;

public class Numeros_Impar {

    public static void main(String[] args) {
        int numero = 1;
```

```
8.          int contador = 0;
9.          while (contador < 50) {
10.             System.out.println(numero);
11.             numero = numero + 2;
12.             contador = contador +1;
13.         }
14.     }
15. }
```

## Resultado Java

```
1.  run:
2.  1
3.  3
4.  5
5.  7
6.  9
7.  11
8.  ...
9.  89
10. 91
11. 93
12. 95
13. 97
14. 99
15. BUILD SUCCESSFUL (total time: 0 seconds)
```

## Ejercicio 2

Supongamos que debemos obtener la suma de los gastos que hicimos en nuestro último viaje, pero no sabemos exactamente cuántos fueron. Construye un algoritmo para conocer cuántos gastos se realizaron.

```
1.  VIATICOS_VIAJE
2.  gasto, total: Real
3.  total←0
4.  gasto←0
5.  Escribir “Ingresa el gasto”;
6.  Leer gasto
7.  Si gasto = 0 Entonces
8.    Escribir “No hubo gastos”
9.  FinSi
10. Mientras gasto <> 0 HACER
11.   total ← total + gasto
12.   Escribir “El total es:”
13.   Escribir total
14.   Escribir “¿Hay más gastos? Ingresa la cantidad”;
15.   Leer gasto
16.   Si gasto = 0 Entonces
17.   Escribir “Ya no hay gastos”
18.   Finsi
19. FinMientras
20. Escribir “El total es: “, TOTAL
21. Fin
```

## PRUEBA DE ESCRITORIO

| VIATICOS_VIAJE | | | | |
|---|---|---|---|---|
| **GASTO** | **TOTAL** | **Salida** | | **IMPRESIÓN FINAL** |
| 0 | 0 | “No hubo gastos” | | |
| 30 | 30 | “El total es “TOTAL” 30” | “¿Hay más gastos? Dame la cantidad” | |
| 100 | 130 | | | |
| 50 | 180 | | | |
| 25 | 205 | | | |
| 0 | 0 | | “Ya no hay gastos” | “El total es “205” |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Determina si un número ingresado es negativo, nulo o
    positivo.
2.  Algoritmo VIATICOS_VIAJE
3.      DEFINIR gasto, total Como REAL;
4.      gasto ← 0;
5.      total ← 0;
6.      ESCRIBIR “Ingrese el gasto: “;
7.      LEER gasto;
8.      SI (gasto = 0) ENTONCES
9.      .   ESCRIBIR “No hubo gastos”;
10.     SINO
11.     .   MIENTRAS (gasto <> 0) ENTONCES
12.     .   .  total ← total + gasto;
13.     .   .   ESCRIBIR “El total es:”;
14.     .   .   ESCRIBIR total;
15.     .   .   ESCRIBIR “¿Hay más gastos? Ingresa la cantidad:”;
16.     .   .   LEER gasto;
17.     .   .   SI (gasto = 0) ENTONCES
18.     .   .   .  ESCRIBIR “Ya no hay gastos”;
19.     .   .   FINSI
20. .   FINMIENTRAS
21.     .   ESCRIBIR “El total es:”;
22.     .   ESCRIBIR total;
23.     FINSI
24. FinAlgoritmo
```

## Ejecutar Paso a Paso

Ejecución iniciada, solicitud e ingreso del primer gasto

```
*** Ejecución Iniciada. ***
Ingrese el gasto:
> 50
```

Impresión del gasto total y solicitud de más gastos

```
El total es:
50
¿Hay más gastos? Ingresa la cantidad:
Ingreso de más gastos
¿Hay más gastos? Ingresa la cantidad:
> 100
```

Impresión del gasto total y solicitud de más gastos

```
El total es:
150
¿Hay más gastos? Ingresa la cantidad:
```

Ingreso de cero gastos y finalización del programa

```
¿Hay más gastos? Ingresa la cantidad:
> 0
Ya no hay más gastos
El total es:
150
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el gasto:
> 50
El total es:
50
¿Hay más gastos? Ingresa la cantidad:
> 100
El total es:
150
¿Hay más gastos? Ingresa la cantidad:
> 0
Ya no hay más gastos
El total es:
150
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Construye un algoritmo para conocer cuántos gastos se realizaron.
package EstructuraRepeticion;

import java.util.Scanner;

public class Viaticos_Viaje {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int gasto = 0, total = 0;

        System.out.println("Ingresa el gasto: ");
        gasto = entrada.nextInt();

        if (gasto == 0) {
            System.out.println("No hubo gastos");
        }
```

```
        while (gasto =/= 0) {
            total = total + gasto;
            System.out.println("El total es: $" + total);
            System.out.println("¿Hay más gastos? Ingresa la cantidad");
            gasto = entrada.nextInt();
            if (gasto == 0) {
                System.out.println("Ya no Hay Gastos");
            }
        }
        System.out.println("El total es: $" + total);

    }

}
```

## Resultado de Java

```
run:
Ingresa el gasto:
50
El total es: $50
¿Hay más gastos? Ingresa la cantidad
100
El total es: $150
¿Hay más gastos? Ingresa la cantidad
0
Ya no Hay Gastos
El total es: $150
BUILD SUCCESSFUL (total time: 5 seconds)
```

## Ejercicio 3

Escriba un algoritmo dado un grupo de números naturales positivos, calcule e imprima el cubo de estos números.

```
1.  Proceso CUBO_NUMEROS_NATURALES
2.  numero, cubo: Real
3.  numero←0
4.  cubo←0
```

Escribir “Ingresa el número natural”

```
1.  Leer numero
2.  Si numero < 0 O numero > 9 entonces
3.     Escribir “El número no es natural”
4.   FinSi
5.  Mientras numero ⩾ 0 y numero ⩽ 9 hacer
6.    cubo ← numero ^ 3
7.    Escribir “El cubo del número es:”
8.    Escribir cubo
9.    Escribir “Ingresa otro número natural”
10.   Leer numero
11.   Si numero < 0 O numero > 9 entonces
12.     Escribir “El número no es natural”
13.   FinSi
14. FinMientras
```

### PRUEBA DE ESCRITORIO

| CUBO_NUMEROS | | |
|---|---|---|
| **numero** | **SI (numero < 0) O (numero > 9) Entonces** | **cubo** |
| ~~0~~ | ~~“El número no es natural”~~ | |
| ~~4~~ | | ~~64~~ |
| 3 | | 27 |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula el cubo de los números ingresados.
2.  Algoritmo CUBO_NUMEROS
3.      DEFINIR numero, cubo Como REAL;
4.      numero ← 0;
5.      cubo ← 0;
6.      ESCRIBIR “Ingrese el número natural: “;
7.      LEER numero;
8.      SI (numero < 0) O (numero > 9) ENTONCES
9.      .   ESCRIBIR “El número no es natural”;
10.     FINSI
11.     MIENTRAS (numero ⩾ 0) Y (numero ⩽ 9) ENTONCES
12.     .   cubo ← numero ^ 3;
13.     .   ESCRIBIR “El cubo del número es:”;
14.     .   ESCRIBIR cubo;
15.     .   ESCRIBIR “Ingresa otro número natural:”;
16.     .   LEER numero;
17.     .   SI (numero < 0) O (numero > 9) ENTONCES
18.     .   .    ESCRIBIR “El número no es natural”;
19. .   FINSI
20.     FINMIENTRAS
21. FinAlgoritmo
```

### Ejecutar Paso a Paso

Solicitud e ingreso de un número natural

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el número natural:
3.  > 3
```

Despliegue del resultado y solicitud de otro número natural

```
El cubo del número es:
27
Ingrese otro número natural:
```

Ingreso de un número no natural

```
Ingrese otro número natural:
> -10
```

Finaliza la ejecución del programa

```
El número no es natural
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el número natural:
> 3
El cubo del número es:
27
Ingrese otro número natural:
> -10
El número no es natural
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
// Calcula el cubo de los números ingresados.
package EstructuraRepeticion;

import java.util.Scanner;
```

```
5.  public class Cubo_Numeros {
6.      public static void main(String[] args) {
7.          Scanner entrada = new Scanner(System.in);
8.          double numero = 0;
9.          double cubo = 0;
10.         System.out.println(“Ingrese el número natural:”);
11.         numero = entrada.nextDouble();
12.         if ((numero < 0) || (numero > 9)) {
13.             System.out.println(“El número no es natural”);
14.         }
15.         while ((numero ⩾ 0) && (numero ⩽ 9)) {
16.             cubo = Math.pow(numero, 3);
17.             System.out.println(“El cubo del número es: “ +
    cubo);
18.             System.out.println(“Ingresa otro número natural: “);
19.             numero = entrada.nextDouble();
20.             if ((numero < 0) || (numero > 9)) {
21.                 System.out.println(“El número no es natural”);
22.             }
23.         }
24.     }
25.
26. }
```

## Resultado Java

```
1.  run:
2.  Ingrese el número natural:
3.  3
4.  El cubo del número es: 27.0
5.  Ingresa otro número natural:
6.  -10
7.  El número no es natural
8.  BUILD SUCCESSFUL (total time: 10 seconds)
```

## Ejercicio 4

Calcule el aumento de sueldo para un grupo de empleados de una empresa teniendo en cuenta el siguiente criterio, si sueldo es inferior a 1000, aumento del 15%, si sueldo es mayor o igual a 1000, aumento del 12%. Imprima el sueldo nuevo del trabajador y el total de nómina de la empresa, considerando este nuevo aumento.

```
1.  Proceso AUMENTO_SUELDO_GRUPO
2.  trabajadores, nomina, sueldo, sueldoNuevo: entero
3.  Inicio
```

Escribir “¿Cuántos trabajadores tiene el grupo?”;

```
1.  Leer trabajadores
2.  Mientras trabajadores > 0
```

Escribir “¿Cuál es el sueldo actual?”;

```
1.    Leer sueldo
2.    Si sueldo < 1000 entonces
3.      sueldoNuevo ← sueldo * 1.15
4.    sino
5.      Si sueldo ⩾ 1000 entonces
6.        sueldoNuevo ← sueldo * 1.12
7.      Fin_si
8.    Fin_si
9.    Escribir “El nuevo sueldo del grupo de trabajadores es: “
10.   Escribir sueldoNuevo
11.   nomina ← (sueldoNuevo * trabajadores)
12.   Escribir “El total de nómina es:”
13.   Escribir nomina
14.   Escribir “¿Cuántos trabajadores tiene el nuevo grupo?”
15.   Leer trabajadores
16. FinMientras
17. Fin
```

## PRUEBA DE ESCRITORIO

| AUMENTO_SUELDO_GRUPO | | | |
|---|---|---|---|
| **trabajadores** | **sueldo** | **sueldoNuevo** | **nomina** |
| ~~0~~ | ~~0~~ | ~~0~~ | ~~0~~ |
| ~~10~~ | ~~1500~~ | ~~1680~~ | ~~16800~~ |
| 5 | 500 | 575 | 2875 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcule el aumento de sueldo para un grupo de empleados de
    una
2.  // empresa teniendo en cuenta el siguiente criterio, si sueldo
    es
3.  // inferior a 1,000 aumento del 15%, si sueldo es mayor o igual
4.  // a 1000 aumento del 12%.
5.  Algoritmo AUMENTO_SUELDO_GRUPO
6.      DEFINIR trabajadores, sueldo, sueldoNuevo, nomina Como REAL;
7.      trabajadores ← 0;
8.      sueldo ← 0;
9.      sueldoNuevo ← 0;
10.     nomina ← 0;
11.     ESCRIBIR "¿Cuántos trabajadores tiene el grupo?";
12.     LEER trabajadores;
13.     MIENTRAS (trabajadores > 0) HACER
14.     .  ESCRIBIR "¿Cuál es el sueldo actual?";
15.     .  LEER sueldo;
16.     .  SI (sueldo < 1000) ENTONCES
17.     .  .   sueldoNuevo ← sueldo * 1.15;
18.     .  SINO
19.     .  .    SI (sueldo ⩾ 1000) ENTONCES
20.     .  .    .   sueldoNuevo ← sueldo * 1.12;
21.     .  .    FINSI
22.     .  FINSI
23.     .  nomina ← sueldoNuevo * trabajadores;
24.     .  ESCRIBIR "El nuevo sueldo del grupo de trabajadores será
    de: ";
25.     .  ESCRIBIR sueldoNuevo;
```

```
26.      .  ESCRIBIR " El total de nómina es: ";
27.      .  ESCRIBIR nomina;
28.      .  ESCRIBIR "¿Cuántos trabajadores tiene el grupo?";
29.      .  LEER trabajadores;
30.    FINMIENTRAS
31. FinAlgoritmo
```

## Ejecutar Paso a Paso

Solicitud e ingreso de trabajadores en el grupo

```
1.  *** Ejecución Iniciada. ***
2.  ¿Cuántos trabajadores tiene el grupo?:
3.  > 10
4.  Solicitud de sueldo del grupo de trabajadores
5.  ¿Cuál es el sueldo actual?:
6.  > 1500
```

Despliegue del nuevo sueldo de trabajadores y el total de nómina

```
1.  El nuevo sueldo del grupo de trabajadores es: 1680.0000000000002
2.  El total de nómina es: $16800.000000000004
```

Solicitud de otro grupo de trabajadores

```
1.  ¿Cuántos trabajadores tiene el grupo?:
2.  > 0
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  ¿Cuántos trabajadores tiene el grupo?:
3.  > 10
```

```
¿Cuál es el sueldo actual?:
> 1500
El nuevo sueldo del grupo de trabajadores es: 1680.0000000000002
El total de nómina es: $16800.000000000004
¿Cuántos trabajadores tiene el grupo?:
> 0
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Calcule el aumento de sueldo para un grupo de empleados de una empresa teniendo
 * en cuenta el siguiente criterio, si sueldo es inferior a 1,000 aumento del 15%,
 * si sueldo es mayor o igual a 1000 aumento del 12%.
 */
package EstructuraRepeticion;

import java.util.Scanner;

public class Aumento_Sueldo_Grupo {

  public static void main(String[] args) {
     Scanner entrada = new Scanner(System.in);
     double trabajadores, sueldo, sueldoNuevo, nomina;

     System.out.println("¿Cuántos trabajadores tiene el grupo?");
     trabajadores = entrada.nextDouble();
     while (trabajadores > 0) {
        System.out.println("¿Cuál es el sueldo actual?");
        sueldo = entrada.nextDouble();
        if (sueldo < 1000) {
           sueldoNuevo = sueldo * 1.15;
           nomina = sueldoNuevo * trabajadores;
           System.out.println("El nuevo sueldo del grupo de trabajadores es: $"
+ sueldoNuevo);
           System.out.println("El total de la nómina es: " +
```

```
26. nomina);
27.         } else {
28.            if (sueldo ⩾ 1000) {
29.               sueldoNuevo = sueldo * 1.12;
30.               nomina = sueldoNuevo * trabajadores;
31.               System.out.println(“El nuevo sueldo del grupo de
    trabajadores es: $” +
32. sueldoNuevo);
33.               System.out.println(“El total de la nómina es: $” +
    nomina);
34.            }
35.         }
36.         System.out.println(“---”);
37.         System.out.println(“¿Cuántos trabajadores tiene el
    grupo?”);
38.         trabajadores = entrada.nextDouble();
39.      }
40.   }
41.
42. }
```

### Resultado Java

```
1.
2.  run:
```

¿Cuántos trabajadores tiene el grupo?

```
3.  10
```

¿Cuál es el sueldo actual?

```
1.  1500
2.  El nuevo sueldo del grupo de trabajadores es:
    $1680.0000000000002
3.  El total de la nómina es: $16800.000000000004
4.  ---
```

¿Cuántos trabajadores tiene el grupo?

```
5.  5
```

¿Cuál es el sueldo actual?

```
1.  500
2.  El nuevo sueldo del grupo de trabajadores es: $575.0
3.  El total de la nómina es: 2875.0
4.  ---
```

¿Cuántos trabajadores tiene el grupo?

```
1.  0
2.  BUILD SUCCESSFUL (total time: 16 seconds)
```

## Ejercicio 5

Realiza un algoritmo que permita al usuario continuar dentro del sistema. Donde pregunte:

```
1.  ¿Desea continuar?»; y como respuesta: S/N
2.
3.  OPCION_CONTINUAR
4.  opcion: Entero
5.  Inicio
6.  opcion← 0
7.  Mientras opcion <> 2
8.     Escribir “¿Desea continuar?”
9.     Escribir “(1) Si.    (2) No.”
10.    Leer opcion
11. Fin_Mientras
12. Fin
```

### PRUEBA DE ESCRITORIO

| OPCION_CONTINUAR | |
|---|---|
| **Opcion** | **Salida** |
| (1) Sí | ¿Desea continuar? |
| (1) Sí | ¿Desea continuar? |
| (2) No | <<Fin>> |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Muestra al usuario un menú y dependiendo de su respuesta
    continúa
2.  // o termina el algoritmo.
3.  Algoritmo OPCION_CONTINUAR
4.      DEFINIR opcion Como ENTERO;
5.      opcion ← 0;
6.      MIENTRAS (opcion <> 2) HACER
```

```
7.      .   ESCRIBIR "¿Deseas continuar?";
8.      .   ESCRIBIR "(1) Si (2)No";
9.      .   LEER opcion;
10.     FINMIENTRAS
11. FinAlgoritmo
```

## Ejecutar Paso a Paso

El programa nos pregunta si deseamos o no terminar, ingresamos la opción 1 (Sí).

```
1.  *** Ejecución Iniciada. ***
2.  ¿Deseas continuar?
3.  (1)Si  (2)No
4.  > 1
```

Al aceptar continuar, el programa imprime en pantalla el mismo cuestionamiento.

```
1.  ¿Deseas continuar?
2.  (1) Si (2) No
3.  > 1
4.  ¿Deseas continuar?
5.  (1) Si (2) No
6.  >
```

Tras ingresar repetidas veces la opción de continuar, el programa continúa desplegando en pantalla el mismo mensaje y mostrando las mismas opciones.

```
1.  ¿Deseas continuar?
2.  (1) Si (2) No
3.  > 1
4.  ¿Deseas continuar?
5.  (1) Si (2) No
```

```
> 1
...
¿Deseas continuar?
(1) Si (2) No
>
```

Al ingresar la opción 2 (No), el programa finaliza.

```
¿Deseas continuar?
(1) Si (2) No
> 2
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
¿Deseas continuar?
(1) Si (2) No
> 1
¿Deseas continuar?
(1) Si (2) No
> 1
¿Deseas continuar?
(1) Si (2) No
> 2
*** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
/*Realiza un algoritmo que permita al usuario continuar dentro
del sistema. Donde pregunte:
  ¿Desea continuar?»; y como respuesta: S/N
 */
package EstructuraRepeticion;

```

```
6.  import java.util.Scanner;
7.
8.  public class Opcion_Continuar {
9.
10.     public static void main(String[] args) {
11.         Scanner entrada = new Scanner (System.in);
12.         int opcion = 0;
13.
14.         while (opcion =≠ 2) {
15.             System.out.println(“¿Desea Continuar?”);
16.             System.out.println(“(1) Si (2)No”);
17.             opcion = entrada.nextInt();
18.         }
19.     }
20.
21. }
```

## Resultado Java

```
1.  run:
2.  ¿Desea Continuar?
3.  (1)Si   (2)No
4.  1
5.  ¿Desea Continuar?
6.  (1)Si   (2)No
7.  1
8.  ¿Desea Continuar?
9.  (1)Si   (2)No
10. 2
11. BUILD SUCCESSFUL (total time: 5 seconds)
```

## Ejercicio 6

Hacer un algoritmo para calcular la suma de los primeros cien números.

```
1.  SUMA_NUMEROS
2.  numero, suma: Entero
3.  Inicio
4.  suma←0
5.  numero←1
6.  Mientras (numero ⩽ 100)
7.     suma ← suma + numero
8.     numero ← numero + 1
9.  FinMientras
10. Escribir “La suma de los primeros 100 números es: ”
11. Escribir suma
12. Fin
```

### PRUEBA DE ESCRITORIO

| SUMA_NUMEROS | | |
|---|---|---|
| **suma** | **numero** | salida |
| ~~0~~ | ~~1~~ | - |
| ~~1~~ | ~~2~~ | - |
| ~~3~~ | ~~3~~ | - |
| ... | ... | - |
| 5050 | 100 | 5050 |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Determina si un número ingresado es negativo, nulo o
    positivo.
2.  Algoritmo SUMA_NUMEROS
3.      DEFINIR numero, suma Como ENTERO;
4.      numero ← 0;
5.      suma ← 0;
```

```
6.      MIENTRAS (numero ⩽ 100) ENTONCES
7.      .   suma ← suma + numero;
8.      .   numero ← numero + 1;
9.      FINMIENTRAS
10.     ESCRIBIR “La suma de los primeros 100 números es: “;
11.     ESCRIBIR suma;
12. FinAlgoritmo
```

### Ejecutar Paso a Paso

Realiza las operaciones pertinentes e imprime en pantalla lo obtenido.

```
1.  *** Ejecución Iniciada. ***
2.  La suma de los primeros 100 números es:
3.  > 5050
4.  *** Ejecución Finalizada. ***
```

### Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  La suma de los primeros 100 números es:
3.  > 5050
4.  *** Ejecución Finalizada. ***
```

### ALGORITMO EN NETBEANS: JAVA

```
1.  // Crear un algoritmo que me diga los números del 1 al 100.
2.
3.  package EstructuraRepeticion;
4.
5.  public class Suma_Numeros {
6.
7.      public static void main(String[] args) {
8.          int numero, suma;
```

```
9.          numero = 1;
10.         suma = 0;
11.         while (numero ⩽ 100) {
12.             suma += numero;
13.             numero++;
14.         }
15.         System.out.println(“La suma de los primeros 100 números
    es: “ +
16. suma);
17.     }
18. }
```

## Resultado Java

```
1.  run:
2.  La suma de los primeros 100 números es: 5050
3.  BUILD SUCCESSFUL (total time: 0 seconds)
```

# Ejercicio 7

Crear un algoritmo que me diga los números del 1 al 1000.

```
1.  UNO_AL_MIL
2.  numero: Entero
3.  numero ← 1
4.  Mientras (numero ⩽ 1000)
5.     Escribir numero
6.     numero ←numero + 1
7.  FinMientras
8.  Fin
```

### PRUEBA DE ESCRITORIO

| NUMEROS_AL_MIL | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **numero** | 1 | 2 | 3 | 4 | ... | 997 | 998 | 1000 |
| Numero ⩽ 1000 | Verdadero | Verdadero | Verdadero | Verdadero | ... | Verdadero | Verdadero | Falso |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Muestra en pantalla los números del 1 al 1,000.
2.  Algoritmo NUMEROS_AL_MIL
3.      DEFINIR numero Como ENTERO;
4.      numero ← 1;
5.      MIENTRAS (numero ⩽ 1000) ENTONCES
6.      .   ESCRIBIR numero;
7.      .   numero ← numero + 1;
8.      FINMIENTRAS
9.  FinAlgoritmo
```

### Ejecutar Paso a Paso

Se imprime en pantalla el valor inicial (1)

```
*** Ejecución Iniciada. ***
1
2
3
4
...
```

El programa continúa imprimiendo en pantalla los resultados de manera secuencial.

```
...
500
501
502
503
...
```

Al llegar a 1000, el algoritmo concluye su ejecución.

```
...
997
998
999
1000
*** Ejecución Finalizada. ***
```

### Ejecutar

```
*** Ejecución Iniciada. ***
1
2
3
4
...
```

```
500
501
502
503
...
997
998
999
1000
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```java
// Crear un algoritmo que muestre los números del 1 al 1000.
package EstructuraRepeticion;

public class Numeros_Al_Mil {

    public static void main(String[] args) {
        int numero = 1;
        while (numero ⩽ 1000) {
            System.out.println(numero);
            numero++;
        }
    }

}
```

## Resultado Java

```
run:
1
2
3
4
...
500
```

```
501
502
503
...
997
998
999
1000
BUILD SUCCESSFUL (total time: 0 seconds)
```

# Ejercicio 8

Escriba un algoritmo que obtenga la suma e imprima los términos de la siguiente serie:

2, 5, 7, 10, 12, 15, 17…1800

```
1.  Serie, contador: entero
2.  Serie←0;
3.  Escribir “A continuación, se imprime una serie de números,
    aumentando +3 y +2 consecutivamente.”
4.  Mientras Serie<1800 hacer
5.    serie = serie + 2
6.    contador = contador + serie
7.    Escribir Serie
8.    serie = serie + 3
9.    contador = contador + serie
10.   Escribir Serie
11. Fin Mientras
12. Escribir “La suma de la serie es: “
13. Escribir contador
14. Fin
```

## PRUEBA DE ESCRITORIO

| SERIE_NUMEROS | |
|---|---|
| **Serie** | **Contador** |
| 0 | 0 |
| 2 | 2 |
| 5 | 7 |
| 7 | 14 |
| … | … |
| 1795 | 645123 |
| 1797 | 646920 |
| 1800 | 648720 |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Escriba un algoritmo que obtenga la suma e imprima los
    términos de
2.  // la siguiente serie: 2, 5, 7, 10, 12, 15, 17…1800 .
3.  Algoritmo SERIE_NUMEROS
4.      DEFINIR serie, contador Como ENTERO;
5.      serie ← 0;
6.      contador ← 0;
7.      ESCRIBIR "A continuación, se imprime una serie de números de
    manera
8.        n+2, n+3";
9.
10.     MIENTRAS (serie < 1800) HACER
11.     .   serie ← serie + 2;
12.     .   contador ← contador + serie;
13.     .   ESCRIBIR serie;
14.     .   serie ← serie + 3;
15.     .   contador ← contador + serie;
16.     .   ESCRIBIR serie;
17.
18.     FINMIENTRAS
19.     ESCRIBIR "La suma de todos los términos de la serie es:";
20.     ESCRIBIR contador;
21. FinAlgoritmo
```

### Ejecutar Paso a Paso

Se ejecuta el programa, lo primero que muestra en pantalla es un mensaje indicando su función.

```
1.  *** Ejecución Iniciada. ***
2.  A continuación, se imprime una serie de números de manera n+2,
    n+3:
```

Imprime en pantalla la serie de números aplicando la fórmula preestablecida.

```
2
5
7
10
12
15
17
20
...
1792
1795
1797
1800
```

Imprime en pantalla la suma de los elementos de la serie y finaliza la ejecución del algoritmo.

```
La suma de todos los términos de la serie es:
648720
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
A continuación, se imprime una serie de números de manera n+2, n+3:
2
5
7
10
12
15
17
20
...
1792
```

```
1795
1797
1800
La suma de todos los términos de la serie es:
648720
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
1 /* Suma e imprime los números de una serie siguiendo el patrón de n+2 y n+3, al igual
   que imprime la suma de todos los términos de la serie,
 */
package EstructuraRepeticion;

public class Serie_Numeros {

    public static void main(String[] args) {
        System.out.println("A continuación se imprime una serie de n+2, n+3");
        int serie, contador;
        serie = 0;
        contador = 0;
        while (serie < 1800) {
            serie = serie + 2;
            contador = contador + serie;
            System.out.println(serie);
            serie = serie + 3;
            contador = contador + serie;
            System.out.println(serie);
        }
        System.out.println("La suma de todos los términos de la serie es: " +
contador);

    }

}
```

## Resultado Java

```
run:
A continuación, se imprime una serie de n+2, n+3
2
5
7
10
12
15
...
1787
1790
1792
1795
1797
1800
La suma de todos los términos de la serie es: 648720
BUILD SUCCESSFUL (total time: 0 seconds)
```

# CICLOS REPETITIVOS: ESTRUCTURA PARA (FOR)

## Ejercicio 1

Escribe un algoritmo que imprima la factorial de un número dado por el usuario.

```
1.  FACTORIAL_NUMERO
2.  numero, resultado, contador: Entero
3.  resultado ← 1
4.  contador ←1
5.  Escribir “Ingrese el número del cual se desea obtener su
    factorial:”
6.  Leer numero
7.  Para contador ← 1 Hasta contador = numero Hacer contador + 1
8.    resultado ← contador * resultado
9.  Fin Para
10. Escribir “El factorial del número ingresado es:”
11. Escribir resultado
12. Fin
```

### PRUEBA DE ESCRITORIO

| FACTORIAL_NUMERO | | |
|---|---|---|
| **numero** | **contador** | **resultado** |
| ~~0~~ | ~~0~~ | ~~0~~ |
| ~~2~~ | ~~1~~ | ~~0~~ |
| ~~2~~ | ~~2~~ | ~~2~~ |
| 3 | 3 | 6 |

### ALGORITMO EN PseInt
### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula el factorial de un número dado por el usuario.
2.  Algoritmo FACTORIAL_NUMERO
3.    DEFINIR numero, resultado, contador Como ENTERO;
4.    resultado ← 1;
5.    ESCRIBIR “Ingrese el número del cuál desea obtener su
    factorial: “;
6.    LEER numero;
7.    PARA contador ← 1 HASTA numero CON PASO 1 HACER
8.    .   resultado ← contador * resultado;
9.    FINPARA
10.   ESCRIBIR “El factorial del número ingresado es: “;
11.   ESCRIBIR resultado;
12. FinAlgoritmo
```

## Ejecutar Paso a Paso

Se inicia la ejecución del algoritmo y nos pide introducir un valor, el cual ingresamos.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el número del cual desea obtener el factorial:
3.  > 8
```

Al ingresar el valor, el algoritmo nos regresa en pantalla un mensaje indicando el resultado obtenido y con esto, se finaliza la ejecución del algoritmo.

```
1.  El factorial del número ingresado es:
2.  40320
3.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el número del cual desea obtener el factorial:
> 8
El factorial del número ingresado es:
40320
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/*Realizar un algoritmo para hallar el Factorial_Numero de
cualquier número.
 */
package EstructuraRepeticion;

import java.util.Scanner;

public class Factorial_Numero {

   public static void main(String[] args) {
      int numero;
      int resultado = 1;
      Scanner entrada = new Scanner(System.in);

      System.out.println("Ingresa el número del cual desea
obtener su factorial: ");
      numero = entrada.nextInt();

      for (int i = 1; i ⩽ numero; i++) {
         resultado = i * resultado;
      }
      System.out.println("El factorial del número ingresado es:
" + resultado);
   }

}
```

## Resultado Java

```
run:
Ingresa el número del cual desea obtener su factorial:
8

El factorial del número ingresado es: 40320
BUILD SUCCESSFUL (total time: 2 seconds)
```

## Ejercicio 2

Crea un algoritmo que imprima en pantalla los números del 1 al 100.

```
1.  SERIE_AL_CIEN
2.  numero: entero
3.  numero ← 1
4.  Para numero ← 1 Hasta numero = 100 Hacer numero + 1
5.   Escribir numero
6.  Fin Para
7.  Fin
```

### PRUEBA DE ESCRITORIO

| SERIE_AL_CIEN |
|---|
| **numero** |
| ~~1~~ |
| ~~2~~ |
| ... |
| ~~99~~ |
| 100 |

### ALGORITMO EN PseInt

### PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Imprime en pantalla los números del 1 al 100.
2.  Algoritmo SERIE_AL_CIEN
3.    DEFINIR numero Como ENTERO;
4.    numero ← 1;
5.    PARA numero ← 1 HASTA 100 CON PASO 1 HACER
6.    .   ESCRIBIR resultado;
7.    FINPARA
8.  FinAlgoritmo
```

## Ejecutar Paso a Paso

Al iniciar el algoritmo, nos imprime en pantalla la serie de números del 1 al 100 sin necesidad de ingresar valores por parte del usuario.

```
*** Ejecución Iniciada. ***
1
2
3
4
...
```

Continúa imprimiendo la serie hasta llegar al número 100, en donde se detiene y finaliza el algoritmo.

```
96
97
98
99
100
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
1
2
3
4
...
50
51
52
53
54
...
96
97
```

```
98
99
100
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```java
//Imprime en pantalla los primeros 100 números
package EstructuraRepeticion;

public class Serie_Al_Cien {

    public static void main(String[] args) {
        for (int numero = 1; numero ⩽ 100; numero++) {
            System.out.println(numero);
        }
    }
}
```

## Resultado Java

```
run:




...


...


0
BUILD SUCCESSFUL (total time: 0 seconds)
```

## Ejercicio 3

Crea un algoritmo que calcule la tabla de multiplicar de cualquier número dado por el usuario.

```
1.  TABLA_MULTIPLICAR
2.  numero, contador, resultado: entero
3.  Escribir “Ingrese la tabla de multiplicar a mostrar:”
4.  Leer numero
5.  Para contador ← 1 Hasta contador ← 10 Con Paso 1 Hacer
6.    resultado ← contador * numero
7.    Escribir numero + “ x ” + contador “ = ” + resultado
8.  Fin Para
9.  Fin
```

### PRUEBA DE ESCRITORIO

| TABLA_MULTIPLICAR | | | | | |
|---|---|---|---|---|---|
| num=1 | | num=5 | | num=10 | |
| contador | resultado | contador | resultado | contador | resultado |
| 1 | 1 | 1 | 5 | 1 | 10 |
| 2 | 2 | 2 | 10 | 2 | 20 |
| 3 | 3 | 3 | 15 | 3 | 30 |
| 4 | 4 | 4 | 20 | 4 | 40 |
| 5 | 5 | 5 | 25 | 5 | 50 |
| 6 | 6 | 6 | 30 | 6 | 60 |
| 7 | 7 | 7 | 35 | 7 | 70 |
| 8 | 8 | 8 | 40 | 8 | 80 |
| 9 | 9 | 9 | 45 | 9 | 90 |
| 10 | 10 | 10 | 50 | 10 | 100 |

**ALGORITMO EN PseInt**
**PRUEBA DE ESCRITORIO EN PseInt**

```
1.  // Imprime en pantalla la tabla de multiplicar del número
    ingresado.
2.  Algoritmo TABLA_MULTIPLICAR
3.    DEFINIR numero, contador, resultado Como ENTERO;
4.    numero ← 1;
5.    ESCRIBIR “Ingrese la tabla de multiplicar a mostrar: “;
6.    LEER numero;
7.    PARA contador ← 1 HASTA 10 CON PASO 1 HACER
8.    .   resultado ← contador * numero;
9.    .   ESCRIBIR numero “ X ” contador “ = ” resultado;
10.   FINPARA
11. FinAlgoritmo
```

## Ejecutar Paso a Paso

Al iniciar el algoritmo, imprime en pantalla la solicitud para ingresar la tabla deseada.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese la tabla de multiplicar a mostrar:
3.  > 5
```

Una vez registrado el valor, el algoritmo imprime en pantalla el resultado correspondiente y finaliza su ejecución.

```
1.  5 X 1 = 5
2.  5 X 2 = 10
3.  5 X 3 = 15
4.  5 X 4 = 20
5.  5 X 5 = 25
6.  5 X 6 = 30
7.  5 X 7 = 35
8.  5 X 8 = 40
```

```
5 X 9 = 45
5 X 10 = 50
*** Ejecución Finalizada. ***
```

**Ejecuv**

```
*** Ejecución Iniciada. ***
Ingrese la tabla de multiplicar a mostrar:
> 5
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Crear un algoritmo que calcule la tabla de multiplicar de cualquier
 * número dado por el usuario.
 */
package EstructuraRepeticion;

import java.util.Scanner;

public class Tabla_Multiplicar {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int numero, resultado;

        System.out.println("Ingresa la tabla de multiplicar a mostrar: ");
```

```
15.         numero = entrada.nextInt();
16.
17.         for (int contador = 1; contador ⩽ 10; contador++) {
18.             resultado = contador * numero;
19.             System.out.println(numero + " X " + contador + " = "
20. + resultado);
21.         }
22.     }
23.
24. }
```

## Resultado Java

```
1.  run:
2.  Ingresa la tabla de multiplicar a mostrar:
3.  5
4.  5 X 1 = 5
5.  5 X 2 = 10
6.  5 X 3 = 15
7.  5 X 4 = 20
8.  5 X 5 = 25
9.  5 X 6 = 30
10. 5 X 7 = 35
11. 5 X 8 = 40
12. 5 X 9 = 45
13. 5 X 10 = 50
14. BUILD SUCCESSFUL (total time: 9 seconds)
```

# Ejercicio 4

Escribe un algoritmo que calcule e imprima la suma de los primeros cien números.

```
1.  SUMA_CIEN
2.  suma, contador: entero
3.  contador ← 0
4.  suma ← 0
5.  Para contador ← 1 Hasta contador = 100 Hacer
6.    suma← suma + contador
7.  Fin Para
8.  Escribir “La suma de los primeros cien números es:”
9.  Escribir suma
10. Fin
```

## PRUEBA DE ESCRITORIO

| SUMA_CIEN | |
|---|---|
| suma | contador |
| ~~0~~ | ~~0~~ |
| ~~0~~ | ~~1~~ |
| ~~1~~ | ~~2~~ |
| ~~3~~ | ~~3~~ |
| ~~6~~ | ~~4~~ |
| ... | ... |
| 4950 | 100 |
| 5050 | |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcula la suma de los primeros cien números.
2.  Algoritmo SUMA_CIEN
3.    DEFINIR suma, contador Como ENTERO;
```

```
4.    suma ← 0;
5.    contador ← 0;
6.    PARA contador ← 1 HASTA 100 CON PASO 1 HACER
7.    .   suma ← suma + contador;
8.    FINPARA
9.    ESCRIBIR “La suma de los primeros cien números es: “;
10.   ESCRIBIR suma;
11. FinAlgoritmo
```

### Ejecutar Paso a Paso

El algoritmo inicia y realiza las operaciones necesarias para mostrar en pantalla el resultado de los primeros cien números, finalizando después.

```
1.  *** Ejecución Iniciada. ***
2.  La suma de los primeros cien números es:
3.  5050
4.  *** Ejecución Finalizada. ***
```

### Ejecutar

```
1.  *** Ejecución Iniciada. ***
2.  La suma de los primeros cien números es:
3.  5050
4.  *** Ejecución Finalizada. ***
```

#### ALGORITMO EN NETBEANS: JAVA

```
1.  /*Realiza un algoritmo para calcular la suma de los primeros
    cien números
2.   */
3.  package EstructuraRepeticion;
4.
5.  public class Suma_Cien {
6.      public static void main(String[] args) {
```

```
7.          int suma = 0;
8.          for (int contador = 1; contador ⩽ 100; contador++) {
9.              suma = suma + contador;
10.         }
11.         System.out.println(“La suma de los primeros cien números
    es: “ +
12. suma);
13.     }
14.
15. }
```

## Resultado Java

```
1.  run:
2.  La suma de los primeros cien números es: 5050
3.  BUILD SUCCESSFUL (total time: 0 seconds)
```

## Ejercicio 5

Escribe un algoritmo que imprima en pantalla las tablas de multiplicar de los números del 1 al 9.

```
1.  TABLAS_MULTIPLICAR
2.  contador, resultado, tabla: entero
3.  contador ← 0
4.  resultado ← 0
5.  Para tabla ← 1 Hasta tabla = 9 Con Paso 1 Hacer
6.   Para contador ← 1 Hasta contador = 10 Con Paso 1 Hacer
7.     resultado← contador* tabla
8.     Escribir tabla " x " contador " = "resultado
9.   Fin Para
10. Fin Para
11. Fin
```

### PRUEBA DE ESCRITORIO

| TABLAS_MULTIPLICAR | | |
|---|---|---|
| **tabla** | **contador** | **resultado** |
| 4 | 1 | 4 |
| | 2 | 8 |
| | 3 | 12 |
| | 4 | 16 |
| | 5 | 20 |
| | 6 | 24 |
| | 7 | 28 |
| | 8 | 32 |
| | 9 | 36 |
| | 10 | 40 |

**ALGORITMO EN PseInt**
**PRUEBA DE ESCRITORIO EN PseInt**

```
1.  // Calcula la suma de los primeros cien números.
2.  Algoritmo SUMA_CIEN
3.    DEFINIR resultado, contador, tabla Como ENTERO;
4.    resultado ← 0;
5.    contador ← 0;
6.    PARA contador ← 1 HASTA 100 CON PASO 1 HACER
7.    .   PARA contador ← 1 HASTA 100 CON PASO 1 HACER
8.    .  .   resultado ← contador * tabla;
9.    .  .   ESCRIBIR tabla " X " contador " = " resultado;
10.   .  FINPARA
11.   FINPARA
12. FinAlgoritmo
```

## Ejecutar Paso a Paso

Al iniciar la ejecución del algoritmo, nos empieza a imprimir las tablas de multiplicar en pantalla.

```
1.  *** Ejecución Iniciada. ***
2.  1 X 1 = 1
3.  1 X 2 = 2
4.  1 X 3 = 3
5.  1 X 4 = 4
6.  1 X 5 = 5
7.  1 X 6 = 6
8.  1 X 7 = 7
9.  1 X 8 = 8
10. 1 X 9 = 9
11. 1 X 10 = 10
12.
13. ...
```

Imprime en pantalla las tablas de multiplicar hasta llegar a la tabla 9, la imprime y finaliza la ejecución del algoritmo.

```
...

9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
9 X 10 = 90
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
1 X 10 = 10

...
...

9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
```

```
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
9 X 10 = 90
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Escribe un algoritmo que imprima en pantalla las tablas de multiplicar de
// los números del 1 al 9.
package EstructuraRepeticion;

public class Tablas_Multiplicar {

   public static void main(String[] args) {
      int resultado = 1;

      for (int tabla = 1; tabla < 10; tabla++) {
         for (int contador = 1; contador ⩽ 10; contador++) {
            resultado = tabla * contador;
            System.out.println(tabla + " X " + contador + " = " +
resultado);
         }
         //Línea divisora para el formato de la salida.
         System.out.println("");
      }
   }

}
```

## Resultado Java

```
run:

1 X 1 = 1
1 X 2 = 2
1 X 3 = 3
1 X 4 = 4
1 X 5 = 5
1 X 6 = 6
1 X 7 = 7
1 X 8 = 8
1 X 9 = 9
1 X 10 = 10

...

9 X 1 = 9
9 X 2 = 18
9 X 3 = 27
9 X 4 = 36
9 X 5 = 45
9 X 6 = 54
9 X 7 = 63
9 X 8 = 72
9 X 9 = 81
9 X 10 = 90

BUILD SUCCESSFUL (total time: 0 seconds)
```

# ESTRUCTURAS COMBINADAS (PARA + SI)

## Ejercicio 1

Realizar un algoritmo para calcular la media de los números pares e impares, sólo se ingresarán 10 números.

```
1.  MEDIA_PARES_IMPARES
2.  numero,  sumaPar, sumaImpar, contador: entero
3.  sumaPar←0
4.  sumaImpar←0
5.  Para contador ← 1 Hasta contador = 10 Con Paso 1 Hacer
6.   Escribir “Ingrese un número: “
7.   Leer numero
8.   Si (numero %= 0) entonces
9.     sumaPar ← sumaPar + numero
10.  Si no
11.    sumaImpar ← sumaImpar + numero
12.  Fin_si
13. Fin_Para
14. sumaImpar ← sumaImpar / 10
15. sumaPar ← sumaPar / 10
16. Escribir “La media de los números pares ingresados es: ”
17. Escribir sumaPar
18. Escribir “La media de los números impares ingresados es: ”
19. Escribir sumaImpar
20. Fin
```

## PRUEBA DE ESCRITORIO

| MEDIA_PARES_IMPARES | | | |
|---|---|---|---|
| **numero** | **sumaPar** | **sumaImpar** | **contador** |
| | 0 | 0 | 0 |
| 2 | 2 | | 1 |
| 3 | | 3 | 2 |
| 7 | | 10 | 3 |
| 4 | 6 | | 4 |
| 8 | 14 | | 5 |
| 3 | | 13 | 6 |
| 5 | | 18 | 7 |
| 5 | | 23 | 8 |
| 2 | 16 | | 9 |
| 4 | 20 | | 10 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  // Calcular la media de los números pares e impares ingresados.
2.  Algoritmo MEDIA_PARES_IMPARES
3.      DEFINIR numero, sumaPar, sumaImpar, contador Como ENTERO;
4.      sumaPar ← 0;
5.      sumaImpar ← 0;
6.      PARA contador ← 1 HASTA 10 CON PASO 1 HACER
7.      .  ESCRIBIR “Ingrese un número entero: «;
8.      .  LEER numero;
9.      .  SI (numero MOD 2 = 0) ENTONCES
10.     .  .  sumaPar ← sumaPar + numero;
11.     .  SINO
12.     .  .  sumaPar ← sumaPar + numero;
13.     .  FINSI
14.     FINPARA
15.     sumaPar ← sumaPar / 10;
16.     sumaImpar ← sumaImpar / 10;
17.     ESCRIBIR “La media de los números pares ingresados es: “;
18.     ESCRIBIR sumaPar;
19.     ESCRIBIR “La media de los números impares ingresados es: “;
20.     ESCRIBIR sumaImpar;
21.     FinAlgoritmo
```

## Ejecutar Paso a Paso

Al iniciar, el algoritmo nos solicita la entrada de un valor.

```
*** Ejecución Iniciada. ***
Ingrese un número entero:
> 5
```

Continúa pidiéndonos valores, en total el algoritmo nos pide 10 valores.

```
Ingrese un número entero:
> 7
Ingrese un número entero:
> 8
Ingrese un número entero:
> 5
...
```

El algoritmo realiza las operaciones pertinentes e imprime el promedio de todos los valores ingresados, divididos entre pares e impares, después de esto, finaliza la ejecución del algoritmo.

```
La media de los números pares ingresados es:
3
La media de los números pares ingresados es:
2
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese un número entero:
> 5
Ingrese un número entero:
> 7
```

```
Ingrese un número entero:
> 8
Ingrese un número entero:
> 5

...

La media de los números pares ingresados es:
3
La media de los números pares ingresados es:
2
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Realizar un algoritmo para calcular la media de los números pares e impares,
 * sólo se ingresará diez números.
 */
package EstructuraRepeticion;

import java.util.Scanner;

public class Media_Pares_Impares {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int numero, sumaPar = 0, sumaImpar = 0;

        for (int contador = 0; contador < 10; contador++) {
            System.out.println("Ingresa un número entero: ");
            numero = entrada.nextInt();

            if (numero % 2 == 0) {
                sumaPar += numero;
            } else {
                sumaImpar += numero;
            }
```

```
        }
        sumaPar = sumaPar / 10;
        sumaImpar = sumaImpar / 10;
        System.out.println(“La media de los números pares ingresados es: “ +
sumaPar);
        System.out.println(“La media de los números impares ingresados es: “ +
sumaImpar);

    }

}
```

## Resultado Java

```
run:
Ingresa un número entero:
5
Ingresa un número entero:
7
Ingresa un número entero:
8
Ingresa un número entero:
5

...

La media de los números pares ingresados es: 3
La media de los números impares ingresados es: 2
BUILD SUCCESSFUL (total time: 16 seconds)
```

# ARREGLOS

## Ejercicio 1

En el centro meteorológico de Argentina se llevan los promedios mensuales de las lluvias caídas en las principales regiones del país. Existen 3 regiones importantes denominadas norte, centro y sur. Realice un algoritmo para calcular lo siguiente:

Promedio anual de la región Centro.

El mes con menor lluvia en la región sur.

La región con mayor lluvia.

```
1.  MES_LLUVIAS
2.  Arreglo: mes[12]
3.  Arreglo: norte[12]
4.  Arreglo: centro[12]
5.  Arreglo: sur[12]
6.  sumaCentro: como Real
7.  auxMes, auxMayor, auxMenor: como Entero
8.  region: como Caracter
9.  mes[1]← “Enero”
10. mes[2]←”Febrero”
11. mes[3]←”Marzo”
12. mes[4]←”Abril”
13. mes[5]←”Mayo”
14. mes[6]←”Junio”
15. mes[7]←”Julio”
16. mes[8]←”Agosto”
17. mes[9]←”Septiembre”
18. mes[10]←”Octubre”
19. mes[11]←”Noviembre”
20. mes[12]←”Diciembre”
21.  sumaCentro ← 0;
22.  auxMes ← 0;
23.  auxMayor ← 0;
```

```
 auxMenor ← 0;
 region ← "";
Escribir "Los porcentajes de lluvias van del 0 al 100%."
PARA i ← 1 Hasta 12 Con Paso 1 Hacer
  Escribir "Ingresa el porcentaje de lluvias para la Zona Norte:"
  Leer norte[i]
  SI auxMayor < norte[i] Entonces
    auxMayor ← norte[i]
    region ← "Norte"
  FIN SI
FINPARA
PARA i ← 1 Hasta 12 Con Paso 1 Hacer
  Escribir " Ingresa el porcentaje de lluvias para la Zona Centro:"
  Leer centro[i]
  sumaCentro ← sumaCentro + centro[i]
  SI auxMayor < centro[i] Entonces
    auxMayor ← centro[i]
    region ← "Centro"
  FINSI
FINPARA
PARA i ← 1 Hasta 12 Con Paso 1 Hacer
  Escribir " Ingresa el porcentaje de lluvias para la Zona Sur:"
  Leer sur[i]
  SI auxMenor > sur[i] Entonces
    auxMenor ← sur[i]
    auxMes ← i
  FINSI
  SI auxMayor < sur [i] Entonces
    auxMayor ← sur [i]
    region ← "Sur"
  FINSI
FINPARA
Escribir "Promedio anual de la región Centro: "
Escribir (sumaCentro / 12)
Escribir "El mes con menor lluvia de la zona Sur es: "
Escribir mes[auxMes]
Escribir "La región con más lluvia es: "
Escribir region
FIN
```

## PRUEBA DE ESCRITORIO

| MES_LLUVIAS | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| mes[ ] | norte[ ] | centro[ ] | sur[ ] | sumaCentro | auxMes | auxMayor | auxMenor | region |
| " "<br>" "<br>...<br>" "<br>" " | 0<br>0<br>...<br>0<br>0 | 0<br>0<br>...<br>0<br>0 | 0<br>0<br>...<br>0<br>0 | 0 | 0 | 0 | 0 | " " |
| "Enero"<br>"Febrero"<br>...<br>"Noviembre"<br>"Diciembre" | 0<br>15<br>18<br>...<br>20<br>25 | 10<br>25<br>30<br>...<br>0<br>24 | 30<br>75<br>80<br>...<br>1<br>12 | 300 | 2 | 80 | 0 | "Centro" |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN Pseint

```
1.  /* Realice un algoritmo para calcular lo siguiente a partir de
    lo
2.   * ingresado por el usuario:
3.   *   Promedio anual de la región Centro.
4.   *   El mes con menor lluvia en la región sur.
5.   *   La región con mayor lluvia.
6.   */
7.  Algoritmo MES_LLUVIAS
8.      DEFINIR sumaCentro Como REAL;
9.      DEFINIR auxMes, auxMayor, auxMenor Como ENTERO;
10.     DEFINIR region Como CARACTER;
11.     DIMENSION mes[12];
12.     DIMENSION norte[12];
13.     DIMENSION centro[12];
```

```
14.     DIMENSION sur[12];
15.     mes[1] ← “Enero”;
16.     mes[2] ← “Febrero”;
17.     mes[3] ← “Marzo”;
18.     mes[4] ← “Abril”;
19.     mes[5] ← “Mayo”;
20.     mes[6] ← “Junio”;
21.     mes[7] ← “Julio”;
22.     mes[8] ← “Agosto”;
23.     mes[9] ← “Septiembre”;
24.     mes[10] ← “Octubre”;
25.     mes[11] ← “Noviembre”;
26.     mes[12] ← “Diciembre”;
27.     sumaCentro ← 0;
28.     auxMes ← 0;
29.     auxMayor ← 0;
30.     auxMenor ← 0;
31.     region ← “”;
32.     ESCRIBIR “Los porcentajes de lluvias van del 0% al 100%.”;
33.     PARA i ← 1 HASTA 12 CON PASO 1 HACER
34.     .  ESCRIBIR “Ingresa el porcentaje de lluvias para
35.          la zona Norte”;
36.     .  LEER norte[ i ];
37.     .  SI (auxMayor < norte[i]) ENTONCES
38.     .  .   auxMayor ← norte[i];
39.     .  .   region ← “Norte”;
40.     .  FINSI
41.     FINPARA
42.     PARA i ← 1 HASTA 12 CON PASO 1 HACER
43.     .  ESCRIBIR “Ingresa el porcentaje de lluvias para
44.          la zona Centro”;
45.     .  LEER centro[ i ];
46.     .  sumaCentro ← sumaCentro + centro[i];
47.     .  SI (auxMayor < centro[i]) ENTONCES
48.     .  .   auxMayor ← centro[i];
49.     .  .   region ← “Centro”;
50.     .  FINSI
51.     FINPARA
52.     PARA i ← 1 HASTA 12 CON PASO 1 HACER
```

```
53.      .  ESCRIBIR "Ingresa el porcentaje de lluvias para
54.           la zona Sur";
55.      .  LEER sur[ i ];
56.      .  SI (auxMenor > sur[i]) ENTONCES
57.      .  .   auxMenor ← sur[i];
58.      .  .   auxMes ← i;
59.      .  FINSI
60.      .  SI (auxMayor < sur[i]) ENTONCES
61.      .  .   auxMayor ← sur[i];
62.      .  .   region ← "Sur";
63.      .  FINSI
64.      FINPARA
65.      ESCRIBIR "Promedio anual de la región centro:";
66.      ESCRIBIR (sumaCentro / 12);
67.      ESCRIBIR "El mes con menor lluvia de la región sur es:";
68.      ESCRIBIR mes[auxMes];
69.      ESCRIBIR "La región con más lluvia es:";
70.      ESCRIBIR region;
71. FinAlgoritmo
```

## Ejecutar Paso a Paso

Ingresamos los 12 elementos de la Zona Norte que corresponden a cada mes del año.

```
1.  *** Ejecución Iniciada. ***
2.  Los porcentajes de lluvias van del 0% al 100%.
3.  Ingresa el porcentaje de lluvias para la zona Norte:
4.  > 50
5.  Ingresa el porcentaje de lluvias para la zona Norte:
6.  > 55
7.  Ingresa el porcentaje de lluvias para la zona Norte:
8.  > 12
9.  ...
10. Ingresa el porcentaje de lluvias para la zona Norte:
11. > 10
```

Ingresamos los 12 elementos de la Zona Centro que corresponden a cada mes del año.

```
Ingresa el porcentaje de lluvias para la zona Centro:
> 80
Ingresa el porcentaje de lluvias para la zona Centro:
> 40
Ingresa el porcentaje de lluvias para la zona Centro:
> 57
...
Ingresa el porcentaje de lluvias para la zona Centro:
> 95
```

Ingresamos los 12 elementos de la Zona Sur que corresponden a cada mes del año.

```
Ingresa el porcentaje de lluvias para la zona Sur:
> 90
Ingresa el porcentaje de lluvias para la zona Sur:
> 30
Ingresa el porcentaje de lluvias para la zona Sur:
> 22
...
Ingresa el porcentaje de lluvias para la zona Sur:
> 3
```

Calcula el promedio anual de la región centro, busca el mes con menor lluvia en la Región Sur e imprime la región con más lluvia.

```
Promedio anual de la región centro:
51.333333333
El mes con menor lluvia de la región sur es:
Junio
La región con más lluvia es:
Centro
*** Ejecución Finalizada. ***
```

## Ejecutar

```
 *** Ejecución Iniciada. ***
Los porcentajes de lluvias van del 0% al 100%.
Ingresa el porcentaje de lluvias para la zona Norte:
> 50
Ingresa el porcentaje de lluvias para la zona Norte:
> 55
Ingresa el porcentaje de lluvias para la zona Norte:
> 12
...
Ingresa el porcentaje de lluvias para la zona Norte:
> 10
Ingresa el porcentaje de lluvias para la zona Centro:
> 80
Ingresa el porcentaje de lluvias para la zona Centro:
> 40
Ingresa el porcentaje de lluvias para la zona Centro:
> 57
...
Ingresa el porcentaje de lluvias para la zona Centro:
> 95
Ingresa el porcentaje de lluvias para la zona Sur:
> 90
Ingresa el porcentaje de lluvias para la zona Sur:
> 30
Ingresa el porcentaje de lluvias para la zona Sur:
> 22
...
Ingresa el porcentaje de lluvias para la zona Sur:
> 3
Promedio anual de la región centro:
51.333333333
El mes con menor lluvia de la región sur es:
Junio
La región con más lluvia es:
Centro
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
1.  /* Realice un algoritmo para calcular lo siguiente a partir de
    lo
2.   * ingresado por el usuario:
3.   *   Promedio anual de la región Centro.
4.   *   El mes con menor lluvia en la región sur.
5.   *   La región con mayor lluvia.
6.   */
7.  package Arreglos;
8.
9.  import java.util.Scanner;
10.
11. public class Mes_Lluvias {
12.
13.   public static void main(String[] args) {
14.      Scanner entrada = new Scanner(System.in);
15.      double sumaCentro = 0;
16.      int auxMes = 0;
17.      int auxMayor = 0;
18.      int auxMenor = 100;
19.      String region = "";
20.      String[] mes = {"Enero", "Febrero", "Marzo", "Abril",
21. "Mayo", "Junio", "Julio", "Agosto",
22.                "Septiembre", "Octubre", "Noviembre",
    "Diciembre"};
23.      int[] norte = new int[12];
24.      int[] centro = new int[12];
25.      int[] sur = new int[12];
26.
27.      System.out.println("Los porcentajes de lluvias van del 0%
    al 100%.");
28.      for (int i = 0; i < 12; i++) {
29.         System.out.println("Ingresa el porcentaje de lluvias de
    la zona Norte:");
30.         norte[i] = entrada.nextInt();
31.         if (auxMayor < norte[i]) {
32.            auxMayor = norte[i];
33.            region = "Zona Norte";
```

```
34.         }
35.      }
36.      for (int i = 0; i < 12; i++) {
37.         System.out.println("Ingresa el porcentaje de lluvias de
    la zona Centro:");
38.         centro[i] = entrada.nextInt();
39.         sumaCentro = sumaCentro + centro[i];
40.         if (auxMayor < centro[i]) {
41.            auxMayor = centro[i];
42.            region = "Zona Centro";
43.         }
44.      }
45.      for (int i = 0; i < 12; i++) {
46.         System.out.println("Ingresa el porcentaje de lluvias de
    la zona Sur:");
47.         sur[i] = entrada.nextInt();
48.         if (auxMenor > sur[i]) {
49.            auxMenor = sur[i];
50.            auxMes = i;
51.         }
52.         if (auxMayor < sur[i]) {
53.            auxMayor = sur[i];
54.            region = "Zona Sur";
55.         }
56.      }
57.      System.out.println("Promedio anual de la región Centro: " +
    (sumaCentro
58. / 12));
59.      System.out.println("El mes con menor lluvia de la región
    Sur es: " +
60. mes[auxMes]);
61.      System.out.println("La región con más lluvia es: " +
    region);
62.   }
63.
64. }
```

## Resultado Java

```
run:
Los porcentajes de lluvias van del 0% al 100%.
Ingresa el porcentaje de lluvias de la zona Norte:
1
Ingresa el porcentaje de lluvias de la zona Norte:
2
Ingresa el porcentaje de lluvias de la zona Norte:
3
Ingresa el porcentaje de lluvias de la zona Norte:
4
...
Ingresa el porcentaje de lluvias de la zona Centro:
13
Ingresa el porcentaje de lluvias de la zona Centro:
14
Ingresa el porcentaje de lluvias de la zona Centro:
15
Ingresa el porcentaje de lluvias de la zona Centro:
16
...
Ingresa el porcentaje de lluvias de la zona Sur:
24
Ingresa el porcentaje de lluvias de la zona Sur:
25
...
Ingresa el porcentaje de lluvias de la zona Sur:
34
Ingresa el porcentaje de lluvias de la zona Sur:
35
Promedio anual de la región Centro: 16.583333333333332
El mes con mayor lluvia de la región Sur es: Enero
La región con más lluvia es: Zona Sur
BUILD SUCCESSFUL (total time: 50 seconds)
```

## Ejercicio 2

Realice un algoritmo para calcular lo que hay que pagar por un conjunto de llamadas telefónicas. Por cada llamada se ingresa el tipo (Internacional, Nacional, Local) y la duración en minutos. El criterio que se sigue para calcular el costo de cada llamada es el siguiente:

Internacional: 3 primeros minutos $7.59, cada minuto adicional es de $3.03.

Nacional: 3 Primeros minutos $1.20, cada minuto adicional $0.48

Local: Las primeras 50 llamadas no se cobran, luego cada llamada cuesta $0.60.

```
1.  LLAMADAS
2.  internacional, nacional, local, extra: como Entero
3.  sumaInter, sumaNacio, sumaLocal: como Real
4.  sumaInter ← 0
5.  sumaNacio ← 0
6.  sumaLocal ← 0
7.  Escribir “Ingresa el número de llamadas Internacionales: “
8.  Leer internacional
9.  Arreglo llamadasInter[internacional]
10. Escribir “Ingresa el número de llamadas Nacionales: “
11. Leer nacional
12. Arreglo llamadasNacio[nacional]
13. Escribir “Ingresa el número de llamadas locales: “
14. Leer local
15. Arreglo LlamadasLocal[local]
16. PARA i ← 1 Hasta internacional Con Paso 1 Hacer
17.   Escribir “Ingresa los minutos de la llamada Internacional: “
18.   Leer llamadasInter[i]
19.   Si (3 ⩾ llamadasInter[i]) Entonces
20.     sumaInter ← sumaInter + 7.59;
21.   SiNo
22.     extra ← llamadasInter[i]- 3
23.     sumaInter ← sumaInter + 7.59 +( extra * 3.3);
24.   FINSI
25. FINPARA
```

```
Para j ← 1 Hasta nacional Con Paso 1 Hacer
  Escribir “Ingresa los minutos de la llamada Nacional: “
  Leer llamadasNacio[j]
  Si (3 ⩾ llamadasNacio[j]) Entonces
    sumaNacio ← sumaNacio + 1.20
  SINO
    extra ← llamadasNacio[j] - 3
    sumaNacio ← sumaNacio + 1.20 + (extra * 0.48)
  FINSI
FINPARA
PARA k ← 1 Hasta local Con Paso 1 Hacer
  Escribir “Ingresar los minutos de la llamada Local: “
  Leer llamadasLocal[k]
  Si (50 < llamadasLocal[k]) Entonces
    extra ← llamadasLocal[k]- 50
    sumaLocal ← sumaLocal + (extra * 0.60)
  FINSI
FINPARA
Escribir “Total a pagar por llamadas Internacionales: “
Escribir sumaInter
Escribir “Total a pagar por llamadas Nacionales: “
Escribir sumaNacio
Escribir “Total a pagar por llamadas Locales: “
Escribir sumaLocal
Escribir “Total a pagar por todas las llamadas: “
Escribir(sumaInter + sumaNacio + sumaLocal)
FIN
```

## PRUEBA DE ESCRITORIO

| LLAMADAS | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| internacional | nacional | local | extra | sumaInter | sumaNacio | sumaLocal | llamadasInter | llamadasNacio | llamadasLocal |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0<br>0<br>...<br>0<br>0 | 0<br>0<br>...<br>0<br>0 | 0<br>0<br>...<br>0<br>0 |
| 10 | 2 | 40 | | 120 | 50 | 250 | 1<br>2<br>3<br>...<br>21<br>20 | 2<br>10 | 1<br>2<br>3<br>...<br>31<br>25 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  /* Calcula lo que hay que pagar por un conjunto de llamadas
2.     telefónicas. */
3.  Algoritmo LLAMADAS
4.      DEFINIR sumaInter, sumaNacio, sumaLocal Como REAL;
5.      DEFINIR internacional, nacional, local, extra Como ENTERO;
6.      sumaInter ← 0;
7.      sumaNacio ← 0;
8.      sumaLocal ← 0;
9.      ESCRIBIR “Ingresa el número de llamadas internacionales:”;
10.     LEER internacional;
11.     DIMENSION llamadasInter[internacional];
12.     ESCRIBIR “Ingresa el número de llamadas nacionales:”;
```

```
13.     LEER nacional;
14.     DIMENSION llamadasNacio[nacional];
15.     ESCRIBIR “Ingresa el número de llamadas locales:”;
16.     LEER local;
17.     DIMENSION llamadasLocal[local];
18.     PARA i ← 1 HASTA internacional CON PASO 1 HACER
19.     .  ESCRIBIR “Ingresa los minutos de la llamada
    internacional”;
20.     .  LEER llamadasInter[i];
21.     .  SI (3 ⩾ llamadasInter[i]) ENTONCES
22.     .  .   sumaInter ← sumaInter + 7.59;
23.     .  SINO
24.     .  .   extra ← llamadasInter[i] - 3;
25.     .  .   sumaInter ← sumaInter + 7.59 + (extra * 3.3);
26.     .  FINSI
27.     FINPARA
28.     PARA j ← 1 HASTA nacional CON PASO 1 HACER
29.     .  ESCRIBIR “Ingresa los minutos de la llamada nacional”;
30.     .  LEER llamadasNacio[j];
31.     .  SI (3 ⩾ llamadasNacio[j]) ENTONCES
32.     .  .   sumaNacio ← sumaNacio + 1.20;
33.     .  SINO
34.     .  .   extra ← llamadasNacio[j] - 3;
35.     .  .   sumaNacio ← sumaNacio + 1.20 + (extra * 0.48);
36.     .  FINSI
37.     FINPARA
38.     PARA k ← 1 HASTA local CON PASO 1 HACER
39.     .  ESCRIBIR “Ingresa los minutos de la llamada nacional”;
40.     .  LEER llamadasLocal[k];
41.     .  SI (50 < llamadasLocal[k]) ENTONCES
42.     .  .   extra ← llamadasLocal[k] - 50;
43.     .  .   sumaLocal ← sumaLocal + (extra * 0.60);
44.     .  FINSI
45.     FINPARA
46.     ESCRIBIR “Total a pagar por llamadas Internacionales: “;
47.     ESCRIBIR sumaInter;
48.     ESCRIBIR “Total a pagar por llamadas Nacionales: “;
49.     ESCRIBIR sumaNacio;
50.     ESCRIBIR “Total a pagar por llamadas Locales:”;
```

```
51.     ESCRIBIR sumaLocal;
52.     ESCRIBIR "Total a pagar por todas las llamadas:";
53.     ESCRIBIR (sumaInter + sumaNacio + sumaLocal);
54. FinAlgoritmo
```

## Ejecutar Paso a Paso

Al principio, nos pide el número de llamadas Internacionales hechas, dependiendo de eso, va a consistir el tamaño del arreglo con el que se va a trabajar, y así más adelante hacer las operaciones del total en costos de llamas.

```
1.  *** Ejecución Iniciada. ***
2.  Ingresa el número de llamadas internacionales:
3.  > 2
```

Luego nos pide el número total de llamadas Nacionales hechas.

```
1.  Ingresa el número de llamadas nacionales:
2.  > 3
```

Después nos pide el número total de llamadas locales realizadas.

```
1.  Ingresa el número de llamadas locales:
2.  > 1
```

Ahora, dependiendo del número de llamadas Internacionales, nos pedirá la duración de cada llamada.

```
1.  Ingresa los minutos de la llamada internacional
2.  > 2
3.  Ingresa los minutos de la llamada internacional
4.  > 3
5.  ...
```

Seguido de eso, dependiendo del número de llamadas Nacionales, nos pedirá la duración de cada llamada.

```
Ingresa los minutos de la llamada nacional
> 2
Ingresa los minutos de la llamada nacional
> 5
...
```

Posteriormente, dependiendo del número de llamadas Locales, nos pedirá la duración de cada llamada.

```
Ingresa los minutos de la llamada local
> 5
...
```

Finalmente, se imprimen en pantalla los resultados y se finaliza el algoritmo.

```
Total a pagar por llamadas internacionales:
15.18
Total a pagar por llamadas nacionales:
5.52
Total a pagar por llamadas locales:
0
Total a pagar por todas las llamadas:
20.7
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingresa el número de llamadas internacionales:
> 2
Ingresa el número de llamadas nacionales:
> 3
```

```
Ingresa el número de llamadas locales:
> 1
Ingresa los minutos de la llamada internacional
> 2
Ingresa los minutos de la llamada internacional
> 3
Ingresa los minutos de la llamada nacional
> 2
Ingresa los minutos de la llamada nacional
> 5
Ingresa los minutos de la llamada nacional
> 5
Ingresa los minutos de la llamada local
> 5
Total a pagar por llamadas internacionales:
15.18
Total a pagar por llamadas nacionales:
5.52
Total a pagar por llamadas locales:
0
Total a pagar por todas las llamadas:
20.7
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Realice un algoritmo para calcular lo que hay que pagar por
un conjunto de llamadas
   telefónicas.  */
package Arreglos;

import java.util.Scanner;

public class Llamadas {

  public static void main(String[] args) {
     Scanner entrada = new Scanner(System.in);
     int internacional, nacional, local, extra;
```

```
12.      double sumaInter = 0, sumaNacio = 0, sumaLocal = 0;
13.
14.      System.out.println(“Ingresa el número de llamadas
    internacionales: “);
15.      internacional = entrada.nextInt();
16.      int[] llamadaInter = new int[internacional];
17.      System.out.println(“Ingresa el número de llamadas
    nacionales: “);
18.      nacional = entrada.nextInt();
19.      int[] llamadaNacio = new int[nacional];
20.      System.out.println(“Ingresa el número de llamadas locales:
    “);
21.      local = entrada.nextInt();
22.      int[] llamadaLocal = new int[local];
23.
24.      for (int i = 0; i < internacional; i++) {
25.         System.out.println(“Ingresa los minutos de la llamada
    internacional: “);
26.         llamadaInter[i] = entrada.nextInt();
27.         if (3 ⩾ llamadaInter[i]) {
28.            sumaInter = sumaInter + 7.59;
29.         } else {
30.            extra = llamadaInter[i] - 3;
31.            sumaInter += 7.59 + (extra * 3.03);
32.         }
33.      }
34.      for (int j = 0; j < nacional; j++) {
35.         System.out.println(“Ingresa los minutos de la llamada
    Nacional: “);
36.         llamadaNacio[j] = entrada.nextInt();
37.         if (3 ⩾ llamadaNacio[j]) {
38.            sumaNacio = sumaNacio + 1.20;
39.         } else {
40.            extra = llamadaNacio[j] - 3;
41.            sumaNacio += 1.20 + (extra * 0.48);
42.         }
43.      }
44.      for (int i = 0; i < local; i++) {
45.         System.out.println(“Ingresa los minutos de la llamada
    Local: “);
46.         llamadaLocal[i] = entrada.nextInt();
```

```
47.         if (50 < llamadaLocal[i]) {
48.            extra = llamadaLocal[i] - 50;
49.            sumaLocal += (extra * 0.60);
50.         }
51.      }
52.
53.      System.out.println(“Total a pagar por llamadas
    Internacionales: “ +
54. sumaInter);
55.      System.out.println(“Total a pagar por llamadas Nacionales:
    “ +
56. sumaNacio);
57.      System.out.println(“Total a pagar por llamadas Locales: “ +
    sumaLocal);
58.      System.out.println(“Total a pagar por todas las llamadas: “
    + (sumaInter +
59. sumaNacio + sumaLocal));
60.     }
61.
62. }
```

## Resultado Java

```
1.  run:
2.  Ingresa el número de llamadas internacionales:
3.  2
4.  Ingresa el número de llamadas nacionales:
5.  3
6.  Ingresa el número de llamadas locales:
7.  1
8.  Ingresa los minutos de la llamada internacional:
9.  2
10. Ingresa los minutos de la llamada internacional:
11. 3
12. Ingresa los minutos de la llamada Nacionales:
13. 2
14. Ingresa los minutos de la llamada Nacionales:
15. 5
16. Ingresa los minutos de la llamada Nacionales:
```

```
5
Ingresa los minutos de la llamada Locales:
5
Total a pagar por llamadas Internacionales: 15.18
Total a pagar por llamadas Nacionales: 5.5200000000000005
Total a pagar por llamadas Locales: 0.0
Total a pagar por todas las llamadas: 20.7
BUILD SUCCESSFUL (total time: 43 seconds)
```

## Ejercicio 3

Se tienen los sueldos de un grupo de 5 empleados de una empresa y necesitamos saber cuántos de estos empleados tienen un sueldo superior al promedio del grupo.

```
1.  SUELDOS
2.  sumaSueldo , sumaEmp: como Entero
3.  media: como Real
4.  arreglo empleados [5]
5.  sumaSueldo ← 0
6.  sumaEmp ← 0
7.  PARA i ←1 Hasta 5 Con Paso 1 Hacer
8.    Escribir “Ingrese el sueldo del trabajador: “
9.    Leer empleados[i]
10.   sumaSueldo = sumaSueldo + empleados[i]
11. FINPARA
12. media = sumaSueldo / 5
13. PARA j ← 1 Hasta 5 Con Paso 1 Hacer
14.   SI (empleados[j] > media) Entonces
15.     sumaEmp = sumaEmp + 1
16.   FINSI
17. FINPARA
18. Escribir “Número de trabajadores superiores al promedio: “
19. Escribir sumaEmp
20. FIN
```

## PRUEBA DE ESCRITORIO

| SUELDOS | | | |
|---|---|---|---|
| **sumaSueldo** | **sumaEmp** | **media** | **empleados** |
| 0 | 0 | 0 | 0<br>0<br>0<br>0<br>0 |
| 1500 | 2 | 300 | 300<br>200<br>500<br>400<br>100 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  /* Calcula lo que hay que pagar por un conjunto de llamadas
2.     telefónicas. */
3.  Algoritmo SUELDOS
4.      DEFINIR sumaSueldo, sumaEmp Como ENTERO;
5.      DEFINIR media Como REAL;
6.      DIMENSION empleados[5];
7.      sumaSueldo ← 0;
8.      sumaEmp ← 0;
9.      PARA i ← 1 HASTA 5 CON PASO 1 HACER
10.     .  ESCRIBIR “Ingrese el sueldo del trabajador:”;
11.     .  LEER empleados[i];
12.     .  sumaSueldo ← sumaSueldo + empleados[i];
13.     FINPARA
14.     media ← sumaSueldo / 5;
15.     PARA j ← 1 HASTA 5 CON PASO 1 HACER
16.     .  SI (empleados[j] > media) ENTONCES
17.     .  .   sumaEmp ← sumaEmp + 1;
18.     .  FINSI
19.     FINPARA
20.     ESCRIBIR “Número de trabajadores superiores al promedio: “;
21.     ESCRIBIR sumaEmp;
```

## Ejecutar Paso a Paso

Como primer dato pedido, el algoritmo solicita introducir el sueldo del primer trabajador.

```
*** Ejecución Iniciada. ***
Ingrese el sueldo del trabajador:
> 100
Y así, sucesivamente hasta completar los 5 sueldos requeridos.
Ingrese el sueldo del trabajador:
> 200
Ingrese el sueldo del trabajador:
> 300
Ingrese el sueldo del trabajador:
> 400
Ingrese el sueldo del trabajador:
> 500
```

Y así, finalmente, hacer los cálculos para conocer cuántos empleados tienen un sueldo superior al promedio de los 5 empleados.

```
Número de trabajadores superiores al promedio:
2
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el sueldo del trabajador:
> 100
Ingrese el sueldo del trabajador:
> 200
Ingrese el sueldo del trabajador:
> 300
Ingrese el sueldo del trabajador:
> 400
Ingrese el sueldo del trabajador:
> 500
```

```
Número de trabajadores superiores al promedio:
2
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
// Define cuantos de los 5 empleados ingresados tiene un sueldo mayor a la media.
package Arreglos;

import java.util.Scanner;

public class sueldos {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int sumaSueldo = 0, sumaEmp = 0;
        double media;

        int[] empleados = new int[5];

        for (int i = 0; i < 5; i++) {
            System.out.println("Ingresa el sueldo del trabajador: ");
            empleados[i] = entrada.nextInt();
            sumaSueldo = sumaSueldo + empleados[i];
        }
        media = sumaSueldo / 5;
        for (int i = 0; i < 5; i++) {
            if (empleados[i] > media) {
                sumaEmp += 1;
            }
        }

        System.out.println("Número de trabajadores superiores al promedio: "
+ sumaEmp);
    }

}
```

## Resultado Java

```
run:
Ingresa el sueldo del trabajador:
100
Ingresa el sueldo del trabajador:
200
Ingresa el sueldo del trabajador:
300
Ingresa el sueldo del trabajador:
400
Ingresa el sueldo del trabajador:
500
Número de trabajadores superiores al promedio: 2
BUILD SUCCESSFUL (total time: 5 seconds)
```

## Ejercicio 4

En un arreglo unidimensional se ha almacenado el número total de toneladas de cereales cosechadas durante cada mes del año anterior. Construye un algoritmo que proporcione la siguiente información:

El promedio anual de toneladas de cosechas

Cuántos meses tuvieron una cosecha superior al promedio anual.

Cuántos meses tuvieron una cosecha inferior al promedio anual.

```
1.  CEREALES
2.  sumaTonelada, mayorMedia, menorMedia: Como Entero
3.  media: Como Real
4.  Arreglo: toneladas[12]
5.  PARA i ← 1 Hasta 12 Con Paso 1 Hacer
6.    Escribir “Ingrese las toneladas del mes:”
7.    Leer toneladas[i]
8.    sumaTonelada ← sumaTonelada + toneladas[i]
9.  FINPARA
10. media ← sumaTonelada / 12
11. PARA i ← 1 Hasta 12 Con Paso 1 Hacer
12.   SI toneladas[i] > media Entonces
13.     mayorMedia ← mayorMedia + 1
14.   SINO
15.     menorMedia ← menorMedia + 1
16.   FINSI
17. FINPARA
18. Escribir “Promedio de toneladas en el año: “
19. Escribir media
20. Escribir “Superiores a la media: “
21. Escribir mayorMedia
22. Escribir “Inferiores a la media: “
23. Escribir menorMedia
24. FIN
```

## PRUEBA DE ESCRITORIO

| CEREALES | | | | |
|---|---|---|---|---|
| toneladas [] | sumaTonelada | mayorMedia | menorMedia | media |
| 0<br>0<br>...<br>0<br>0 | 0 | 0 | 0 | 0 |
| 1<br>2<br>...<br>11<br>12 | 78 | 6 | 6 | 6.5 |

## ALGORITMO EN PseInt
## RUEBA DE ESCRITORIO EN PseInt

```
1.  /* Construye un algoritmo que proporcione la siguiente
    información:
2.   *  El promedio anual de toneladas de cosechas
3.   *  Cuántos meses tuvieron una cosecha superior al promedio
    anual.
4.   *  Cuántos meses tuvieron una cosecha inferior al promedio
    anual.
5.   */
6.  Algoritmo CEREALES
7.      DEFINIR sumaTonelada, mayorMedia, menorMedia Como ENTERO;
8.      DEFINIR media Como REAL;
9.      DIMENSION toneladas[12];
10.     PARA i ← 1 HASTA 12 CON PASO 1 HACER
11.     .  ESCRIBIR “Ingrese las toneladas del mes:”;
12.     .  LEER toneladas[i];
13.     .  sumaTonelada ← sumaTonelada + toneladas[i];
14.     FINPARA
15.     media ← sumaTonelada / 12;
16.     PARA j ← 1 HASTA 12 CON PASO 1 HACER
17.     .  SI (toneladas[j] > media) ENTONCES
18.     .  .   mayorMedia ← mayorMedia + 1;
19.     .  SINO
```

```
20.      .  .   menorMedia ← menorMedia + 1;
21.      .  FINSI
22.      FINPARA
23.      ESCRIBIR "Promedio de toneladas en el año: ";
24.      ESCRIBIR media;
25.      ESCRIBIR "Superiores a la media: ";
26.      ESCRIBIR mayorMedia;
27.      ESCRIBIR "Inferiores a la media: ";
28.      ESCRIBIR menorMedia;
29.  FinAlgoritmo
```

## Ejecutar Paso a Paso

Una vez iniciado el programa, pedirá las toneladas registradas desde el mes 1 (enero) hasta el mes 12 (diciembre).

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese las toneladas del mes:
3.  > 1
```

Repite la operación las 12 veces requeridas por el programa.

```
1.  Ingrese las toneladas del mes:
2.  > 2
3.  Ingrese las toneladas del mes:
4.  > 3
5.  ...
6.  Ingrese las toneladas del mes:
7.  > 11
8.  Ingrese las toneladas del mes:
9.  > 12
```

Por último, calcula el promedio de las toneladas del año entero, así como los meses que sobrepasan el promedio y los que están por debajo del promedio.

```
Promedio de toneladas en el año:
6.5
Superiores a la media: 6
Inferiores a la media: 6
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese las toneladas del mes:
> 1
Ingrese las toneladas del mes:
> 2
Ingrese las toneladas del mes:
> 3
...
Ingrese las toneladas del mes:
> 11
Ingrese las toneladas del mes:
> 12
Promedio de toneladas en el año:
6.5
Superiores a la media: 6
Inferiores a la media: 6
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Construye un algoritmo que proporcione la siguiente
información:
 * El promedio anual de toneladas de cosechas
 * Cuántos meses tuvieron una cosecha superior al promedio
anual.
 * Cuántos meses tuvieron una cosecha inferior al promedio
anual.
 */
package Arreglos;
```

```
7.  import java.util.Scanner;
8.
9.  public class Cereales {
10.
11.     public static void main(String[] args) {
12.         Scanner entrada = new Scanner(System.in);
13.         int sumaTonelada = 0;
14.         int mayorMedia = 0;
15.         int menorMedia = 0;
16.         double media;
17.         int[] toneladas = new int[12];
18.         for (int i = 0; i < 12; i++) {
19.             System.out.println("Ingrese las toneladas del
    mes:");
20.             toneladas[i] = entrada.nextInt();
21.             sumaTonelada = sumaTonelada + toneladas[i];
22.         }
23.         media = sumaTonelada / 12;
24.         for (int i = 0; i < 12; i++) {
25.             if (toneladas[i] > media) {
26.                 mayorMedia = mayorMedia + 1;
27.             } else {
28.                 menorMedia = menorMedia + 1;
29.             }
30.         }
31.         System.out.println("Promedio de toneladas en el año: " +
    media);
32.         System.out.println("Superiores a la media: " +
    mayorMedia);
33.         System.out.println("Inferiores a la media: " +
    menorMedia);
34.     }
35. }
```

## Resultado Java

```
1.  run:
2.  Ingrese las toneladas del mes:
3.  1
```

```
4.  Ingrese las toneladas del mes:
5.  2
6.  Ingrese las toneladas del mes:
7.  3
8.  ...
9.  Ingrese las toneladas del mes:
10. 10
11. Ingrese las toneladas del mes:
12. 11
13. Ingrese las toneladas del mes:
14. 12
15. Promedio de toneladas en el año: 6.0
16. Superiores a la media: 6
17. Inferiores a la media: 6
18. BUILD SUCCESSFUL (total time: 11 seconds)
```

## Ejercicio 5

Realice un algoritmo con arreglos unidimensionales para las ventas de todo un mes. La tienda desea conocer aquellas ventas de mayores a $ 1000 y aquéllas que son entre $800 y $1000 y las que son inferiores a $800.

```
1.  VENTAS
2.  cantidad, mayores, entre, menores: como Entero
3.  Escribir “Ingresa la cantidad de ventas del mes: “
4.  Leer cantidad
5.  Arreglo ventas[cantidad]
6.  PARA i ← 1 Hasta cantidad Con Paso 1 Hacer
7.    Escribir “Ingresa el monto de la venta: “
8.    Leer ventas[i]
9.    SI ventas[i] < 1000 Entonces
10.     SI ventas[i] ⩾ 800 Entonces
11.       entre ← entre +1
12.     SINO
13.       menores ← menores + 1
14.     FINSI
15.   SINO
16.     mayores ← mayores +1
17.   FINSI
18. FINPARA
19. Escribir “Ventas mayores a $1000:  “
20. Escribir mayores
21. Escribir “Ventas entre $1000 y $800: “
22. Escribir entre
23. Escribir “Ventas menores a $800: “
24. Escribir menores
25. FIN
```

## PRUEBA DE ESCRITORIO

| VENTAS | | | | |
|---|---|---|---|---|
| cantidad | ventas | entre | menores | mayores |
| 0 | 0<br>0<br>...<br>0<br>0 | 0 | 0 | 0 |
| 4 | 100<br>200<br>3000<br>4000 | 0 | 2 | 2 |

## ALGORITMO EN PseInt
## PRUEBA DE ESCRITORIO EN PseInt

```
1.  /* Realice un algoritmo con arreglos unidimensionales para las
    ventas
2.   * de todo un mes. La tienda desea conocer aquellas ventas de
    mayores
3.   * a $ 1000 y aquellas que son entre $800 y $1000 y las que son
4.     inferiores a $800. */
5.  Algoritmo VENTAS
6.      DEFINIR cantidad, mayores, entre, menores Como ENTERO;
7.      ESCRIBIR “Ingrese la cantidad de ventas del mes:”;
8.      LEER cantidad;
9.      DIMENSION ventas[cantidad];
10.     PARA i ← 1 HASTA cantidad CON PASO 1 HACER
11.     .  ESCRIBIR “Ingrese el monto de la venta:”;
12.     .  LEER ventas[i];
13.     .  SI (ventas[i] ⩽ 1000) ENTONCES
14.     .  .  SI (ventas[i] ⩾ 800) ENTONCES
15.     .  .  .  entre ← entre + 1;
16.     .  .  SINO
17.     .  .  .  menores ← menores + 1;
18.     .  .  FINSI
19.     .  SINO
20.     .  .   mayores ← mayores + 1;
```

```
21.     .  FINSI
22.     FINPARA
23.     ESCRIBIR “Ventas mayores a $1000: “;
24.     ESCRIBIR mayores;
25.     ESCRIBIR “Ventas entre $1000 y $800: “;
26.     ESCRIBIR entre;
27.     ESCRIBIR “Ventas menores a $800: “;
28.     ESCRIBIR menores;
29. FinAlgoritmo Ejecutar Paso a Paso
```

El programa inicia y solicita al usuario la cantidad de ventas realizadas durante todo el mes.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el total de ventas del mes:
3.  > 3
```

Continua con la solicitud de los montos de las ventas realizadas en el mes

```
1.  Ingrese el monto de la venta:
2.  > 500
3.  Ingrese el monto de la venta:
4.  > 3500
5.  Ingrese el monto de la venta:
6.  > 1000
```

Se hacen las operaciones correspondientes para sacar los montos mayores a 1000, los montos entre 1000 y 800, para finalmente acabar con los menores a 800.

```
1.  Ventas mayores a $1000:
2.  1
3.  Ventas entre $1000 y $800:
4.  1
5.  Ventas menores a $800:
6.  1
7.  *** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el total de ventas del mes:
> 3
Ingrese el monto de la venta:
> 500
Ingrese el monto de la venta:
> 3500
Ingrese el monto de la venta:
> 1000
Ventas mayores a $1000:
1
Ventas entre $1000 y $800:
1
Ventas menores a $800:
1
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/* Realice un algoritmo con arreglos unidimensionales para las ventas de todo un mes.
 * La tienda desea conocer aquellas ventas de mayores a $ 1 000 y aquellas que son
 * entre $800 y $1000 y las que son inferiores a $800.
 */
package Arreglos;

import java.util.Scanner;

public class Ventas {

    public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int mayores = 0, entre = 0, menores = 0;
        int cantidad;
        System.out.println(“Ingresa el total de ventas del mes:”);
        cantidad = entrada.nextInt();
```

```
        int[] ventas = new int[cantidad];

        for (int i = 0; i < cantidad; i++) {
            System.out.println("Ingresa el monto de la venta:");
            ventas[i] = entrada.nextInt();

            if (ventas[i] ⩽ 1000) {
                if (ventas[i] ⩾ 800) {
                    entre += 1;
                } else {
                    menores += 1;
                }
            } else {
                mayores += 1;
            }
        }
        System.out.println("Ventas mayores a $1000 :" +
mayores);
        System.out.println("Ventas entre $1000 y $800: " +
entre);
        System.out.println("Ventas menores a $800: " + menores);
    }
}
```

## Resultado Java

```
run:
Ingresa el total de ventas del mes:
3
Ingresa el monto de la venta:
500
Ingresa el monto de la venta:
3500
Ingresa el monto de la venta:
1000
Ventas mayores a $1000 :1
Ventas entre $1000 y $800: 1
Ventas menores a $800: 1
BUILD SUCCESSFUL (total time: 6 seconds)
```

## Ejercicio 6

Hacer la suma de dos arreglos unidimensionales, elemento a elemento y almacenar el resultado en un tercer arreglo.

```
1.  SUMA_ARREGLOS
2.   tamanio: como real
3.   Escribir “Ingresa el número de elementos del arreglo: “
4.   Leer tamanio
5.   Arreglo arreglo1[tamanio]
6.   Arreglo arreglo2[tamanio]
7.   Arreglo arreglo3[tamanio]
8.   PARA i ←1 Hasta tamanio Con Paso 1 Hacer
9.     Escribir “Ingresa el valor del primer arreglo: “
10.    Leer arreglo1[i]
11.    Escribir “Ingresa el valor del segundo arreglo: “
12.    Leer arreglo2[i]
13.    arreglo3[i]← arreglo1[i] + arreglo2[i]
14.  FINPARA
15.  PARA i ← 1 Hasta tamanio Con Paso 1 Hacer
16.   Escribir “La suma de la posición “ ,i, “ es:  “ ,arreglo3[i]
17.  FINPARA
18. FIN
```

### PRUEBA DE ESCRITORIO

| SUMA_ARREGLOS | | | |
|---|---|---|---|
| tamanio | arreglo1 | arreglo2 | arreglo3 |
| 0 | 0 | 0 | 0 |
| 2 | 0<br>0 | 0<br>0 | 0<br>0 |
| 2 | 1<br>2 | 3<br>4 | 4<br>6 |

**ALGORITMO EN PseInt**
**PRUEBA DE ESCRITORIO EN PseInt**

```
1.  /* Realice la suma de dos arreglos unidimensionales, elemento a
2.   * elemento y almacenar el resultado en un tercer arreglo.
3.   */
4.  Algoritmo SUMA_ARREGLOS
5.      DEFINIR tamanio Como ENTERO;
6.      ESCRIBIR “Ingrese el número de elementos del arreglo:”;
7.      LEER tamanio;
8.      DIMENSION arreglo1 [tamanio];
9.      DIMENSION arreglo2 [tamanio];
10.     DIMENSION arreglo3 [tamanio];
11.     PARA i ← 1 HASTA tamanio CON PASO 1 HACER
12.     .  ESCRIBIR “Ingrese el valor del primer arreglo:”;
13.     .  LEER arreglo1[i];
14.     .  ESCRIBIR “Ingrese el valor del segundo arreglo:”;
15.     .  LEER arreglo2[i];
16.     .  arreglo3[i] ← arreglo1[i] + arreglo2[i];
17.     FINPARA
18.     PARA i ← 1 HASTA tamanio CON PASO 1 HACER
19.     .  ESCRIBIR “La suma de los valores del primer y segundo
20.     .    arreglo es:”;
21.     .   ESCRIBIR arreglo3[i];
22.     FINPARA
23. FinAlgoritmo
```

## Ejecutar Paso a Paso

Se define el tamaño del arreglo.

```
1.  *** Ejecución Iniciada. ***
2.  Ingrese el tamaño de elementos del arreglo:
3.  > 2
```

Luego se introduce el valor para la posición 1 del arreglo 1.

```
Ingrese el valor del primer arreglo:
> 3
```

Luego se introduce el valor para la posición 1 del arreglo 2.

```
Ingrese el valor del segundo arreglo:
> 2
```

Y así sucesivamente, hasta completar el tamaño ingresado.

```
Ingrese el valor del primer arreglo:
> 4
Ingrese el valor del segundo arreglo:
> 3
...
```

Realiza la suma de las posiciones 1 y 2, para luego mostrarlas en pantalla.

```
La suma de los valores del primer y segundo arreglo es:
5
La suma de los valores del primer y segundo arreglo es:
7
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
Ingrese el tamaño de elementos del arreglo:
> 2
Ingrese el valor del primer arreglo:
> 3
Ingrese el valor del segundo arreglo:
```

```
> 2
Ingrese el valor del primer arreglo:
> 4
Ingrese el valor del segundo arreglo:
> 3
La suma de los valores del primer y segundo arreglo es:
5
La suma de los valores del primer y segundo arreglo es:
7
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/*Hacer la suma de dos arreglos unidimensionales elemento a elemento y almacenar
 el resultado en un tercer arreglo
*/
package Arreglos;

import java.util.Scanner;

public class Suma_Arreglos {

   public static void main(String[] args) {
        Scanner entrada = new Scanner(System.in);
        int tamanio;
        System.out.println("Ingresa el número de elementos del arreglo: ");
        tamanio = entrada.nextInt();
        int[] arreglo1 = new int[tamanio];
        int[] arreglo2 = new int[tamanio];
        int[] arreglo3 = new int[tamanio];

        for (int i = 0; i < tamanio; i++) {
            System.out.println("Ingresa el valor del primer arreglo: ");
            arreglo1[i] = entrada.nextInt();
            System.out.println("Ingresa el valor del segundo arreglo: ");
            arreglo2[i] = entrada.nextInt();
```

```
            arreglo3[i] = arreglo1[i] + arreglo2[i];
        }

        for (int i = 0; i < tamanio; i++) {
            System.out.println("La suma de los valores del primer y segundo arreglo es: "
+ arreglo3[i]);
        }
    }

}
```

## Resultado Java

```
run:
Ingresa el número de elementos del arreglo:
2
Ingresa el valor del primer arreglo:
3
Ingresa el valor del segundo arreglo:
2
Ingresa el valor del primer arreglo:
4
Ingresa el valor del segundo arreglo:
3
La suma de los valores del primer y segundo arreglo es: 5
La suma de los valores del primer y segundo arreglo es: 7
BUILD SUCCESSFUL (total time: 31 seconds)
```

# Ejercicio 7

Construye un arreglo unidimensional que calcule la suma de los números pares comprendidos entre 50 y 100.

```
1.  SUMA_PAR
2.   sumaPar, contador Como Entero
3.   Arreglo: pares[26]
4.   contador ← 0;
5.   PARA i ← 50 Hasta 100 Con Paso 1 Hacer
6.     SI i % 2 == 0 Entonces
7.       pares[contador] ← i
8.       sumaPar ← sumaPar + i
9.       contador ← contador + 1
10.    FINSI
11.  FINPARA
12.  Escribir "La suma de los números pares es: "
13.  Escribir sumaPar
14. FIN
```

## PRUEBA DE ESCRITORIO

| SUMA_PAR | | |
|---|---|---|
| sumaPar | contador | pares [ ] |
| 0 | 0 | 0<br>0<br>...<br>0<br>0 |
| 1950 | 25 | 50<br>52<br>...<br>98<br>100 |

**ALGORITMO EN PseInt**
**PRUEBA DE ESCRITORIO EN PseInt**

```
1.  // Calcule la suma de los números pares comprendidos entre 50 y
    100.
2.  Algoritmo SUMA_PAR
3.      DEFINIR sumaPar, contador Como ENTERO;
4.      DIMENSION pares [26];
5.      PARA i ← 50 HASTA 100 CON PASO 1 HACER
6.      .  SI (i MOD 2 = 0) ENTONCES
7.      .  .  pares[contador] ← i;
8.      .  .  sumaPar ← sumaPar + i;
9.      .  .  contador ← contador + 1;
10.     .  FINSI
11.     ESCRIBIR "La suma de números pares es:";
12.     ESCRIBIR sumaPar;
13.     FINPARA
14. FinAlgoritmo
```

## Ejecutar Paso a Paso

Se inicia el programa y se realizan las operaciones necesarias para mostrar los valores pares entre 50 y 100. El programa muestra en pantalla los números, además de la suma de estos.

```
1.  *** Ejecución Iniciada. ***
2.  50
3.  52
4.  54
5.  56
6.  58
7.  ...
8.  90
9.  92
10. 94
11. 96
12. 98
```

```
100
La suma de los números pares es:
1950
*** Ejecución Finalizada. ***
```

## Ejecutar

```
*** Ejecución Iniciada. ***
50
52
54
56
58
...
90
92
94
96
98
100
La suma de los números pares es:
1950
*** Ejecución Finalizada. ***
```

## ALGORITMO EN NETBEANS: JAVA

```
/*Construye un arreglo unidimensional que calcule la suma de los
números pare
  comprendidos entre 50 y 100.
 */
package Arreglos;

public class Suma_Par {

    public static void main(String[] args) {
        int[] pares = new int[26];
```

```
10.         int sumaPar = 0;
11.         int contador = 0;
12.
13.         for (int i = 50; i < 101; i++) {
14.
15.             if (i % 2 == 0) {
16.                 pares[contador] = i;
17.                 sumaPar += i;
18.                 contador++;
19.                 System.out.println(i);
20.             }
21.         }
22.         System.out.println("La suma de números impares es: " +
    sumaPar);
23.     }
24.
25. }
```

## Resultado Java

```
1.  run:
2.  50
3.  52
4.  54
5.  56
6.  58
7.  ...
8.  90
9.  92
10. 94
11. 96
12. 98
13. 100
14. La suma de números impares es: 1950
15. BUILD SUCCESSFUL (total time: 0 seconds)
```

# Referencias bibliográficas


CORREA U, Guillermo. Desarrollo de Algoritmos y sus aplicaciones. ISBN: 958-600-109-1. Editorial Mc-Graw Hill.

FARRELL, Joyce (2013). Introducción a la programación. ISBN: 978-1-133-52651-3. Editorial Thomson.

GENOVA, Miguel (1991). Introducción a la Ingeniería de Sistemas. Universidad Nacional Abierta de Venezuela. UNA.

GRECH, Pablo (2001). Introducción a la Ingeniería, Un enfoque a través de Diseño. Segunda edición. Editorial PEARSON.

JOYANES A Luis (1990). Problemas de metodología de la programación. ISBN: 9788476154625. Editorial Mc-Graw Hill.

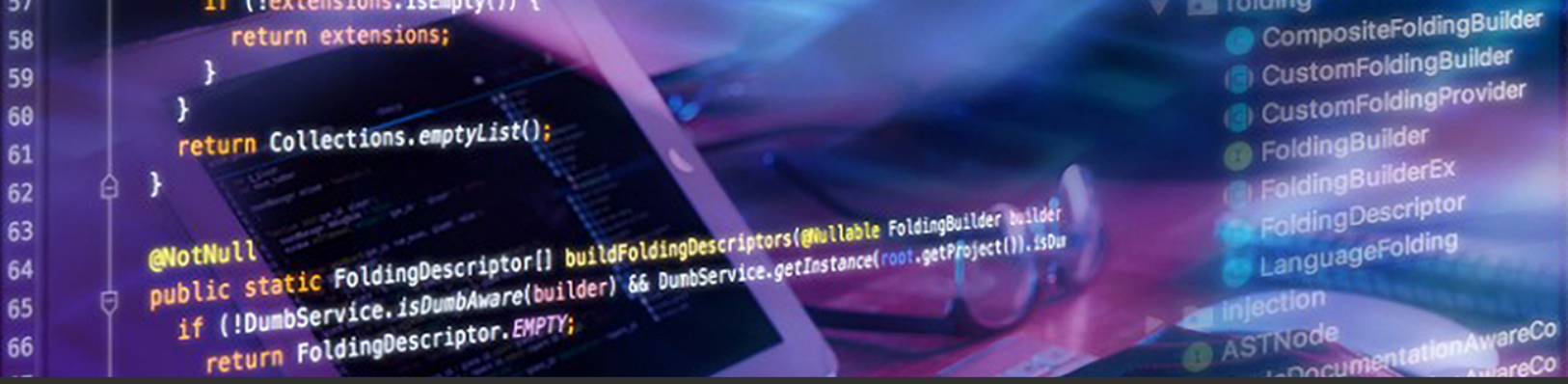
JOYANES A Luis, Lucas Sánchez & Ignacio Zahonero (2007). Estructura de datos C+. Editorial Mc-Graw Hill Interamericana de España. S. A. U. Pp 15, España.

LONG, Larry (1995). Introducción a las computadoras y al procesamiento de información. Prentice Hall Hispanoamericana. ISBN: 9789688804698.

NORTON, Peter (2006). Introducción a la Informática. Editorial Mc-Graw Hill.

SANDERS, Donald (1992). Informática Presente y Futuro, 3a. ed. Mc-Graw-Hill.

SHANKER, Stuart. (1987). "Wittgenstein versus Turing on nature of Church's thesis". Notre Dame Journal of Formal Logic, Vol. 28, pp. 615-649.




Mientras se iniciaban los trabajos de este libro, en el proceso de revisión de la literatura y el estado del arte, nos percatamos de la gran variedad de autores y referencias que existen sobre el tema. Sin embargo, la idea concebida para abordarlo era diferente, sin dejar de mencionar que las descripciones problemáticas de las instrucciones son transcripciones de diversos autores.

La idea inicial de este libro nace, a partir de un grupo de profesores que impartimos las asignaturas de introducción a la programación y programación en Java, quienes somos parte del Cuerpo Académico “Aplicación y enseñanza de la ingeniería de software” en la Licenciatura en Ingeniería de Software. Nos percatamos de la necesidad de crear y compilar una serie de ejercicios prácticos, básicamente un problemario y en este sentido, una inspiración para ello fue el autor reconocido Osvaldo Cairó con su Metodología de la Programación 1 , así como diversos compendios de autores de la web.

9 786079 011994

HESS grupo editorial

Universidad Veracruzana

ISBN: 978-607-9011-99-4