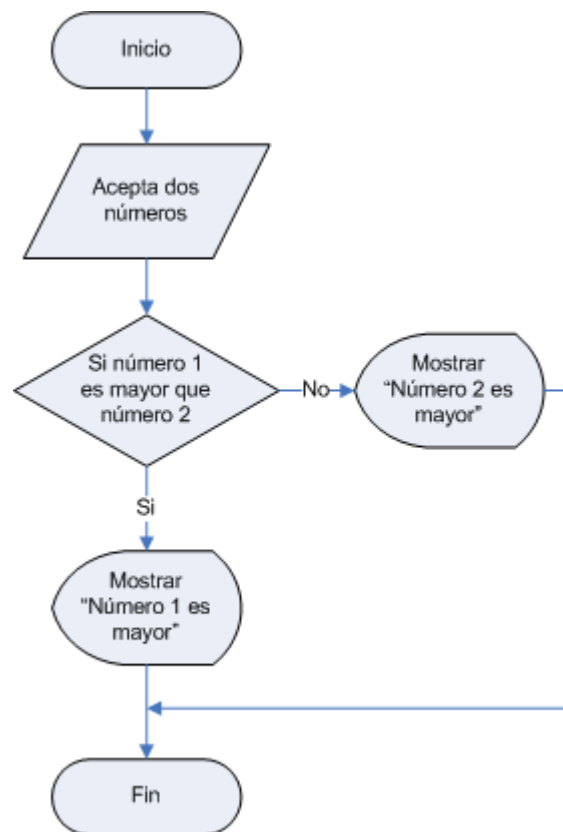


# Java 2 v5.0

## Fundamentos de Programación



Eric Gustavo Coronel Castillo

## **Java 2 v5.0 – Fundamentos de Programación**

Esta obra es de uso exclusivo del CEPS-UNI para el curso  
Java 2 – Fundamentos del Lenguaje de Programación.

Derechos Reservados © Junio-2006 Eric Gustavo Coronel Castillo  
Segunda Edición

# Fundamentos de Programación con Java 2

## Indice

---

### Lección 01

Algoritmos.....	2
¿Qué es un Algoritmo? .....	2
Características de un Algoritmo.....	2
Niveles de un Algoritmo.....	4
Representación de Algoritmos .....	5
Diagramas de Flujo.....	6
Definición.....	6
Símbolos .....	6
Ejemplos.....	8
Reglas de un Diagrama de Flujo.....	11
Ventajas de los Diagramas de Flujo .....	11
Desventajas de los Diagramas de Flujo.....	12
Tablas de Decisiones.....	13
Definición.....	13
Ejemplo .....	14
Ventajas de las Tablas de Decisiones .....	15
Desventajas de las Tablas de Decisiones .....	16
Pseudocódigo .....	16
Definición.....	16
Ejemplos.....	17
Ventajas del Pseudocódigo.....	17
Limitaciones del Pseudocódigo.....	17
Ejercicios .....	18
Ejercicio 01 .....	18
Ejercicio 02 .....	18
Ejercicio 03 .....	18
Ejercicio 04 .....	19
Ejercicio 05 .....	19
Ejercicio 06 .....	19
Ejercicio 07 .....	19

## Lección 02

Proceso de Instalación .....	22
Archivo de Instalación.....	22
Proceso de Instalación.....	23
La Carpeta bin.....	28
Probando Java .....	29
Carpeta de Trabajo .....	29
Establecer la Variable de Entorno PATH.....	29
Ejecutando el Primer Programa.....	31

## Lección 03

Introducción a Variables .....	34
Tipos de Datos .....	35
Declaración de Variables .....	36
Asignación de Valores a Variables .....	38
Asignación Directa .....	38
Instrucción de Lectura .....	39
Operadores.....	40
Operadores Aritméticos .....	40
Operadores Relacionales .....	41
Operadores Lógicos.....	42
Precedencia de Operadores.....	43
Ejemplos .....	46
Ejemplo 1 .....	46
Ejemplo 2 .....	47
Ejemplo 3 .....	48
Ejemplo 4 .....	50
Ejercicios .....	52
Ejercicio 1 .....	42
Ejercicio 2 .....	52
Ejercicio 3 .....	52
Ejercicio 4 .....	52
Ejercicio 5 .....	52
Ejercicio 6 .....	52

## Lección 04

Estructura: if .....	54
Estructuras if Simple .....	54
Estructura if...else .....	57
Estructura if...else Anidada .....	61
Estructura: switch...case .....	65
Ejercicios .....	69
Ejercicio 1 .....	69
Ejercicio 2 .....	69
Ejercicio 3 .....	69
Ejercicio 4 .....	69
Ejercicio 5 .....	69
Ejercicio 6 .....	69
Ejercicio 7 .....	69
Ejercicio 8 .....	70
Ejercicio 9 .....	70
Ejercicio 10 .....	70
Ejercicio 11 .....	70
Ejercicio 12 .....	71
Ejercicio 13 .....	71
Ejercicio 14 .....	71

## Lección 05

Estructura: while.....	74
Ejemplo 1 .....	75
Ejemplo 2 .....	78
Estructura: for .....	81
Ejemplo 4 .....	82
Ejercicios Propuestos .....	85
Ejercicio 1 .....	85
Ejercicio 2 .....	85
Ejercicio 3 .....	85
Ejercicio 4 .....	85
Ejercicio 5 .....	85
Ejercicio 6 .....	85
Ejercicio 7 .....	85
Ejercicio 8 .....	85
Ejercicio 9 .....	85
Ejercicio 10 .....	86
Ejercicio 11 .....	86
Ejercicio 12 .....	86
Ejercicio 13 .....	86
Ejercicio 14 .....	86
Ejercicio 15 .....	86
Ejercicio 16 .....	86
Ejercicio 17 .....	86
Ejercicio 18 .....	87
Ejercicio 19 .....	87
Ejercicio 20 .....	87
Ejercicio 21 .....	87
Ejercicio 22 .....	87
Ejercicio 23 .....	87
Ejercicio 24 .....	88
Ejercicio 25 .....	88
Ejercicio 26 .....	88

## Lección 06

Enfoque Modular a la Programación .....	90
Ejemplo 1 .....	91
Procedimientos .....	92
Declarar, Definir e Invocar Procedimientos .....	92
Ejemplo 2 .....	93
Parámetros de los Procedimientos .....	97
Ejemplo 3 .....	98
Funciones .....	100
Declarar, Definir e Invocar Funciones .....	100
Parámetros de las Funciones .....	101
Ejemplo 4 .....	102
Alcance de las Variable .....	104
Alcance Local .....	104
Alcance Global .....	105
Alcance de lo Parámetros .....	106
Ejercicios .....	107
Ejercicio 1 .....	107
Ejercicio 2 .....	107
Ejercicio 3 .....	107
Ejercicio 4 .....	107
Propuesta Adicional .....	107

## **Bibliografía**

---

### **Lenguaje de Programación Java2 V5.0**

Autor: Eric Gustavo Coronel Castillo

### **Programación con Java2**

Autor: Joel Carrasco Muñoz

### **La Biblia de Java 2 v5.0**

Autor: Herbert Schildt

### **Documentación Oficial de Java**

Autor: Sun Microsystems Inc.



# Lección 01

## Algoritmos

### Contenido

- Algoritmos
- Representación de Algoritmos
- Ejercicios

## Algoritmos

---

### ¿Qué es un Algoritmo?

Un algoritmo es un conjunto finito de instrucciones o pasos que sirven para ejecutar una tarea o resolver un problema. En su vida diaria y profesional también ejecuta una secuencia de pasos para lograr una tarea dada. Por ejemplo, para mirar una película en el cine, usted necesita comprar los tickets y luego mirar la película en el cine. Usted no puede entrar directamente a la sala del cine sin comprar los tickets.

Otro ejemplo de una secuencia de pasos se puede considerar en una tienda por departamento. En una tienda por departamento se automatizó la facturación por los productos comprados por los clientes, los siguientes son los pasos que se realizan:

- El cliente compra algunos productos.
- El vendedor Sergio provee al computador información de la venta, como el nombre del cliente, el nombre del producto, y la cantidad del producto.
- Después de que el vendedor proporcione la información requerida, una factura que contiene el valor total de los productos se presenta en la pantalla.

Usted notará que el algoritmo sigue ciertos pasos para lograr la tarea. Los pasos se pueden categorizar en las tres fases siguientes:

1. En la fase de la entrada, la información relacionada con la compra de los productos se proporciona a la computadora.
2. En la fase de proceso, el computador procesa la información proporcionada para calcular el valor total por la compra de los productos.
3. En la fase de salida, el resultado del cálculo hecho en la fase de proceso se muestra en la pantalla.

Estas tres fases juntas forman el ciclo Entrada-Proceso-Salida (EPS). Cada vez que se genere una nueva factura se repite el ciclo EPS.

### Características de un Algoritmo

Un algoritmo tiene las siguientes cinco características:

- Un algoritmo acaba después de un número de pasos predeterminado.
- En cada paso de un algoritmo se especifica claramente las acciones a realizar.
- Los pasos en un algoritmo especifican operaciones básicas. Estas operaciones pueden incluir cálculos matemáticos, funciones de entrada/salida de datos y comparaciones lógicas.
- El algoritmo tendría que aceptar la entrada de datos en un formato definido, antes de que pueda procesarse con las instrucciones dadas.

- Un algoritmo genera una o más salidas tras el procesamiento de la entrada de datos.

Los pasos en un algoritmo se escriben en el orden en que son ejecutados. A continuación tenemos algunos ejemplos de algoritmos.

- Este algoritmo representa la lógica del ejemplo discutido en la sección anterior para la generación automática de una factura.

Paso 1: Inicio

Paso 2: Aceptar el nombre del producto

Paso 3: Aceptar la cantidad comprada

Paso 4: Leer el precio del producto y su descuento desde la base de datos

Paso 5: Calcular el precio total como un producto de la cantidad que compró y precio del producto.

Paso 6: Calcular el precio de descuento deduciendo el valor del descuento del precio total.

Paso 7: Mostrar los cálculos obtenidos

Paso 8: Fin

- Este algoritmo acepta los puntajes sobre cuatro temas como entrada y exhibe el puntaje total para un estudiante.

Paso 1: Inicio

Paso 2: Acepta el puntaje obtenido en Fundamentos de Programación

Paso 3: Acepta el puntaje obtenido en Modelamiento de Datos

Paso 4: Acepta el puntaje obtenido en Programación Orientada a Objetos

Paso 5: Acepta el puntaje obtenido en SQL Server

Paso 6: Calcula el puntaje total como la suma de todos los puntajes parciales

Paso 7: Mostrar el puntaje total

Paso 8: Fin

## Niveles de un Algoritmo

Los niveles de un algoritmo son:

**Nivel Macro** Un algoritmo que contenga pasos sin mucho detalle sobre un proceso se llama algoritmo de nivel macro.

**Nivel Micro** Un algoritmo que contenga pasos detallados sobre un proceso se llama algoritmo de nivel micro.

Por ejemplo, una compañía de televisión puede proporcionar pasos breves usando un algoritmo nivel macro en el manual del usuario de la televisión para localizar averías de sonido como sigue:

Paso 1: Inicio

Paso 2: Asegúrese de que el botón Mute no esté presionado

Paso 3: Asegúrese que el nivel de volumen sea el adecuado

Paso 4: Si el sonido sigue siendo inaudible, llame al ingeniero de la televisión

Paso 5: Fin

O, una compañía de televisión puede proporcionar los pasos detallados para solucionar el mismo problema usando un algoritmo de nivel micro:

Paso 1: Inicio

Paso 2: Tomar el control remoto

Paso 3: Chequear si el símbolo Mute parpadea sobre la pantalla de la televisión, presionar el botón Mute sobre el control remoto para habilitar el sonido.

Paso 4: Si el sonido aún sigue inaudible, incremente el volumen usando el control de volumen del control remoto.

Paso 5: Si aún no hay sonido, llame al ingeniero de televisión.

Paso 6: Fin

## Representación de Algoritmos

---

Los algoritmos pueden ser representados mediante:

- Diagramas de flujo
- Tablas de decisiones
- Pseudocódigos



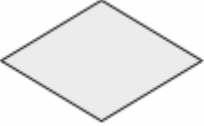

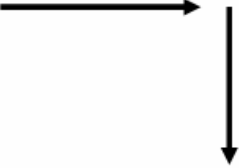

## Diagramas de Flujo




### Definición

Los diagramas de flujo son representaciones gráficas de algoritmos. Un diagrama de flujo consta de símbolos, que representan los pasos o etapas del algoritmo. Cada símbolo representa un tipo de actividad.

### Símbolos

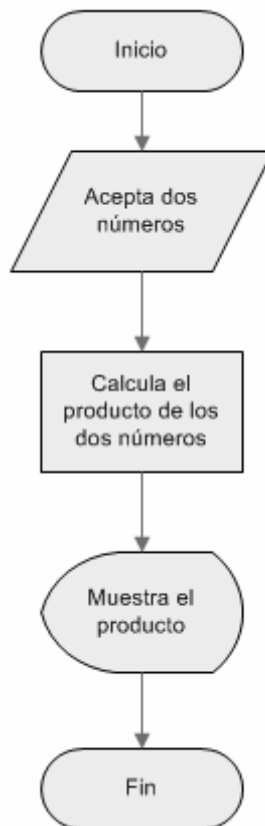
Los diferentes símbolos usados en un diagrama de flujo son:

Símbolo	Paso ó Actividad
	<p><i>Entrada/Salida</i></p> <p>Este símbolo representa una entrada ó salida.</p>
	<p><i>Proceso</i></p> <p>Este símbolo representa un proceso de una entrada, tal como la suma de dos números.</p>
	<p><i>Decisión</i></p> <p>Este símbolo representa una condición con la cual se debe tomar una decisión.</p>
	<p><i>Procedimiento/Subrutina</i></p> <p>Este símbolo representa la llamada a un procedimiento o subrutina predefinido compuesto de pasos que no son parte de este diagrama.</p> <p>ó</p> <p>Un programa grande puede ser dividido en subprogramas pequeños llamados procedimientos o subrutinas. Este simbolo representa la llamada a un procedimiento o subrutina desde el programa principal. El procedimiento o subrutina es completamente desdrito en un diagrama de flujo diferente.</p>
	<p><i>Línea de flujo</i></p> <p>Este símbolo representa los enlaces de un símbolo con otro y ayuda a entender la secuencia de los pasos a seguir para completar una tarea. Este símbolo indica el flujo del diagrama de flujo desde arriba hacia abajo o de la izquierda a la derecha.</p>
	<p><i>Inicio y Fin</i></p> <p>Este símbolo representa el inicio y fin del diagrama de flujo.</p>

<b>Símbolo</b>	<b>Paso ó Actividad</b>
	<p><i>Conector en Página</i></p> <p>Un diagrama de flujo se puede dividir en partes cuando muchas líneas del flujo lo hacen ilegible. Este símbolo representa la conexión entre estas partes del un diagrama de flujo en una misma página. Este símbolo es etiquetado con letras en mayúsculas, por ejemplo A.</p>
	<p><i>Conector Fuera de Página</i></p> <p>Este símbolo representa la conexión entre las partes de un diagrama de flujo en páginas separadas. Esto ayuda a prevenir confusión respecto a la secuencia de un diagrama de flujo cuando este abarca múltiples páginas. Este símbolo es etiquetado con números, por ejemplo 1.</p>
	<p><i>Visualización</i></p> <p>Este símbolo representa la salida usando la instrucción mostrar.</p>

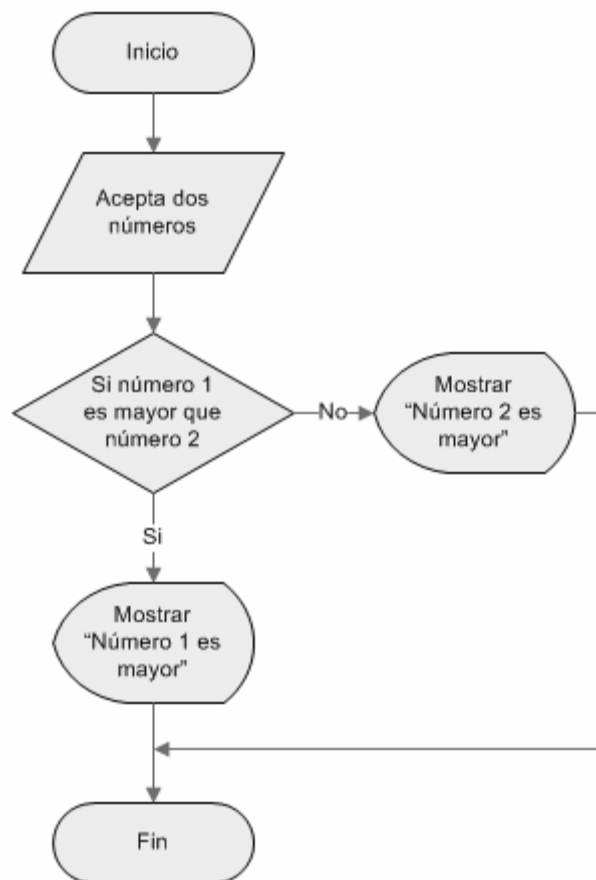
## Ejemplos

- El siguiente diagrama de flujo acepta dos números, calcula el producto y muestra el resultado.

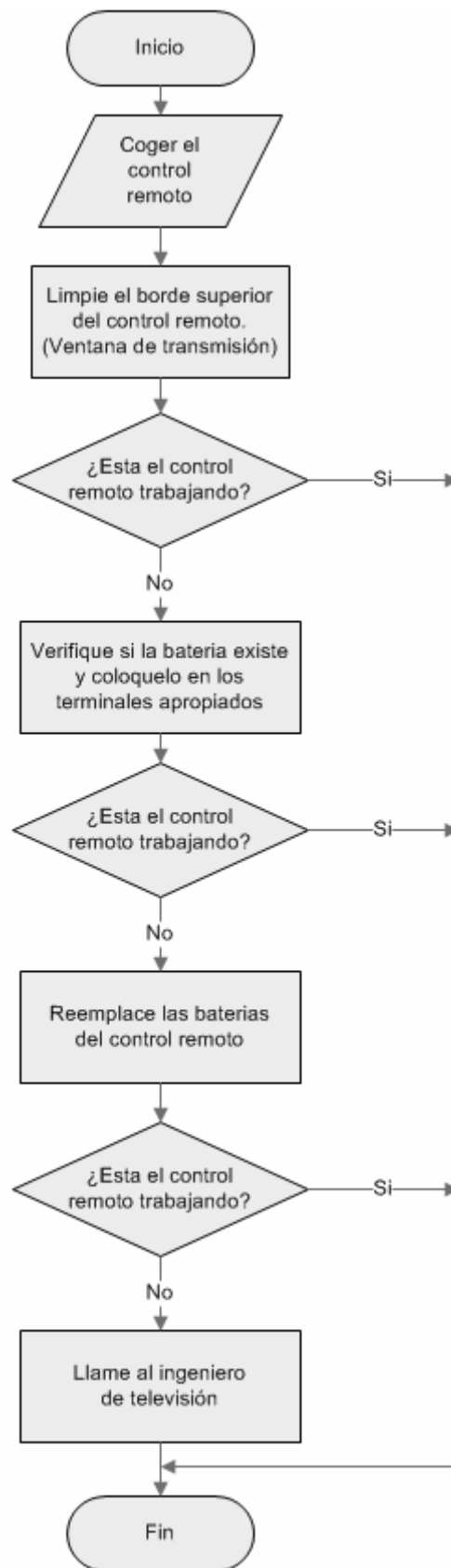




- El siguiente diagrama de flujo acepta dos números, y muestra el mayor de ellos después de compararlos.



- El siguiente es el diagrama de flujo dado en el manual de una televisión proporciona las recomendaciones para resolver el problema de mal funcionamiento del control remoto.



## Reglas de un Diagrama de Flujo

El American National Standards Institute (ANSI) recomienda un número de reglas a cumplir en el dibujo de diagramas de flujo. Algunas de estas reglas y pautas se muestran a continuación:

- La lógica completa de un diagrama de flujo debería representarse usando los símbolos estándares.
- El diagrama de flujo debería ser claro, preciso y de fácil interpretación.
- Los diagramas de flujo solo puede tener un punto de inicio y un punto de término.
- Los pasos en un diagrama de flujo deberían seguir el enfoque de arriba a abajo o de izquierda a derecha.
- Todas las entradas de datos necesarias deberían exponerse en un orden lógico.
- Los símbolos de inicio y fin deberían tener una sola línea de flujo.
- Los símbolos de entrada, procesamiento, salida y visualización de datos deberían tener dos líneas de flujo conectadas, una previa al símbolo y otra posterior al símbolo.
- El símbolo de decisión debería tener una línea de flujo conectada previo al símbolo y dos líneas de flujo conectadas posterior al símbolo para cada posible solución.

## Ventajas de los Diagramas de Flujo

Las ventajas de los diagramas de flujo son:

- Los diagramas de flujo es el mejor método de comunicar lógica.
- Los diagramas de flujo ayudan a analizar los problemas eficazmente.
- Los diagramas de flujo actúan como guía durante la fase de diseño del programa.
- Es más fácil depurar errores de lógica usando un diagrama de flujo.
- Los diagramas de flujo ayudan a mantener los programas.

## Desventajas de los Diagramas de Flujo

Las desventajas de los Diagramas de flujo son:

- Un diagrama de flujo largo puede extenderse sobre múltiples páginas, lo cual reduce su legibilidad.
- Como los símbolos de los diagramas de flujo no pueden escribirse, el dibujo de un diagrama de flujo usando cualquier herramienta gráfica lleva mucho tiempo.
- Los cambios hechos en un solo paso pueden ocasionar tener que volver a dibujar el diagrama de flujo completo.
- Un diagrama de flujo representando un algoritmo complejo puede tener demasiadas líneas de flujo. Esto reduce su legibilidad y llevará mucho tiempo dibujarlo y entender su lógica.

## Tablas de Decisiones

### Definición

Las tablas de decisiones representan algoritmos que implican una toma de decisiones compleja.

Una tabla de decisiones consta de cuatro componentes, Código de Condición, Condición Aplicada, Código de Acción y Acción Tomada, tal como se muestra en la siguiente tabla:

Código de Condición	Condiciones Aplicadas
Código de Acción	Acción Tomada

<b>Código de Condición</b>	Consta de las condiciones en base a las cuales se toma una decisión.
<b>Condiciones Aplicadas</b>	Este componente contiene las condiciones alternativas.
<b>Código de Acción</b>	Contiene las acciones a tomar según la combinación de condiciones especificadas en el componente Condiciones Aplicadas.
<b>Acción Tomada</b>	Consta de las alternativas de la acción como S o N. Aquí, S o N especifica si debe ejecutarse o no la acción mencionada en el Código Acción.

## Ejemplo

Para entender cada componente del cuadro de decisión, considere un ejemplo de desarrollo de una tabla de decisión para representar la elección de un candidato a solicitar el puesto de recepcionista en ABC Ltd. Los criterios de elección para aceptar los candidatos son:

- El candidato debe ser mujer
- El candidato debe tener mas de dos años de experiencia de trabajo
- El estado civil del candidato debe ser soltero

La tabla de decisión por el ejemplo es:

<b>Criterios de elección para el puesto de recepcionista en ABC Ltd.</b>								
<b>Código de Condición</b>	<b>Condiciones Aplicadas</b>							
El candidato es mujer.	Y	Y	Y	Y	N	N	N	N
El candidato tiene mas de dos años de experiencia trabajando.	Y	N	Y	N	Y	N	Y	N
El estado civil del candidato es soltero.	N	N	Y	Y	N	N	Y	Y
<b>Código de Acción</b>	<b>Acción Tomada</b>							
Candidata aceptada	N	N	Y	N	N	N	N	N

Aquí, el Código de Condición especifica los criterios elegidos para el puesto de recepcionista. La sección Condiciones Aplicadas contiene dos alternativas, Y cuando satisface la condición y N cuando no satisface la condición. Con dos alternativas para cada condición, el número de combinaciones posibles es  $2^3$ , lo cual significa 8 combinaciones tal como se muestra en la tabla de decisión, 8 columnas en la sección Condiciones Aplicadas. La sección Código de Acción define la acción para los candidatos aceptados. La sección Acción Tomada consiste de dos alternativas, Y para tomar la acción y N para no tomar la acción.

Para crear una tabla de decisiones, deberían considerarse los puntos siguientes:

- Especificar un nombre apropiado para la tabla, describiendo su objetivo.
- Escribir la sentencia/sentencias de la condición en Código de Condición, en base a las cuales se tomará una decisión.
- Hacer un listado de todas las combinaciones correspondientes a las condiciones especificadas en la sección del Código de Condición.
- Especificar todas las acciones que pueden hacerse en conjunción con cada combinación.

## **Ventajas de las Tablas de Decisiones**

Las ventajas de las tablas de decisiones son:

- Son útiles en casos en los que hay que representar algoritmos complejos con muchas derivaciones.
- Se usan tablas de decisiones como herramienta alternativa para simplificar las tareas de toma de decisiones que conlleva el procesamiento.

## **Desventajas de las Tablas de Decisiones**

Las desventajas de las tablas de decisiones son

- Una tabla de decisiones solo pueden representar tareas de toma de decisiones implicadas en un procesamiento.
- Los principiantes prefieren los diagramas de flujo a las tablas de decisiones para representar la secuencia completa de pasos para llevar a cabo una tarea.

## Pseudocódigo


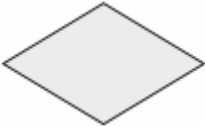

### Definición

El pseudocódigo representa un algoritmo en el lenguaje inglés. Se usa como alternativa a un diagrama de flujo. Las sentencias usadas en el pseudocódigo son simples y están escritas de manera secuencial.

En el pseudocódigo se emplean una serie de claves:

- begin**      Se utiliza para indicar el inicio de un pseudocódigo.
- end**        Se utiliza para indicar el final del pseudocódigo.
- accept**     Se utiliza para obtener una entrada del usuario.
- display**    Se utiliza para presentar los resultados.
- if...else**   Se utiliza para representar condiciones.

Las palabras utilizadas en el pseudocódigo corresponden con símbolos del diagrama de flujo, tal como se ilustra a continuación:

<i><b>Símbolo del Diagrama de Flujo</b></i>	<i><b>Palabra del Pseudocódigo</b></i>
	accept
	<i>if ... else</i>
	begin, end



## Ejemplos

- El siguiente ejemplo de pseudocódigo acepta dos números, calcula su producto, y muestra el resultado.

```
begin
  accept número 1
  accept número 2
  calcular el producto
  display el producto
end
```

- El siguiente ejemplo de pseudocódigo acepta dos números y muestra el mayor de ellos.

```
begin
  accept número 1
  accept número 2
  if número 1 es mayor que número 2
    display número 1
  else
    display número 2
  end
end
```

## Ventajas del Pseudocódigo

Las ventajas del pseudocódigo son:

- Es de escritura más fácil y rápida que la de un diagrama de flujo.
- Puede detectar errores y aplicar fácilmente los cambios.
- No tiene que rescribirse si se hacen cambios.
- Puede convertirse a un programa usando cualquier lenguaje de programación.

## Limitaciones del Pseudocódigo

Las limitaciones del uso de pseudocódigo son:

- El pseudocódigo no proporciona una representación gráfica de un algoritmo.
- Un pseudocódigo que represente demasiadas condiciones anidadas puede ser de difícil comprensión

## Ejercicios

---

### Ejercicio 01

Los campos de la vacunación necesitan ser organizados para proporcionar la vacunación gratis a los niños por debajo de cinco años de la edad. Para organizar estos campos de vacunación, es necesaria realizar una encuesta a la población. Esta encuesta ayudará al personal del hospital a determinar el número aproximado de vacunas que deben ser provistas a los campos. Se darán vacunaciones gratis a los niños que están debajo de cinco años la edad, no tienen ninguna enfermedad, y nacen en familias por debajo de la línea de pobreza. Asumirán a las familias que tienen una renta anual menor de \$4500 para estar debajo de la línea de la pobreza.

¿Qué técnica usted utilizará para representar el algoritmo para este problema? Dé un análisis razonado para su opción, y represente el algoritmo usando la técnica seleccionada.

### Ejercicio 02

Global Manufactures Company mantiene y repara coches y tiene muchos talleres por todo el mundo. Cuando un coche es traído para servicio a un taller de Global Manufacturers Company, este es enviado al departamento de mantenimiento. Sin embargo, si se encuentra dañado y es traído para ser reparado, se envía al departamento de reparación. Después de terminar el mantenimiento o reparación del coche, un supervisor lo examina y el coche se devuelve al cliente.

Desarrolle el diagrama de flujo que represente el algoritmo para este problema.

### Ejercicio 03

La gerencia de PeruDev decide otorgar un Subsidio por Alquiler de Vivienda (SAV) a sus empleados. Los criterios para calcular el importe de SAC son:

- Si el salario básico de un empleado es mayor que 10,000 Nuevos Soles, la suma por concepto de SAV otorgado al empleado será del 30% del salario básico.
- Si el salario básico de un empleado es igual ó menor que 10,000 Nuevos Soles, la suma por concepto de SAV otorgado al empleado será del 20% del salario básico.

Desarrolle el pseudocódigo que represente el algoritmo para este problema.

## Ejercicio 04

La Universidad de Pacherez publica los resultados finales de los exámenes en su Web Site. Para ver los resultados, los estudiantes necesitan ingresar sus códigos en la Web Site. Si la calificación de un estudiante es superior al 50 por ciento, se publicará el mensaje Aprobado, junto con sus notas. Si no, se publica el mensaje Desaprobado.

Desarrollar un diagrama de flujo y su respectivo pseudocódigo que represente el algoritmo para este problema.

## Ejercicio 05

Una librería está ofreciendo descuentos en algunos de sus libros. Los libros están divididos en dos categorías, A y B. El descuento es solo para los libros de la categoría A. Si un cliente compra un libro de la categoría A, se le otorga un descuento de 10%.

Desarrollar el pseudocódigo que represente el algoritmo para este problema.

## Ejercicio 06

Sergio, un docente de ISIL, necesita calcular el puntaje promedio de los estudiantes en la clase de SQL Server. El código y puntaje de todos los estudiantes son:

Código	Puntaje
A001	45
A002	78
A003	56
A004	89
A005	70

Crear un algoritmo que representa la lógica para calcular el puntaje promedio de la clase.

## Ejercicio 07

Una tienda de electrónica está ofreciendo un descuento en los televisores. El descuento ofrecido se basa en el tamaño de la televisión. Para una televisión de 14 pulgadas, se ofrece un 10% de descuento, mientras que para televisiones de 21 pulgadas, se está ofreciendo un descuento de 20%.

Represente el problema usando pseudocódigo.

# Apuntes

# Lección 02

## Instalación de Java

### Contenido

- Proceso de Instalación
- La Carpeta bin
- Probando Java

## Proceso de Instalación

### Archivo de Instalación

En este caso instalaremos Java 2 JDK versión 5.0, el archivo de instalación se detalla a continuación.

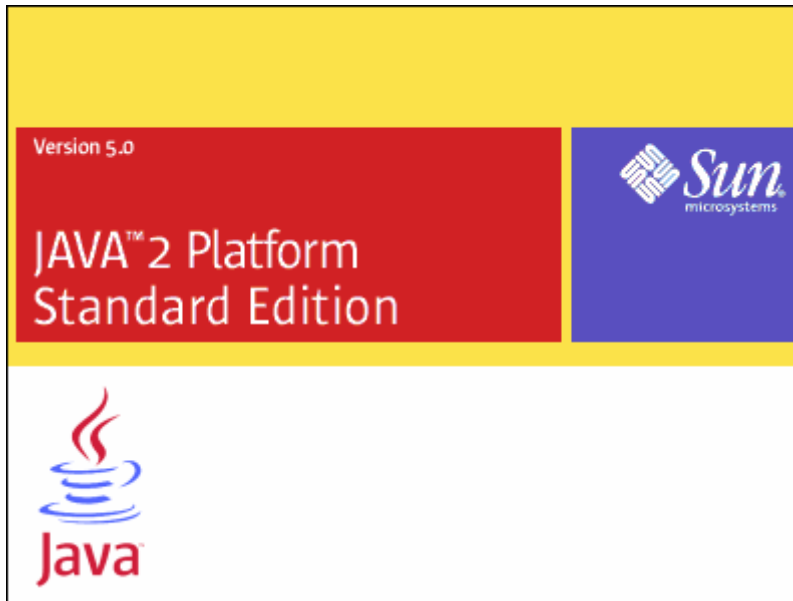
<b>Página Web</b>	<a href="http://java.sun.com/j2se/1.5.0/download.jsp">http://java.sun.com/j2se/1.5.0/download.jsp</a>
<b>Archivo</b>	jdk-1_5_0_06-windows-i586-p.exe
<b>Tamaño</b>	59.86 MB

La página Web de donde debe bajar este archivo se muestra en la siguiente figura:

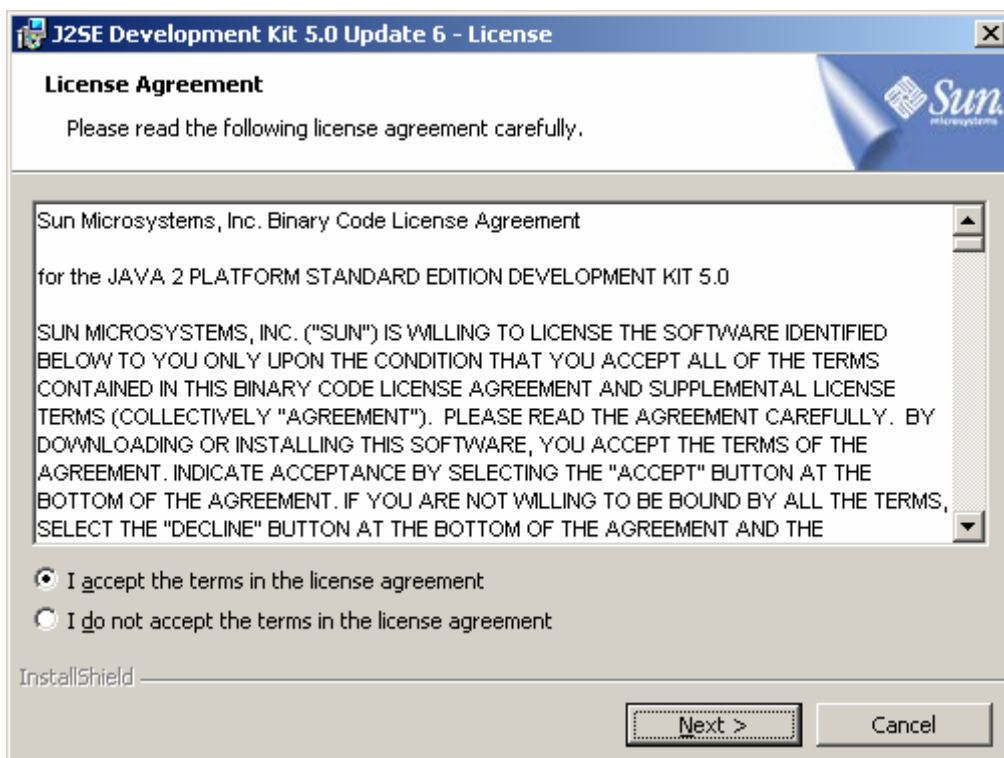


## Proceso de Instalación

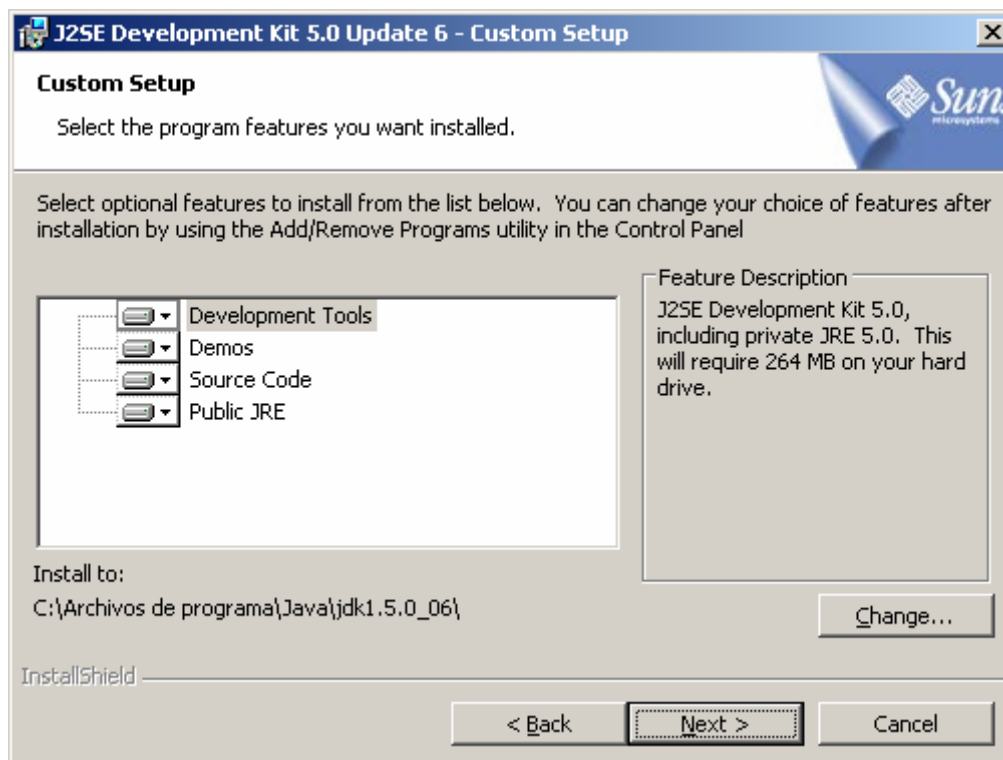
1. Ejecute el archivo de instalación, obtendrá la siguiente ventana de bienvenida.



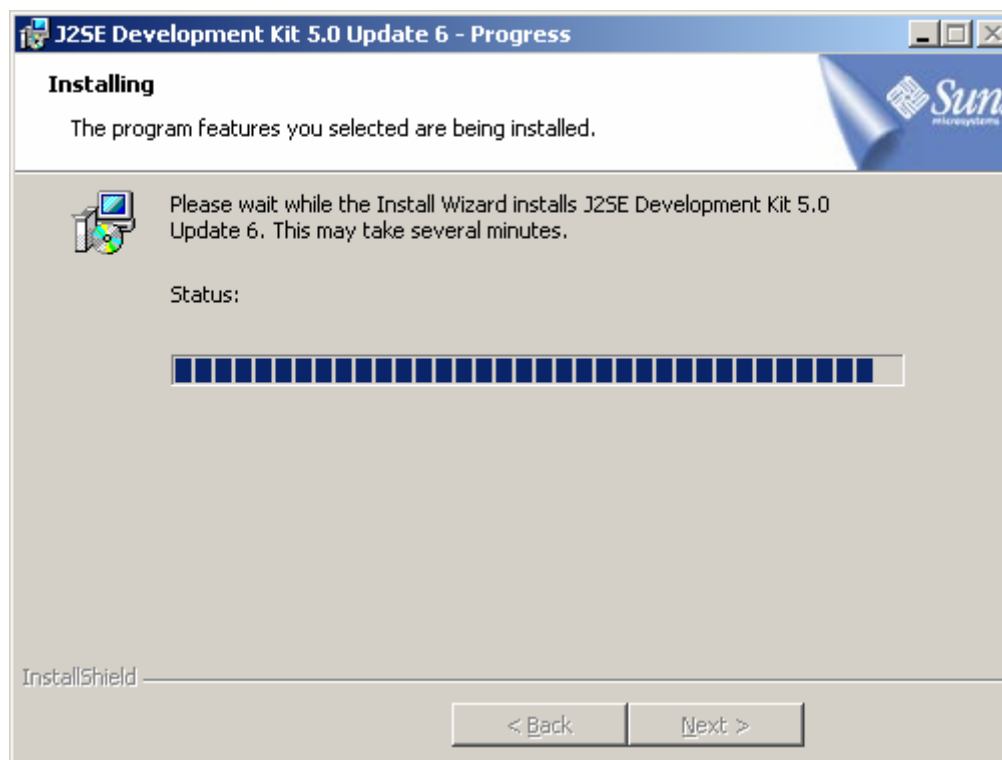
2. Después de preparar la instalación tendrá la ventana donde debe aceptar las condiciones de licenciamiento. Debe hacer clic en el botón Next.



3. En la siguiente ventana debe seleccionar los productos a instalar y el directorio donde serán instalados, por defecto están marcados todos los productos. Le sugiero que deje las opciones por defecto y solo haga clic en el botón Next.

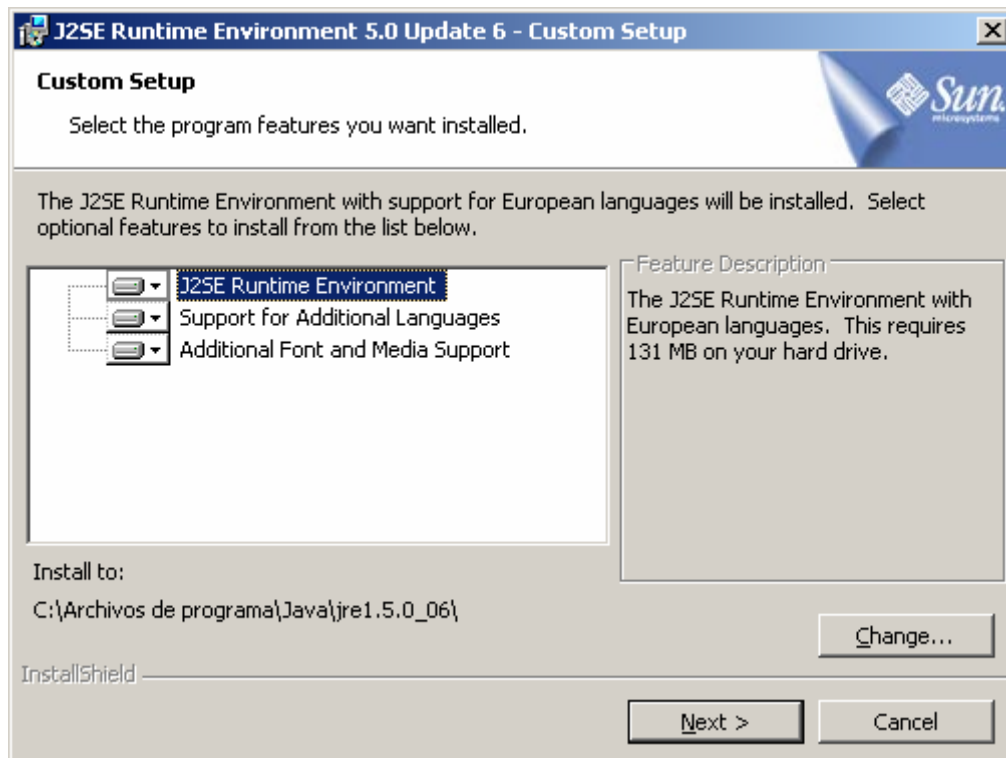


4. A continuación se inicia el proceso de instalación, esto puede tardar varios minutos.

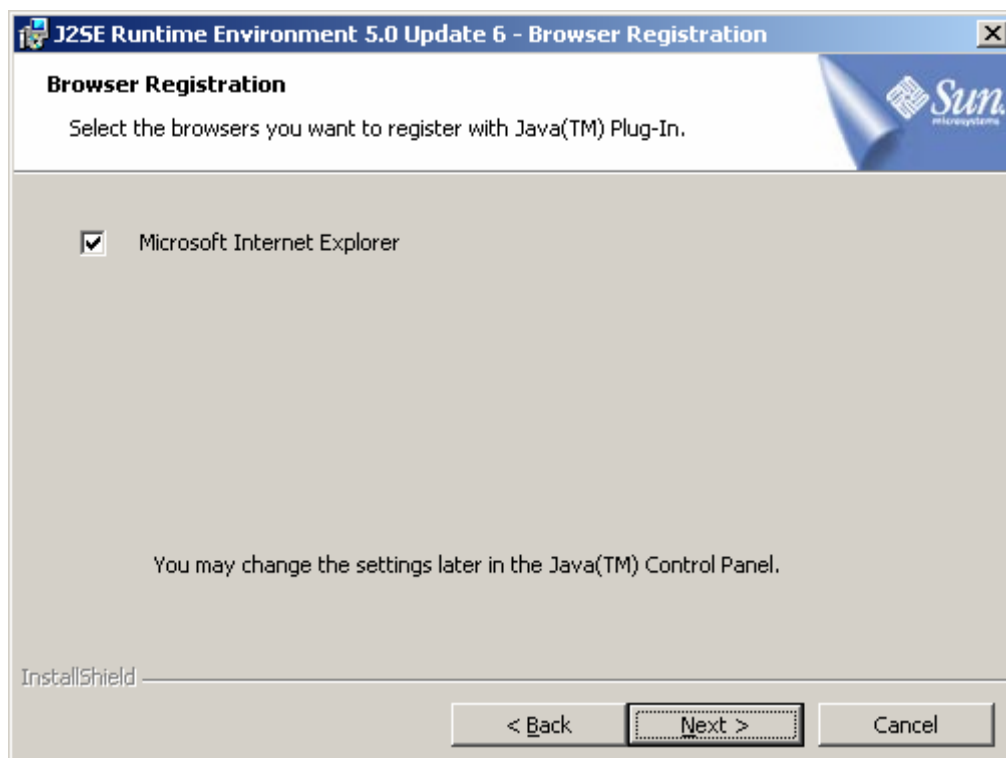




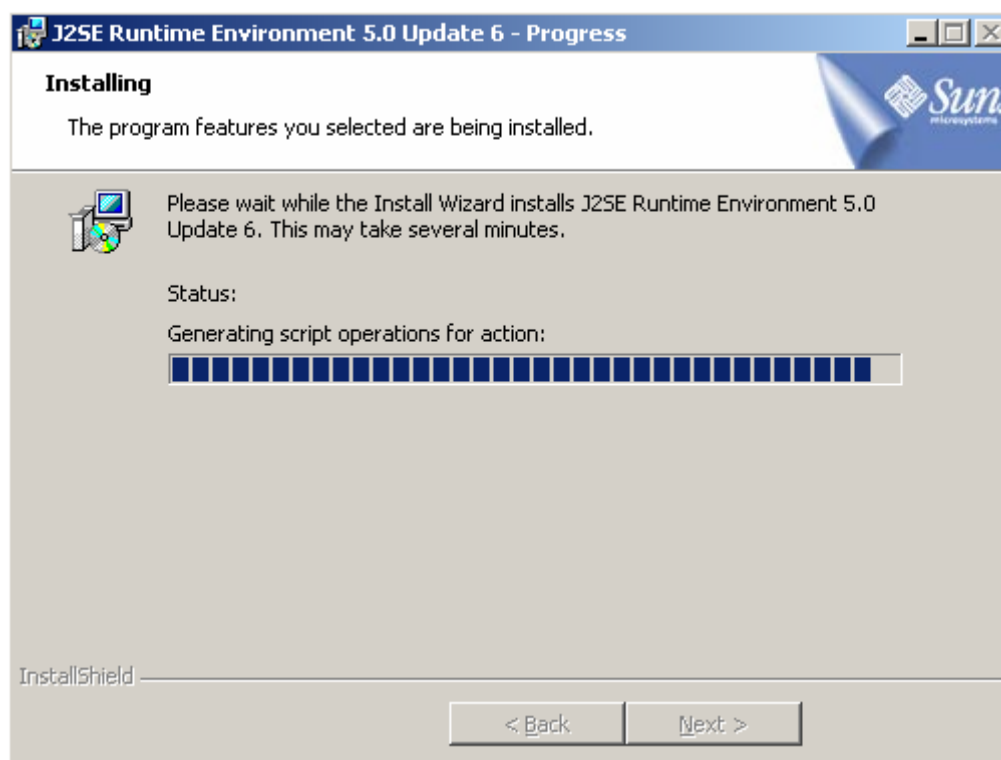
5. En la siguiente ventana, tiene la posibilidad de seleccionar características adicionales, deje las opciones por defecto, y haga clic en el botón Next.



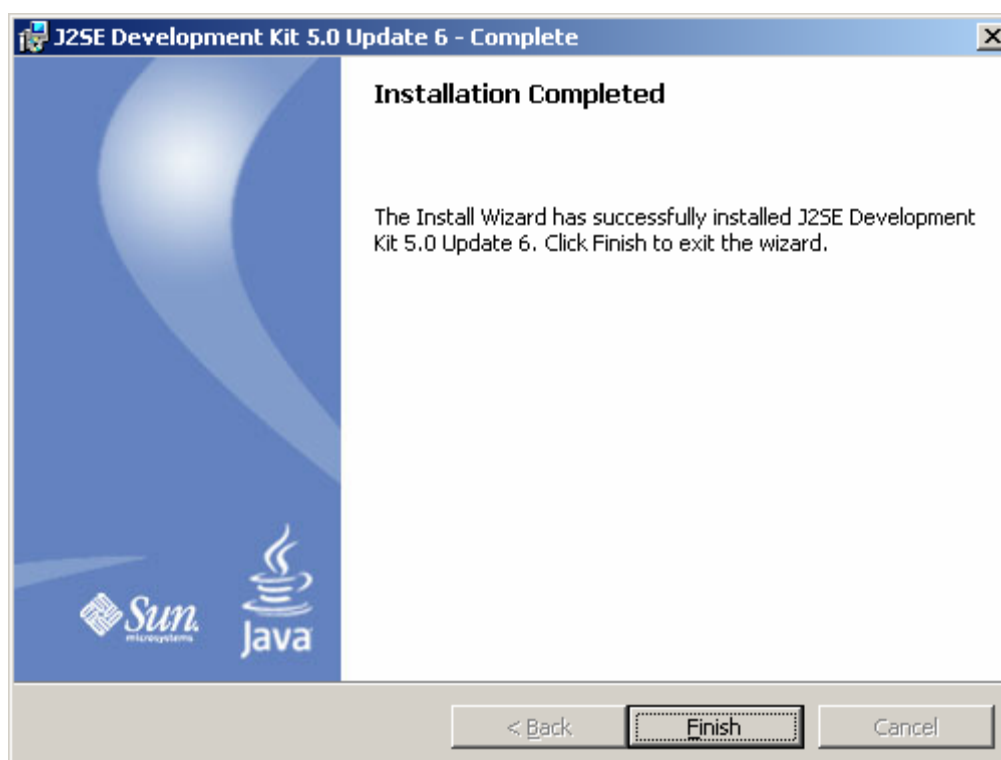
6. En la siguiente ventana, debe seleccionar el navegador que utilizaremos, luego debe hacer clic en el botón Next.



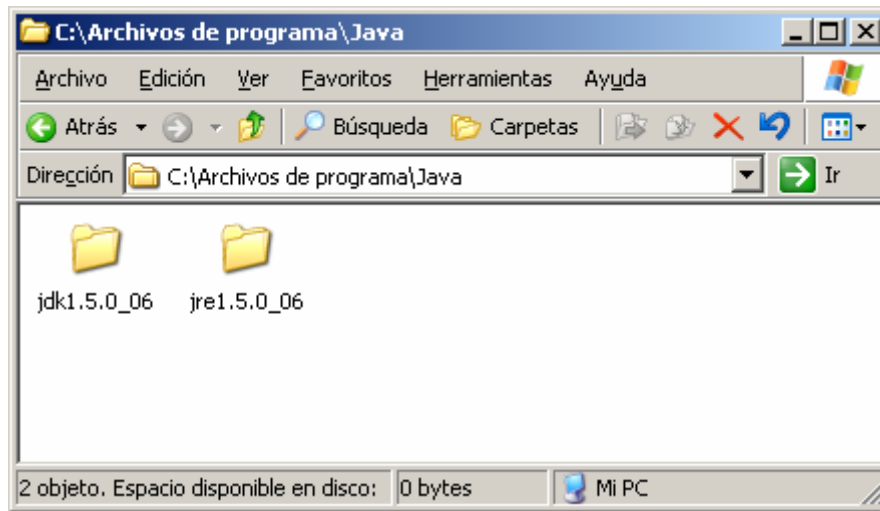
7. En la siguiente ventana, continua la instalación de las opciones seleccionadas en el paso anterior.



8. Finalmente llegamos a la finalización de la instalación, solo queda hacer clic en el botón Finish.



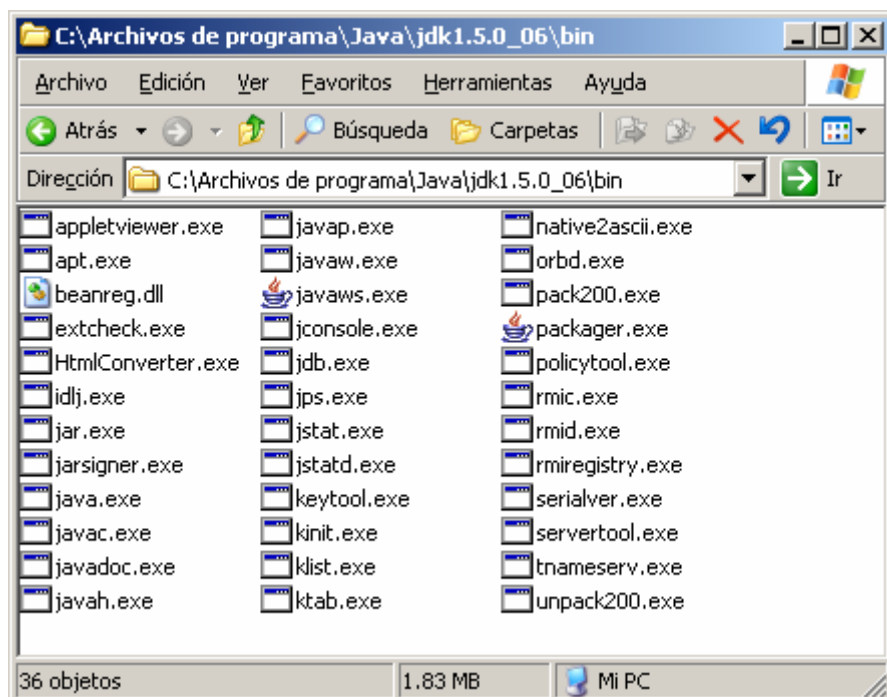
Al finalizar el proceso de instalación podemos verificar que se han creado dos carpetas, tal como se puede apreciar en la siguiente figura.



<b>jdk1.5.0_06</b>	Esta carpeta contiene las herramientas de desarrollo, por ejemplo el compilador de java, el depurador, el runtime y ejemplos sobre varios temas del lenguaje Java.
<b>jre1.5.0_06</b>	Esta carpeta contiene las librerías, la maquina virtual y otros componentes para ejecutar applets y programas desarrollados en Java.

## La Carpeta bin

La carpeta bin contiene las diferentes herramientas de desarrollo de la plataforma Java.



A continuación se describen las principales herramientas.

<b>javac.exe</b>	Compilador de los programas .java.
<b>java.exe</b>	Maquina virtual de java, ejecuta los bytecode (.class).
<b>javadoc.exe</b>	Permite documentar los programas.
<b>appletviewer.exe</b>	Permite ejecutar Applets.
<b>jdb.exe</b>	Permite depurar los programas.

## Probando Java

### Carpeta de Trabajo

La carpeta de trabajo para este laboratorio es:

C:\j2n100\lab01

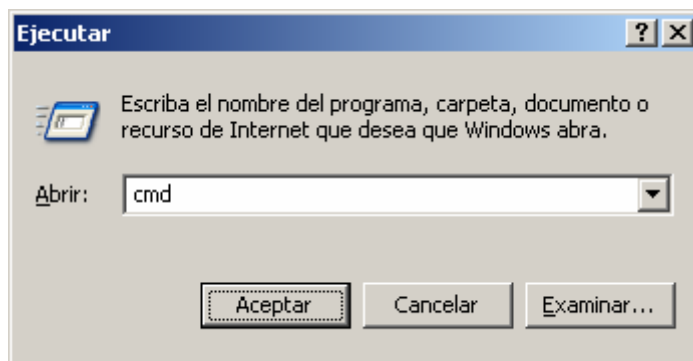
Y la carpeta donde están las herramientas de Java es:

C:\Archivos de programa\Java\jdk1.5.0\_06\bin

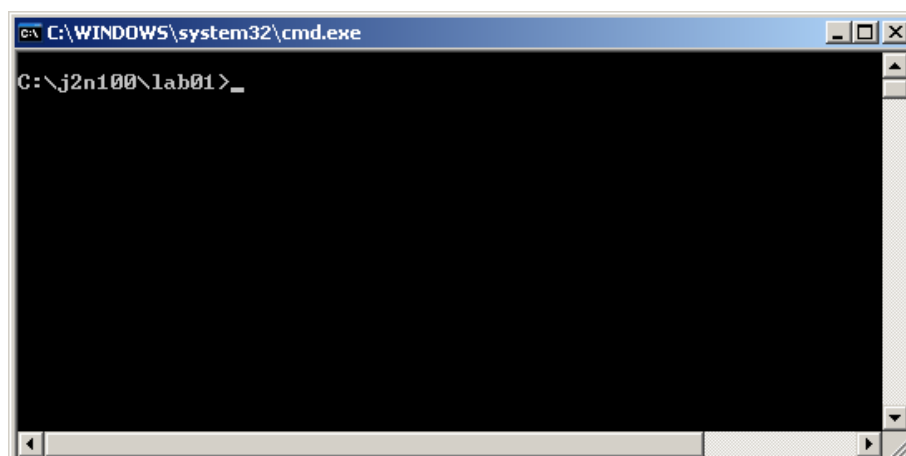
### Establecer la Variable de Entorno PATH

La variable de entorno PATH tiene las rutas donde el sistema operativo busca las aplicaciones cuando no las encuentra en la carpeta actual, por lo tanto debe contener la ruta donde se encuentran las herramientas de Java.

1. Cargar la consola de comando, para lo cual ejecute el comando `cmd` desde el dialogo Ejecutar.



2. En la consola, debe ubicarse en la carpeta `C:\j2n100\lab01`.



3. Ahora debe verificar si la variable de entorno `PATH` contiene la ruta de la carpeta `bin`.

```
C:\j2n100\lab01>set path [Enter]

Path=C:\oracle\product\10.2.0\db_1\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDO
WS
\System32\Wbem

PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
```

Observe que no tiene la ruta de la carpeta `bin` de Java.

4. Ahora debe agregar la ruta de la carpeta `bin` a la variable de entorno `PATH`. Ejecute el siguiente comando:

```
C:\j2n100\lab01>set path=C:\Archivos de programa\Java\jdk1.5.0_06\bin;%path%
[Enter]

C:\j2n100\lab01>set path [Enter]

Path=C:\Archivos de programa\Java\jdk1.5.0_06\bin;C:\oracle\product\10.2.0\db_1\
bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem

PATHEXT=.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
```

Después de haber establecido el valor de la variable de entorno `PATH`, esta en condiciones de ejecutar sus primeros ejemplos.

---

**Nota**

La configuración de la variable de entorno `PATH` de esta manera, solo es valida en la actual consola, no afecta al resto del sistema. Si abre otra consola tendrá que volver a configurar la variable de entorno `PATH`.

---

## Ejecutando el Primer Programa

1. Usando el bloc de notas proceda a crear el siguiente programa y grábelo con el nombre Prog0101.java en la carpeta C:\j2n100\lab01.

Prog0101.java
<pre>public class Prog0101 {      public static void main(String[] srgs) {          System.out.println("Java esta listo");         System.out.println("Ahora debo ponerme las pilas");      }  }</pre>

2. Ahora debe compilar el programa y crear el bytecode.

```
C:\j2n100\lab01>javac Prog0101.java    [Enter]
```

3. Puede verificar que se ha creado el archive Prog0101.class.

```
C:\j2n100\lab01>dir    [Enter]
El volumen de la unidad C es WinXP
El número de serie del volumen es: 8C64-E123

Directorio de C:\j2n100\lab01

18/02/2006  11:40 p.m.  <DIR>          .
18/02/2006  11:40 p.m.  <DIR>          ..
18/02/2006  11:40 p.m.                471 Prog0101.class
18/02/2006  11:35 p.m.                184 Prog0101.java
                2 archivos             655 bytes
                2 dirs  11,652,476,928 bytes libres
```

4. Ahora puede ejecutar el programa.

```
C:\j2n100\lab01>java Prog0101    [Enter]
Java esta listo
Ahora debo ponerme las pilas
```

Si ha llegado hasta este punto, a logrado ejecutar su primer programa en Java

# Apuntes



# Lección 03

## Variables y Expresiones

### Contenido

- Introducción a Variables
- Operadores
- Ejemplos
- Ejercicios

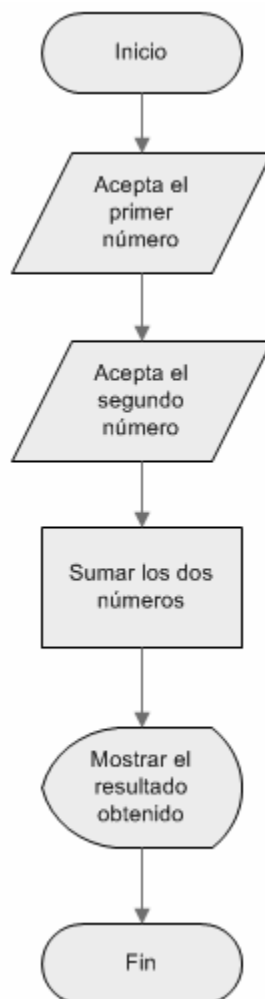
## Introducción a Variables

---

La memoria interna del computador se utiliza para almacenar los datos de entrada proporcionados por el usuario, las instrucciones para tratar estos datos y el resultado del proceso o datos de salida. La memoria consta de diversas localizaciones en las cuales se almacenan los datos. A estas ubicaciones de la memoria se les denomina variables. A los valores que se almacenan en las variables se los denomina literales. Los literales representan valores que pueden clasificarse en dos categorías:

- Valores numéricos, como 25, 78 y 90.45
- Valores carácter como "Hola", "X", "E001" y "1988". Los literales carácter siempre van entre comillas (" ")

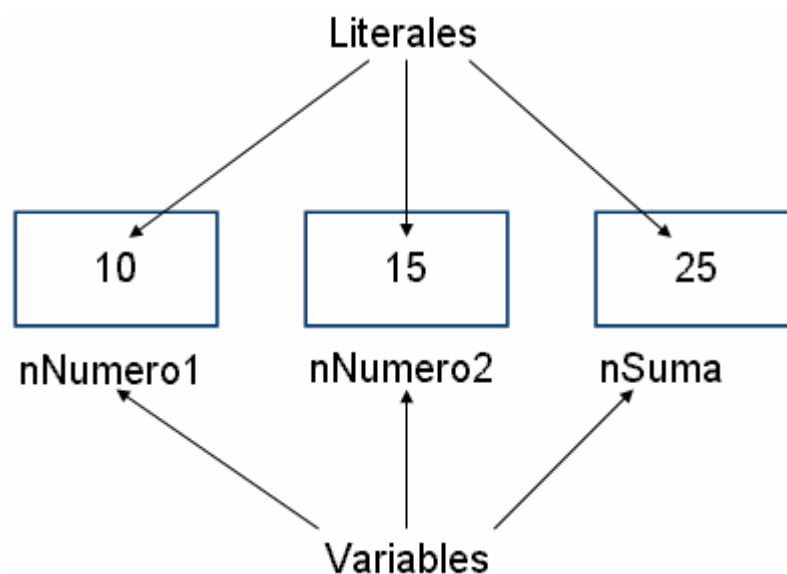
Para entender como un computador procesa los datos, considere el siguiente problema donde dos números son ingresados y el resultado se muestra sobre la pantalla. El diagrama de flujo usado para representar la lógica de la solución del problema es el siguiente:



Cuando las instrucciones son ejecutadas, el valor del primer número es aceptado y almacenado en la memoria. De manera similar, el valor del segundo número es también aceptado y almacenado en la memoria. El computador hace referencia a los números almacenado en memoria, calcula la suma, y almacena el resultado obtenido

en una localización diferente de la memoria. El computador hace referencia al resultado almacenado en memoria, para mostrarlo en la pantalla. Por lo tanto, el computador necesita identificar las localizaciones de memoria para almacenar los valores o recuperar los valores almacenados.

Las localizaciones cuando el primer número, el segundo número, y el resultado son almacenados pueden ser referenciadas como `nNumero1`, `nNumero2` y `nSuma` respectivamente. Cada vez que el conjunto de instrucciones son ejecutadas; los valores de `nNumero1`, `nNumero2`, y `nSuma` variará, dependiendo de los valor ingresados por el usuario. Por consiguiente, `nNumero1`, `nNumero2`, y `nSuma` son conocidos como variables. Si el usuario inicializa `nNumero1` y `nNumero2` con 10 y 15 respectivamente dentro del programa, los valores 10 y 15 no cambian. Por lo tanto, los valores almacenados en estas variables son conocidas como literales.



## Tipos de Datos

El número de bytes que debe reservarse para las diferentes variables depende del tipo de valor que éstas almacenan. Por lo tanto, hay una necesidad de clasificar los tipos de datos que puedan ser almacenados en la memoria. Este tipo de valor es denominado tipo de dato. Los tipos de datos están clasificados en:

- **Numéricos:** Las variables de tipo de dato numérico solo pueden contener números. Por ejemplo; la edad de una persona, el precio de un producto. Estas variables pueden almacenar números de coma flotante y pueden ser usadas dentro de cálculos.
- **Carácter:** Las variables de tipo de dato carácter pueden contener una combinación de letras, números, y caracteres especiales. Por ejemplo; el nombre de una persona ó la dirección postal. Estas variables no pueden ser usadas dentro de cálculos.

## Declaración de Variables

Es necesario declarar una variable antes de ser usada dentro de un programa. Cuando se declara una variable, una posición de memoria definida se le asigna a la variable. La declaración de una variable asigna un nombre a la variable y especifica el tipo de dato que la variable puede almacenar.

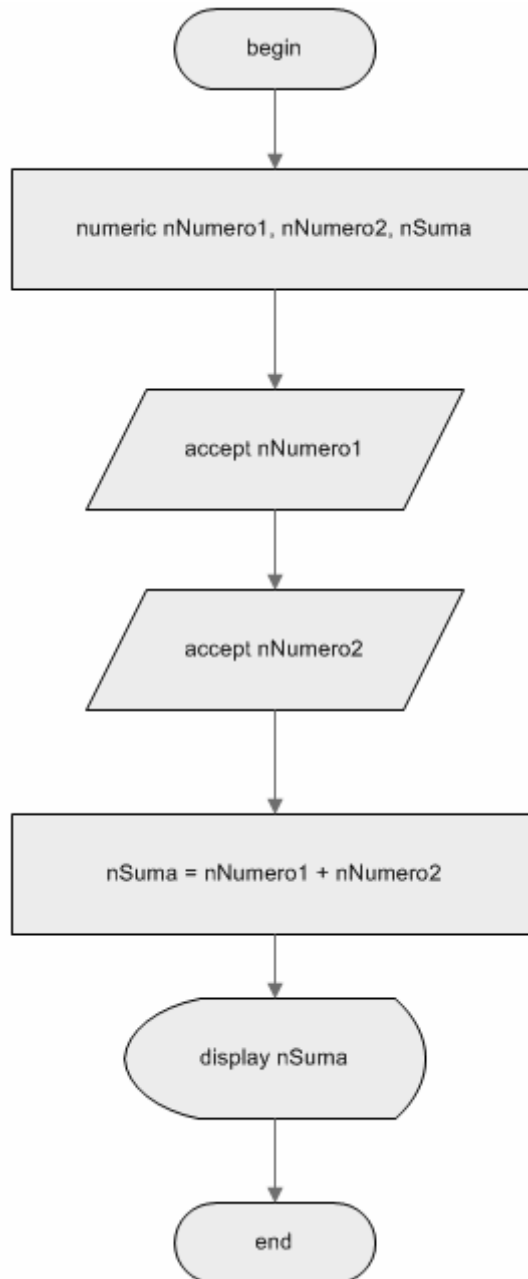
Por ejemplo, se necesita desarrollar un pseudocódigo para aceptar el código, nombre, edad, y teléfono de un cliente. Para aceptar el código y nombre, se necesita declarar dos variables de tipo carácter. Es buena práctica declarar una variable con un nombre que pueda describir su propósito. Por lo tanto, para nuestro ejemplo, se sugiere declarar las dos variables con los nombres `clicod` y `clinom` en lugar de `cc` y `cn`. De manera similar, es necesario declarar dos variables numéricas con los nombres `edad` y `telefono` para aceptar la edad y teléfono del cliente. La declaración de estas variables es como sigue:

```
character clicod
character clinom
numeric edad
numeric telefono
```

También se puede declarar variables del mismo tipo en una simple sentencia como se ilustra a continuación:

```
character clicod, clinom
numeric edad, telefono
```

A continuación tenemos el diagrama de flujo completo para leer dos números y mostrar el resultado.



En este diagrama de flujo, las variable nNumero1, nNumero2, y nSuma son declaradas en el primer paso. En el segundo paso, los valores de los dos números son aceptados desde el usuario. Posteriormente, el resultado de la suma de nNumero1 y nNumero2 es almacenado en la variable nSuma. Finalmente, se muestra el valor de nSuma.

Aunque no hay convenciones para dar nombre a las variables, las siguientes pautas pueden resultar útiles:

- La primera letra del nombre de la variable podría indicar el tipo de dato de la variable. Por ejemplo, puede ser “c” o “n” para indicar una variable carácter o numérica, respectivamente. Algunos ejemplos son cNombre y nEdad.
- El nombre de la variable debería describir con claridad el propósito de la variable. Por ejemplo, nNota es una variable numérica para guardar la nota del alumno.
- El nombre de la variable no debería contener espacios o símbolos tales como: ! @ # \$ % ^ & \* ( ) { } [ ] . , : ; “ ’ / y \. Se puede utilizar el carácter de subrayado cuando sea necesario insertar un espacio en el nombre de una variable, como por ejemplo, nSalario\_Básico.
- Si el nombre de la variable está compuesto por varias palabras sin espacios entre ellas, la primera letra de cada palabra debería ir en mayúscula para facilitar la lectura.

## Asignación de Valores a Variables

Cualquier variable necesita que se le asigne un valor antes de utilizarla. Esto es necesario para asegurarnos que la memoria asignada a la variable este inicializada con un valor valido.

Existen dos métodos para asignar valores a una variable dentro de un algoritmo:

- Asignación directa
- Instrucción de lectura

### Asignación Directa

Se puede asignar valores a las variables siguiendo el método de la asignación directa utilizando el signo igual (=).

#### Sintaxis:

```
nombre_variable = valor
```

Los siguientes son algunos ejemplos:

```
numeric nAltura, nEdad, nContador
character cCodigo

nAltura = 180
nEdad = 40
nContador = 0
cCodigo = "16A87E"
```

## Instrucción de Lectura

Se puede asignar valores a variables por medio de la instrucción de lectura.

### Sintaxis

```
accept nombre_variable
```

Los siguientes son algunos ejemplos:

```
character cNombre  
numeric nEdad  
  
display "Ingrese su nombre"  
accept cNombre  
display "Ingrese su edad"  
accept nEdad
```

## Operadores

---

Los operadores determinan el tipo de operación que se quiere realizar con los elementos de una expresión. En una expresión, el elemento sobre el cual actúa un operador se llama operando. Por ejemplo, en la expresión,  $a + b$ ,  $a$  y  $b$  son conocidos como operandos.

Los operadores pueden ser clasificados en las siguientes categorías:

- Operadores aritméticos
- Operadores Relacionales
- Operadores lógicos

### Operadores Aritméticos

Los operadores aritméticos, como su nombre lo indica, son utilizados para realizar cálculos aritméticos. Algunos de los operadores aritméticos más comunes son los siguientes:

Operador	Descripción	Ejemplo
Suma (+)	Suma los operandos	$c = a + b$
Resta (-)	Resta el operando derecho del operando izquierdo	$c = a - b$
Multiplicación (*)	Multiplica los operandos	$c = a * b$
División (/)	Divide el operando izquierdo por el operando derecho	$c = a / b$
Modulo (%)	Calcula el residuo de una división entera	$c = a \% b$

El siguiente pseudocódigo representa una operación usando el operador modulo:

```
begin
  numeric nNum1, nNum2, nNum3
  nNum1 = 15
  nNum2 = 2
  nNum3 = nNum1 % nNum2
  display nNum3
end
```

En este pseudocódigo, a las variables `nNum1` y `nNum2` se le asigna 15 y 2 respectivamente. La salida del pseudocódigo será 1, el cual es el residuo de la división entera entre `nNum1` y `nNum2`.



## Operadores Relacionales

Se puede comparar dos operandos con el operadores relacional. Cuando dos operandos son comparados usando estos operadores, el resultado es un valor lógico, TRUE o FALSE.

Son seis operadores relacionales. La siguiente tabla muestra los operadores relacionales:

Operador	Descripción	Ejemplo	Explicación
=	Evalúa si los operandos son iguales	$a = b$	Retorna TRUE si los valores son iguales y FALSE en caso contrario.
!=	Evalúa si los operandos son diferentes	$a \neq b$	Retorna TRUE si los valores son diferentes y FALSE en caso contrario.
>	Evalúa si el operando de la izquierda es mayor que el operando de la derecha	$a > b$	Retorna TRUE si a es mayor que b y FALSE en caso contrario
<	Evalúa si el operando de la izquierda es menor que el operando de la derecha	$a < b$	Retorna TRUE si a es menor que b y FALSE en caso contrario
>=	Evalúa si el operando de la izquierda es mayor o igual que el operando de la derecha.	$a \geq b$	Retorna TRUE si a es mayor o igual que b y FALSE en caso contrario
<=	Evalúa si el operando de la izquierda es menor o igual que el operando de la derecha	$a \leq b$	Retorna TRUE si a es menor o igual que b y FALSE en caso contrario

## Operadores Lógicos

Los operadores lógicos son usados para combinar los resultados de expresiones que contienen operadores relacionales.

A continuación tenemos una tabla que describe los operadores lógicos:

Operador	Descripción	Ejemplo
AND	Lógica AND	$a < 5 \text{ AND } b > 10$
OR	Lógica OR	$a < 5 \text{ OR } b > 10$
NOT	Lógica NOT	$\text{NOT } a = 5$

En la siguiente tabla se describe como trabaja el operador AND y el operador OR:

Expresión 1	Expresión 2	Valor de la Expresión Combinada	
		AND	OR
TRUE	TRUE	TRUE	TRUE
TRUE	FALSE	FALSE	TRUE
FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE

## Precedencia de Operadores

Cada operador tiene una precedencia asociada. Se utiliza esta característica para determinar la manera en que es evaluada una expresión que implica más de un operador. Por ejemplo, consideremos la siguiente expresión:

```
nResultado = nNum1 + nNum2 * nNum3 / 45
```

Para obtener el resultado correcto de tal expresión, es necesario saber la prioridad o precedencia de cada operador.

La precedencia tiene diferentes niveles desde 1 hasta 8. Estos niveles determinan el orden de evaluación de la expresión. Cada operador pertenece a uno a un solo nivel y más de un operador pueden pertenecer al mismo nivel. Los operadores de más alta precedencia se evalúan primero. Los operadores del mismo nivel de precedencia se evalúan de izquierda a derecha en una expresión. Esto se conoce como asociatividad. La tabla siguiente lista los operadores en orden decreciente de precedencia.

Operador	Descripción	Asociatividad	Nivel de Precedencia
( )	Paréntesis		1
!	Lógica NOT		2
*	Multipliación	De Izquierda a Derecha	3
/	División		
%	Módulo		
+	Suma	De Izquierda a Derecha	4
-	Resta		
<	Menor que	De Izquierda a Derecha	5
<=	Menor o Igual que		
>	Mayor que		
>=	Mayor o Igual que		
=	Igual que	De Izquierda a Derecha	6
!	Diferente que		
AND	Lógica "Y"	De Izquierda a Derecha	7
OR	Lógica "O"	De Izquierda a Derecha	8

Esta tabla muestra la precedencia y la asociatividad de los operadores. El orden de precedencia y la asociatividad de los operadores tienen que ser evaluados mientras construimos la expresión para obtener la salida deseada. Por ejemplo, consideremos la siguiente expresión:

```
nResultado = nNum1 + nNum2 * nNum3 / 45
```

De acuerdo a las reglas de precedencia, el operador de multiplicación, "\*", tiene la precedencia mas alta que al operador suma "+" y el operador de división "/". Por lo tanto la multiplicación de nNum1 y nNum2 se ejecuta primero. Asumiendo que los valores de nNum1 es 8, nNum2 es 9 y nNum3 es 10, el resultado de la expresión será:

```
nResultado = 8 + 90 / 45
```

El operador "/" tiene precedencia alta que "+". Por lo tanto, 90/45 es evaluado primero. El resultado de la expresión es:

```
nResultado = 8 + 2
```

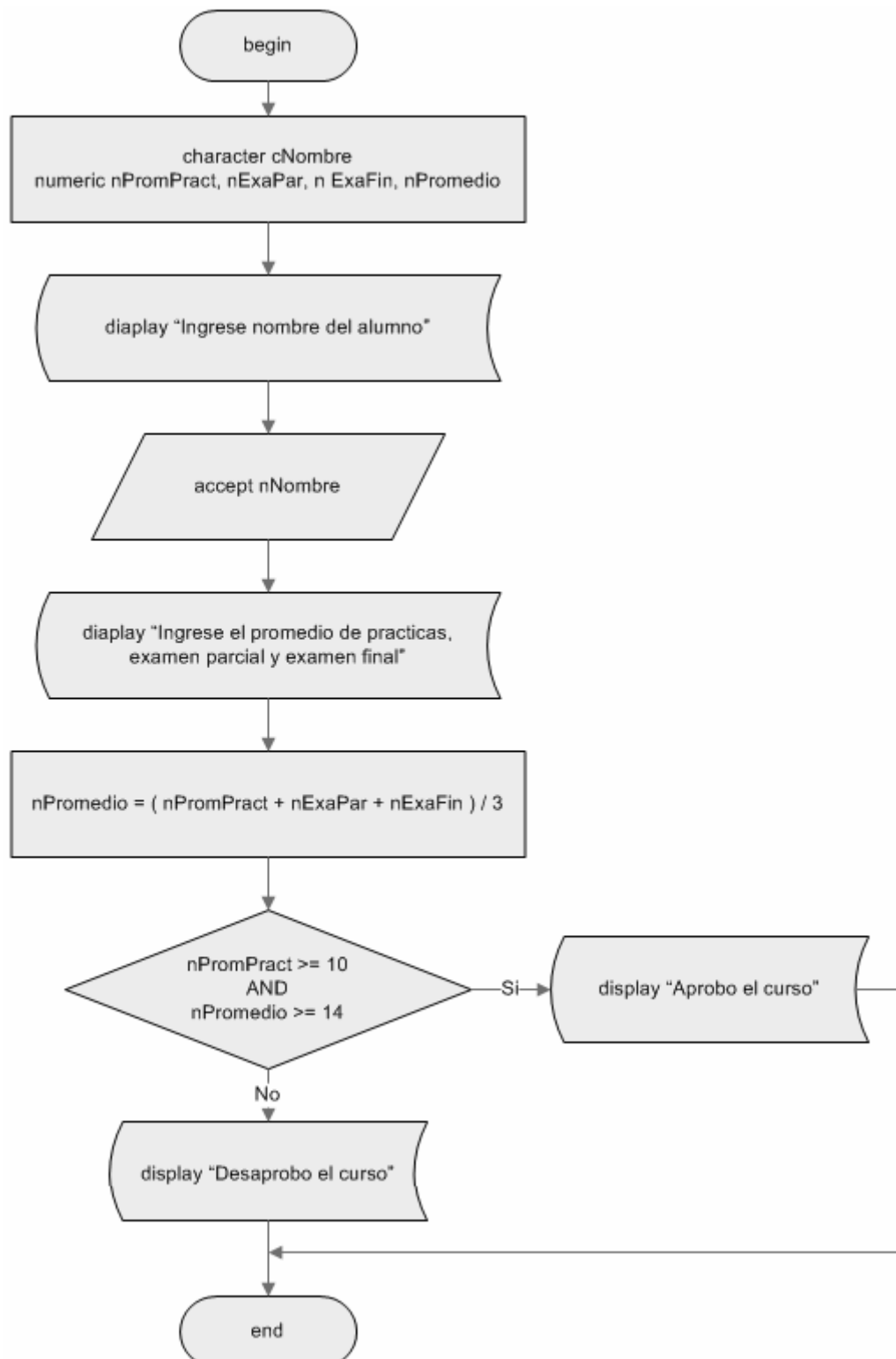
El resultado final es:

```
nResultado = 10
```

En la expresión anterior, supongamos que el propósito era primero sumar los números nNum1 y nNum2, y después multiplicar su resultado con nNum3. Finalmente, dividir el resultado por 45. Para resolver este propósito, se puede cambiar la orden de precedencia usando el operador paréntesis "()", tal como sigue:

```
nResultado = ( nNum1 + nNum2 ) * nNum3 / 45
```

El siguiente diagrama de flujo representa la lógica para determinar si un alumno aprueba el curso de Base de Datos:



El diagrama de flujo lee el nombre del alumno, y las notas obtenidas en el curso. Luego calcula el promedio. Para que el alumno apruebe el curso debe cumplirse dos condiciones, la primera que la nota mínima en promedio de practicas debe ser 10, y el promedio final del curso debe ser mínimo 14.

## Ejemplos

---

### Ejemplo 1

Determinar la suma de los N primeros números enteros de acuerdo a la siguiente formula:

$$Suma = \frac{N * (N + 1)}{2}$$

#### Pseudocódigo

```
begin  
  
    numeric nN, nSuma  
    display "Ingrese el valor de N"  
    accept nN  
    nSuma = n * (n + 1) / 2  
    display nSuma  
  
end
```

#### Programación en Java

```
import java.util.Scanner;  
  
public class Ejemplo01 {  
  
    public static void main(String[] args){  
  
        Scanner teclado = new Scanner(System.in);  
        int n, suma;  
        System.out.print("Ingrese el valor de N: ");  
        n = teclado.nextInt();  
        suma = n * (n+1) / 2;  
        System.out.println("La suma es: " + suma);  
  
    }  
  
}
```

## Ejemplo 2

Crear un programa para encontrar el Área de un Círculo, según la siguiente formula:

$$Area = \pi * R^2$$

### Pseudocódigo

```
begin  
  
    numeric radio, area  
    display "Ingrese el valor del radio"  
    accept radio  
    area = 3.141516 * radio * radio  
    display area  
  
end
```

### Programación en Java

```
import java.util.Scanner;  
  
public class Ejemplo02 {  
  
    public static void main(String[] args) {  
  
        Scanner teclado = new Scanner(System.in);  
        int radio;  
        double area;  
        System.out.print("Ingrese el valor del radio: ");  
        radio = teclado.nextInt();  
        area = Math.PI * radio * radio;  
        System.out.println("El area es: " + area);  
  
    }  
  
}
```

## Ejemplo 3

Un docente del instituto San Ignacio de Loyola lleva a cabo una prueba de SQL Server para tres estudiantes: Marcelo, Carlos y Manuel. El ha de comparar el puntaje obtenido por los tres estudiantes, y mostrar el nombre del estudiante que sacó el mayor puntaje. El siguiente pseudocódigo representa el algoritmo correspondiente a este problema.

### Pseudocódigo

```
begin

    numeric nNota1, nNota2, nNota3
    display "Ingrese la nota de Marcelo"
    accept nNota1
    display "Ingrese la nota de Carlos"
    accept nNota2
    display "Ingrese la nota de Manuel"
    accept nNota3
    if nNota1 > nNota2 AND nNota1 > nNota3
        display "Marcelo tiene la nota mas alta"
    if nNota2 > nNota1 AND nNota2 > nNota3
        display "Carlos tiene la nota mas alta"
    if nNota3 > nNota1 AND nNota3 > nNota2
        display "Manuel tiene la nota mas alta"

end
```



## Programación en Java

```
import java.util.Scanner;

public class Ejemplo03 {

    public static void main(String[] args) {

        // Variables
        Scanner teclado = new Scanner(System.in);
        int nota1, nota2, nota3;

        // Lectura de Datos
        System.out.print("Ingrese la nota de Marcelo: ");
        nota1 = teclado.nextInt();
        System.out.print("Ingrese la nota de Carlos: ");
        nota2 = teclado.nextInt();
        System.out.println("Ingrese la nota de Manuel: ");
        nota3 = teclado.nextInt();

        // Proceso
        if(nota1 > nota2 & nota1 > nota3)
            System.out.println("Marcelo tiene la nota mas alta");
        if(nota2 > nota1 & nota2 > nota3)
            System.out.println("Carlos tiene la nota mas alta");
        if(nota3 > nota1 & nota3 > nota1)
            System.out.println("Marcelo tiene la nota mas alta");

    }

}
```

## Ejemplo 4

Pacherrez Delivery International presta servicios de mensajería nacional e internacional. Las tarifas del servicio de mensajería son calculadas de acuerdo al peso y destino de los paquetes. La siguiente tabla muestra las distancias y la correspondiente tarifa de entrega.

Distancia en Km.	Costo en Soles por Kilo
0 a 500	50
501 a 10,000	100
mas de 10,0000	500

Desarrollar el programa para calcular y mostrar las tarifas de entrega en un pseudocódigo. Las sentencias en el pseudocódigo deberán leer la distancia a cubrir para enviar el paquete, y el peso del mismo, y también calcular la suma total que cobrará Pacherrez Delivery International por el envío del paquete.

### Pseudocódigo

```
begin

    numeric nDistancia, nPeso, nCosto
    display "Ingrese la distancia: "
    accept nDistancia
    display "Ingrese el peso del paquete: "
    accept nPeso
    if( nDistancia >= 0 AND nDistancia <= 500 )
        nCosto = nPeso * 50
    if( nDistancia >= 501 AND nDistancia <= 10,000 )
        nCosto = nPeso * 100
    if( nDistancia > 10,000 )
        nCosto = nPeso * 500
    display nCosto

end
```

## Programación en Java

```
import java.util.Scanner;

public class Ejemplo04 {

    public static void main(String[] args) {

        // Variables del Programa
        Scanner teclado = new Scanner(System.in);
        int distancia, peso, costo=0;

        // Lectura de Datos
        System.out.print("Ingrese distancia: ");
        distancia = teclado.nextInt();
        System.out.print("Ingrese peso: ");
        peso = teclado.nextInt();

        // Proceso
        if(distancia >= 0 & distancia <= 500)
            costo = peso * 50;
        if(distancia >= 501 & distancia <= 10000)
            costo = peso * 100;
        if(distancia > 10000)
            costo = peso * 500;

        // Reporte
        System.out.println("Costo = " + costo);

    }

}
```

## Ejercicios

---

### Ejercicio 1

Crear un programa que calcule la media aritmética de 3 números enteros positivos.

### Ejercicio 2

Crear un programa que calcule el sueldo Neto a pagar a un empleado, considerando los siguientes puntos:

- El pago por horas trabajadas, equivale a 25 dólares.
- El empleado debe abonar el 5% del pago por impuesto.
- Observación: El tipo de cambio debe ser ingresado manualmente.

### Ejercicio 3

Un proyecto internacional sobre clima y temperaturas, iniciado por el Departamento Meteorológico, requiere un listado conteniendo el nombre de tres ciudades con sus temperaturas. Los datos se registran en grados Celsius, y se requiere en grados Fahrenheit.

Desarrollar el programa que permita mostrar los datos en Fahrenheit.

### Ejercicio 4

Desarrollar un programa para calcular el importe que pagaría una promoción para ir de excursión al Cuzco, los datos de entrada son: número de alumnos, costo por persona.

### Ejercicio 5

Desarrollar un programa para encontrar el tiempo que se demora en desplazarse un auto de la ciudad A hacia la ciudad B, los datos de entrada son: la distancia en Km y la velocidad en Km.

### Ejercicio 6

Desarrollar un programa para determinar si el precio de venta de un artículo es barato ó caro, se considera caro si la ganancia es superior al 20% del precio de costo.

# Lección 04

## Estructuras Condicionales

### Contenido

- Estructura: if
- Estructura: switch...case
- Ejercicios

## Estructura: if

---

La estructura if permite tomar decisiones dentro de un programa. El resultado de una decisión determina la secuencia en que el programa puede ejecutar las instrucciones. La decisión se toma en base al resultado de una comparación.

Los tres tipos principales de if son los siguientes:

- Estructuras if simples
- Estructuras if...else
- Estructuras if...else anidadas

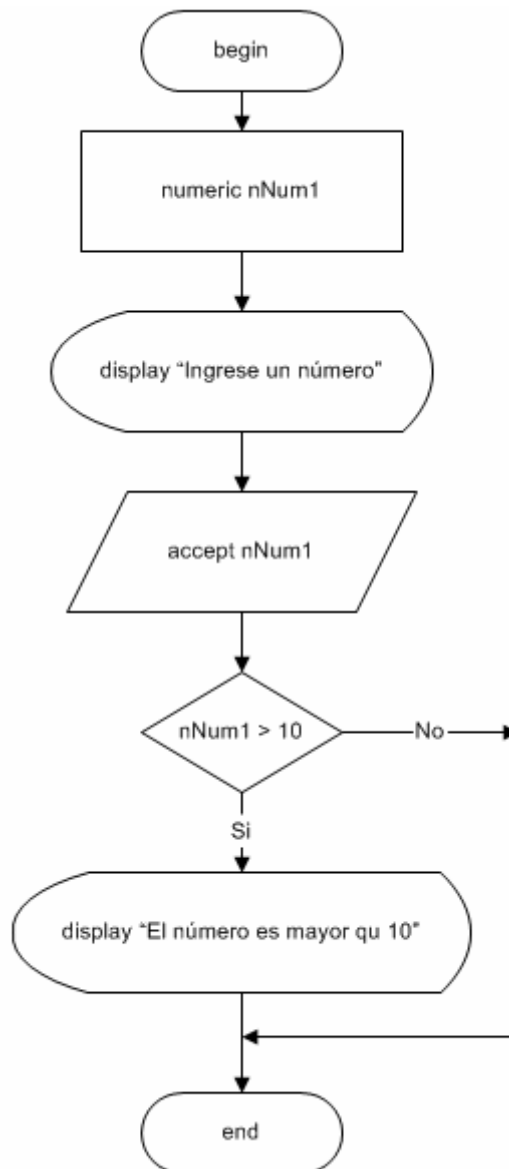
### Estructuras if Simple

#### Sintaxis

```
if ( condición )  
begin  
  
    // Sentencias  
  
end
```

Si la condición especificada en la estructura if simple es verdadera (true), las sentencias contenidas dentro del bloque if son ejecutadas.

El siguiente diagrama de flujo, muestra la logica para determinar si el número ingresado es mayor que 10 y muestra un mensaje respectivo.



En el siguiente pseudocódigo se ilustra el uso de la estructura if simple. Este pseudocódigo determina si un número ingresado es mayor que 10 y muestra un mensaje respectivo.

```
begin  
  
    numeric nNum1  
    display "Ingrese un número"  
    accept nNum1  
    if ( nNum1 > 10 )  
    begin  
        display nNum1  
        display "El número es mayor que 10"  
    end  
  
end
```

El siguiente es el programa en Java:

```
import java.util.Scanner;  
  
public class Ejemplo01 {  
  
    public static void main(String[] args) {  
  
        Scanner teclado = new Scanner( System.in );  
        int num;  
        num = teclado.nextInt();  
        if ( num > 10 ) {  
            System.out.println(num);  
            System.out.println("El número es mayor que 10");  
        }  
  
    }  
  
}
```



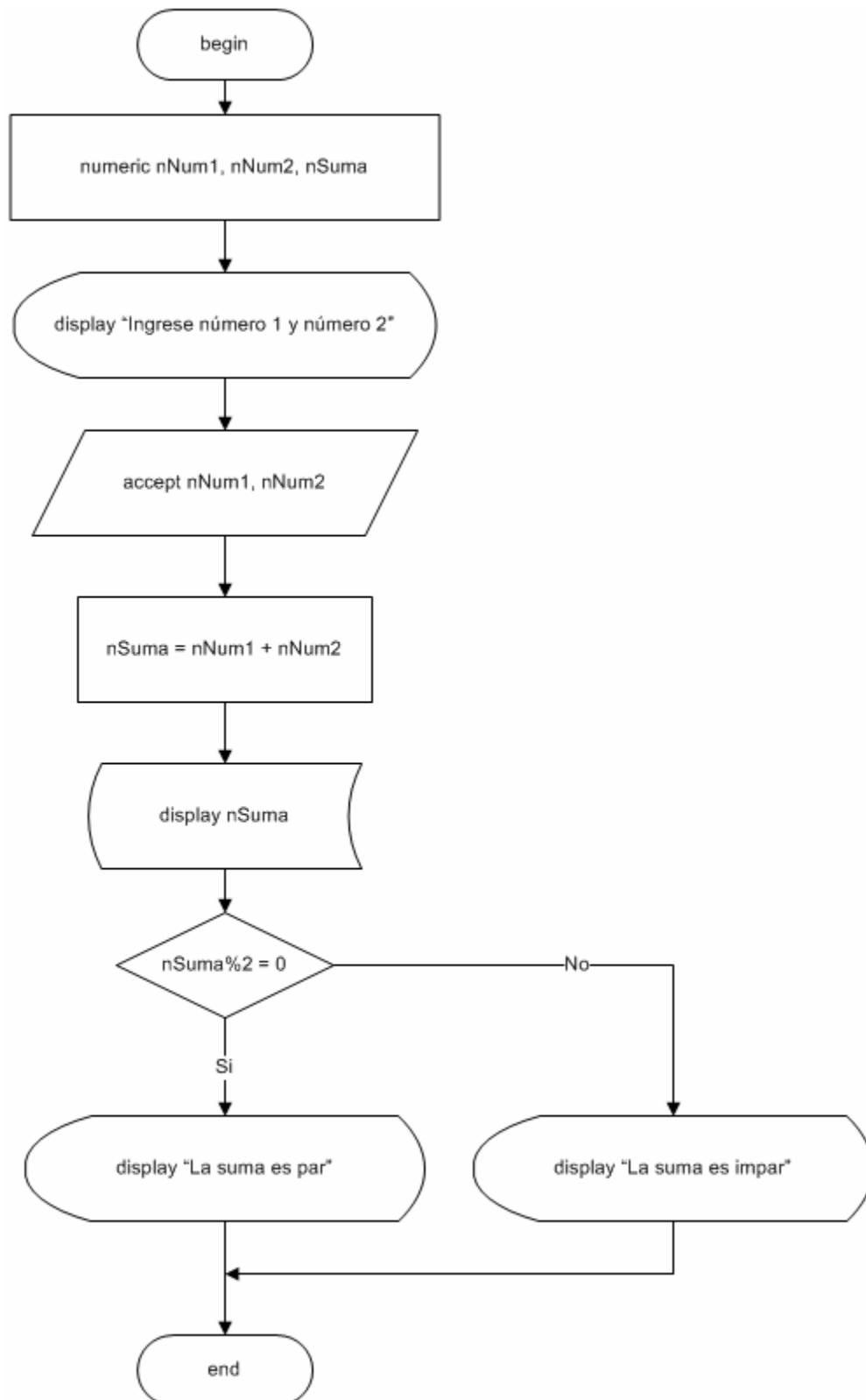
## Estructura if...else

### Sintaxis

```
If ( condición )  
begin  
    // Sentencias  
  
end  
else  
begin  
    // Sentencias  
end
```

En la estructura if...else, si la condición especificada es verdadera (true), las sentencias contenidas dentro del bloque if son ejecutadas. Si la condición es falsa (false), las sentencias contenidas dentro del bloque else son ejecutadas.

El siguiente diagrama de flujo, muestra la lógica para encontrar la suma de dos números y determinar si es par ó impar.



El mismo algoritmo puede también ser representado usando pseudocódigo, tal como se muestra a continuación:

```
begin

    // Variables del Programa
    numeric nNum1, nNum2, nSuma

    // Lectura de Datos
    display "Ingrese número 1 y número2"
    accept nNum1
    accept nNum2

    // Proceso
    nSuma = nNum1 + nNum2

    // Reporte
    display nSuma
    if ( nSuma % 2 = 0 )
    begin
        display "La suma es par"
    end
    else
    begin
        display "La suma es impar"
    end
end
```

El siguiente es el programa en Java:

```
import java.util.Scanner;

public class Ejemplo03 {

    public static void main(String[] args) {

        // Variables del Programa
        Scanner teclado = new Scanner( System.in );
        int num1, num2, suma;

        // Lectura de Datos
        System.out.print("Ingrese número 1: ");
        num1 = teclado.nextInt();
        System.out.print("Ingrese número 2: ");
        num2 = teclado.nextInt();

        // Proceso
        suma = num1 + num2;

        // Reporte
        System.out.println("Suma: " + suma);
        if (suma % 2 == 0) {
            System.out.println("La suma es Par");
        } else {
            System.out.println("La suma es Impar");
        }
    }
}
```

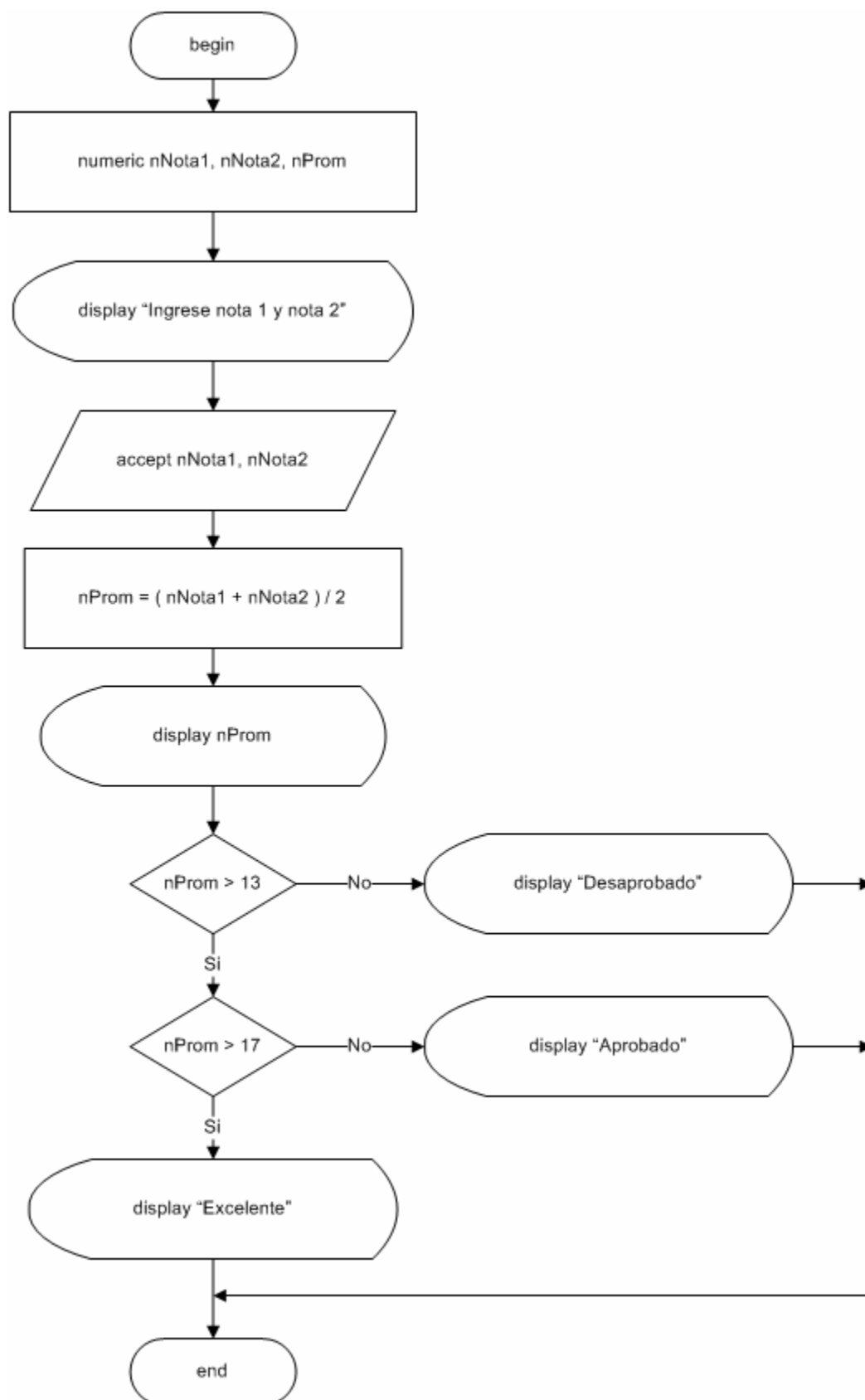
## Estructura if...else Anidada

En algunos casos, se necesita representar una condición que están basada en otra condición. En este caso, se puede usar una estructura if...else dentro de otra estructura if...else. Esta construcción es conocida como if...else anidado. El número de estructuras condicionales y el nivel de anidamiento dependen de la complejidad del problema. Por lo tanto, no hay un límite o regla para especificar el número de estructuras condicionales que se pueden anidar.

Por ejemplo, considerar un algoritmo para aceptar la nota de un alumno en dos cursos y calcular su promedio. El algoritmo muestra el promedio obtenido con el mensaje "Aprobado", "Desaprobado", o "Excelente". El criterio para el mensaje esta dado por:

- Excelente, si el promedio es mayor que 17
- Aprobado, si el promedio es mayor que 13
- Desaprobado, si el promedio es menor o igual que 13

El diagrama de flujo para este problema es el siguiente:



El mismo algoritmo puede también ser representado con estructuras if...else anidadas, tal como se muestra a continuación:

```
begin

    // Datos del Programa
    numeric nNota1, nNota2, nProm

    // Lectura de Datos
    display "Ingrese la nosta de los dos cursos"
    accept nNota1, nNota2

    // Proceso
    nProm = ( nNota1 + nNota2 ) / 2

    // Reporte
    display nProm
    if ( nProm > 13 )
    begin
        if( nProm > 17 )
        begin
            display "Excelente"
        end
        else
        begin
            display "Aprobado"
        end
    end
    else
    begin
        diaplay "Desaprobado"
    end

end
```

El siguiente es el programa en Java:

```
import java.util.Scanner;

public class Ejemplo04 {

    public static void main(String[] args) {

        // Variables del Programa
        Scanner teclado = new Scanner( System.in );
        double nota1, nota2, prom;

        // Lectura de Datos
        nota1 = teclado.nextDouble();
        nota2 = teclado.nextDouble();

        // Proceso
        prom = (nota1 + nota2) / 2;

        // Reporte
        System.out.println("Promedio: " + prom);
        if(prom > 13.0){
            if(prom > 17){
                System.out.println("Excelente");
            } else {
                System.out.println("Aprobado");
            }
        } else {
            System.out.println("Desaprobado");
        }
    }
}
```



## Estructura: switch...case

Otra estructura condicional disponible es, la estructura `switch...case`. Esta es usada cuando hay varios valores para una variable que deben ser evaluados.

La estructura `switch...case` permite seleccionar una opción desde un conjunto de alternativas. Esta estructura consiste de una sentencia `switch`, un número de sentencias `case`, y una sentencia por defecto.

### Sintaxis

```
switch ( expresión )
begin

    case constante_1:
        sentencia(s)
        break

    case constante_2:
        sentencia(s)
        break

    case constante_3:
        sentencia(s)
        break

    . . .
    . . .

    default:
        sentencia(s)

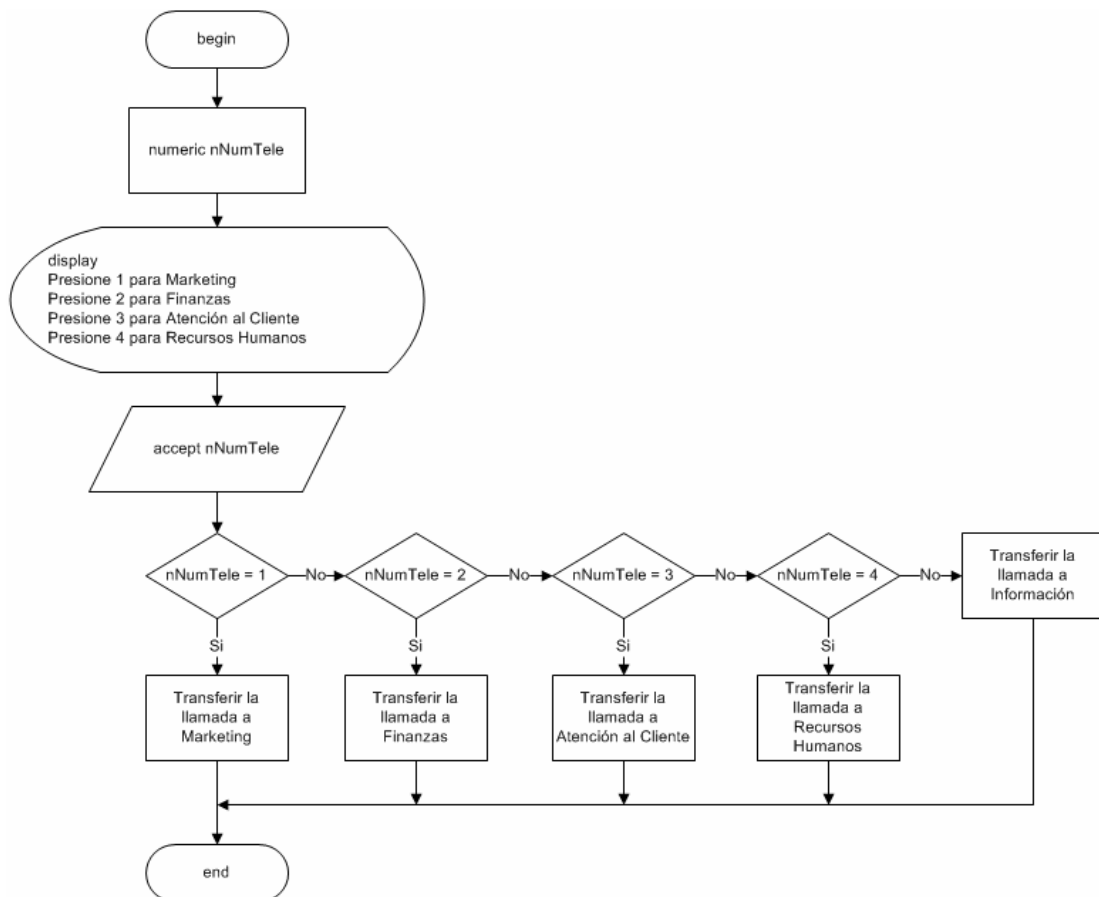
end
```

Cuando el pseudocódigo que usa la estructura `switch...case` es ejecutado, la expresión especificada en la palabra clave `switch` es evaluada. El valor de la expresión es igualado con los valores constantes especificados en cada caso (palabra clave `case`). Cuando el valor de la expresión es igual a uno de los valores constantes, las instrucciones dentro del bloque de ese caso son ejecutadas. Si ningún caso es igual al resultado de la expresión, las instrucciones del bloque `default` son ejecutadas.

La instrucción `break` es usada para salir del caso después que las instrucciones son ejecutadas. La sección `default` es opcional. La expresión especificada en `switch` puede ser numérica o carácter.

Considere el siguiente ejemplo para la estructura `switch...case`. Claudia esta escribiendo el algoritmo para transferir automáticamente las llamadas telefónicas para varios departamentos tales como Marketing, Finanzas, Atención al Cliente, Recursos Humanos, e Información.

El diagrama de flujo para el proceso es el siguiente:



El pseudocódigo para el mismo proceso usando la estructura `switch...case` es el siguiente:

```
begin

    // Variables
    numeric nNumTele

    // Lectura de Datos
    display  "Para comunicarse con Marketing presione 1, para Finanzas presione 2,
             para Atención al cliente presione 3, para Recursos Humanos presione 4.
             De lo contrario la llamada se transferirá al departamento de Información"
    accept nNumTele

    // Proceso
    switch ( nNumTele)
    begin

        case 1:
            Transferir la llamada al Departamento de Marketing
            break

        case 2:
            Transferir la llamada al Departamento de Finanzas
            break

        case 3:
            Transferir la llamada del Departamento de Atención al Cliente
            break

        case 4:
            Transferir la llamada al Departamento de Recursos Humanos
            break

        default:
            Transferir la llamada al Departamento de Información

    end

end
```

El siguiente es el programa en Java:

```
import java.util.Scanner;

public class Ejemplo05 {

    public static void main(String[] args) {

        // Variables
        Scanner teclado = new Scanner( System.in );
        int numTele;

        // Datos
        System.out.println("Para comunicarse con Marketing presione 1, para Finanzas presione 2,");
        System.out.println("para Atención al cliente presione 3, para Recursos Humanos presione 4.");
        System.out.println("De lo contrario la llamada se transferirá al departamento de Información");
        numTele = teclado.nextInt();

        // Proceso
        switch(numTele){

            case 1:
                System.out.println("Su llamada ha sido transferida a Marketing");
                break;
            case 2:
                System.out.println("Su llamada ha sido transferida a Finanzas");
                break;
            case 3:
                System.out.println("Su llamada ha sido transferida a Atención al cliente");
                break;
            case 4:
                System.out.println("Su llamada ha sido transferida a Recursos Humanos");
                break;
            default:
                System.out.println("Su llamada ha sido transferida a Información");

        }

    }

}
```

## Ejercicios

### Ejercicio 1

El restaurante "El Sabor Norteño" ofrece un descuento del 10 % para consumos de hasta S/.100.00 y un descuento de 20% para consumos mayores, para ambos casos se aplica un impuesto del 19%. Determinar el importe a pagar por lo consumido, mostrando todos los importes (subtotal, impuesto, y total).

### Ejercicio 2

Elabore un programa para determinar si un número entero A es divisible por otro B. Considere que un número es divisible por otro si y solo si el residuo de la división es cero.

### Ejercicio 3

Debido a los excelentes resultados, el restaurante "El Sabor Norteño" decide ampliar sus ofertas de acuerdo a la siguiente escala de consumo. Determinar el importe a pagar por lo consumido, mostrando todos los importes.

Consumo (S/.)	Descuento (%)
Mayor a 200	30
Mayor a 100	20
Hasta 100	10

### Ejercicio 4

Elabore un programa que encuentre el mayor de 3 números dados.

### Ejercicio 5

Crear un programa que compruebe si un número ingresado es Par o Impar.

### Ejercicio 6

Crear un programa que determine el Menor número de 5 números ingresados.

### Ejercicio 7

Crear un programa que calcule el sueldo neto de un trabajador según el número de horas trabajadas, considerando que si excede a 40 horas se le paga 15% más del pago por hora solo por las horas extras y si el sueldo excede a 2000, tiene que abonar un impuesto de 5% del sueldo bruto.

## Ejercicio 8

El rendimiento de un alumno se califica según lo siguiente:

Calificación	condición
Bueno	si su promedio esta entre 16 y 20
Regular	si su promedio esta entre 11 y 15
Deficiente	si su promedio esta entre 6 y 10
Pésimo	si su promedio esta entre 0 y 5

Crear un programa que lea el promedio de un alumno y diga cuál es su rendimiento.

## Ejercicio 9

La comisión sobre las VENTAS de un empleado es como sigue:

Comisión	Condición
No hay comisión	VENTAS menores a 50
10%	VENTAS entre 50 y 500
20%	VENTAS mayores que 500

Crear un programa que lea el importe de las VENTAS de un empleado y calcule el importe de su Comisión.

## Ejercicio 10

Crear un programa tipo calculadora que permita ingresar dos números y una letra que indica la operación a realizar (S, R, M, D).

## Ejercicio 11

En una Universidad se ha establecido los siguientes puntajes de ingreso a sus respectivas facultades:

Facultad	Puntaje Mínimo
Sistemas	100
Electrónica	90
Industrial	80
Administración	70

De acuerdo al puntaje obtenido por un postulante determinar la facultad a la cual ingresó o dar un mensaje correspondiente para el caso que no ingrese.

## Ejercicio 12

Crear un programa que lea un número que represente a un mes y muestre el nombre del mes al que corresponde, por ejemplo 1 es Enero.

## Ejercicio 13

Crear un programa que ingresado una fecha determine a que estación pertenece (Verano, Otoño, Invierno, Primavera).

## Ejercicio 14

Crear un programa que lea el día y el mes de nacimiento de una persona, y determine a que signo pertenece.

# Apuntes



# Lección 05

## Estructuras Repetitivas

### Contenido

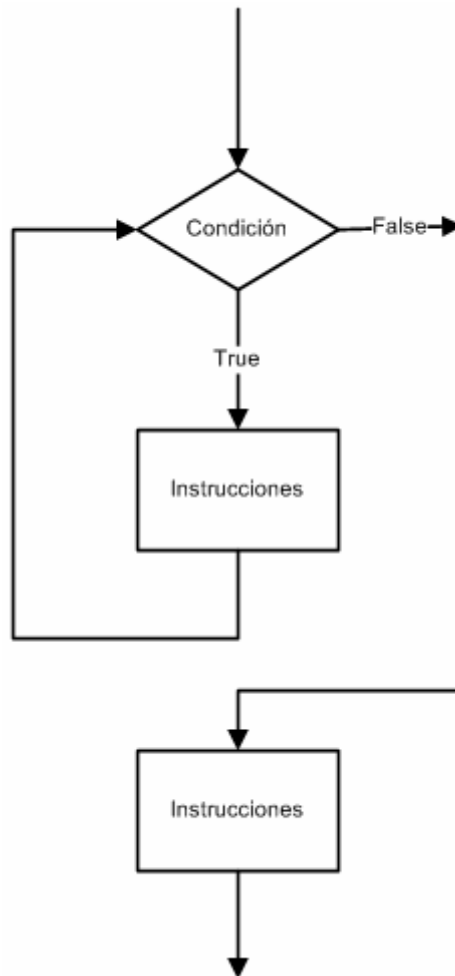
- Estructura: while
- Estructura: for
- Ejercicios Propuestos

## Estructura: while

---

La estructura `while` ejecuta un grupo de instrucciones mientras se cumple una condición. Para que se ingrese al cuerpo del bucle, debe cumplirse la condición, si la primera vez que la condición evaluada da como resultado **false**, el bucle no se ejecuta ninguna vez.

### Diagrama de Flujo



### Sintaxis

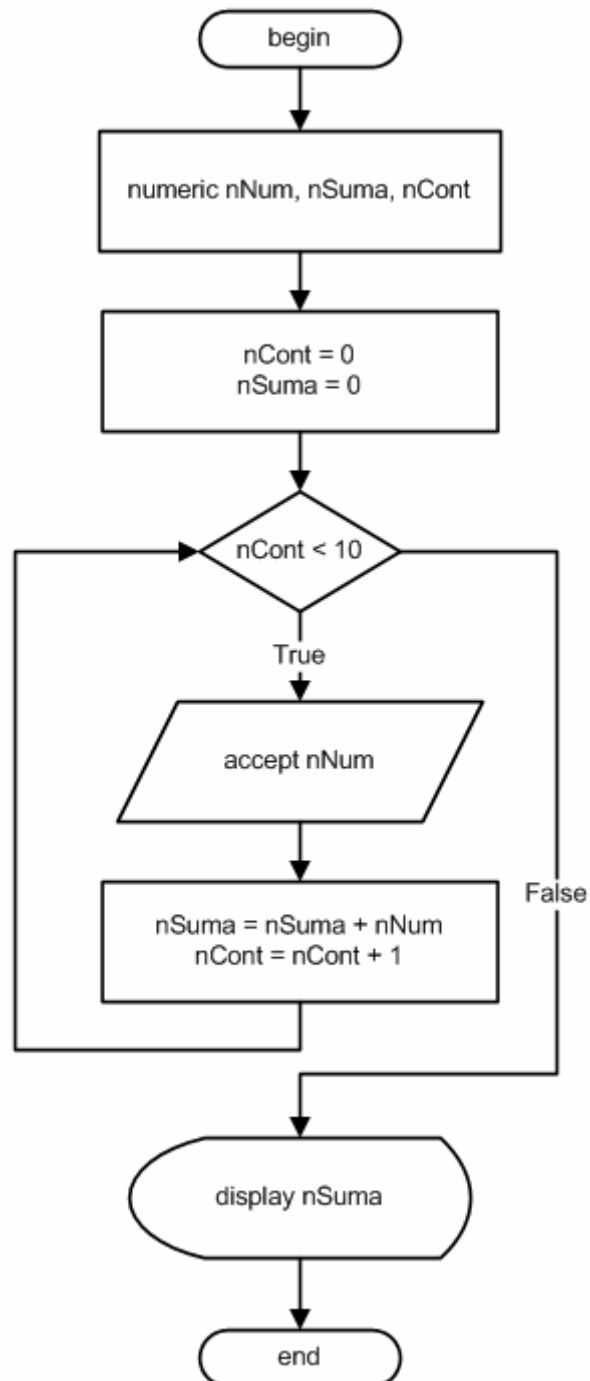
```
while ( condición )  
begin  
    // Sentencias  
end
```

## Ejemplo 1

### Enunciado

Encontrar la suma de 10 números ingresados por teclado.

### Diagrama de Flujo



**Pseudocódigo**

```
begin

    // Datos del Programa
    numeric nNum, nSum, nCont

    // Proceso
    nCont = 0
    nSuma = 0
    while ( nCont < 10 )
    begin
        display "Ingrese número entero"
        accept nNum
        nSuma = nSuma + nNum
        nCont = nCont + 1
    end

    // Reporte
    display "La suma es:" + nSuma

end
```

La siguiente tabla se puede utilizar para probar el funcionamiento del bucle:

Iteración	nNum	nSuma	nCont
Valores Iniciales	null	0	0
1	15	15	1
2	10	25	2
3	20	45	3
4	30	75	4
5	10	85	5
6	15	100	6
7	20	120	7
8	10	130	8
9	25	155	9
10	15	<b>170</b>	10

Se puede comprobar que el valor de la suma final es 170.

**Programación en Java**

```
import java.util.Scanner;

public class Ejemplo1 {

    public static void main(String[] args) {

        // Variables del programa
        Scanner teclado = new Scanner(System.in);
        int num, suma, cont;

        // Proceso
        cont = 0;
        suma = 0;
        while( cont < 10 ){
            System.out.print("Ingrese numero entero: ");
            num = teclado.nextInt();
            suma = suma + num;
            cont = cont + 1;
        }

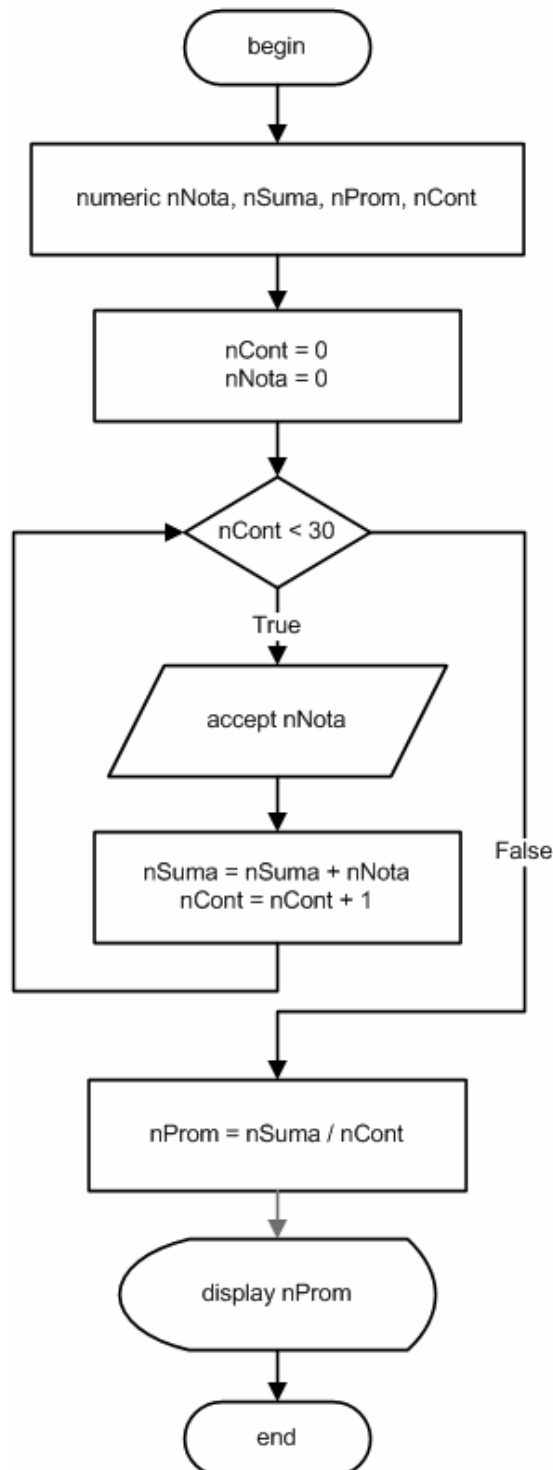
        // Reporte
        System.out.println("La suma es: " + suma);

    }
}
```

## Ejemplo 2

Ernesto es docente del curso de *Fundamentos Web* y quiere calcular la nota promedio de su clase, se sabe que son 20 alumnos.

### Diagrama de Flujo



**Pseudocódigo**

```
begin

    // Datos del programa
    numeric nNota, nSuma, nProm, nCont

    // Inicialización de Variables
    nSuma = 0
    nCont = 0

    // Proceso
    while( nCont < 30 )
    begin
        display "Ingrese Nota:"
        accept nNota
        nSuma = nSuma + nNota
        nCont = nCont + 1
    end
    nProm = nSuma / nCont

    // Reporte
    display "La nota promedio es:"
    display nProm

end
```

## **Programación en Java**

```
import java.util.Scanner;

public class Ejemplo2 {

    public static void main(String[] args) {

        // Variables del Programa
        Scanner teclado = new Scanner(System.in);
        int nota, suma, prom, cont;

        // Inicialización de Variables
        suma = 0;
        cont = 0;

        // Proceso
        while( cont < 3 ){
            System.out.println("Ingrese nota:");
            nota = teclado.nextInt();
            suma = suma + nota;
            cont = cont + 1;
        }
        prom = suma / cont;

        // Reporte
        System.out.println("La nota promedio es: " + prom);

    }

}
```



## Estructura: for

El bucle **for** se utiliza cuando el número de iteraciones del bucle se conoce de manera anticipada.

El bucle **for** tiene la siguiente sintaxis:

```
for( expr_inicialización; expr_evaluación; expt_incremento/decremento )  
begin  
    // Instrucciones  
end
```

Se compone de las siguientes tres partes separadas por punto y coma:

- **Expresión de inicialización:** La variable numérica se inicializa con un valor.
- **Expresión de evaluación:** La condición se comprueba al comienzo de la repetición del bucle. Cuando la expresión es falsa, el bucle finaliza.
- **Expresión de incremento/decremento:** El valor de la variable se incrementa ó decrementa.

Por ejemplo, si queremos mostrar el mensaje **Alianza Campeón** 10 veces, el bucle sería:

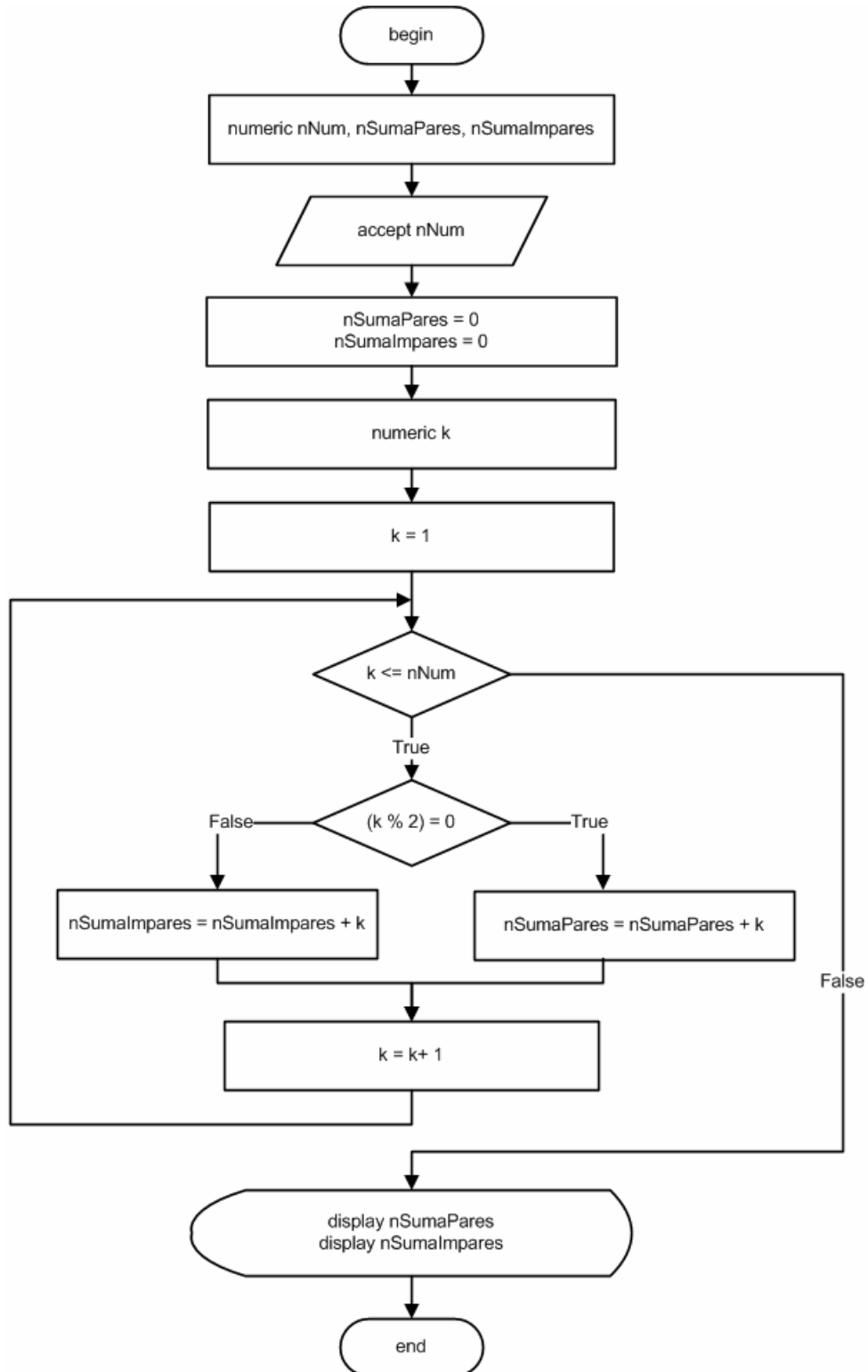
```
for( nCont = 1; nCont <= 10; nCont = nCont + 1 )  
begin  
    display "Alianza Campeón"  
end
```

En un bucle **for**, la inicialización de una variable, la evaluación de la condición y el aumento del valor de la variable quedan especificados en una sentencia.

### Ejemplo 3

Desarrollar un programa para encontrar la suma de los números pares e impares comprendidos entre 1 y **num**. El valor de **num** es el dato de entrada.

#### Diagrama de Flujo



**Pseudocódigo**

```
begin

    // Datos del Programa
    numeric nNum, nSumaPares, nSumaImpares, k

    // Lectura de Datos
    display "Ingrese el valor de N:"
    accept nNum

    // Inicialización de Variables
    nSumaPares = 0
    nSumaImpares = 0

    // Proceso
    for( k=1; k <= nNum; k = k + 1 )
    begin
        if( (k % 2) = 0 )
        begin
            nSumaPares = nSumaPares + k
        end
        else
        begin
            nSumaImpares = nSumaImpares + k
        end
    end

    // Reporte
    display "La suma de los números pares es: "
    display nSumaPares
    display "La suma de los números impares es: "
    display nSumaImpares

end
```

## **Programación en Java**

```
import java.util.Scanner;

public class Ejemplo4 {

    public static void main(String[] args) {

        // Variables del Programa
        Scanner teclado = new Scanner(System.in);
        int num, sumaPares, sumaImpares;

        // Lectura de num
        System.out.println("Ingrese el valor de num: ");
        num = teclado.nextInt();

        // Inicialización de variables
        sumaPares = 0;
        sumaImpares = 0;

        // Proceso
        for( int k = 1; k <= num; k++ ){
            if( (k%2) == 0 )
                sumaPares += k;
            else
                sumaImpares += k;
        }

        // Reporte
        System.out.println("Suma de Pares: " + sumaPares);
        System.out.println("Suma de Impares: " + sumaImpares);

    }

}
```

## Ejercicios Propuestos

---

### Ejercicio 1

Crear un programa que permita ingresar un número y muestre el número en forma invertida, por ejemplo al ingresar 12345 deberá mostrarlo en forma invertida 54321.

### Ejercicio 2

Dado un número positivo, mostrar los divisores del número ingresado.

### Ejercicio 3

Crear un programa que permita convertir un número de Base 10 a Base 16.

### Ejercicio 4

Escribir un programa que permita imprimir los 10 primeros números múltiplos de 7.

### Ejercicio 5

Escribir un programa que dado un número entero, muestre el menor y el mayor dígito que lo conforma.

### Ejercicio 6

El factorial de un número es:

$$n! = 1 \times 2 \times 3 \times 4 \times \dots \times n$$

Escribir un programa que permita calcular el factorial de un número.

### Ejercicio 7

Un número es primo cuando es divisible por uno y por sí mismo. Escribir un programa que permita evaluar si un número es primo.

### Ejercicio 8

Escribir un programa que muestre los divisores primos de un número.

### Ejercicio 9

Escribir un programa que muestre los **N** primeros términos de la serie de fibonacci.

## Ejercicio 10

Escribir un programa que imprima la tabla de multiplicar de un número.

## Ejercicio 11

Escribir un programa que muestre los **N** primeros números múltiplos de 5 que no son múltiplos de 3, y además debe mostrar la suma de todos ellos.

## Ejercicio 12

Escribir un programa para determinar los números múltiplos de **P** que hay entre **M** y **N**, donde: **M** < **N**.

## Ejercicio 13

Escribir un programa que determine cuantos dígitos tiene un número.

## Ejercicio 14

Escribir un programa que permita determinar si dos números son amigos.

Dos números son amigos si la suma de sus divisores de uno de ellos es igual al otro y viceversa, por ejemplo 220 y 284 son amigos:

Divisores de 220 son:  $1 + 2 + 4 + 5 + 10 + 11 + 20 + 22 + 44 + 55 + 110 = 284$

Divisores de 284 son:  $1 + 2 + 4 + 71 + 142 = 220$

## Ejercicio 15

Desarrollar un programa que permita calcular la suma de los cuadrados de los primeros **N** números.

## Ejercicio 16

Desarrollar un programa que permita calcular la suma de los cubos de los **N** primeros números.

## Ejercicio 17

Crear un algoritmo que indique si un número es cubo perfecto (anstrong) o no, se dice que un número es cubo perfecto si al sumar los cubos de sus dígitos dan el mismo número, por ejemplo 153, cubos de sus dígitos  $1^3 + 5^3 + 3^3 = 153$ , por lo tanto el número 153 es cubo perfecto.

## Ejercicio 18

Desarrollar un programa para obtenga el cociente y el residuo de una división mediante restas sucesivas.

Por ejemplo si el dividendo es 3989 y el divisor es 1247, entonces:

Iteración	Operación
1	$3989 - 1247 = 2742$ R(1)
2	$2742 - 1247 = 1495$ R(2)
3	$1495 - 1247 = 248$ R(3)

Ya no se puede seguir restando, pues 248 es menor a 1247, entonces el cociente es el número de iteraciones (3) y el residuo es el valor de la última resta (248).

## Ejercicio 19

Escribir un programa para convertir un número de base 10 a base 2.

## Ejercicio 20

Escribir un programa para obtener el MCD (máximo común divisor) de dos números, utilice el método EUCLIDES (divisiones sucesivas).

## Ejercicio 21

Escribir un programa para obtener el MCD (máximo común divisor) de dos números, utilice el método Factorización Simultanea.

**Recuerde:** El máximo común divisor es el divisor mayor común de todos ellos.

## Ejercicio 22

Escribir un programa para obtener el MCM (mínimo común múltiplo) de dos números, utilice descomposición simultanea.

**Recuerde:** El mínimo común múltiplo es el múltiplo menor común de todos ellos.

## Ejercicio 23

Escriba un programa que calcule, la suma de la siguiente serie:

$$\frac{1}{2} + \frac{2}{3} + \frac{3}{4} + \frac{4}{N}$$

Hasta el número entero positivo N ingresado.

## Ejercicio 24

Escriba un programa que calcule, la suma de la siguiente serie:

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{6} + \frac{1}{N}$$

Hasta el número entero positivo N ingresado.

## Ejercicio 25

Crear un programa que determine si un número es perfecto o no.

Un número es perfecto si la suma de sus divisores es igual al número, por ejemplo 6 tiene como divisores 1, 2 y 3, entonces  $1 + 2 + 3 = 6$  el número 6 es perfecto, si el número es 9 tiene como divisores 1, 3, entonces  $1 + 3 = 4$  no es perfecto.

## Ejercicio 26

Desarrollar un programa que permita encontrar el cuadrado de un número usando la siguiente sumatoria:

$$N^2 = 1 + 3 + 5 + \dots + (2N - 1)$$



# Lección 06

## Procedimientos y Funciones

### Contenido

- Enfoque Modular a la Programación
- Procedimientos
- Funciones
- Alcance de las Variable
- Ejercicios

## Enfoque Modular a la Programación

---

En las etapas iniciales, los programas se consideraban aceptables siempre y cuando funcionaran y cumplieran con su labor.

A mediados de los 60, el movimiento para incluir estructuras especialmente diseñadas en los lenguajes de programación motivó que los programadores adoptasen un punto de vista más disciplinado al crear programas.

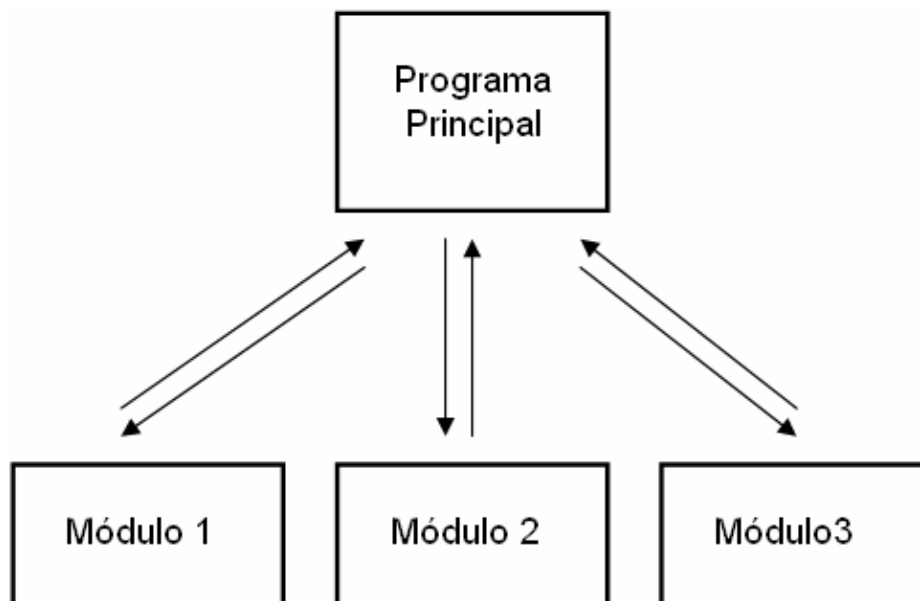
La programación estructurada incluye características diseñadas no sólo para resolver el problema inmediato sino también para que la lógica sea clara.

Algunas técnicas también aportan formas de dividir programas largos y continuos en una serie de módulos individuales que están relacionados entre sí de una manera determinada.

Una aplicación consta habitualmente de tareas integradas. Casi todas las aplicaciones están diseñadas como un grupo de módulos pequeños. Estos módulos son completos y pueden integrarse en un programa principal.

Los módulos son bloques construidos dentro de un programa. Cada módulo puede ser llamado desde el programa principal.

En un lenguaje de programación, estos módulos también se denominan subrutinas, subprogramas o procedimientos.

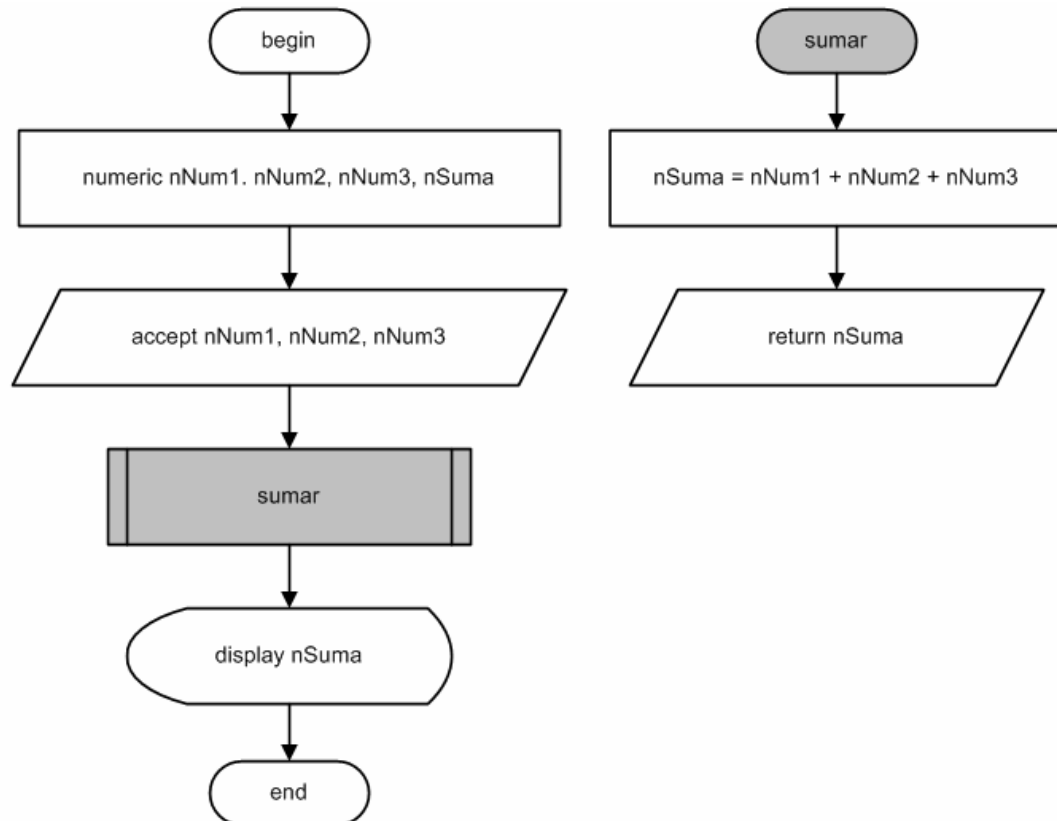


## Ejemplo 1

Consideremos una situación en la cual se necesita aceptar el ingreso de tres números, y mostrar la suma de estos tres números.

Podemos crear un diagrama de flujo simple que acepte tres números, los sume, y muestre el resultado.

Otro método es dividir la aplicación y crear un módulo separado para calcular la suma de los tres números. Este módulo puede ser integrado dentro del programa principal.



La ejecución del programa inicia con la instrucción **begin**. El programa se ramifica cuando el módulo **sumar** es llamado, y el control se transfiere al módulo **sumar**. Los números son sumados dentro del módulo, y el control es retornado al programa principal usando la instrucción **return**. El resto del programa se ejecuta secuencialmente. Cada módulo finaliza con una instrucción **return** que permite retornar el control al programa principal.

## Procedimientos

---

En el enfoque modular, uno de los métodos de representación de un módulo es el procedimiento. Un procedimiento es un conjunto de instrucciones que ejecutan una tarea específica. El funcionamiento de un procedimiento se conoce bajo el nombre **mecanismo llamada-retorno**.

Los siguientes pasos forman parte del mecanismo llamada-retorno:

- Se llama a un procedimiento.
- El conjunto de instrucciones almacenadas dentro del procedimiento se ejecutan.
- El control es devuelto al código de llamada.

### Declarar, Definir e Invocar Procedimientos

Los procedimientos deben declararse y definirse antes de ser llamados. La declaración de procedimientos es similar a la declaración de cualquier otra variable en un pseudocódigo.

Los procedimientos se declaran con la sintaxis siguiente:

```
procedure <nombre_procedimiento>
```

El procedimiento se define cuando éste queda expresado en su cuerpo. La sintaxis de la definición del procedimiento es:

```
procedure <nombre_procedimiento>
begin
    //el conjunto de sentencias del procedimiento
end
```

La llamada a un procedimiento también se denomina invocación a un procedimiento. El método para invocar un procedimiento se conoce como llamada al procedimiento.

La sintaxis de una llamada al procedimiento es:

```
call <nombre_procedimiento>
```

Por ejemplo, el procedimiento testProcedure puede ser invocado en el pseudocódigo principal usando la siguiente instrucción:

```
call testProcedure
```

## Ejemplo 2

### Enunciado

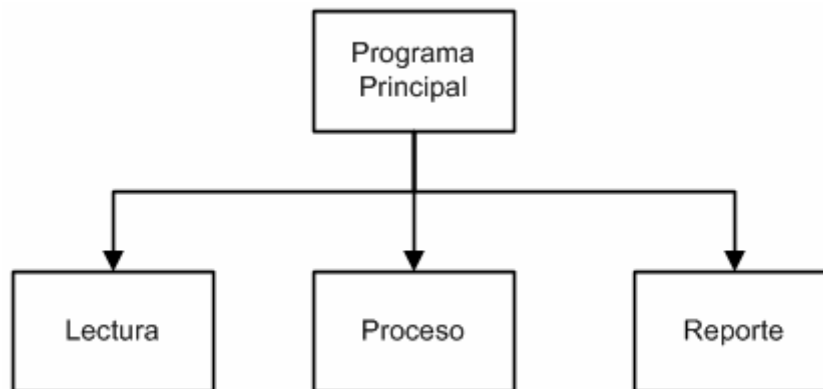
Desarrollar un programa para encontrar el promedio de un alumno. Se sabe que son tres notas y se elimina la más baja.

### Planteamiento

La solución del problema se enfoca en tres procesos perfectamente identificados:

1. Lectura de notas, el que desarrollaremos en un procedimiento de nombre **Lectura**.
2. Proceso de las notas, el que desarrollaremos en un procedimiento de nombre **Proceso**.
3. Reporte del resultado obtenido, el que desarrollaremos en un procedimiento de nombre **Reporte**.

Todos los procedimientos se invocaran en forma secuencial desde el programa principal, el siguiente grafico ilustra el planteamiento descrito.



## **Pseudocódigo**

```
// Datos del programa
numeric nNota1, nNota2, nNota3, nProm

begin

    call Lectura
    call Proceso
    call Reporte

end

procedure Lectura
begin

    display "Nota 1: "
    accept nNota1
    display "Nota 2: "
    accept nNota2
    display "Nota 3: "
    accept nNota3

end

procedure Proceso
begin

    numeric menor
    menor = nNota1
    if(menor > nNota2)
    begin
        menor = nNota2
    end
    if(menor > nNota3)
    begin
        menor = nNota3
    end
end

end

procedure Reporte
begin

    display "Nota 1: "
    display nNota1
    display "Nota 2: "
    display nNota2
    display "Nota 3: "
    display nNota3
    display "Promedio:"
    display nProm

end
```

## **Programación en Java**

En Java no existen funciones y procedimientos, lo que tenemos son métodos, si queremos que un método se comporte similar a un procedimiento, no debe retornar ningún resultado, esto se especifica con la palabra reservada **void**, para nuestro caso la sintaxis es:

```
private static void nombre_método ( argumentos ) {

    // instrucciones

}
```

El programa es el siguiente:

```
import java.util.Scanner;

public class Ejemplo2 {

    private static int nota1, nota2, nota3, prom;

    public static void main(String[] args) {

        lectura();
        proceso();
        reporte();

    }

    private static void lectura(){

        Scanner teclado = new Scanner(System.in);
        System.out.println("Ingreso de Notas");
        System.out.println("Nota 1:");
        nota1 = teclado.nextInt();
        System.out.println("Nota 2:");
        nota2 = teclado.nextInt();
        System.out.println("Nota 3:");
        nota3 = teclado.nextInt();

    }

    private static void proceso(){

        int menor;
        menor = nota1;
        if(menor>nota2) menor = nota2;
        if(menor>nota3) menor = nota3;
        prom = (nota1 + nota2 + nota3 - menor) / 2;

    }

}
```

```
private static void reporte(){  
    System.out.println("Reporte");  
    System.out.println("Nota 1: " + nota1);  
    System.out.println("Nota 2: " + nota2);  
    System.out.println("Nota 3: " + nota3);  
    System.out.println("Promedio: " + prom);  
}  
}
```



## Parámetros de los Procedimientos

Los parámetros son como un puente entre el procedimiento y el código que lo invoca. Los parámetros comprenden los datos utilizados y procesados por un procedimiento. Los parámetros pueden ser variables de tipo numérico ó carácter. Los parámetros se utilizan para realizar las siguientes tareas:

- Enviar datos a un procedimiento
- Recuperar datos de un procedimiento

La sintaxis para especificar los parámetros en la declaración de un procedimiento es la siguiente:

```
procedure <nombre_procedimiento>  
  ( input <tipo> <nombre_parametro>, output <tipo> <nombre_parametro> )
```

Los parámetros de tipo **input** son usados para enviar datos al procedimiento, y los parámetros de tipo **output** son usados para recuperar datos desde el procedimiento.

## Ejemplo 3

### Enunciado

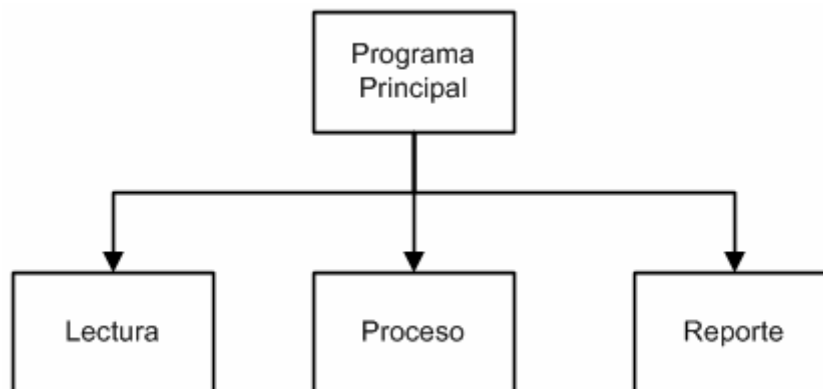
Desarrollar un programa que permita calcular el MCD y MCM de dos números.

### Planteamiento

La solución del problema se enfoca en tres procesos perfectamente identificados:

1. Lectura de los dos números que se quieren procesar, este proceso se desarrollará con un procedimiento de nombre **Lectura**, este procedimiento tendrá dos parámetros de salida.
2. Proceso de los números ingresados para obtener el MCD y MCM, este proceso se desarrollará con un procedimiento de nombre **Proceso**, este procedimiento tendrá dos parámetros de entrada y dos de salida.
3. Reporte de resultados obtenidos, este proceso se desarrollará con un procedimiento de nombre **Reporte**, este procedimiento tendrá cuatro parámetros de entrada.

Todos los procedimientos se invocaran en forma secuencial desde el programa principal, el siguiente grafico ilustra el planteamiento descrito.



### Pseudocódigo

```
begin
    // Datos del programa
    numeric nNum1, nNum2, nMCD, nMCM

    // Llamada a procedimientos
    call Lectura ( nNum1, nNum2 )
    call Proceso ( nNum1, nNum2, nMCD, nMCM )
    call Reporte ( nNum1, nNum2, nMCD, nMCM )

end
```

```
procedure Lectura ( output numeric nN1, output numeric nN2 )
begin

    display "Ingrese Número 1:"
    accept nN1
    display "Ingrese Número 2:"
    accept nN2

end

procedure Proceso Lectura
( input numeric nN1, input numeric nN2, output numeric nMCD, output numeric nMCM )
begin

    numeric a, b
    a = nN1
    b = nN2
    while ( a <> b )
    begin
        if ( a > b )
        begin
            a = a - b
        end
        else
        begin
            b = b - a
        end
    end
    nMCD = a
    nMCM = nN1 * nN2 / nMCD

end

procedure Reporte
( input numeric nN1, input numeric nN2, input numeric nMCD, input numeric nMCM )
begin

    display "Número 1: "
    display nN1
    display "Número 2: "
    display nN2
    display "MCD: "
    display nMCD
    display "MCM: "
    display nMCM

end
```

La implementación en Java no es posible por que no existen parámetros de salida en la implementación de los métodos.

## Funciones

---

Una función es un bloque de sentencias que realizan una tarea específica. Los principios básicos de las funciones y de los procedimientos son muy similares, por lo que se pueden intercambiar.

La diferencia entre una función y un procedimiento, es que la función retorna un valor al programa que la invoca, mientras que un procedimiento no retorna ningún valor.

Las funciones en un programa interactúan entre sí pasando y recibiendo datos. Las funciones también operan bajo el mecanismo llamada-retorno. Estos son los pasos del mecanismo llamada-retorno utilizando funciones:

- La función es invocada.
- Se ejecuta el conjunto de instrucciones dentro de la función.
- La control es retornado al programa que invoca la función junto con el valor de retorno.

### Declarar, Definir e Invocar Funciones

Las funciones se declaran como cualquier otro procedimiento o variable. El formato de la declaración es:

```
function <nombre_función>
```

Las funciones se definen como los procedimientos. La única diferencia es que contienen una sentencia **return** al final. La sintaxis de la definición de una función es:

```
function <nombre_función>
begin
    // las sentencias de la función
    return // La función devuelve un valor
end
```

Una vez que se declara la función, ésta puede ser invocada desde el programa. El método para invocar funciones se conoce como llamada a la función. La sintaxis de la llamada a la función es:

```
resultado = call <nombre_función>
```

## Parámetros de las Funciones

Los parámetros de la función forman una interfaz entre la función y el código de llamada.

Las funciones aceptan valores en forma de parámetros. Si se especifican los parámetros, estos deberían estar separados por comas.

A diferencia de los procedimientos, las funciones utilizan parámetros sólo para recibir datos desde el código de llamada.

Devuelven valores al código de llamada por medio de la sentencia **return**. Por lo tanto, las funciones sólo tienen parámetros de entrada de datos.

La sintaxis para definir parámetros en la declaración de una función es la siguiente:

```
function <nombre_funcion> ( <tipo> <parametro1>, . . . )
```

En Java no tenemos funciones, en su defecto debemos crear un método que retorne un valor, la sintaxis a utilizar es:

```
private static tipo_retorno nombre_método ( argumentos )
```

El siguiente gráfico muestra la representación del funcionamiento de una función:



## Ejemplo 4

### Enunciado

Desarrollar un programa para imprimir la tabla de multiplicar de un número.

### Planteamiento

La solución de este problema se enfoca en la construcción de una función que retorne el producto de dos números, luego esta función se utilizará en el proceso para imprimir la tabla de multiplicar.

### Pseudocódigo

```
begin

    // Datos del Programa
    numeric nNum, k, p;

    // Lectura del número
    display "Tabla del: "
    accept nNum;

    // Proceso
    display "Tabla del: " + nNum
    for( k = 1; k <= 12; k = k + 1 )
    begin
        p = call producto( k, nNum )
        display k + " * " + nNum + " = " + p
    end

end

function producto( numeric a, numeric b )
begin

    numeric p
    p = a * b
    return p

end
```

**Programación en Java**

```
import java.util.Scanner;

public class Ejemplo4 {

    public static void main(String[] args) {

        // Variables
        int n, p;
        Scanner teclado = new Scanner(System.in);

        // Lectura de Dato
        System.out.println("Ingrese el valor de N:");
        n = teclado.nextInt();

        // Proceso
        System.out.println("Tabla del: " + n);
        for(int k=1; k <= 12; k++)
            System.out.println(k + " * " + n + " = " + producto(k, n));

    }

    private static int producto ( int a, int b ){

        int c;
        c = a * b;
        return c;

    }

}
```

## Alcance de las Variable

---

Las variables pueden ser declaradas dentro ó fuera del bloque **begin .. end** de un pseudocódigo principal, de un procedimiento o de una función.

Dependiendo del lugar en el que se declaran las variables, éstas tienen dos tipos de alcance:

- Alcance Local
- Alcance Global

### Alcance Local

Las variables que se declaran dentro del bloque **begin .. end** de un pseudocódigo principal, de una función ó de un procedimiento tienen alcance local.

Las variables que tienen alcance local se denominan variables locales o internas.

Durante la ejecución del programa:

- Las variables se crean cuando se ejecuta la sentencia que declara la variable interna en el bloque **begin .. end**.
- Las variables expiran cuando el control sale del bloque **begin .. end**.

El siguiente ejemplo describe el alcance de las variables locales:

```
begin
    numeric num
    display "Ingrese un número:"
    accept num
    if ( num > 10 )
        begin
            numeric resto
            resto = num % 2
            display resto
        } Alcance de
        la variable resto
    end
end
```

El siguiente ejemplo ilustra el alcance de una variable dentro de una función:

```
function producto( numeric a, numeric b )
begin
    numeric p
    p = a * b
    return p
} Alcance de
la variable p
end
```



## Alcance Global

Las variables que pueden utilizarse en cualquier parte del pseudocódigo tienen alcance global y se denominan variables globales.

Las variables globales se declaran fuera de la función, del procedimiento o del bloque **begin .. end** del pseudocódigo principal.

El siguiente pseudocódigo ilustra el alcance de una variable global:

```
numeric a      // Variable Global
caracter b     // Variable Global

begin

    numeric c   // Variable Local
    . . .
    . . .

end

procedure proceso
begin

    numeric d   // Variable Local
    . . .
    . . .

end
```

De este pseudocódigo podemos afirmar:

- Las variables **a** y **b** son globales, por lo tanto se pueden utilizar en cualquier parte del programa, dentro de cualquier función, y dentro de cualquier procedimiento.
- La variable **c** es local al bloque **begin .. end**, por lo tanto, no puede ser utilizada fuera de este bloque.
- La variable **d** es local al procedimiento **proceso**, por lo tanto no puede ser utilizada fuera del procedimiento.

## Alcance de lo Parámetros

Los parámetros de un procedimiento o función son variables a las que sólo se puede acceder desde dentro del procedimiento o la función.

Esto significa que los parámetros actúan como las variables locales de una función o procedimiento.

Este es el alcance de los parámetros:

- Los parámetros existen sólo dentro de la función o procedimiento para el que se han definido. No se puede acceder a ellos desde fuera de la función o procedimiento.
- Mantienen su valor mientras se ejecuta la función o procedimiento.
- Los parámetros se inicializan cada vez que se llama a la función o al procedimiento.

## Ejercicios

---

### Ejercicio 1

Desarrollar un programa para averiguar el mayor de tres números, debe tener en cuenta lo siguiente:

- Los números deben ser ingresados por el usuario.
- Desarrollar un función para obtener el mayor de dos números.
- Utilizar la función del punto anterior en el proceso para encontrar el mayor de los tres números.

### Ejercicio 2

Basándose en la lógica del Ejercicio 1, desarrollar un programa que permita calcular el promedio de un alumnos, se sabe que son 4 notas y se promedia las tres mejores.

### Ejercicio 3

Desarrollar un programa que calcule el factorial de lo **N** primeros números, debe tener en cuenta lo siguiente:

- El valor de **N** debe ser ingresado por el usuario.
- Elabore una función para calcular el factorial de un número.
- La función del paso anterior debe ser utilizada en el programa principal ó dentro de un procedimiento.

### Ejercicio 4

Desarrollar un programa para averiguar si los números en un rango de **[M,N]** son primos, debe tener en cuenta lo siguiente:

- Los valores de **M** y **N** deben ser ingresados por el usuario.
- Desarrolle un función que reciba como parámetro un número, y retorne 1 si es primo y 0 si no lo es.
- La función del paso anterior debe ser utilizada en el proceso para análisis del rango de números.

### Propuesta Adicional

Los ejercicios propuestos en la Lección 05, debe reprogramarlos utilizando el enfoque modular, utilizando funciones y/o procedimientos.

# Apuntes