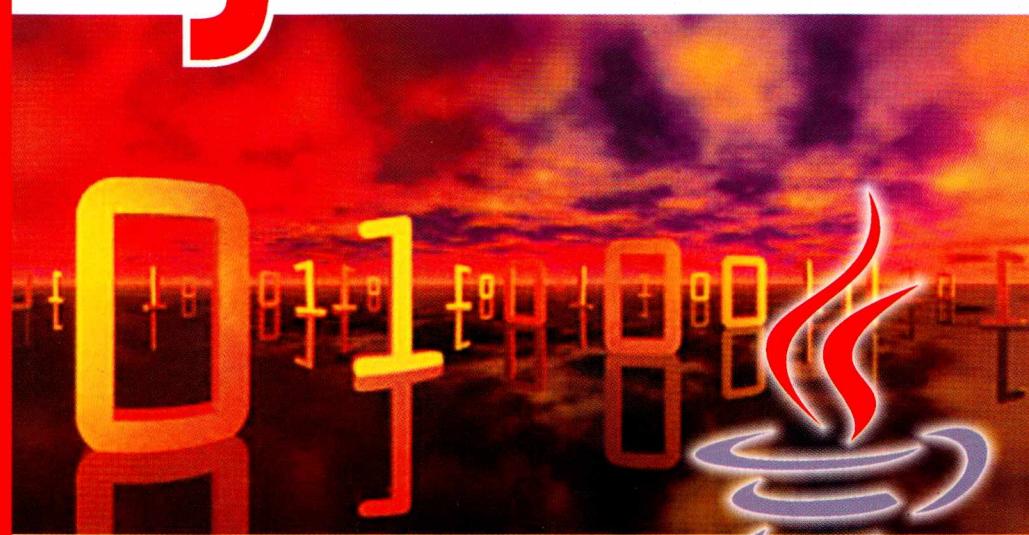


# ALGORITMOS Y DIAGRAMAS DE FLUJO APLICADOS EN

Paso a  
Paso

# java2

Incluye  
CD ROM  
  
Videos  
Tutoriales



GRUPO  
EDITORIAL

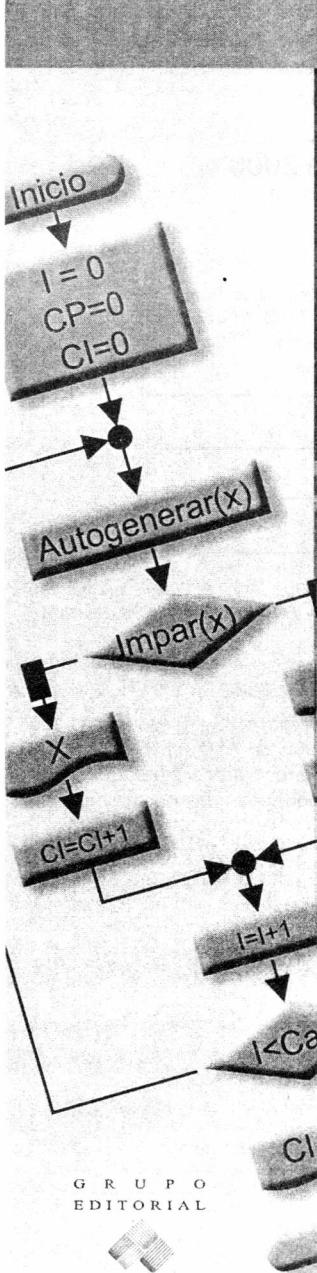


Megabyte®

[FREELIBROS.ORG](http://FREELIBROS.ORG)



Robert Jaime Pantigoso Silva



GRUPO  
EDITORIAL



Megabyte

**Algoritmos y Diagramas de  
Flujo aplicados en**

# java2



*Robert Jaime Pantigoso Silva*



# Megabyte s.a.c.

GRUPO EDITORIAL

Primera Edición, Octubre 2006

**Área** : Computación e Informática

Hecho el Depósito Legal en la Biblioteca Nacional del Perú

Nº 2006 - 5843 (Ley Nº 26905 / D.S. Nº 017-98-ED)

R.U.C. Nº 20507993444

**ISBN:** 9972 - 821 - 84 - 6

---

© Edward Aburto Correa

**Gerente General**

---

© Robert Jaime, Pantigoso Silva

**Autor**

---

© Gean Carlo Apolinario García

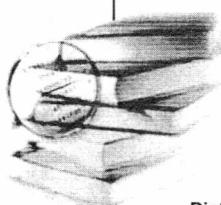
**Diseño de Carátula**

---

## *Algoritmos y Diagramas de Flujo aplicados en Java 2*

### **Derechos Reservados / Decreto Ley 822**

Prohibida la reproducción total o parcial de este libro, su tratamiento informático la transmisión de ninguna otra forma o por cualquier otro medio ya sea electrónico, mecánico, por fotocopia, por registro u otros métodos sin permiso previo y por escrito de los titulares de Copyright.



### **Distribución y Ventas**

Jr. Rufino Torrico 889 of. 208 - Cercado de Lima

**Telefax:** 332-4110 **Nextel:** 407\*4515

[www.grupomegabyte.com](http://www.grupomegabyte.com)

[www.editorialmegabyte.com](http://www.editorialmegabyte.com)

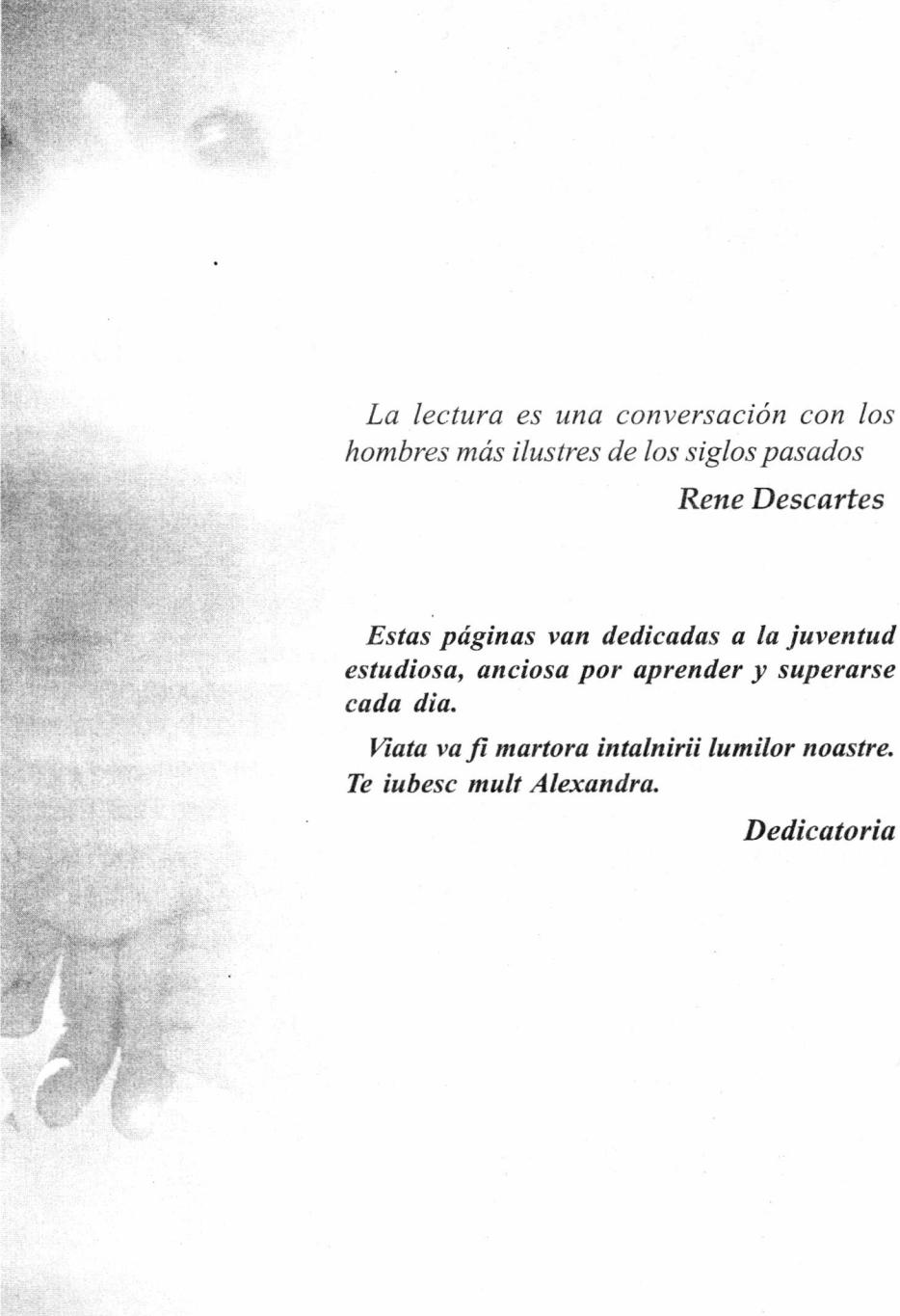
[ventas@grupomegabyte.com](mailto:ventas@grupomegabyte.com)

[ventas@editorialmegabyte.com](mailto:ventas@editorialmegabyte.com)

### **Cta. Banco de Crédito**

S/. 191-12591005-0-86

S/. 191-12591006-1-87



*La lectura es una conversación con los  
hombres más ilustres de los siglos pasados*

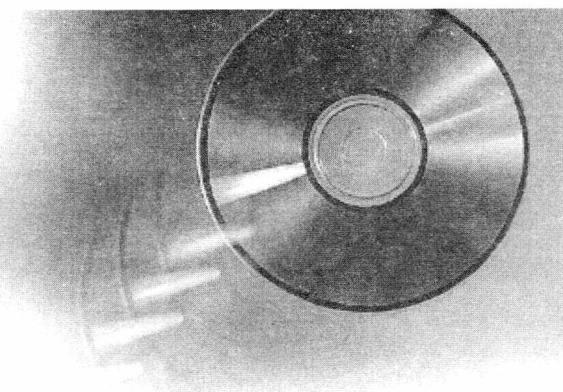
*Rene Descartes*

*Estas páginas van dedicadas a la juventud  
estudiosoa, anciosa por aprender y superarse  
cada dia.*

*Viata va fi martora intalnirii lumilor noastre.  
Te iubesc mult Alexandra.*

*Dedicatoria*

**FREELIBROS.ORG**



## Contenido del CD

El Cd adjunto a este libro contiene todos los programas desarrollados en el transcurso del texto, con la finalidad de que el lector compruebe la veracidad de lo mostrado en el libro.

Todos los ejemplos desarrollados en el libro han sido expresamente hechos por el autor.

La codificación en Java también a sido desarrollada por el autor y puede ser ejecutado con el compilador 1.5 de Java, el cual esta en Cd.

Para comodidad del lector también le ofrecemos el JCreator a fin de poder ejecutar cada uno de los ejemplos que les brindamos.

El Cd contiene videos los cuales explican la manera de crear un applet de Java en JCreator, así mismo la instalación de cada uno de estos software.



# ÍNDICE

---

---

## Capítulo I: Conceptos Básicos

Conceptos básicos de algoritmia.....	17
Algoritmos.....	17
Características de los algoritmos.....	17
Programa.....	17
Diagrama de flujo.....	18
Variables.....	19
Constantes.....	19
Expresiones.....	19
Operadores.....	20
Operadores Aritméticos.....	20
Operadores relacionales.....	20
Operadores Lógicos.....	20
ENTRADA / SALIDA de datos.....	21
Asignaciones.....	22
Declaración de variables.....	22
Pseudocódigo.....	23
Programación Orientada a Objetos (POO).....	23

Objetos.....	24
Clases.....	24
Creando un programa en java.....	24
Ejemplos desarrollados de algoritmos.....	25
Ejemplo 1.1: Área de un triángulo.....	25
Ejemplo 1.2: Área de una circunferencia.....	28
Ejemplo 1.3: Perímetro y superficie de un rectángulo.....	30
Ejemplo 1.4: Área de un triángulo a través de sus lados.....	33
Ejemplo 1.5: cuadrado y cubo de un número entero.....	35
Ejemplo 1.6: Promedio de calificaciones.....	37
Problema 1.7: transformación de unidades de medida y peso.....	40
Problema 1.8: Intercambio de tres valores.....	42
Problema 1.9: Inverso de un número de tres cifras.....	45
Problema 1.10: Cambio de un billete.....	48
Ejercicios propuestos.....	51

## **Capítulo II: Estructuras lógicas selectivas**

Introducción.....	57
Estructura Si...Entonces (Selección simple).....	57
Ejemplo 2.1: Calificación de un alumno I.....	58
Ejemplo 2.2: Aumento de sueldo I.....	60
Estructura Si...Entonces...Si no (Alternativa doble).....	63
Ejemplo 2.3: Calificación de alumno II.....	63
Ejemplo 2.4: Aumento de sueldo II.....	66
Anidamiento de estructuras condicionales.....	67
Ejemplo 2.5: Validación de nota.....	69
Ejemplo 2.6: Verificar si un número es par o impar.....	71
Estructura de selección múltiple.....	74
Ejemplo 2.7: Solucionar la función especificada.....	74
Ejemplo 2.8: Números en forma ascendente.....	77

Ejemplo 2.9: Obtener el mayor de tres números.....	80
Ejemplo 2.10: Número capicúa.....	83
Ejemplo 2.11: Estado de un alumno según nota obtenida.....	86
Ejemplo 2.12: Aumento de sueldo según categoría.....	89
Ejemplo 2.13: Formato de fecha.....	92
Ejemplo 2.14: Dia siguiente de una fecha dada.....	96
Ejercicios propuestos.....	99

## **Capítulo III: Estructuras lógicas repetitivas**

Introducción.....	105
Estructura <i>Hacer Mientras</i> .....	105
Ejemplo 3.1: Tabla de multiplicar de un número.....	107
Ejemplo 3.2: Números pares e impares de números autogenerador.....	107
Ejemplo 3.3: Calcular la serie: $100 + 98 + 96 + 94 + \dots + 0$ .....	112
Estructura <i>Mientras</i> .....	114
Ejemplo 3.4: Seno, coseno y tangente de ángulos.....	114
Ejemplo 3.5: Obtener una pirámide de asteriscos(*).....	117
Ejemplo 3.6: Promedio de calificaciones.....	121
Ejemplo 3.7: Invertir un número positivo.....	123
Estructura <i>Desde</i> .....	127
Ejemplo 3.8Ejemplo 3.8: Suma de los 20 números naturales.....	128
Ejemplo 3.9Ejemplo 3.9: Identificar si un número es primo.....	130
Ejemplo 3.10: Promedio de un alumno eliminado la menor.....	133
Ejemplo 3.11: Menor y mayor de una lista de N números.....	137
Ejemplo 3.12: Progresión 5, 8, 11, 14, 17, 20, 23,..n.....	140
Ejemplo 3.13: Hallar el factorial de un número.....	143
Ejemplo 3.14: Números primos gemelos.....	145

## **Capítulo IV: Vectores y matrices**

Vectores.....	155
---------------	-----

Lectura y escritura de un vector.....	156
Ejemplo 4.1: Suma de los elementos de un arreglo.....	157
Ejemplo 4.2: Media aritmética de un vector.....	160
Ordenamiento de un vector.....	162
Método de la burbuja.....	162
Método de ordenamiento de la burbuja mejorada.....	166
Método de ordenamiento por Inserción.....	167
Método de Selección.....	168
Método de Shell.....	169
Búsqueda binaria en un vector.....	170
Ejemplos desarrollados.....	172
Ejemplo 4.3: Operaciones con vectores.....	172
Ejemplo 4.4: Operaciones con matrices.....	178
Ejemplo 4.5: Operaciones con cadenas.....	183

## **Apéndice A: Uso de funciones para desarrollar programas**

Introducción.....	189
Funciones.....	189
Ejemplo de función.....	190
Procedimientos.....	191
Ámbito de las variables.....	191
Variable local.....	192
Variable global.....	192
Paso de parámetros.....	193
Paso de parámetros por valor.....	194
Paso por referencia (dirección) o variable.....	195
Funciones y Procedimientos como parámetros.....	195

## **Apéndice B: Uso básico del JCreator**

# Introducción

---

Este libro esta orientado a las personas que empiezan en el mundo de la programación. Los algoritmos son la base de todo sistema de cómputo, por lo que se ha querido brindarle los conceptos básicos apoyados en una solución de cómputo, motivo por el cual todos los ejemplos están desarrollados en Java 2.

Al final del desarrollo del libro el lector estará en capacidad de diseñar algoritmos mediante ejercicios prácticos, con el objetivo de adquirir una mentalidad de programador.

Ya hemos mencionado que para el desarrollo de los algoritmos usaremos el Java 2, el cual es un lenguaje de programación orientado a objetos, además de ser versátil y muy potente. En java podemos realizar tanto aplicaciones, es decir programas para PC, como applet los cuales nos permite realizar tareas desde un browser como Internet Explorer o FireFox, entre otros.

El realizar un algoritmo y luego codificarlo en un lenguaje de programación, en nuestro caso Java 2, no es más que una simple traducción y en algunos casos una adaptación del algoritmo al lenguaje específico. Mientras que el algoritmo es mas parecido al lenguaje normal que diariamente hablamos, un programa ya codificado procesa y ejecuta las ideas plasmadas en el algoritmo.

Estoy seguro que este libro les ayuda en el ingreso al mundo de la programación, el cual, si bien es cierto, es muy grande y complejo, teniendo muy en claro los conceptos básicos, podremos resolver cualquier problema sin importar su complejidad.



# Capítulo I

## Conceptos Básicos

**Este capítulo contiene:**

Conceptos básicos de algoritmia,  
seudocódigo y diagramas de flujo

Elementos presentes en un algoritmo  
como variables, constantes,  
expresiones

Tipo de operadores

Declaración de variables y asignaciones

Ejemplos desarrollados

Ejemplos propuestos





## **Conceptos básicos de algoritmia**

### **Algoritmos**

Los algoritmos constituyen un listado de instrucciones que indican el camino a seguir para dar solución a un problema.

Podriamos decir que un algoritmo es la suma de una parte lógica mas una parte de control, en donde la parte lógica especifica el conocimiento en la solución del problema y la parte de control es la estrategia para solucionar el problema.

### **Características de los algoritmos**

- ❖ Un algoritmo no debe de ser ambiguo.
- ❖ Debe de tener una secuencia inicial
- ❖ Cada paso deberá tener una secuencia sucesiva y única es decir que deben indicar claramente el camino a seguir en la solución del problema.
- ❖ El algoritmo debe de ser siempre eficiente y dar una solución al problema o de lo contrario deberá dar un mensaje que diga "Sin solución"

### **Programa**

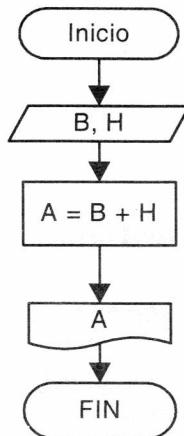
Un programa define un "algoritmo", porque constituye el conjunto de

instrucciones que forman el algoritmo (ya codificados en un lenguaje de programación) y que se dan a una computadora para solucionar un problema específico.

## Diagrama de flujo

El diagrama de flujo es la representación gráfica de dicha secuencia de instrucciones que conforman el algoritmo.

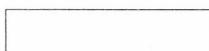
El siguiente es un ejemplo de un diagrama de flujo para sumar dos variables B y H, el resultado es almacenado en la variable A.



Los símbolos mas comunes y los cuales usaremos son:



Terminal: Se usa para indicar el inicio o fin de un diagrama de flujo



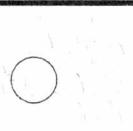
Proceso: Cualquier tipo de operación que pueda originar cambio de valor, operaciones aritméticas, de transformaciones, etc.



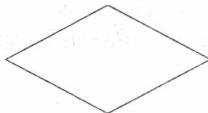
Entrada/Salida: Se usa para indicar el ingreso o salida de datos.



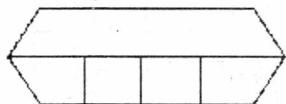
Salida: Se utiliza para mostrar datos, será el símbolo usado en este texto.



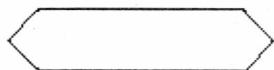
Conejero: Sirve para enlazar dos partes cualesquiera de un diagrama a través de un conector en la salida y otro conector en la entrada.



Salida: Indica operaciones lógicas o de comparación entre datos y en función del resultado de la misma determina cual de los distintos caminos alternativos del programa se debe seguir.



En caso de: Usado para indicar varias acciones posibles según sea un dato de entrado al control.



Desde: Estructura repetitiva q indica un ciclo de N repeticiones de una o mas acciones.

## Variables

Son los elementos que se utilizan para contener datos de distintos tipos: números, letras, cadenas de caracteres, valores lógicos, etc. El valor contenido en una variable puede cambiar a lo largo de la ejecución de un programa.

## Constantes

Son elementos que contienen datos, el valor asignado a una constante es fijo y no se puede cambiar durante toda la ejecución de un programa.

## Expresiones

Las expresiones son combinaciones de constantes, variables, símbolo de operación, paréntesis y nombres de funciones especiales.

Por ejemplo

$$a + ( b + 3 ) / c$$

Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones implicadas.

Una expresión consta de operadores y operandos. Según sea el tipo de datos que manipulan, se clasifican las expresiones en:

Aritméticas, Relacionales y Lógicas

## Operadores

### Operadores Aritméticos

Los operadores aritméticos nos permiten, básicamente, hacer cualquier operación aritmética (suma, resta, multiplicación y división). En la siguiente tabla se muestran los operadores de los que se dispone.

Operador	Java	Acción	Ejemplo	Result.
-	-	Resta	5-2	3
+	+	Suma	5+2	7
*	*	Multiplicación	5*2	10
/	/	División	5/2	2.5
^	pow	potencia	5^2 pow(5,2)	25
MOD	%	Resto de una división de dos números enteros	5 MOD 2 5 % 2	1
DIV	/	Parte entera de la división de dos números enteros	5 DIV 2 5/2	2

El operador MOD nos devuelve el residuo de una división entera, mientras que el operador DIV permite realizar una división entre dos números enteros, allí radica la diferencia con el operador "/".

### Operadores relacionales

Al igual que en matemáticas, estos operadores nos permitirán evaluar las relaciones (igualdad, mayor, menor, etc) entre un par de operandos (en principio, pensemos en números). Los operadores relacionales de los que disponemos son:

Operador	Java	Acción
>	>	Mayor que
<	<	Menor que
=	==	Igual que
>=	>=	Mayor igual que
<=	<=	Menor igual que
<> o !=	!=	Diferente que

### Operadores Lógicos

Los operadores lógicos producen un resultado **booleano**, y sus operandos son también valores lógicos o asimilables a ellos (los valores numéricos son asimilados a **cierto** o **falso** según su valor sea cero o distinto de cero).

Los operadores lógicos son tres; dos de ellos son binarios, el último (negación) es unario.

### **AND (Y)**

Produce un resultado con valor de verdad (true) cuando ambos operandos tienen valor de verdad true; en cualquier otro caso el resultado tendrá un valor de verdad false.

Sintaxis: operando1 **AND** operando2

### **OR (O)**

Produce un resultado con valor de falso (false) cuando ambos operadores tienen valores falsos; en cualquier otro caso el resultado tendrá un valor de verdad.

Sintaxis: operando1 **OR** operando2

### **NOT (NO)**

Invierte el valor de verdad de operando.

Sintaxis: **NOT** operando

## **Prioridad de los Operadores**

Los operadores deben ser evaluados según la siguiente prioridad

- 1.- ( )
- 2.- ^
- 3.- \*, /, Mod, Not
- 4.- +, -, And
- 5.- >, <, >=, <=, <>, =, Or

## **ENTRADA / SALIDA de datos**

Los dispositivos de entrada/salida permiten que el usuario interactúe con la máquina. Por medio de los dispositivos de entrada el usuario ingresa los datos a procesar en el sistema y los dispositivos de salida presentan los resultados en un formato legible.

### **Instrucción LEER (setText())**

Nos permite el ingreso de datos al computador, el dato leido debe ser almacenado en una variable, la sintaxis es la siguiente:

Leer variable

En java usaremos la setText() para obtener el valor ingresado de un control.

### **Instrucción ESCRIBIR(getText())**

Esta instrucción permite reportar resultados de un proceso, también se usa para enviar un mensaje al operario. La sintaxis es la siguiente:

ESCRIBIR Variable

ESCRIBIR ‘Texto’

ESCRIBIR Expresión

Veamos unos ejemplos, según sean las sintaxis anteriores respectivamente

ESCRIBIR Resultado

Esta instrucción devuelve el contenido de la variable Resultado

ESCRIBIR ‘Prepárese para el ingreso de datos’

Esta instrucción muestra al usuario el contenido de los apóstrofes, nótese que para poder emitir un texto este debe ir encerrado en apóstrofes

ESCRIBIR 4\*n

Esta instrucción primero calcula la expresión 4\*n y luego muestra ese resultado.

La función getText nos permitirá mostrar datos en un control.

## **Asignaciones**

Una asignación consiste en darle un determinado valor a una variable o constante, por ejemplo en la siguiente sentencia observamos que a la variable A, se le da el valor de 5.

A = 5

De manera similar podemos tener la siguiente asignación

A = 4 + (3 \* Y^2)

vemos que una expresión a sido asignada a la variable A

Algunos autores usan el símbolo  $\leftarrow$  en vez de igual (=) para una asignación.

## **Declaración de variables**

Mediante la declaración de variables describimos las características de las mismas. La sintaxis que usaremos es la siguiente:

Nombre\_de\_variable : Tipo

Entiéndase por tipo, al tipo de dato de la variable

## **Tipos de datos escalares**

Son aquellos tipos de datos cuyos miembros están compuestos por un solo ítem (dato). Los tipos de datos escalares nativos son aquellos tipos de datos escalares que ya están implementados en el lenguaje junto a sus respectivas operaciones.

Tipo	Java	Tamaño
byte	byte	1 bytes (8 bits)
entero corto	short	2 bytes (16 bits)
entero	int	4 bytes (32 bits)
entero largo	long	8 bytes (64 bits)
real	float	4 bytes (32 bits)
doble	double	8 bytes (64 bits)
carácter	char	2 bytes (16 bits)

## **Pseudocódigo**

Diremos que una notación es un pseudocódigo si mediante ella podemos describir el algoritmo, utilizando palabras y frases del lenguaje natural sujetas a unas determinadas reglas.

Todo pseudocódigo debe posibilitar la descripción de:

- ❖ Instrucciones de entrada / salida.
- ❖ Instrucciones de proceso.
- ❖ Sentencias de control del flujo de ejecución.
- ❖ Acciones compuestas, a refinar posteriormente.

Para entender mejor estos conceptos veamos algunos ejemplos.

## **Programación Orientada a Objetos (POO)**

La programación orientada a objetos intenta tener una semejanza al modo de pensar del hombre, debido a la forma en que se manejan las abstracciones que representan las entidades del dominio del problema, y las propiedades como la jerarquía o el encapsulamiento.

A diferencia de la programación estructurada, el elemento básico es el objeto, el cual es la representación de un concepto en el programa. El objeto contiene los datos y las operaciones para manipular dichos datos.

Java incorpora el uso de la orientación a objetos como uno de los pilares básicos de su lenguaje.

## Objetos

Podemos definir objeto como el "encapsulamiento de un conjunto de operaciones (métodos) que pueden ser invocados externamente, y de un estado que recuerda el efecto de los servicios". [Piattini et al., 1996].

Un objeto consta de:

Tiempo de vida: La duración de un objeto en un programa siempre está limitada en el tiempo. La mayoría de los objetos sólo existen durante una parte de la ejecución del programa. Son creados por un constructor y finalizan mediante un destructor

Estado: Todo objeto posee un estado, definido por sus *atributos*. Con él se definen las propiedades del objeto, y el estado en que se encuentra en un momento determinado de su existencia.

Comportamiento: Todo objeto ha de presentar una interfaz, definida por sus *métodos*, para que el resto de objetos que componen los programas puedan interactuar con él.

## Clases

Las clases son abstracciones que representan a un conjunto de objetos con un comportamiento e interfaz común. Es la implementación de un tipo de objeto (considerando los objetos como instancias de las clases).

Una clase no es más que una plantilla para la creación de objetos. Cuando se crea un objeto (*instanciación*) se ha de especificar de qué clase es el objeto instanciado, para que el compilador comprenda las características del objeto.

Los *métodos* son las funciones mediante las que las clases representan el comportamiento de los objetos. En dichos métodos se modifican los valores de los atributos del objeto, y representan las capacidades del objeto (en muchos textos se les denomina *servicios*).

## Creando un programa en java

Podemos usar cualquier un simple editor de texto para crear un programa java, pero lo mas recomendable es usar los editores de textos java especializados, entre ellos tenemos el JCreator, Sun **Java Studio Creator**, **Java editor**, Java Builder, entre otros.

El compilador utilizado para compilar los ejemplo es el JDK version 1.5.0\_09 en cual puede ser descargado gratuitamente de la página de java.

En el cd adjunto le brindamos el compilador y el JCreator los cuales son de libre distribución.

## Ejemplos desarrollados de algoritmos

### Ejemplo 1.1

Elabore un algoritmo y su pseudocódigo para calcular e imprimir el área de un triángulo.

#### Solución

Declarar Variables

b,h : real

Entrada: (datos a introducir al computador)

Base = b

Altura = h

Operación: calcular área del triángulo (Base x Altura)/2

Salida: (Resultado que mostrará el computador)

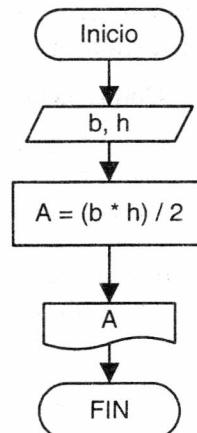
#### Pseudocódigo:

1. Iniciar proceso
2. Declarar variables  
    h : real  
    b : real  
    A : real
3. LEER b
4. LEER h
5. A = ( b \* h ) / 2
6. ESCRIBIR A
7. Terminar el proceso

Este pseudocódigo es fácil pero daremos una pequeña explicación de ello.

1. Se inicia el proceso, todo programa o bloque de programa, debe tener un indicativo de inicio, ya sea inicio de bloque o inicio de programa.
2. Declaramos las variables a usar, nótese que las variables h y b son de tipo real, de eso se deduce que el resultado de una operación entre estas dos variables también sea del tipo real, por eso A es del tipo real.

Las siguiente línea, tiene el mismo resultado para la declaración de las variables



b, h y A.

b,h,A : Real

3. Leemos el valor de la variables b, por ejemplo digamos el valor introducido sea 5
4. Leemos el valor de la variable h, para este ejemplo ingresemos 7.
5. Calculamos la expresión  $(b*h)/2$  y dicho resultado lo almacenamos en la variable A, entonces será:  $A = (5*7)/2$ , por lo tanto tendremos  $A = 17.5$
6. Escribimos el resultado de la operación, en este caso será 7.
7. Cuando un proceso o programa se inicia, debemos indicar su fin, caso contrario, dicho proceso puede ejecutarse infinitamente, es decir al ser traducido a un lenguaje de programación

Una vez elaborado el algoritmo, el problema a programar está prácticamente resuelto, ya que lo único que faltaría sería la codificación correspondiente en el lenguaje de programación que se esté usando, técnicamente un algoritmo es codificable en cualquier lenguaje de programación.

Primero veamos cual es el diagrama de flujo de este ejercicio y luego su codificación en JAVA

### **Codificación en Java**

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo1_1 extends Applet implements
ActionListener {
    //declaramos los objetos a usar
    Label lblBase, lblAltura, lblArea,Resul;
    TextField B,h;
    Button Ejecutar;

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (210, 120);
        /*ubicamos y dibujamos los objetos*/
        //Etiqueta Base
        lblBase=new Label("Base");
```

```
lblBase.setBounds(10,15,42,20);
add(lblBase);
//Etiqueta Altura
lblAltura=new Label("Altura");
lblAltura.setBounds(10,40,42,20);
add(lblAltura);
//Etiqueta Area
lblArea=new Label("Area");
lblArea.setBounds(10,65,42,20);
add(lblArea);
//Resultado
Resul=new Label(" ");
Resul.setBounds(75,65,42,20);
add(Resul);
//Base
B=new TextField();
B.setBounds(75,15,90,20);
add(B);
//Altura
h=new TextField();
h.setBounds(75,40,90,20);
add(h);
//boton ejecutar
Ejecutar=new Button("Ejecutar");
Ejecutar.setBounds(50,100,100,20);
Ejecutar.addActionListener(this);
add(Ejecutar);
}
public void actionPerformed(ActionEvent e){
    //Declarar variables
    double Base,Altura,AArea;
    //Leer datos y convertir
    Base=Double.parseDouble(B.getText());
    Altura=Double.parseDouble(h.getText());
    //Cálculos
    AArea=(Base*Altura)/2;
    //Mostrar resultado
    Resul.setText("= " + AArea);
```

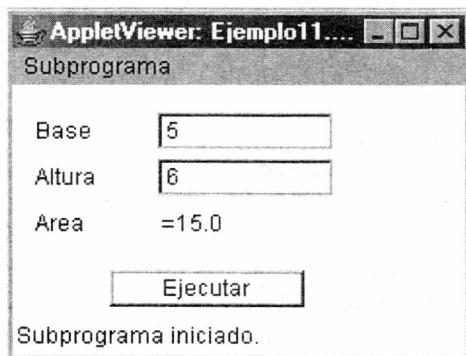
Las tres primeras líneas del programa en java hacen referencia a las librerías que deben usar para la ejecución de los comandos usados.

B.getText() nos permite obtener el valor del control TextField B, por otro lado Double.parseDouble() nos permite convertir el dato ingresado al tipo double.

La declaración de las variables son del tipo double, se hubiera podido usar float en ese caso la lectura y conversión de datos seria:

```
Base=Float.parseFloat(B.getText());  
Altura=Float.parseFloat(h.getText());
```

La vista del programa en java es el siguiente



## Ejemplo 1.2

Construya un pseudocódigo que dado el radio, calcule e imprima el área de una circunferencia.

### Consideraciones:

La superficie de una circunferencia se calcula aplicando la siguiente fórmula:

$$\text{Área} = \pi * \text{radio}^2$$

### Solución

Declarar Variables R : Real

Declarar constante PI = 3.1416

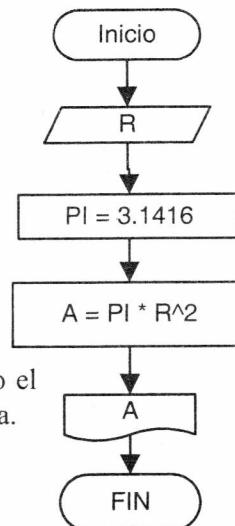
Ingresar radio R

Calcular la superficie de la circunferencia.

Mostrar resultados obtenidos.

### Pseudocódigo

1. Iniciar proceso
2. Declarar variables  
    R,A : Real
3. Declarar constante  
    PI : Real
4. Asignar PI = 3.1416
5. LEER R
6. Calcular A = PI \* R^2
7. ESCRIBIR A
8. Terminar el proceso



Nótese que estamos declarando una constante, para ello el valor de PI no podrá cambiar en la ejecución del programa.

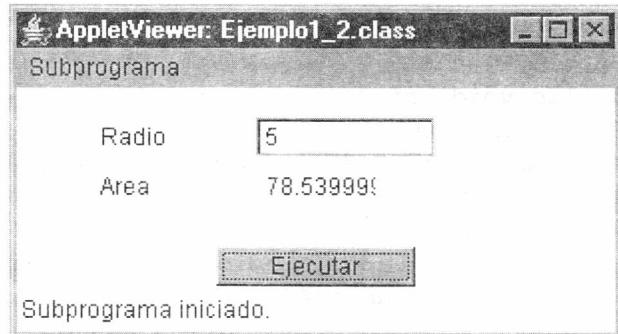
### Codificación en java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import java.math.*;

public class Ejemplo1_2 extends Applet implements
ActionListener{
    Label lbl1, lbl2, A;
    TextField R;
    Button Ejecutar;
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (300, 100);
        //Etiquetas
        lbl1=new Label("Radio");
        lbl1.setBounds(40,15,60,20);
        add(lbl1);
        lbl2=new Label("Area");
        lbl2.setBounds(40,40,60,20);
        add(lbl2);
        //Resultado
        A=new Label("");
        A.setBounds(120,40,60,20);
        add(A);
        //TXT
```

```
R=new TextField();
R.setBounds(120,15,90,20);
add(R);
//botón ejecutar
Ejecutar=new Button("Ejecutar");
Ejecutar.setBounds(100,80,100,20);
Ejecutar.addActionListener(this);
add(Ejecutar);
}
public void actionPerformed(ActionEvent e){
    //Declarar variables
    double AA,BB,RR;
    final double PI=3.1416;
    //Leer datos
    RR=Double.parseDouble(R.getText());
    //Calcular y mostrar datos
    AA = PI * Math.pow(RR, 2);
    A.setText(" "+AA);
}
}
```

En java usamos la palabra reservada **final** para declarar una constante, cuyo valor no cambiará a lo largo del programa.



### Ejemplo 1.3

Construya un pseudocódigo, que dados como datos la base y la altura de un rectángulo, calcule el perímetro y la superficie del mismo.

#### Consideraciones:

- ❖ La superficie de un rectángulo se calcula aplicando la siguiente fórmula:

$$\text{Superficie} = \text{base} * \text{altura}$$

❖ El perímetro se calcula como:

$$\text{Perímetro} = 2 * (\text{base} + \text{altura})$$

### Solución:

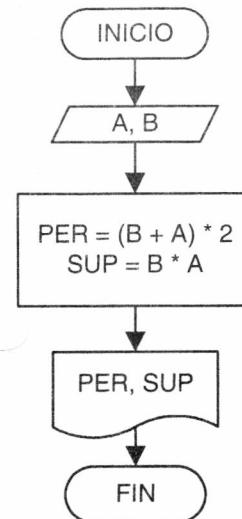
Declarar Variables B, A : Reales

Ingresar base y altura (B y A)

Calcular la superficie y perímetro del rectángulo.

### Pseudocódigo

1. Iniciar proceso
2. Declarar variables  
    B, A : Real  
    PER, SUP : Real
3. LEER B, A
4. Calcular PER = (B + A) \* 2
5. Calcular SUP = B \* A
6. ESCRIBIR PER, SUP
7. Terminar el proceso



SUP: Variable de tipo real. Almacena la superficie del rectángulo

PER: Variable de tipo real. Almacena el perímetro del rectángulo.

### Codificación en Java

```

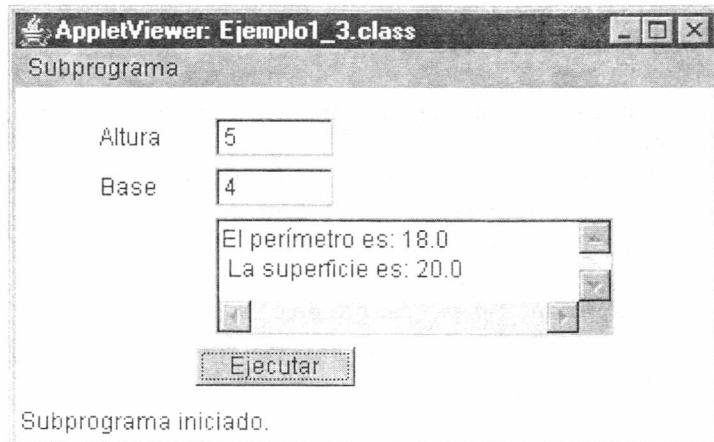
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo1_3 extends Applet implements
ActionListener{
    Label lbl1=new Label("Altura");
    Label lbl2=new Label("Base");
    TextField Altura=new TextField();
    TextField Base=new TextField();
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");

    public void init() {
        setLayout(null);
  
```

```
        setBackground(java.awt.Color.white);
        resize (350, 180);
        //controles
        lbl1.setBounds(40,15,60,20);
        lbl2.setBounds(40,40,60,20);
        Altura.setBounds(100,15,60,20);
        Base.setBounds(100,40,60,20);
        R.setBounds(100,65,200,60);
        Ejecutar.setBounds(90,130,80,20);
        Ejecutar.addActionListener(this);
        add(lbl1);add(lbl2);add(Altura);add(Base);
        add(R);add(Ejecutar);
    }
    public void actionPerformed(ActionEvent ac){
        float B, A, PER, SUP;
        A=Float.parseFloat(Altura.getText());
        B=Float.parseFloat(Base.getText());
        PER = (B + A) * 2;
        SUP = B * A;
        R.append("El perímetro es: "+PER);
        R.append("\n La superficie es: "+SUP);
    }
}
```

Nótese que hemos usado una notación diferente para declarar e inicializar los controles java, en mi opinión es mejor y mas compacta que la mostrada en los dos ejemplo anteriores, pero ambas son válidas.



## Ejemplo 1.4

Construya un algoritmo, que dados los tres lados de un triángulo, pueda determinar su área. Esta la calculamos aplicando la siguiente fórmula:

$$\text{Area} = \sqrt{S * (S - L1) * (S - L2) * (S - L3)}$$

En donde S es:  $S = (L1 + L2 + L3) / 2$

### Solución

Declarar Variables

L1, L2, L3, S, Area : Real

Leer L1, L2, L3

Realizar el cálculo del área

Escribir resultado

L1, L2, L3, son variables de tipo real, que representan los lados de un triángulo.

### Pseudocódigo

1. Iniciar proceso
2. Declarar variables  
L1, L2, L3, S, Area : Real
3. Calcular  $S = (L1 + L2 + L3) / 2$
4. Calcular Area =  $(S * (S - L1) * (S - L2) * (S - L3))^{0.5}$
5. ESCRIBIR Area
7. Terminar el proceso

S se utiliza como una variable auxiliar para el cálculo del área.

AREA almacena el área del triángulo.

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

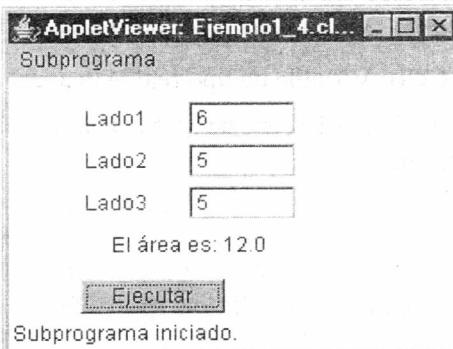
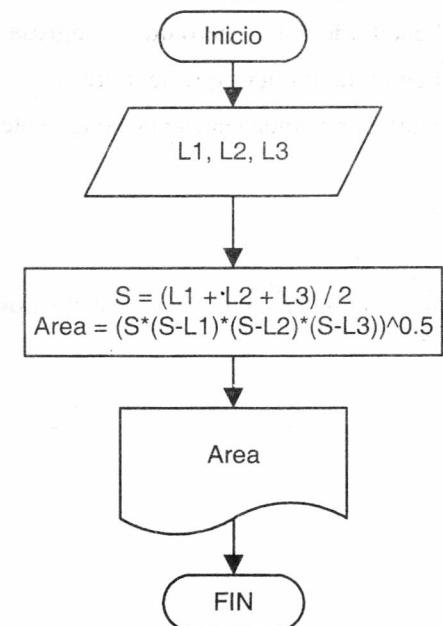
public class Ejemplo1_4 extends Applet implements
ActionListener{
    Label lbl1=new Label("Lado1");
    Label lbl2=new Label("Lado2");
    Label lbl3=new Label("Lado3");
    TextField Lado1=new TextField();
    TextField Lado2=new TextField();
```

```
TextField Lado3=new TextField();
Label R=new Label();
Button Ejecutar=new Button("Ejecutar");

public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (250, 140);
    //controles
    lbl1.setBounds(40,15,60,20);
    lbl2.setBounds(40,40,60,20);
    lbl3.setBounds(40,65,60,20);
    Lado1.setBounds(100,15,60,20);
    Lado2.setBounds(100,40,60,20);
    Lado3.setBounds(100,65,60,20);
    R.setBounds(55,90,200,20);
    Ejecutar.setBounds(40,120,80,20);
    Ejecutar.addActionListener(this);

    add(lbl1);
    add(lbl2);
    add(lbl3);
    add(Lado1);
    add(Lado2);
    add(Lado3);
    add(R);
    add(Ejecutar);
}

public void actionPerformed(ActionEvent ac){
    float L1, L2, L3, S;
    double AArea;
    L1=Float.parseFloat(Lado1.getText());
    L2=Float.parseFloat(Lado2.getText());
    L3=Float.parseFloat(Lado3.getText());
    S = (L1 + L2 + L3) / 2;
    AArea = Math.sqrt(S*(S-L1)*(S-L2)*(S-L3));
    R.setText("El área es: "+AArea);
}
}
```



### Ejemplo 1.5

Escriba un pseudocódigo que permita calcular e imprimir el cuadrado y el cubo de un número entero positivo NUM.

#### Solución:

Declarar Variables    NUM : Entero

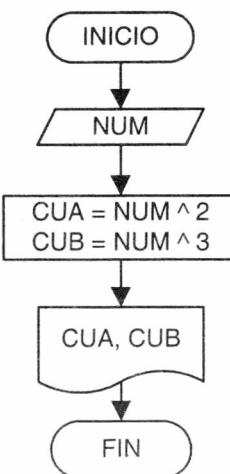
Ingresar el número.

Calcular el cuadrado y cubo del número ingresado.

Mostrar los resultados obtenidos.

#### Pseudocódigo

1. Iniciar proceso
2. Declarar variables  
NUM : Entero  
CUA, CUB : Real
3. LEER NUM
4. Calcular CUA = NUM ^ 2
5. Calcular CUB = NUM ^ 3
6. ESCRIBIR CUA, CUB
7. Terminar el proceso



CUA: Variable de tipo real. Almacena el cuadrado del número que se ingresa.

CUB: Variable de tipo real. Almacena el cubo del número que se ingresa.

El cálculo del cubo de NUM, también lo hubiéramos podido realizar de la siguiente manera:

```
Calcular CUB = NUM * NUM * NUM  
Calcular CUB = (NUM ^ 2) * NUM  
Calcular CUB = CUA * NUM
```

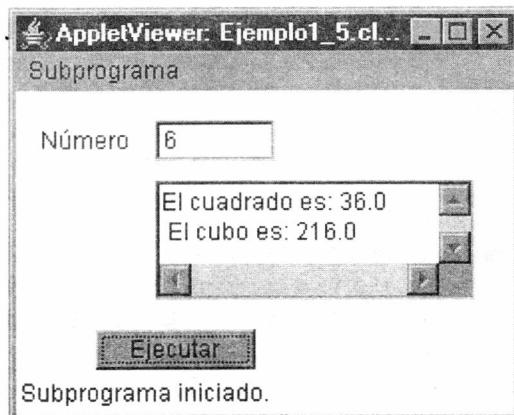
Este último es válido ya que anteriormente en la línea 4, CUA ya tiene el valor de la expresión NUM^2.

### Codificación en Java

```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
  
public class Ejemplo1_5 extends Applet implements  
ActionListener{  
    Label lbl1=new Label("Número");  
    TextField Numero=new TextField();  
    TextArea R=new TextArea();  
    Button Ejecutar=new Button("Ejecutar");  
  
    public void init() {  
        setLayout(null);  
        setBackground(java.awt.Color.white);  
        resize (250, 140);  
        //controles  
        lbl1.setBounds(10,15,60,20);  
        Numero.setBounds(70,15,60,20);  
        R.setBounds(70,45,160,60);  
        Ejecutar.setBounds(40,120,80,20);  
        Ejecutar.addActionListener(this);  
        add(lbl1);add(Numero);add(R);add(Ejecutar);  
    }  
    public void actionPerformed(ActionEvent ac){  
        int NUM;  
        double CUA, CUB;  
        NUM=Integer.parseInt(Numero.getText());  
        CUA = Math.pow(NUM,2);
```

```
CUB = Math.pow(NUM, 3);  
R.append("El cuadrado es: "+CUA);  
R.append("\n El cubo es: "+CUB);  
}  
}
```

Otra vez usamos la función Math.pow() para hallar tanto el cuadrado como el cubo del número.



## Ejemplo 1.6

Dada el código de matrícula y 5 calificaciones de un alumno obtenidas a lo largo del semestre; construya un pseudocódigo imprima la matrícula del alumno y el promedio de sus calificaciones.

### Solución:

Declarar Variables

CODIGO : Entero largo

C1, C2, C3, C4, C5, PRO : Real

Ingresar el código de matrícula y sus calificaciones.

Calcular el promedio de las calificaciones.

Reportar la matrícula y el promedio obtenido.

### Pseudocódigo

1. Iniciar proceso
2. Declarar variables

```
CODIGO : Entero largo
C1, C2, C3, C4, C5, PRO : Real
3. LEER Mat, C1, C2, C3, C4, C5
4. Calcular PRO = (C1 + C2 + C3 + C4 + C5) / 5
4. ESCRIBIR MAT, PRO
5. Terminar el proceso
```

CODIGO es una variable de tipo entero, que representa la matrícula del alumno.

PRO: Variable de tipo real. Almacena el promedio de las calificaciones del alumno.

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;

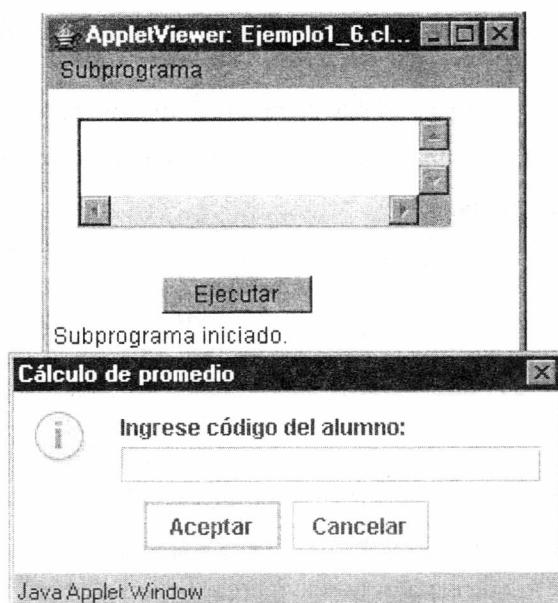
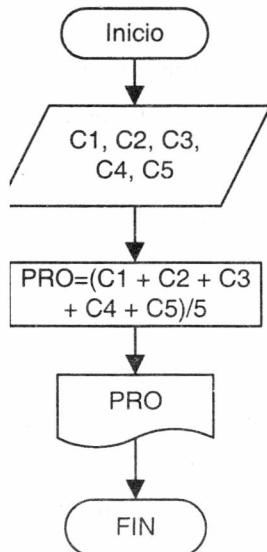
public class Ejemplo1_6 extends Applet implements
ActionListener{
    TextArea R=new TextArea();
    Button Ejecutar=new Button("Ejecutar");

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 140);
        //controles
        R.setBounds(15,15,200,60);
        Ejecutar.setBounds(60,100,80,20);
        Ejecutar.addActionListener(this);
        add(R);add(Ejecutar);
    }
    public void actionPerformed(ActionEvent ac){
        long CODIGO;
        float C1, C2, C3, C4, C5, PRO;
        CODIGO =
Integer.parseInt(JOptionPane.showInputDialog(null,
"Ingresa código del alumno:","Cálculo de promedio",
",JOptionPane.INFORMATION_MESSAGE));
        C1 =
Integer.parseInt(JOptionPane.showInputDialog(null,
```

```

    "Ingrese nota 1;","Cálculo de promedio
    ", JOptionPane.INFORMATION_MESSAGE));
    C2 =
    Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
    nota 2;","Cálculo de promedio
    ", JOptionPane.INFORMATION_MESSAGE));
    C3 =
    Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
    nota 3;","Cálculo de promedio
    ", JOptionPane.INFORMATION_MESSAGE));
    C4 =
    Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
    nota 4;","Cálculo de promedio
    ", JOptionPane.INFORMATION_MESSAGE));
    C5 =
    Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
    nota 5;","Cálculo de promedio
    ", JOptionPane.INFORMATION_MESSAGE));
    PRO=(C1 + C2 + C3 + C4 + C5)/5;
    R.append("El alumno: "+CODIGO);
    R.append("\nTiene de promedio: "+PRO);
}
}

```



Note el uso de la función **JOptionPane.showInputDialog** en cual nos permite ingresar texto mediante una caja de diálogo de java. Si no deseamos usar esta caja de diálogo podemos crear 5 controles **TextField** para ingresar cada uno de las notas del alumno.

### **Problema 1.7**

Escriba un pseudocódigo, que dado el nombre de un dinosaurio, su peso y su longitud, expresados estos dos últimos en libras y pies, respectivamente; escriba el nombre del dinosaurio, su peso expresado en kilogramos y su longitud expresada en metros.

#### **Consideraciones:**

- ❖ Para convertir de libras a kilogramos, multiplica por 0.4535924
- ❖ Para convertir de pies a metros, multiplicar por 0.3048006

#### **Solución:**

Declarar Variables: NOM, PES, LON

Ingresar nombre, peso y longitud

Transformar cantidades a kilogramos y metros respectivamente.

Mostrar resultados

Donde: NOM es una variable de tipo cadena de caracteres, que expresa el nombre el dinosaurio.

PES una variable de tipo real, que expresa el peso del dinosaurio en libras.

LON es una variable de tipo real, que expresa la longitud del dinosaurio en pies.

#### **Pseudocódigo**

1. Iniciar proceso
2. Declarar variables  
    NOM : Cadena de caracteres  
    PES, LON : Real
3. LEER NOM, PES, LON
4. Calcular PES = PES \* 0.4535924
5. Calcular LON = LON \* 0.3048006
6. ESCRIBIR NOM, PES, LON
7. Terminar el proceso

Nótese que no hemos declarado variables de almacenamiento para los resultados, por lo que el valor que se ingreso de la variable PES será reemplazado con el de la expresión PES \* 0.4535924, lo mismo sucede con la variable LON, por ejemplo:

sea el valor ingresado de PES = 500 luego de ejecutar la línea 4 el nuevo valor de PES será  $500 * 0.4535924$  ósea 226.7962, de manera análoga si LON = 250 pies es igual a 76.20 metros.

Nótese aquí el uso de una variable de tipo cadena de carácter y también el uso de cin para poder leer cualquier tipo de variable indistintamente.

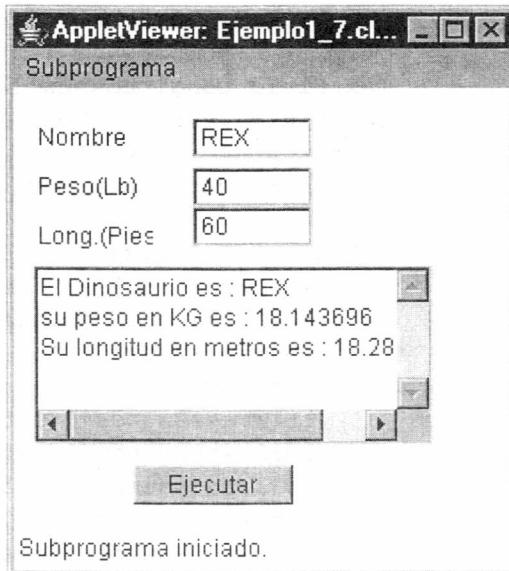
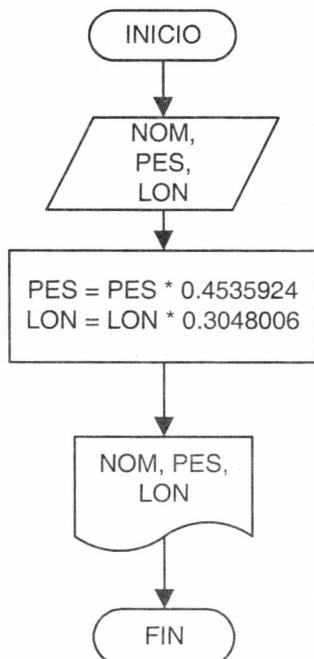
### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo1_7 extends Applet implements
ActionListener{
    Label lbl1=new Label("Nombre");
    Label lbl2=new Label("Peso(Lb)");
    Label lbl3=new Label("Long.(Pies)");
    TextField Nombre=new TextField();
    TextField Longitud=new TextField();
    TextField Peso=new TextField();
    TextArea R=new TextArea();
    Button Ejecutar=new Button("Ejecutar");

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 220);
        //controles
        lbl1.setBounds(10,15,60,20);
        lbl2.setBounds(10,40,60,20);
        lbl3.setBounds(10,65,60,20);
        Nombre.setBounds(90,15,60,20);
        Peso.setBounds(90,40,60,20);
        Longitud.setBounds(90,60,60,20);
        R.setBounds(10,90,200,90);
        Ejecutar.setBounds(60,190,80,20);
        Ejecutar.addActionListener(this);
        add(lbl1);add(lbl2);add(lbl3);
        add(Nombre);add(Longitud);add(Peso);
        add(R);add(Ejecutar);
    }
}
```

```
public void actionPerformed(ActionEvent ac){  
    String NOM;  
    float PES, LON;  
    NOM=Nombre.getText();  
    PES=Float.parseFloat(Peso.getText());  
    LON=Float.parseFloat(Longitud.getText());  
    PES *= 0.4535924;  
    LON *= 0.3048006;  
    R.append("El Dinosaurio es : "+NOM);  
    R.append("\nsu peso en KG es : "+PES);  
    R.append("\nSu long en metros es :" +LON);  
}  
}
```



### Problema 1.8

Construya un algoritmo que sea capaz de intercambiar el valor de tres variables, de tal manera que sean las variables A, B, C, el valor de B se almacene en A, B obtenga el valor de C y C el valor de A.

### Solución

Este ejercicio es muy interesante, analicemos, con valores reales, por ejemplo:

A= 2; B= 5; C=8

veamos como resultaría el intercambio de variables

- 1) Si A = B tendríamos A=5
- 2) Si B= C tendríamos B= 8
- 3) Si C=A tendríamos C= 2

Revisando esta solución concluimos que las líneas 1 y 2 son correctas pero la tercera no, esto es debido que al haberse ejecutado la línea 1 el valor de A es 5, y al llegar a la línea 3 el valor correcto de C sería 5 y no 2 como lo planteamos. El valor anterior de A ya no existe después de la ejecución de la línea 1, entonces ¿como hacemos para resolver esto?, pues es simple, debemos usar una variable auxiliar en donde podamos almacenar el primer valor de A, de esa manera no perderemos este dato, por lo tanto siguiendo con el ejemplo tendríamos lo siguiente

- 1) AUX = A tendríamos AUX=2 ,aquí almacenamos el valor de A
- 2) Si A = B tendríamos A=5
- 3) Si B= C tendríamos B= 8
- 4) Si C=AUX tendríamos C= 2

De esta manera los valores intercambiados de las variables serían A=5, B=8, C=2.

### **Pseudocódigo**

1. Iniciar proceso
2. Declarar variables  
A, B, C, AUX : Entero
3. Calcular AUX = A
4. Calcular A=B
5. Calcular B=C
6. Calcular C=AUX
7. ESCRIBIR A, B, C
8. Terminar el proceso

Podemos ver claramente el uso de la variable AUX, la cual nos permite almacenar el valor de A, por lo tanto AUX debe ser del mismo tipo que las otras variables.

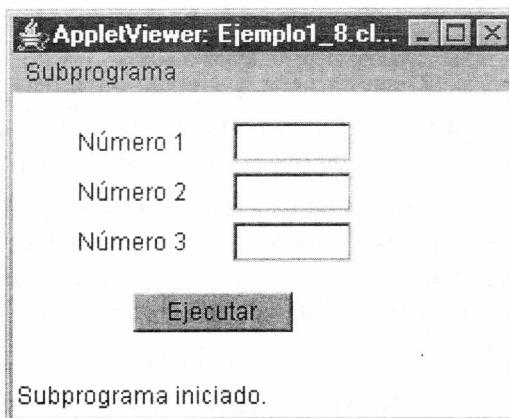
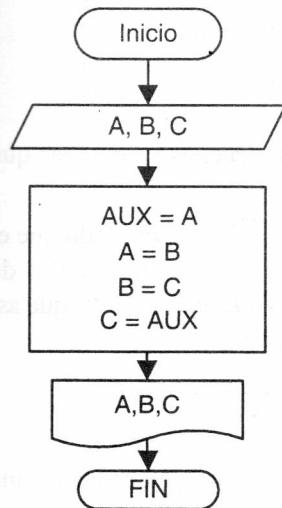
### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo1_8 extends Applet implements
ActionListener{
    Label lbl1=new Label("Número 1");
    Label lbl2=new Label("Número 2");
    Label lbl3=new Label("Número 3");
    TextField AA=new TextField();
    TextField BB=new TextField();
    TextField CC=new TextField();
    Button Ejecutar=new Button("Ejecutar");

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 140);
        //controles
        lbl1.setBounds(30,15,60,20);
        lbl2.setBounds(30,40,60,20);
        lbl3.setBounds(30,65,60,20);
        AA.setBounds(110,15,60,20);
        BB.setBounds(110,40,60,20);
        CC.setBounds(110,65,60,20);
        Ejecutar.setBounds(60,100,80,20);
        Ejecutar.addActionListener(this);
        add(lbl1);add(lbl2);add(lbl3);
        add(AA);add(BB);add(CC);add(Ejecutar);
    }
    public void actionPerformed(ActionEvent ac){
        int A, B, C, AUX;
        A=Integer.parseInt(AA.getText());
        B=Integer.parseInt(BB.getText());
        C=Integer.parseInt(CC.getText());
        //Intercambiar valores
        AUX = A;
        A = B;
        B = C;
```

```
C = AUX;  
AA.setText(String.valueOf(A));  
BB.setText(String.valueOf(B));  
CC.setText(String.valueOf(C));  
}  
}
```



### Problema 1.9

Construya un pseudocódigo que ingresado un número entero de 3 cifras, se obtenga como resultado el número ingresado y el inverso de dicho número.

### Solución

Este es un ejemplo interesante, podríamos resolver leyendo el número dígito por dígito y luego simplemente imprimirlo invertido, pero no es una buena solución. Para resolver este ejercicio necesitaremos de las funciones DIV y MOD. El algoritmo sería el siguiente:

Declarar Variables

Invertir el número

Escribir el número normal y el número invertido

Bueno es un algoritmo muy sencillo, veamos su implementación en Pseudocódigo.

### Pseudocódigo

1. Iniciar proceso

2. Declarar Variables  
NUM, NUMINV, U, D, C, AUX : Entero
3. Calcular AUX = NUM
4. Calcular C = NUM DIV 100
5. Calcular NUM = NUM MOD 100
6. Calcular D = NUM DIV 10
7. Calcular U = NUM MOD 10
8. Calcular NUMINV = U\*100 + D\*10 + C
9. ESCRIBIR AUX, NUMINV
10. Terminar el proceso

Las variables U, D, C, están referidos a las unidades, decenas y centenas que tiene un número de tres cifras, respectivamente.

Veamos como funciona este pseudocódigo con un ejemplo, suponiendo que el número ingresado es NUM=734, por ahora no vamos a entrar en detalles de consistenciar si el valor leido es o no de tres cifras, por lo que asumiremos que así es. Entonces al ejecutar el algoritmo tendríamos lo siguiente:

El primer paso es guardar el valor del número ingresado

$$3. \text{AUX} = \text{NUM} \implies \text{AUX} = 734$$

Luego ejecutamos la línea 4 para poder extraer el primer valor, recordemos que el operador DIV realiza una división entera.

$$4. \text{C} = \text{NUM DIV } 100 = 734 \text{ DIV } 100 = 7 \implies \text{C} = 7$$

Una vez extraído el primer valor es necesario eliminarlo del número para poder seguir con los otros, esto lo logramos con el uso del operador MOD, el cual nos devuelve el residuo de una división entera.

$$5. \text{NUM} = \text{NUM MOD } 100 = 734 \text{ MOD } 100 = 34 \implies \text{NUM} = 34$$

El nuevo valor de NUM ahora es 34, de aquí ya es mas fácil usamos otra vez DIV y MOD para obtener lo números que nos faltan

$$6. \text{D} = \text{NUM DIV } 10 = 34 \text{ DIV } 10 = 3 \implies \text{D} = 3$$

$$7. \text{U} = \text{NUM MOD } 10 = 34 \text{ DIV } 10 = 4 \implies \text{U} = 4$$

Ya tenemos el número descompuesto, ahora solo nos queda realizar una operación sencilla para unirlo, solo que el resultado ahora será el número original 734 invertido, ósea debemos obtener 437.

$$8. \text{NUMINV} = \text{U}*100 + \text{D}*10 + \text{C} = 4*100 + 3*10 + 7 \implies \text{NUMINV} = 437$$

El ejercicio nos pide, devolver el número original y el invertido, recordemos que

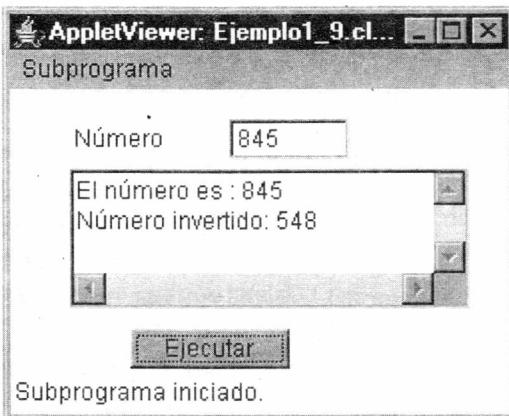
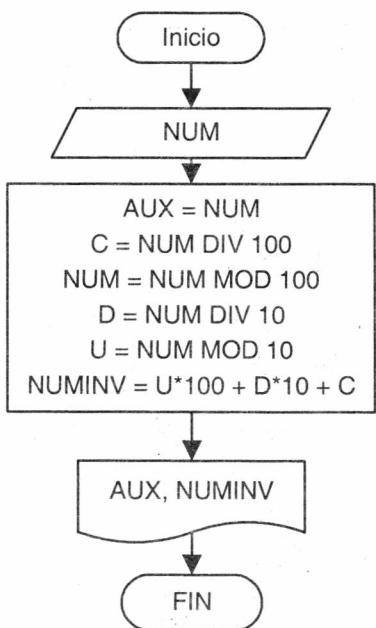
el número original fue almacenado en la variable AUX, ya que NUM, después de del proceso, terminó con el valor de 34.

## 9. ESCRIBIR AUX, NUMINV

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo1_9 extends Applet implements
ActionListener{
    Label lbl1=new Label("Número");
    TextField Numero=new TextField();
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize(250, 140);
        //controles
        lbl1.setBounds(30,15,60,20);
        Numero.setBounds(110,15,60,20);
        R.setBounds(30,40,200,70);
        Ejecutar.setBounds(60,120,80,20);
        Ejecutar.addActionListener(this);
        add(lbl1);add(Numero);
        add(R);add(Ejecutar);
    }
    public void actionPerformed(ActionEvent ac){
        int NUM, NUMINV, U, D, C, AUX;
        NUM=Integer.parseInt(Numero.getText());
        AUX = NUM;
        C = NUM/100;
        NUM = NUM%100;
        D = NUM/10;
        U = NUM%10;
        NUMINV = U*100 + D*10 + C;
        R.append("El número es : "+AUX);
        R.append("\nNúmero invertido: "+NUMINV);
    }
}
```

Nótese en Java el uso del operador / en vez de Div, esto es posible solo si los números son de tipo entero, decidí a eso podemos obtener la parte entera de la división de ambos números.



## Problema 1.10

Escriba un algoritmo que calcule el número mínimo de billetes 20, 10, 5, 1 dólares, que se necesita para cambiar un cheque. Considere que el valor del cheque es un número entero.

### Solución

Este ejemplo es muy sencillo, el uso de los operadores DIV y MOD nos ayudará en la resolución del problema, el cual consiste en cambiar un cheque, en el número óptimo de billetes de cuatro denominaciones distintas (20, 10, 5, 1), este es un ejemplo típico en algoritmia y programación para principiantes, su algoritmo es el siguiente:

LEER Importe del cheque

Calcular número máximo de billetes en denominaciones de 20.

Calcular número máximo de billetes en denominaciones de 10.

Calcular número máximo de billetes en denominaciones de 5.

Calcular número máximo de billetes en denominaciones de 1.

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
CANT, B20, B10, B5, B1 : Entero
3. Calcular B20 = CANT DIV 20
4. Calcular CANT = CANT MOD 20
5. Calcular B10 = CANT DIV 10
6. Calcular CANT = CANT MOD 10
7. Calcular B5 = CANT DIV 5
8. Calcular B1 = CANT MOD 5
9. ESCRIBIR B20, B10, B5, B1
10. Terminar el proceso

B20, B10, B5, B1, representan la cantidad de billetes de 20, 10, 5 y 1 dólares, respectivamente, en los cuales será cambiado el cheque.

Veamos como se ejecuta este pseudocódigo, suponiendo que el valor del cheque es 4152, tenemos lo siguiente:

3.  $B20 = \text{CANT DIV } 20 = 4152 \text{ DIV } 20 \implies B20 = 207$
4.  $\text{CANT} = \text{CANT MOD } 20 = 4152 \text{ DIV } 20 \implies \text{CANT} = 12$
5.  $B10 = \text{CANT DIV } 10 = 12 \text{ DIV } 10 \implies B10 = 1$
6.  $\text{CANT} = \text{CANT MOD } 10 = 12 \text{ MOD } 10 \implies 2$
7.  $B5 = \text{CANT DIV } 5 = 2 \text{ DIV } 5 \implies B5 = 0$
8.  $\text{B1} = \text{CANT MOD } 5 = 2 \text{ MOD } 5 \implies \text{B1} = 2$

Por lo tanto el cheque de 4152 dólares se cambiaría de la siguiente manera:

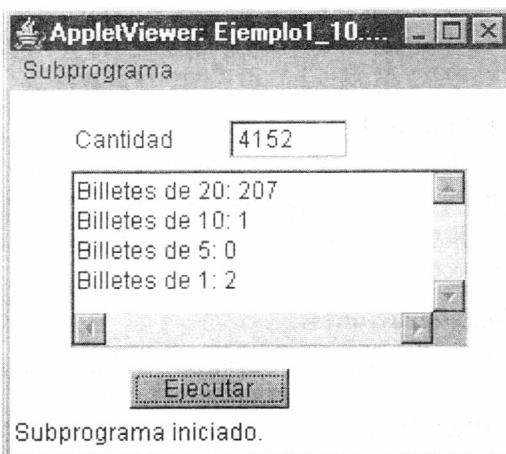
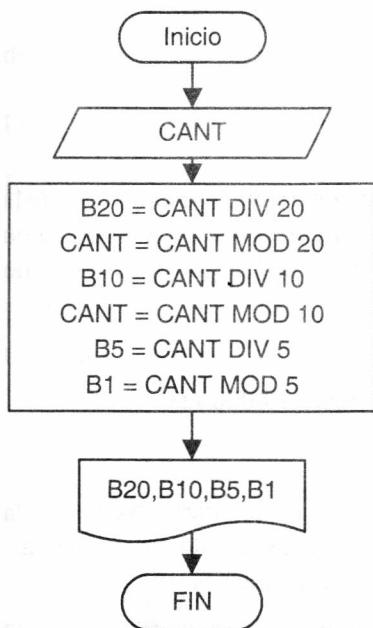
- 207 billetes de 20 dólares
- 1 billete de 10 dólares
- 0 billetes de 5 dólares
- 2 billetes de 1 dolar

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
```

```
public class Ejemplo1_10 extends Applet implements
ActionListener{
    Label lbl1=new Label("Cantidad");
    TextField Cantidad=new TextField();
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 160);
        //controles
        lbl1.setBounds(30,15,60,20);
        Cantidad.setBounds(110,15,60,20);
        R.setBounds(30,40,200,90);
        Ejecutar.setBounds(60,140,80,20);
        Ejecutar.addActionListener(this);
        add(lbl1);
        add(Cantidad);
        add(R);
        add(Ejecutar);
    }
    public void actionPerformed(ActionEvent ac){
        int CANT, B20, B10, B5, B1;
        CANT=Integer.parseInt(Cantidad.getText());
        //operaciones
        B20 = CANT/20;
        CANT = CANT%20;
        B10 = CANT/10;
        CANT = CANT%10;
        B5 = CANT/5;
        B1 = CANT%5;
        //resultados
        R.append("Billetes de 20: "+B20);
        R.append("\nBilletes de 10: "+B10);
        R.append("\nBilletes de 5: "+B5);
        R.append("\nBilletes de 1: "+B1);
    }
}
```



## Ejercicios propuestos

### Ejercicio 1

Construya un pseudocódigo, que dado los datos enteros A y B, escriba el resultado de la siguiente expresión:

$$\frac{(A + B)^2}{3}$$

### Ejercicio 2

En una Casa de Cambio necesitan construir un programa tal que dado como dato una cantidad expresada en dólares, convierta esa cantidad a nuevos soles.

### Ejercicio 3

Construya un algoritmo, que dado el radio y la altura de un cilindro, calcule e imprima el área y su volumen.

#### Consideraciones:

- ❖ El volumen de un cilindro lo calculamos aplicando la siguiente fórmula:

$$\text{Volumen} = \pi * \text{radio}^2 * \text{altura}$$

❖ La superficie del cilindro lo calculamos como:

$$\text{Area} = 2 * \pi * \text{radio} * \text{altura}$$

## Ejercicio 4

Una persona compró una estancia en un país sudamericano. La extensión de la estancia está especificada en acres. Construya un algoritmo, tal que dado como dato la extensión del campo en "acres", calcule e imprima la extensión del mismo en hectáreas.

### Consideraciones:

❖ 1 acre es igual a 4 047 m<sup>2</sup>

❖ 1 hectárea tiene 10 000 m<sup>2</sup>

## Ejercicio 5

En las olimpiadas de invierno el tiempo que realizan los participantes en la competencia de velocidad en pista, se mide en minutos, segundos y centésimas. La distancia que recorren, por otra parte, se expresa en metros.

Construya un pseudocódigo que calcule la velocidad de los participantes, en kilómetros por hora, de las diferentes competencias.

### Consideraciones:

❖ El tiempo debemos expresarlo en segundos, por lo que para hacerlo aplicaremos la siguiente fórmula:

$$\text{TIEMSEG} = \text{Minutos} * 60 + \text{Segundos} + \text{Centesimas} / 100$$

❖ Luego podemos calcular la velocidad, expresada en metros sobre segundos:

$$\text{VELOMS} = \frac{\text{Distancia (metros)}}{\text{TIEMSEG (Segundos)}}$$

❖ Para obtener la velocidad en kilómetros por hora, aplicamos la siguiente fórmula:

$$\text{VELOKH} = \text{VELOMS} * \frac{3600 \quad (\text{Kilómetros})}{1000 \quad (\text{Hora})} = \text{VELOMS} * 3.6 \text{ KM/H}$$

## Ejercicio 6

Construya un algoritmo que sea capaz de intercambiar el valor de cinco variables, de tal manera que sean las variables A, B, C, D, E el valor de C se almacene en A, D obtenga el valor de B, B el valor de E, E sea igual a A y C posea el valor de D.

### Ejercicio 7

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

#### Consideraciones:

- ❖ Un cono tiene la siguiente forma:



- ❖ El área de la base se calcula con base en la siguiente fórmula:

$$AB = \pi * \text{RADIO}^2$$

- ❖ El área lateral se calcula:

$$AL = \pi * \text{RADIO} * \text{GENE}$$

- ❖ El área total se calcula como:

$$AT = AB + AL$$

- ❖ El volumen se calcula de esta forma:

$$VOL = \frac{1}{3} * AB * ALTU$$

### Ejercicio 8

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

#### Consideraciones:

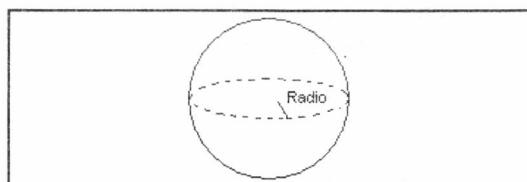
- ❖ El área de una esfera la calculamos de esta forma:

$$AREA = 4 * \pi * \text{RADIO}^2$$

❖ El volumen de una esfera lo calculamos de esta forma:

$$\text{VOL} = \frac{4}{3} * \text{Pi} * \text{RADIO}^3$$

❖ Una esfera tiene la siguiente forma:



### Ejercicio 9

Construya un pseudocódigo que calcule la distancia entre dos puntos dados P1 y P2.

#### Consideraciones:

❖ Para calcular la distancia "D" entre dos puntos dados P1 y P2 aplicamos la siguiente fórmula:

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

# Capítulo

# Estructuras Lógicas

# Selectivas

Este capítulo contiene:

Estructura Si...Entonces (Selección simple)

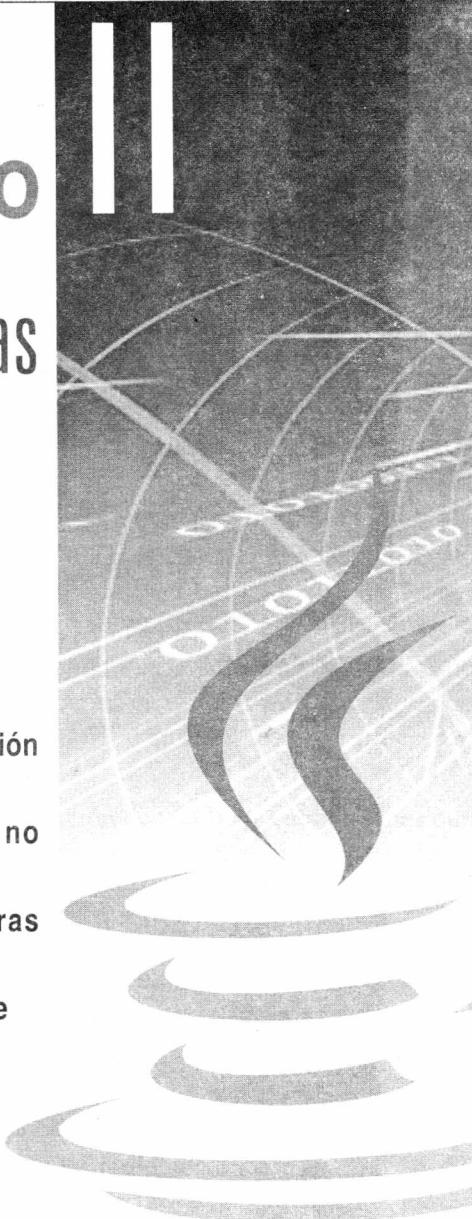
Estructura Si...Entonces...Si no (Alternativa doble)

Anidamiento de Estructuras condicionales

La estructura de selección múltiple

Ejemplos desarrollados

Ejemplos propuestos





## Introducción

Las estructuras lógicas selectivas se encuentran en la solución algorítmica de casi todo tipo de problemas. La utilizamos cuando en el desarrollo de la solución de un problema debemos **tomar una decisión**, para establecer un proceso o señalar un camino alternativo a seguir.

Esta toma de decisión se basa en la evaluación de una o más condiciones que nos señalarán como alternativa o consecuencia, la rama a seguir.

### Estructura Si...Entonces (Selección simple)

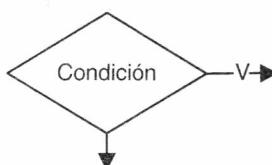
La estructura selectiva **Si...Entonces** permite que el flujo del algoritmo siga por un camino específico si se cumple una condición o conjunto de condiciones. Si al evaluar la condición (o condiciones) el resultado es verdadero, entonces se ejecuta(n) cierta(s) operación(es). Luego se continúa con la secuencia normal del diagrama

La sintaxis para este tipo de estructura es la siguiente:

```
Si condición entonces
    Hacer operación
Fin_Si(Fin del condicional)
```

La *condición* es una expresión lógica. Y *operación* puede ser una operación simple o un grupo de operaciones.

El diagrama de flujo de la condicional Si simple es:



Si la condición es falsa no hace ninguna acción. Veamos unos ejemplos de una sentencia selectiva simple.

### Ejemplo 2.1

Construya un pseudocódigo tal, que dado como dato la calificación de un alumno en un examen, escriba "Aprobado" en caso de que esa calificación fuese mayor que 10.5.

#### Algoritmo

LEER nota

Comprobar si nota es mayor que 10.5

Escribir resultado si esta aprobado

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
Nota : Real
3. LEER Nota
4. Si Nota > 10.5 Entonces  
    4.1. Escribir "Aprobado"
5. Fin\_Si
6. Terminar el proceso

En la línea 4 realizamos la comprobación de la condicional para este ejercicio, solo en el caso de ser verdadero, se ejecuta la línea 4.1, mostrando el mensaje "Aprobado", caso contrario se ejecuta la línea 5, por lo tanto el pseudocódigo no muestra ningún resultado, de ser falsa la condicional.

Nótese que para cada sentencia Si...Entonces, se debe colocar un fin de sentencia mediante Fin\_Si

#### Codificación en Java

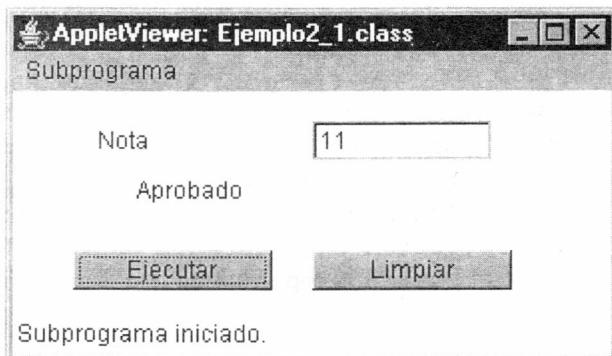
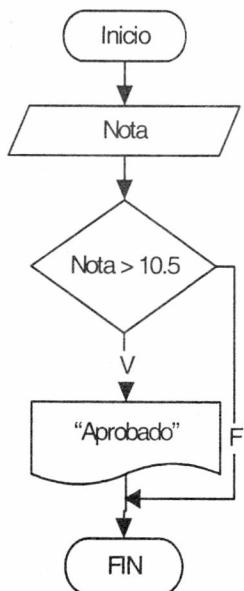
```
import java.awt.*;  
import java.applet.*;
```

## **Capítulo II: Estructuras lógicas selectivas**

```
import java.awt.event.*;

public class Ejemplo2_1 extends Applet implements
ActionListener{
    Label lbl1=new Label("Nota");
    Label R=new Label();
    TextField N= new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (300, 110);
        //controles
        lbl1.setBounds(40,15,60,20);
        N.setBounds(150,15,90,20);
        R.setBounds(60,40,100,20);
        Ejecutar.setBounds(30,80,100,20);
        Limpiar.setBounds(150,80,100,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl1);add(R);add(N);
        add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            float Nota;
            Nota=Float.parseFloat(N.getText());
            if (Nota > 10.5)
                R.setText("Aprobado");
        }
        if(str.equals("Limpiar")){
            R.setText("");
            N.setText("");
            N.requestFocus();
        }
    }
}
```



Note q en la codificación en Java la condicional Si es reemplazada por If luego viene la condición encerrada entre paréntesis; luego de eso vienen las acciones a realizar delimitadas por llaves las cuales indican el inicio y el fin del bloque de la condicional If. Para nuestro ejemplo solo se emite un mensaje de "Aprobado"

Por otro lado vemos claramente en el diagrama de flujo si la condición es falsa la secuencia del diagrama nos lleva a un punto en donde el flujo termina, para este caso en particular.

Hemos agregado a nuestro formulario dos controles button uno Ejecutar y otro Limpiar, anteriormente solo usamos un solo botón lo q no implicaba ningún tipo de programación a dicho botón simplemente agregarle la acción y ya, ahora debemos indicar que diferencia entre estos botón y según esto ejecute una acción específica. Por lo tanto primero ejecutamos la línea de comando:

```
String str=ac.getActionCommand();
```

mediante la cual podemos obtener la etiqueta del control button pulsado, luego de eso solo debemos usar la condicional if para comprobar si dicha etiqueta es igual a "Ejecutar" o "Limpiar". El botón limpiar borra el contenido de los controles especificados.

## Ejemplo 2.2

Dado como dato el sueldo de un trabajador, aplíquele un aumento del 17% si su sueldo es inferior a \$ 1000. Imprima en este caso, el nuevo sueldo del trabajador.

### Algoritmo

Leer Sueldo

## **Capítulo II: Estructuras lógicas selectivas**

---

Comprobar si sueldo cumple con la condición

Si cumple la condición, realizar el aumento y reportar nuevo sueldo

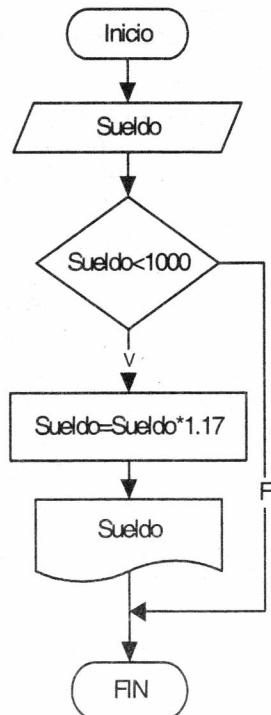
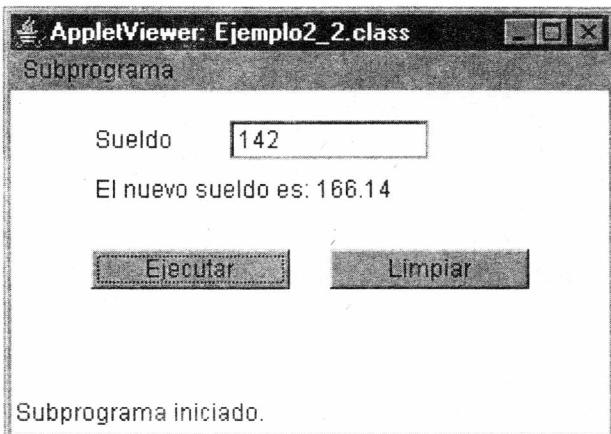
### **Pseudocódigo**

1. Iniciar proceso
2. Declarar Variables  
    Sueldo : Real
3. LEER Sueldo
4. Si Sueldo < 1000 Entonces  
    4.1. Sueldo = Sueldo \*1.17  
    4.2. Escribir Sueldo
5. Fin\_Si
6. Terminar el proceso

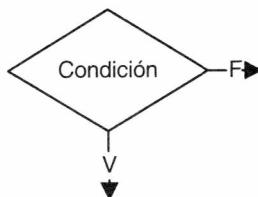
### **Codificación en Java**

```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
  
public class Ejemplo2_2 extends Applet implements  
ActionListener{  
    Label lbl1=new Label("Sueldo");  
    Label R=new Label("");  
    TextField S=new TextField();  
    Button Ejecutar=new Button("Ejecutar");  
    Button Limpiar=new Button("Limpiar");  
  
    public void init() {  
        setLayout(null);  
        setBackground(java.awt.Color.white);  
        resize (300, 150);  
        //controles  
        lbl1.setBounds(40,15,60,20);  
        S.setBounds(110,15,100,20);  
        R.setBounds(40,40,200,20);  
        Ejecutar.setBounds(40,80,100,20);  
        Limpiar.setBounds(160,80,100,20);  
        Ejecutar.addActionListener(this);  
        Limpiar.addActionListener(this);  
        add(lbl1);add(R);add(S);  
        add(Ejecutar);add(Limpiar);  
    }  
}
```

```
public void actionPerformed(ActionEvent ac){  
    String str=ac.getActionCommand();  
    if(str.equals("Ejecutar")){  
        //Declarar variables  
        double Sueldo;  
        //Leer datos  
        Sueldo=Double.parseDouble(S.getText());  
        //Calcular y mostrar datos  
        if (Sueldo < 1000){  
            Sueldo *= 1.17;  
            R.setText("Nuevo sueldo es:  
                      "+Sueldo);  
        }  
    }  
    if(str.equals("Limpiar")){  
        R.setText("");  
        S.setText("");  
        S.requestFocus();  
    }  
}
```



## Estructura Si...Entonces...Si no (Alternativa doble)



Como se puede ver la selección simple es limitada aunque muchas veces necesaria, por otro lado la alternativa doble nos permite tomar decisiones en ambos sentidos, es decir cuando la sentencia de comparación sea verdadero o cuando sea falso, en otras palabras cuando la respuesta de la comparación sea verdadera se ejecutarán una o más acciones, así mismo si la respuesta es falsa se ejecutarán acciones diferentes. Veamos el pseudocódigo el cual explica mejor el concepto de alternativa doble.

```
Si Condición Entonces
    Acciones1
sino
    Acciones2
Fin_Si {Fin del condicional}
```

Donde:

Condición expresa la condición o conjunto de condiciones a evaluarse.

Acciones1 expresa la operación o conjunto de operaciones que se van a realizar si la condición resulta verdadera.

Acciones2 expresa la operación o conjunto de operaciones que se van a realizar si la condición resulta falsa.

### Ejemplo 2.3

Construya un algoritmo, que dado como dato la calificación de un alumno en un examen, escriba "Aprobado" si su calificación es mayor que 10.5 y "Reprobado" en caso contrario.

### Solución

Nótese que este ejercicio, es más completo que el Ejemplo 2.1. El algoritmo es el siguiente:

Leer Nota

Comprobar si la Nota es mayor a 10.5, reportar "Aprobado"

Si no, Reportar "Reprobado"

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
Nota : Real
3. LEER Nota
4. Si Nota > 10.5 Entonces  
    4.1. Escribir 'Aprobado'
5. Si no  
    5.1. Escribir 'Reprobado'
6. Fin\_Si
7. Terminar el proceso

En este ejemplo se asume que la nota es válida, entiéndase nota válida si está entre 0 y 20, incluso.

### Codificación en Java

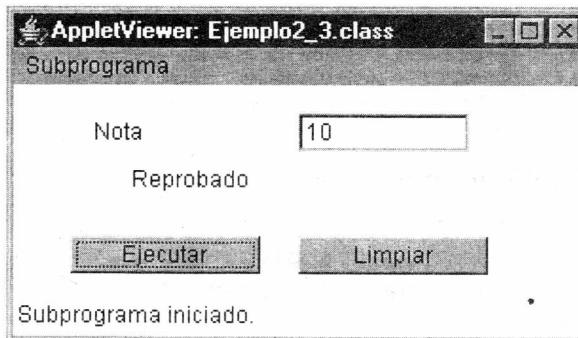
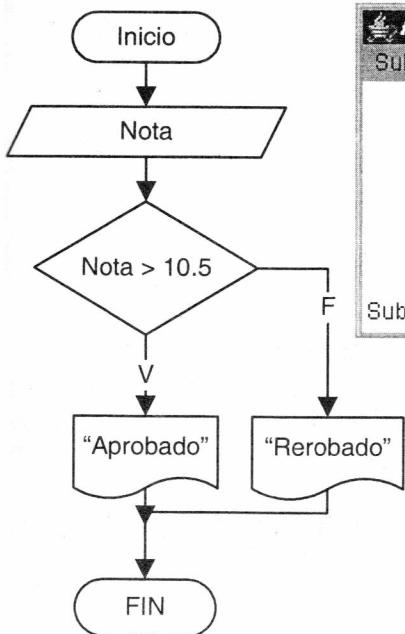
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo2_3 extends Applet implements
ActionListener{
    Label lbl1=new Label("Nota");
    Label R=new Label();
    TextField N= new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (300, 110);
        //controles
        lbl1.setBounds(40,15,60,20);
        N.setBounds(150,15,90,20);
        R.setBounds(60,40,100,20);
        Ejecutar.setBounds(30,80,100,20);
        Limpiar.setBounds(150,80,100,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
```

## Capítulo II: Estructuras lógicas selectivas

```
add(lbl1); add(R); add(N);
add(Ejecutar); add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        float Nota;
        Nota=Float.parseFloat(N.getText());
        if (Nota > 10.5)
            R.setText("Aprobado");
        else
            R.setText("Reprobado");
    }
    if(str.equals("Limpiar")){
        R.setText("");
        N.setText("");
        N.requestFocus();
    }
}
}
```



En la codificación en Java ahora tenemos la palabra "else" el reemplaza al "Si no" del algoritmo, por lo tanto se ejecutará else solo si la condición no se cumple.

Podemos obviar los paréntesis en la función If solo si ejecuta una línea de código, en caso contrario es necesario el uso de los paréntesis

## Ejemplo 2.4

Construya un algoritmo, que dado como dato el sueldo de un trabajador, le aplique un aumento del 17% si su sueldo es inferior a S/. 1000 y 12% en caso contrario. Imprima el nuevo sueldo del trabajador.

### Solución

Leer Sueldo

Comprobar si sueldo cumple con la condición

Si cumple la condición, incrementar el 17%

Si no, incrementar el 12 %

### Algoritmo

1. Iniciar proceso
2. Declarar Variables  
    Sueldo : Real
3. LEER Sueldo
4. Si Sueldo < 1000 Entonces  
        4.1. Sueldo = Sueldo \*1.17
5. Si no  
        5.2. Sueldo = Sueldo \*1.12
6. Fin\_Si
7. Escribir Sueldo
8. Terminar el proceso

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo2_4 extends Applet implements
ActionListener{
    Label lb11=new Label("Sueldo");
    Label R=new Label("");
    TextField S=new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");

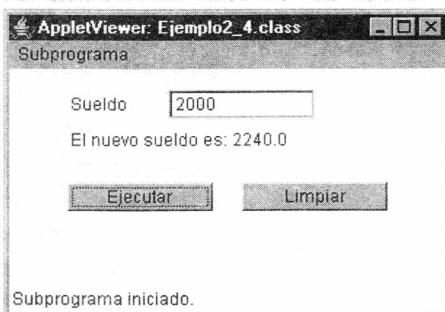
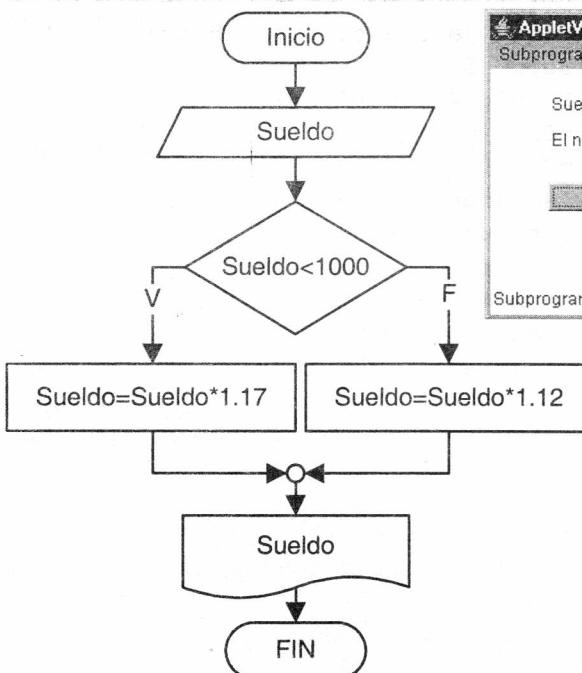
    public void init() {
        setLayout(null);
```

```
setBackground(java.awt.Color.white);
resize (300, 150);
lbl1.setBounds(40,15,60,20);
S.setBounds(110,15,100,20);
R.setBounds(40,40,200,20);
Ejecutar.setBounds(40,80,100,20);
Limpiar.setBounds(160,80,100,20);
Ejecutar.addActionListener(this);
Limpiar.addActionListener(this);
add(lbl1);add(R);add(S);
add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        double Sueldo;
        Sueldo=Double.parseDouble(S.getText());
        if (Sueldo < 1000)
            Sueldo *= 1.17;
        else
            Sueldo *= 1.12;
        R.setText("El nuevo sueldo es:
                  "+Sueldo);
    }
    if(str.equals("Limpiar")){
        R.setText("");
        S.setText("");
        S.requestFocus();
    }
}
}
```

Nótese en el diagrama de flujo el uso de pequeño circulo en donde se encuentran los flujos de la condicional, este es un conector de tal manera que indica que sea verdad o falso la condición, luego de ejecutar las acciones correspondientes el flujo continua hasta el conector y luego su camino hacia Fin

### **Anidamiento de Estructuras condicionales**

A menudo tendrá la necesidad de anidar una o más estructuras condicionales, ya sean simples o dobles, o combinaciones de ellas.



Se dice que las estructuras están anidadas, cuando hay unas dentro de ellas, esto lo veremos muy a menudo. Por ejemplo veamos el siguiente pseudocódigo

```

Si Condición1 Entonces
  Si Condición2 Entonces
    Acciones1
  Fin_si
Fin_Si
  
```

Aquí observamos que primero se debe cumplir la Condición1, para luego evaluar la Condición2, solo al cumplirse esta última, se procederá a la ejecución de Acciones1.

```

Si Condición1 Entonces
  Acciones1
  Si Condición2 Entonces
    Acciones2
  Si no
    Acciones3
  Fin_si
Fin_Si
  
```

En este ejemplo, una vez cumplida la Condición1, realizamos la Acción1 y luego

evaluamos la Condición2, dependiendo el resultado de la Condición2, se procederá a realizar Acciones2 o Acciones3.

Podemos tener muchas combinaciones de anidamiento, pero en el fondo todos se manejan de la misma manera. Veamos con unos ejemplos.

### **Ejemplo 2.5**

Implemente la validación de la nota ingresada en el Ejemplo 2.3.

#### **Solución**

En el Ejemplo 2.3, mencionamos que se suponía una nota válida, en la práctica veremos que nada se supone, todo ingreso de dato debe estar correctamente validado. Para esto, las estructuras condicionales nos ayudan en esta tarea.

Leer nota

Si es nota válida

    Comprobar si la Nota es mayor a 10.5, reportar "Aprobado"

        Si no, Reportar "Reprobado"

En caso de no ser una nota válida, mandarle un mensaje al usuario.

#### **Pseudocódigo**

1. Iniciar proceso
2. Declarar Variables  
    Nota : Real
3. LEER Nota
4. Si Nota $\geq$ 0 Y Nota $\leq$ 20 Entonces
  - 4.1. Si Nota > 10.5 Entonces
    - 4.1.1. Escribir "Aprobado"
  - 4.2. Si no
    - 5.2.1. Escribir "Reprobado"
  - 4.3. Fin\_Si
5. Si no
  - 5.1. Escribir "Nota no es valida"
6. Fin\_Si
7. Terminar el proceso

Una simple línea de código, puede hacer mucha diferencia. La línea 4 muestra el uso del operador Y, esto obliga a que tanto la condición Nota $\geq$ 0 y la condición Nota $\leq$ 20 sean verdaderas, para que la expresión "Nota $\geq$ 0 Y Nota $\leq$ 20" sea verdadera, solo así podemos decir que es una nota válida, en caso contrario, se reportará una nota no válida.

## Codificación en Java

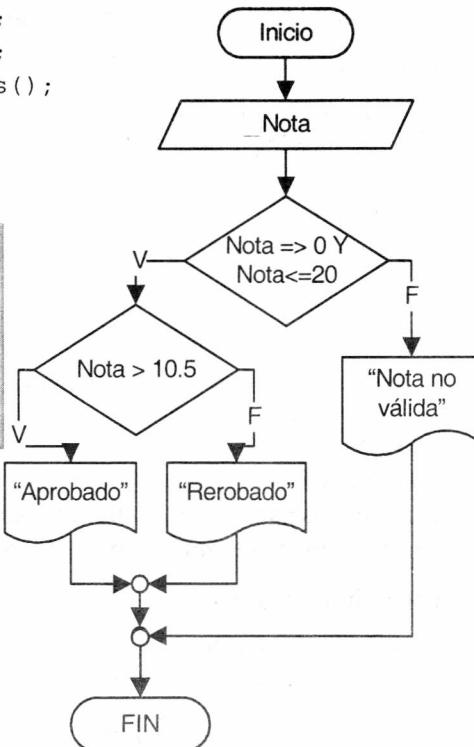
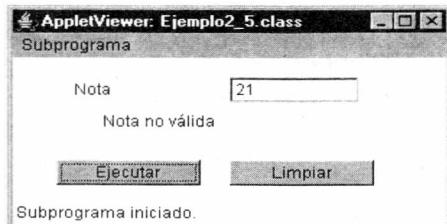
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo2_5 extends Applet implements
ActionListener{
    Label lbl1=new Label("Nota");
    Label R=new Label();
    TextField N= new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (300, 110);
        lbl1.setBounds(40,15,60,20);
        N.setBounds(150,15,90,20);
        R.setBounds(60,40,100,20);
        Ejecutar.setBounds(30,80,100,20);
        Limpiar.setBounds(150,80,100,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl1);add(R);add(N);
        add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            float Nota;
            Nota=Float.parseFloat(N.getText());
            if (Nota>=0 & Nota<=20)
                if (Nota > 10.5)
                    R.setText ("Aprobado");
                else
                    R.setText ("Reprobado");
            else
                R.setText ("Nota no válida");
        }
        if(str.equals("Limpiar")){
    
```

```

        R.setText(" ");
        N.setText("");
        N.requestFocus();
    }
}
}

```



### Ejemplo 2.6

Se requiere la implementación de un pseudocódigo que dado un número entero positivo mayor que cero de como resultado si dicho número es par o impar. El ejercicio requiere la validación del dato de entrada.

### Solución

El problema especifica la validación del dato de entrada, eso quiere decir que si el número ingresado no es mayor a cero, se dará fin a la ejecución del programa o se reportará como número no válido (depende del programador). El algoritmo sería el siguiente:

Leer Dato

Comprobar dato de entrada, si es número entero positivo

Si dato es válido, realizar la comprobación para ver si es par

    Si es Par, reportar par

    Si no, reportar impar

Si el dato no es válido, fin del algoritmo

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables
  - Dato : Entero
3. LEER Dato
4. Si Dato > 0 Entonces
  - 4.1. Si (Dato MOD 2)=0 Entonces
    - 4.1.1. Escribir 'Número ingresado es Par'
  - 4.2. Si no
    - 4.2.1. Escribir 'Número ingresado es Impar'
  - 4.3. Fin\_Si
5. Fin\_Si
6. Terminar el proceso

Un número es par cuando al dividirlo entre dos, es una división exacta, es decir no deja residuo, de allí que usamos el operador MOD.

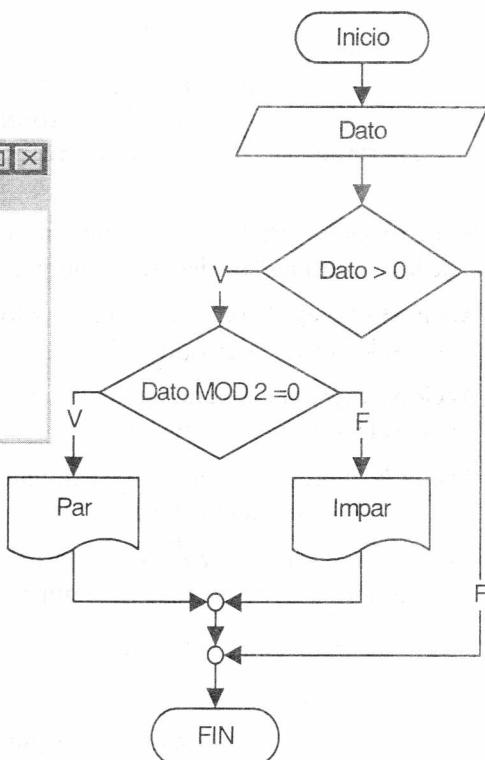
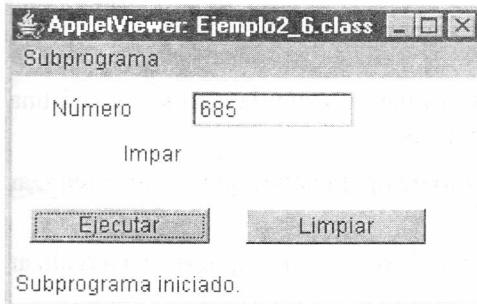
### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo2_6 extends Applet implements ActionListener{
    Label lbl1=new Label("Número");
    Label R=new Label();
    TextField Num=new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (300, 150);
        lbl1.setBounds(40,15,60,20);
        Num.setBounds(120,15,90,20);
        R.setBounds(80,40,100,20);
        Ejecutar.setBounds(30,80,100,20);
        Limpiar.setBounds(150,80,100,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
```

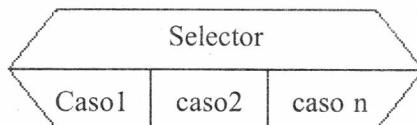
## Capítulo II: Estructuras lógicas selectivas

```
add(lbl1); add(R); add(Num);
add(Ejecutar); add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        double Dato;
        Dato=Double.parseDouble(Num.getText());
        if (Dato>0)
            if ((Dato%2)==0)
                R.setText("Par");
            else
                R.setText("Impar");
    }
    if(str.equals("Limpiar")){
        R.setText("");
        Num.setText("");
        Num.requestFocus();
    }
}
```



## La estructura de selección múltiple

La estructura de selección múltiple permite que el flujo del programa se bifurque por varias ramas en el punto de la toma de decisión(es), esto en función del valor que tome el selector. Así, si el selector toma el valor 1 se ejecutará la acción 1, si toma el valor 2 se ejecutará la acción 2, si toma el valor N se realizará la acción N, y si toma un valor distinto de los valores comprendidos entre 1 y N, se continuará con el flujo normal del diagrama realizándose la acción N+1.



A continuación presentamos el pseudocódigo que ilustra esta estructura selectiva.

**En caso de Selector**

Caso Valor1: Hacer Acción1

Caso Valor2: Hacer Acción2

.

.

.

Caso ValorN: Hacer AcciónN

En otro caso : AccionN+1

**Fin\_Caso** {Fin del condicional}

Donde:

**Selector** es la variable o expresión, a evaluarse, según la cual se tomará una de las "múltiples" decisiones o alternativas.

**Acción1** expresa la operación o conjunto de operaciones que se van a realizar si el selector toma el valor1.

**Acción2** expresa la operación o conjunto de operaciones que se van a realizar si el selector toma el valor 2.

**AcciónN** expresa la operación o conjunto de operaciones que se van a realizar si el selector toma el valor N.

La estructura selectiva **En caso de** es muy flexible, lo que permite aplicarla de diferentes formas. Obsérvense los siguientes aplicaciones.

### Ejemplo 2.7

Construya un pseudocódigo, que dado como datos dos variables de tipo entero (NUM, V), obtenga el resultado de la siguiente función:

100*V	Si NUM = 1
100^V	Si NUM = 2
100/V	Si NUM = 3
0	Para cualquier otro valor de NUM

## Solución

El algoritmo para este caso es simple:

Declarar variables

Leer NUM, V

Según sea el caso de NUM, realizar la acción correspondiente

## Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    NUM, V : Entero  
    VAL : Real
3. Leer NUM, V
4. En caso de NUM
  - 4.1. Caso 1 : Calcular VAL = 100\*V
  - 4.2. Caso 2 : Calcular VAL = 100^V
  - 4.3. Caso 3 : Calcular VAL = 100/V
  - 4.4. En otro caso: Calcular VAL = 0
5. Fin\_Caso
6. Escribir VAL
7. Terminar el proceso

## Codificación en Java

```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
  
public class Ejemplo2_7 extends Applet implements  
ActionListener{  
    Label lbl1=new Label("Num");  
    Label lbl2=new Label("V");  
    Label R=new Label();  
    TextField Numero=new TextField();  
    TextField VV=new TextField();  
    Button Ejecutar=new Button("Ejecutar");
```

```
Button Limpiar=new Button("Limpiar");
public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (300, 120);
    lbl1.setBounds(40,15,60,20);
    lbl2.setBounds(40,40,80,20);
    R.setBounds(40,65,200,20);
    Numero.setBounds(120,15,90,20);
    VV.setBounds(120,40,90,20);
    Ejecutar.setBounds(30,95,100,20);
    Limpiar.setBounds(150,95,100,20);
    Ejecutar.addActionListener(this);
    Limpiar.addActionListener(this);
    add(lbl1);add(lbl2);add(R);add(Numero);
    add(VV);add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        double V;
        int NUM;
        V=Double.parseDouble(VV.getText());
        NUM=Integer.parseInt(Numero.getText());
        switch(NUM){
            case 1 : V = 100*V;break;
            case 2 : V = Math.pow(100,V);break;
            case 3 : V = 100/V;break;
            default: V = 0;
        }
        R.setText("El resultado es: "+V);
    }
    if(str.equals("Limpiar")){
        VV.setText("");
        Numero.setText("");
        R.setText("");
        Numero.requestFocus();
    }
}
```

Switch es lo mismo que usar "En caso de" en el pseudocódigo. El uso de break en la codificación Java nos permite que una vez evaluada una condición se termine la secuencia del flujo Switch en caso contrario puede seguir evaluando otras condiciones y producir un resultado erróneo.

## Ejemplos desarrollados

A continuación presentamos una serie de problemas resueltos.

### Ejemplo 2.8

Dados los datos A, B y C, que representan números enteros diferentes, construya un pseudocódigo para escribir estos números en forma descendente.

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
A, B, C : Entero
3. Leer A, B, C
4. Sí A > B Entonces
  - 4.1. Si A > C Entonces
    - 4.1.1. Si B > C Entonces
      - 4.1.1.1. Escribir A, B, C
    - 4.1.2. Si no
      - 2.1.2.1. Escribir A, C, B
  - 4.1.3. Fin\_Si{Fin del condicional del paso 2.1.1}
  - 4.2. Si no
    - 4.2.1. Escribir C, A, B
  - 4.3. Fin\_Si {Fin del condicional del paso 2.1}
  5. Si no
    - 5.1. Si B > C Entonces
      - 5.1.1. Si A > C Entonces
        - 5.1.1.1. Escribir B, A y C
      - 5.1.2. Si no
        - 5.1.2.1. Escribir B, C y A
    - 5.1.3. Fin\_Si {Fin del condicional del paso 3.1.1}
    - 5.2. Si no
      - 5.2.1. Escribir C, B y A
    - 5.3. Fin\_Si {Fin del condicional del paso 3.1}
    6. {Fin del condicional del paso 3}
    7. Terminar el proceso

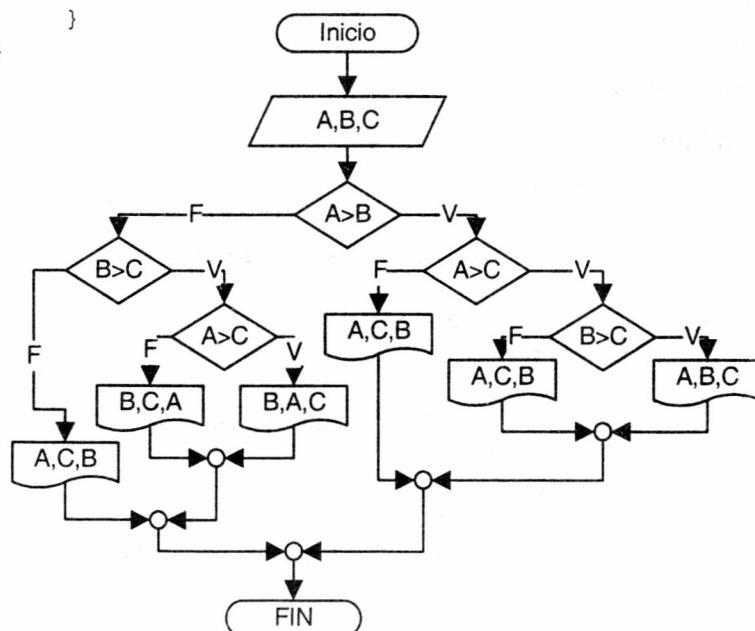
## Codificación en Java

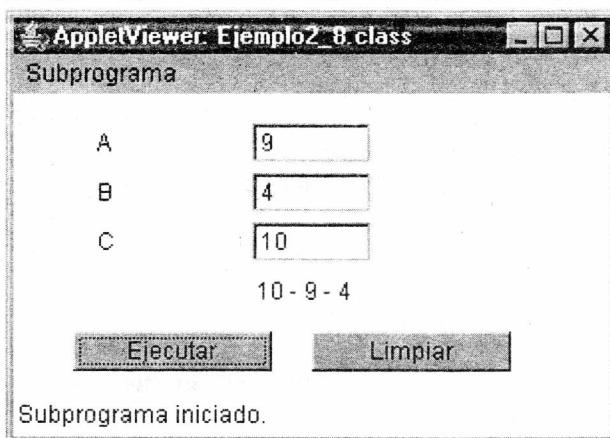
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo2_8 extends Applet implements
ActionListener{
    Label lbl1=new Label("A");
    Label lbl2=new Label("B");
    Label lbl3=new Label("C");
    Label R=new Label();
    TextField AA=new TextField();
    TextField BB=new TextField();
    TextField CC=new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (300, 150);
        lbl1.setBounds(40,15,60,20);
        lbl2.setBounds(40,40,60,20);
        lbl3.setBounds(40,65,60,20);
        R.setBounds(120,90,250,20);
        AA.setBounds(120,15,60,20);
        BB.setBounds(120,40,60,20);
        CC.setBounds(120,65,60,20);
        Ejecutar.setBounds(30,120,100,20);
        Limpiar.setBounds(150,120,100,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl1);add(lbl2);add(lbl3);add(R);add(AA);
        add(BB);add(CC);add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int A,B,C;
            A=Integer.parseInt(AA.getText());
            B=Integer.parseInt(BB.getText());
            C=Integer.parseInt(CC.getText());
        }
    }
}
```

```

        if (A > B) {
            if (A > C)
                if (B > C)
                    R.setText(A+" - "+B+" - "+C);
                else
                    R.setText(A+" - "+C+" - "+B);
            else
                R.setText(C+" - "+A+" - "+B);
        }else{
            if (B > C)
                if (A > C)
                    R.setText(B+" - "+A+" - "+C);
                else
                    R.setText(B+" - "+C+" - "+A);
            else
                R.setText(C+" - "+B+" - "+A);
        }
    }
    if(str.equals("Limpiar")){
        AA.setText("");BB.setText("");
        CC.setText("");r.setText("");
        AA.requestFocus();
    }
}
}

```





## Ejemplo 2.9

Escribir un programa que lea tres números enteros por teclado y muestre por pantalla el mayor de los tres.

### Solución

Declarar Variables

Leer números

Calcular el mayor

Reportar resultado

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
A, B, C, Mayor : Entero
3. Leer A, B, C
4. Si A > B Y A > C Entonces  
    4.1. Hacer Mayor = A
5. Si no  
    5.1. Si B > A Y B > C Entonces  
        5.1.1 Hacer Mayor = B
- 5.2. Si no  
        5.2.1. Si C > A Y C > B Entonces  
            5.1.1 Hacer Mayor = C
- 5.2.2. Fin\_Si
- 5.3. Fin\_Si

6. Fin\_Si
7. Escribir 'El número mayor es: ', Mayor
- 8.Terminar el proceso

### Codificación en Java

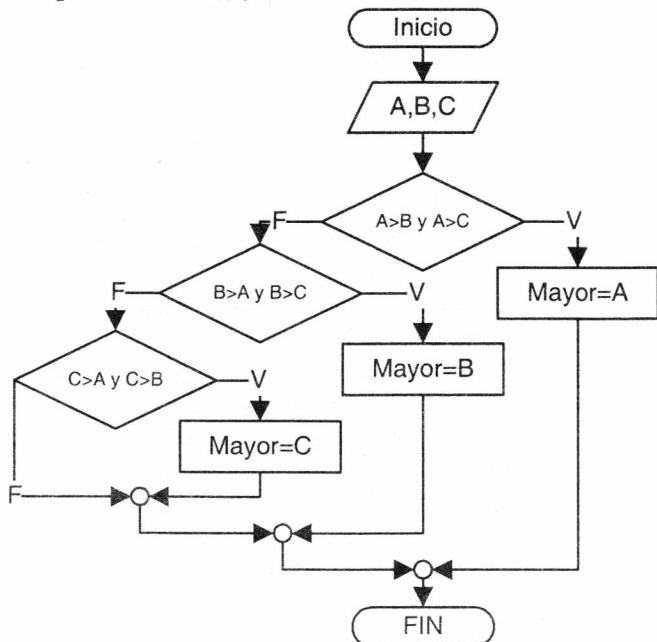
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

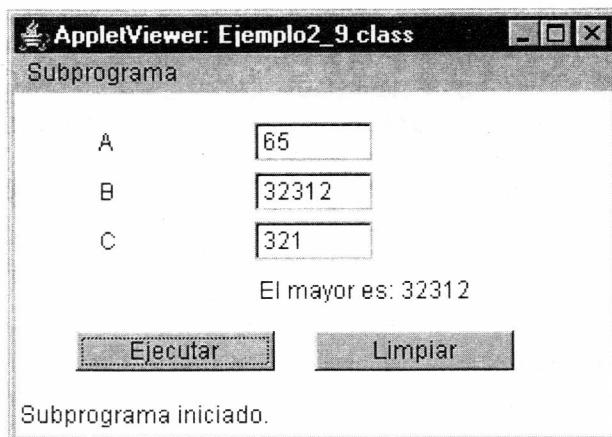
public class Ejemplo2_9 extends Applet implements
ActionListener{
Label lbl1=new Label("A");
Label lbl2=new Label("B");
Label lbl3=new Label("C");
Label R=new Label();
TextField AA=new TextField();
TextField BB=new TextField();
TextField CC=new TextField();
Button Ejecutar=new Button("Ejecutar");
Button Limpiar=new Button("Limpiar");
public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (300, 150);
    lbl1.setBounds(40,15,60,20);
    lbl2.setBounds(40,40,60,20);
    lbl3.setBounds(40,65,60,20);
    R.setBounds(120,90,250,20);
    AA.setBounds(120,15,60,20);
    BB.setBounds(120,40,60,20);
    CC.setBounds(120,65,60,20);
    Ejecutar.setBounds(30,120,100,20);
    Limpiar.setBounds(150,120,100,20);
    Ejecutar.addActionListener(this);
    Limpiar.addActionListener(this);
    add(lbl1);add(lbl2);add(lbl3);add(R);
    add(AA);add(BB);add(CC);
    add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac) {
    String str=ac.getActionCommand();
```

```

if(str.equals("Ejecutar")){
    int A,B,C,Mayor;
    Mayor=0;
    A=Integer.parseInt(AA.getText());
    B=Integer.parseInt(BB.getText());
    C=Integer.parseInt(CC.getText());
    if (A>B && A>C)
        Mayor = A;
    else
        if (B>A && B>C)
            Mayor = B;
        else
            if (C>A && C>B)
                Mayor = C;
    R.setText("El mayor es: "+Mayor);
}
if(str.equals("Limpiar")){
    AA.setText("");
    BB.setText("");
    CC.setText("");
    R.setText("");
    AA.requestFocus();
}
}
}

```





### Ejemplo 2.10

Construya un algoritmo que permita saber si un número entero de 4 cifras, es capicúa.

#### Solución

Un número capicúa es cuando al ser leido de izquierda a derecha y de derecha a izquierda, nos debe dar el mismo número. Por ahora usaremos el método de invertir un número para poder ver si al invertirlo es igual al original.

Declarar variables

Leer número

Invertir número

Comparar número invertido con el original

Reportar resultados

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    N , Aux, : Entero  
    U, D, C, UM : Entero
3. Leer N
4. Hacer Aux = N
5. Calcular     UM = Aux Div 1000  
                  Aux = Aux Mod 1000  
                  C = Aux Div 100

```
Aux = Aux Mod 100
D = Aux Div 10
U = Aux Mod 10
6. Calcular Aux = U*1000 + D*100 + C*10 + UM
7 Si N = Aux Entonces
    7.1. Escribir 'Número es capicúa'
8. Si no
    8.1. Escribir 'Número no es capicúa'
9. 4. Fin_Si
10. Terminar el proceso
```

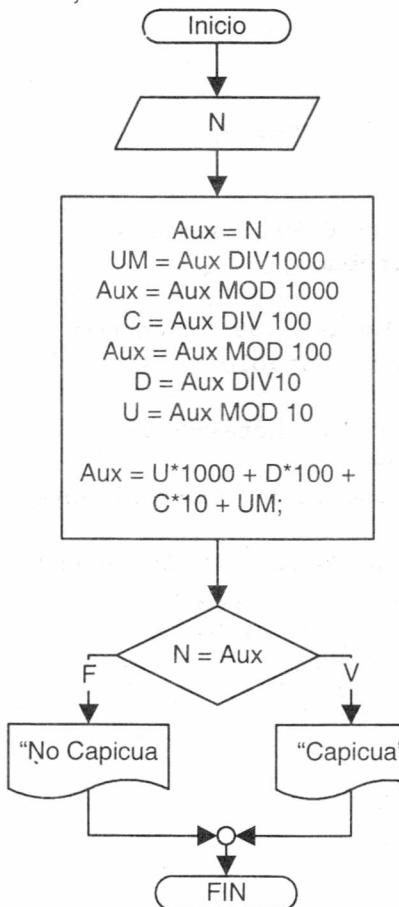
### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo2_10 extends Applet implements
ActionListener{
Label lbl1=new Label("Número");
Label R=new Label();
TextField Numero=new TextField();
Button Ejecutar=new Button("Ejecutar");
Button Limpiar=new Button("Limpiar");
public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (300, 100);
    lbl1.setBounds(40,15,60,20);
    R.setBounds(110;40,100,20);
    Numero.setBounds(110,15,90,20);
    Ejecutar.setBounds(30,70,100,20);
    Limpiar.setBounds(150,70,100,20);
    Ejecutar.addActionListener(this);
    Limpiar.addActionListener(this);
    add(lbl1);add(R);add(Numero);
    add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac) {
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        int N,Aux,UM,C,D,U;
        N=Integer.parseInt(Numero.getText());
```

```

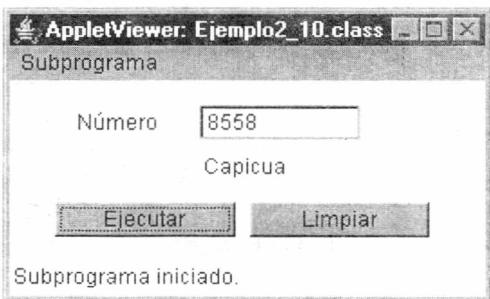
        Aux = N;UM = Aux/1000;  Aux = Aux%1000;
        C = Aux/100;Aux = Aux%100;D = Aux/10;
        U = Aux%10;Aux = U*1000+D*100+C*10+UM;
        if (N == Aux) R.setText("Capicua");
        else R.setText("No capicua");
    }
    if(str.equals("Limpiar")){
        Numero.setText("");
        R.setText("");
        Numero.requestFocus();
    }
}
}

```



U, D, C, UM son variables usadas para la descomposición del número

Nótese que la variable auxiliar (Aux) primero la usamos para realizar la descomposición del número y luego nos sirve para almacenar el número invertido, esta metodología es muy usada, consta en usar una misma variable para diferentes usos, de esa manera no debemos de declarar una variable específica para cada acción a realizar. Solo se deben de declarar las variables necesarias.



## Ejemplo 2.11

Escribir un programa que lea una nota de un examen por teclado y devuelva la calificación que tiene. La calificación podrá ser: Suspenso (0-4.99), Aprobado (5-6.99), Notable (7-8.99), Sobresaliente (9-9.99) o Matricula de Honor (10).

### Solución

Declarar Variables

Leer nota

Calcular calificación y reportar resultados

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    Nota : Real
3. Leer Nota
4. Si Nota >= 0 Y Nota <= 4.99 Entonces  
    4.1. Escribir 'Suspenso'
5. Si no  
    5.1. Si Nota >= 5 Y Nota <= 6.99 Entonces  
        5.1.1. Escribir 'Aprobado'  
    5.2. Si no  
        5.2.1. Si Nota>=7 Y Nota<=8.99 Entonces  
            5.2.1.1. Escribir 'Notable'  
        5.2.2. Si no  
            5.2.2.1. Si Nota>=9 Y Nota<=9.99 Entonces  
                5.2.2.1.1. Escribir 'Sobresaliente'  
            5.2.2.2. Si no  
                5.2.2.2.1. Si Nota = 10 Entonces  
                    5.2.2.2.1.1 Escribir  
                        'Matricula de honor'  
                5.2.2.2.2. Fin\_Si  
            5.2.2.3. Fin\_Si  
        5.2.3. Fin\_Si  
    5.3. Fin\_Si
6. Fin\_Si
7. Terminar el proceso

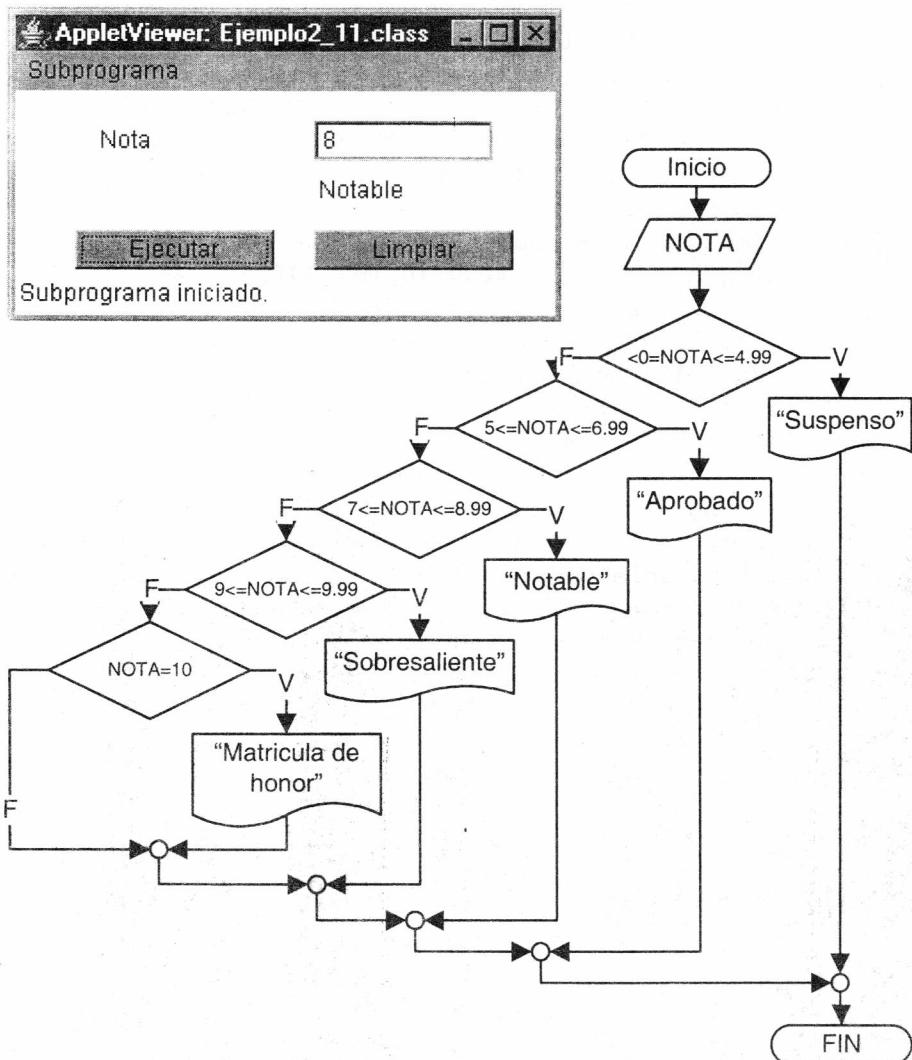
### Codificación en Java

```
import java.awt.*;
```

## **Capítulo II: Estructuras lógicas selectivas**

```
import java.applet.*;
import java.awt.event.*;
public class Ejemplo2_11 extends Applet implements
ActionListener{
    Label lbl1=new Label("Nota");
    Label R=new Label("");
    TextField N=new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (300, 90);
        lbl1.setBounds(40,15,60,20);
        R.setBounds(150,40,100,20);
        N.setBounds(150,15,90,20);
        Ejecutar.setBounds(30,70,100,20);
        Limpiar.setBounds(150,70,100,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl1);add(R);add(N);
        add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            double Nota;
            Nota=Double.parseDouble(N.getText());
            if (Nota>=0 && Nota<=4.99)
                R.setText("Suspensó");
            else
                if (Nota>=5 && Nota<=6.99)
                    R.setText("Aprobado");
                else
                    if (Nota>=7 && Nota<=8.99)
                        R.setText("Notable");
                    else
                        if (Nota>=9 && Nota<=9.99)
                            R.setText("Sobresaliente");
                        else
                            if (Nota==10)
```

```
        R.setText("Matricula de honor");
    }
    if(str.equals("Limpiar")){
        N.setText("");
        R.setText("");
        N.requestFocus();
    }
}
```



### Ejemplo 2.12

Dado como dato el tiempo de servicio de un trabajador, considere un aumento del 15% si la categoría del trabajador es A, un 12% en caso la categoría sea B, Si la categoría es C, un aumento del 10% y para la categoría D se aumentará \$15. Imprima el sueldo con el aumento incorporado, la categoría y el tiempo de servicio del trabajador.

La categoría esta dada por la siguiente tabla

Categoría	Años
A	20-30
B	De 15 a 20
C	De 10 a 15
D	de 0 a 10

### Solución

Nótese que para poder hallar el aumento correspondiente primero debemos de hallar la categoría. El algoritmo para este ejercicio es el siguiente:

Declarar Variables

Leer años de servicio y el sueldo

Obtener categoría de trabajador

Según sea la categoría, realizar el aumento

Mostrar resultado

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    TS, S : Real  
    Cate : Caracter
3. Leer TS, S
4. Si TS >= 0 Y TS <10 Entonces  
    4.1. Cate = 'D'
5. Si no  
    5.1. Si TS >= 10 Y TS <15 Entonces  
        5.1.1. Cate = 'C'  
    5.2. Si no  
        5.2.1. Si TS >= 15 Y TS <20 Entonces  
            5.2.1.1. Cate = 'B'  
        5.2.2. Si no

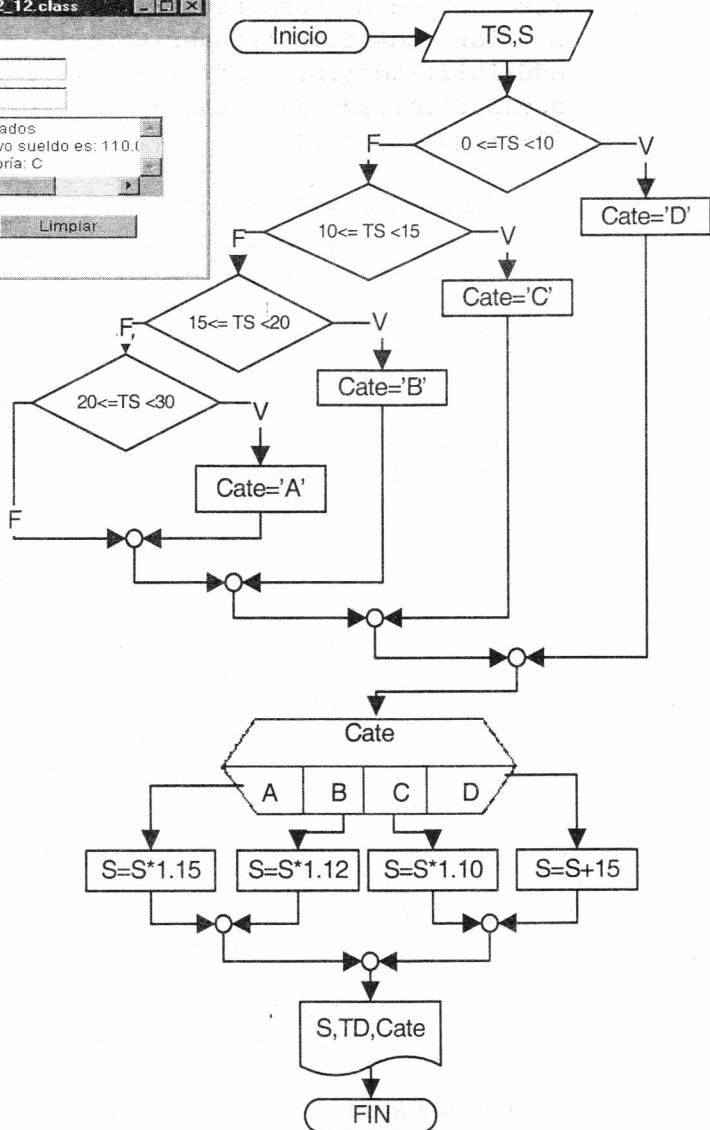
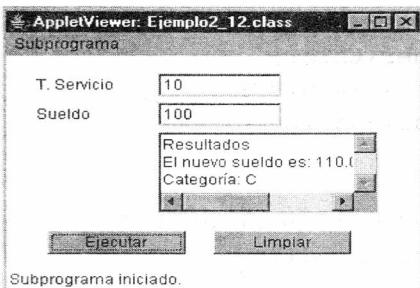
- 5.2.2.1. Si  $TS \geq 20$  Y  $TS \leq 30$  Entonces
  - 5.2.2.1.1. Cate = 'A'
  - 5.2.2.2. Fin\_Si
  - 5.2.3. Fin\_Si
- 5.3. Fin\_Si
6. Fin\_Si
7. En caso de Cate
  - 7.1. Caso 'A' :  $S = S * 1.15$
  - 7.2. Caso 'B' :  $S = S * 1.12$
  - 7.3. Caso 'C' :  $S = S * 1.10$
  - 7.4. Caso 'D' :  $S = S + 15$
8. Fin\_Caso
9. Escribir 'El nuevo sueldo es: ', S  
    'La categoría es: ', Cate  
    'Tiempo de servicio es: ', TS
10. Terminar el proceso

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo2_12 extends Applet implements
ActionListener{
Label lbl1=new Label("T. Servicio");
Label lbl2=new Label("Sueldo");
TextField TTS=new TextField();
TextField Sueldo=new TextField();
TextArea R=new TextArea("Resultados");
Button Ejecutar=new Button("Ejecutar");
Button Limpiar=new Button("Limpiar");
public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (300, 180);
    lbl1.setBounds(20,15,60,20);
    lbl2.setBounds(20,40,60,20);
    TTS.setBounds(110,15,90,20);
    Sueldo.setBounds(110,40,90,20);
    R.setBounds(110,65,160,70);
    Ejecutar.setBounds(30,150,100,20);
    Limpiar.setBounds(150,150,100,20);
```

## Capítulo II: Estructuras lógicas selectivas

```
Ejecutar.addActionListener(this);
Limpiar.addActionListener(this);
add(lbl1);add(lbl2);add(R);add(TTS);
add(Sueldo);add(Ejecutar);add(Limpiar);
TTS.requestFocus();
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        double S;
        int TS;
        char Cate=' ';
        TS=Integer.parseInt(TTS.getText());
        S=Double.parseDouble(Sueldo.getText());
        if (TS >= 0 && TS <10)
            Cate = 'D';
        else
            if (TS >= 10 && TS <15) Cate = 'C';
            else
                if (TS >= 15 && TS <20)
                    Cate = 'B';
                else
                    if (TS>=20 && TS<=30)
                        Cate = 'A';
        switch(Cate){
            case 'A' : S = S * 1.15;break;
            case 'B' : S = S * 1.12;break;
            case 'C' : S = S * 1.10;break;
            case 'D' : S = S + 15;
        }
        R.append("\nEl nuevo sueldo es: "+S);
        R.append("\nCategoría: "+Cate);
    }
    if(str.equals("Limpiar")){
        Sueldo.setText("");
        R.setText("Resultados");
        TTS.setText("");
        TTS.requestFocus();
    }
}
```



### Ejemplo 2.13

Diseñe un algoritmo que ingresada una fecha en formato DD/MM/AAAA, reporte la fecha como: "Es DD del mes MM del año AAAA". Debe suponerse que la fecha ingresada es válida.

### Solución

Se nos pide que ingresado la fecha 15/06/2006, nos de como resultado: Es 15 de Junio del año 2006. El único inconveniente es obtener el mes en letras, la cual es una tarea sencilla. El algoritmo es el siguiente:

Declarar variables

Leer fecha

Convertir mes a letras

Reportar fecha

### Pseudocódigo

1. Iniciar proceso

2. Declarar Variables

    DD, MM, AA : Entero

    Mes : Cadena de caracteres

3. Leer DD, MM, AA

4. En caso de MM

    4.1. Caso 1 : Mes = 'Enero'

    4.2. Caso 2 : Mes = 'Febrero'

    4.3. Caso 3 : Mes = 'Marzo'

    4.4. Caso 4 : Mes = 'Abril'

    4.5. Caso 5 : Mes = 'Mayo'

    4.6. Caso 6 : Mes = 'Junio'

    4.7. Caso 7 : Mes = 'Julio'

    4.8. Caso 8 : Mes = 'Agosto'

    4.9. Caso 9 : Mes = 'Setiembre'

    4.10. Caso 10 : Mes = 'Octubre'

    4.11. Caso 11 : Mes = 'Noviembre'

    4.12. Caso 12 : Mes = 'Diciembre'

5. Fin\_Caso

6. Escribir 'Es ', DD, 'de ', Mes, 'del año ', AA

7. Terminar el proceso

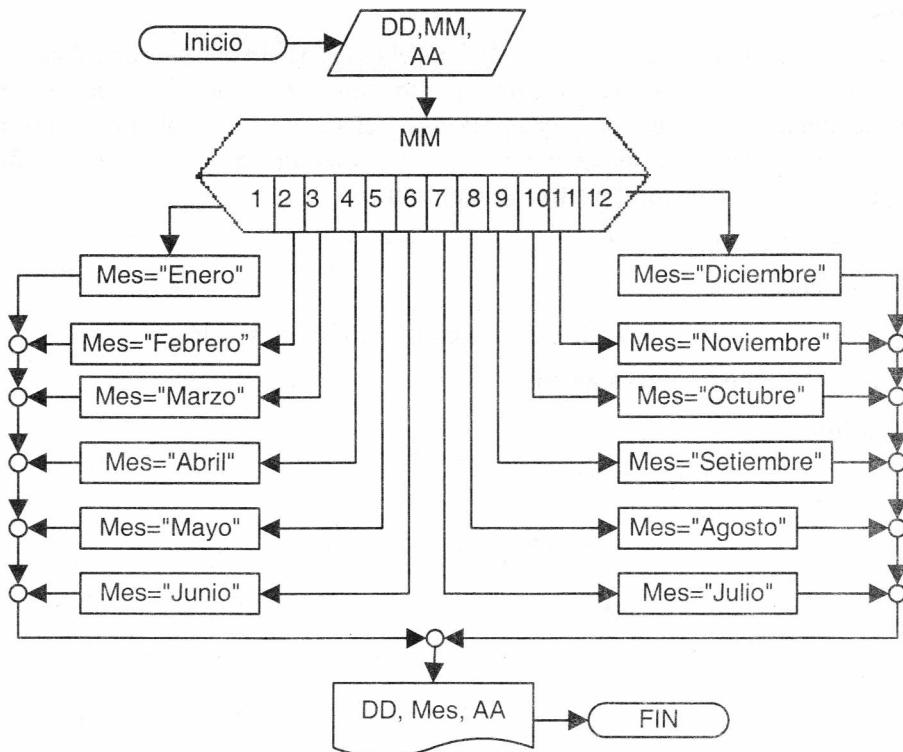
### Codificación en Java

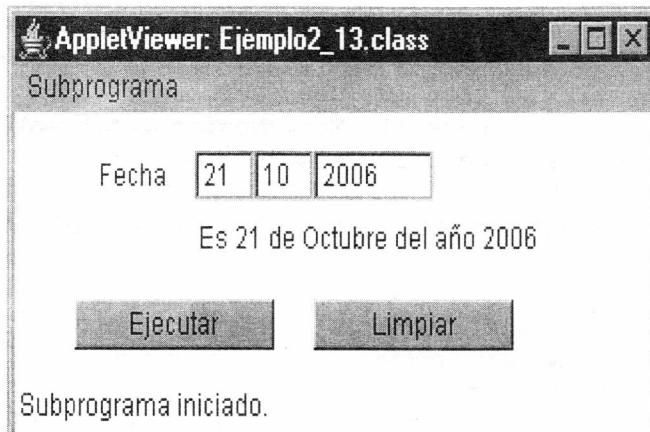
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
```

```
public class Ejemplo2_13 extends Applet implements
```

```
ActionListener{
    Label lbl1=new Label("Fecha");
    Label R=new Label("");
    TextField D=new TextField();
    TextField M=new TextField();
    TextField A=new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (320, 150);
        lbl1.setBounds(40,15,40,20);
        R.setBounds(90,40,250,20);
        D.setBounds(90,15,30,20);
        M.setBounds(120,15,30,20);
        A.setBounds(150,15,60,20);
        Ejecutar.setBounds(30,120,100,20);
        Limpiar.setBounds(150,120,100,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl1);add(R);add(D);add(M);
        add(A);add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int DD, MM, AA;
            String Mes="";
            DD=Integer.parseInt(D.getText());
            MM=Integer.parseInt(M.getText());
            AA=Integer.parseInt(A.getText());
            switch(MM){
                case 1 : Mes="Enero";break;
                case 2 : Mes="Febrero";break;
                case 3 : Mes="Marzo";break;
                case 4 : Mes="Abril";break;
                case 5 : Mes="Mayo";break;
                case 6 : Mes="Junio";break;
                case 7 : Mes="Julio";break;
```

```
        case 8 : Mes="Agosto";break;
        case 9 : Mes="Setiembre";break;
        case 10 :Mes="Octubre";break;
        case 11 :Mes="Noviembre";break;
        case 12 :Mes="Diciembre";break;
    }
    R.setText("Es "+DD+" de "+Mes+" del año "+AA);
}
if(str.equals("Limpiar")){
    D.setText("");
    M.setText("");
    A.setText("");
    R.setText("");
    D.requestFocus();
}
}
```





### Ejemplo 2.14

Diseñe un algoritmo que permita calcular el día siguiente de una fecha dada. Asumir que la fecha ingresada es válida.

#### Solución

Al calcular el día siguiente de una fecha, debemos de ver que sucede cuando el día ingresado es el último día del mes, por lo tanto el siguiente día deberá ser el primer día del mes siguiente, y que pasa si la fecha ingresada es el último día del último mes del año, el día siguiente sería el primer día del primer mes del año siguiente. Por lo tanto el algoritmo será el siguiente:

Declarar variables

Leer fecha

Obtener número máximo de días que contienen los meses del año.

Calcular y Reportar día siguiente

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
DD, MM, AA : Entero
3. Leer DD, MM, AA
4. En caso de MM
  - 4.1. Caso 1 : NDias = 31
  - 4.2. Caso 2 :
    - 4.2.1. Si (AA Mod 4 = 0 Y AA MOD 400 = 0) O

```
(AA Mod 400 =0) Entonces
    4.2.1.1. Hacer NDias = 29.
    4.2.2. Si no
        4.2.2.1. Hacer NDias = 28.
    4.3. Caso 3 : NDias = 31
    4.4. Caso 4 : NDias = 30
    4.5. Caso 5 : NDias = 31
    4.6. Caso 6 : NDias = 30
    4.7. Caso 7 : NDias = 31
    4.8. Caso 8 : NDias = 31
    4.9. Caso 9 : NDias = 30
    4.10. Caso 10 : NDias = 31
    4.11. Caso 11 : NDias = 30
    4.12. Caso 12 : NDias = 31
5. Fin_Caso
6. DD = DD + 1
7. Si DD > NDias Entonces
    7.1. Calcular DD = 1
        MM = MM + 1
    7.2. Si MM > 12 Entonces
        7.2.1. Calcular MM = 1
            AA = AA + 1
    7.3. Fin_Si
8. Fin_Si
9. Escribir 'Dia siguiente es: ', DD, MM, AA
10. Terminar el proceso
```

### Codificación en Java

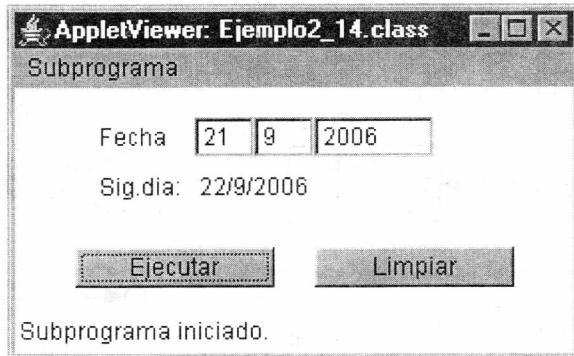
```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;

public class Ejemplo2_14 extends Applet implements
ActionListener{
    Label lbl1=new Label("Fecha");
    Label lbl2=new Label("Sig.dia:");
    Label R=new Label();
    TextField D=new TextField();
    TextField M=new TextField();
    TextField A=new TextField();
    Button Ejecutar=new Button("Ejecutar");
```

```
Button Limpiar=new Button("Limpiar");
public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (280, 110);

    lbl1.setBounds(40,15,40,20);
    lbl2.setBounds(40,40,50,20);
    D.setBounds(90,15,30,20);
    M.setBounds(120,15,30,20);
    A.setBounds(150,15,60,20);
    R.setBounds(90,40,250,20);
    Ejecutar.setBounds(30,80,100,20);
    Limpiar.setBounds(150,80,100,20);
    Ejecutar.addActionListener(this);
    Limpiar.addActionListener(this);
    add(lbl1);add(lbl2);add(R);add(D);
    add(M);add(A);add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        int DD, MM, AA, NDias=0;
        DD=Integer.parseInt(D.getText());
        MM=Integer.parseInt(M.getText());
        AA=Integer.parseInt(A.getText());
        switch(MM){
            case 1 : NDias = 31;
            case 2 : if ((AA%4==0 && AA%400==0) ||
                        (AA%400==0))
                NDias = 29;
            else
                NDias = 28;
            break;
            case 3 : NDias = 31;break;
            case 4 : NDias = 30;break;
            case 5 : NDias = 31;break;
            case 6 : NDias = 30;break;
            case 7 : NDias = 31;break;
            case 8 : NDias = 31;break;
            case 9 : NDias = 30;break;
```

```
case 10 : NDias = 31;break;
case 11 : NDias = 30;break;
case 12 : NDias = 31;break;
}
DD++;
if (DD > NDias) {
    DD = 1; MM++;
    if (MM > 12) {
        MM = 1; AA++;
    }
}
R.setText (DD+ "/" +MM+ "/" +AA);
}
if(str.equals("Limpiar")){
    D.setText("");
    M.setText("");
    A.setText("");
    R.setText("");
    D.requestFocus();
}
}
}
```



## Ejercicio propuestos

### Ejercicio 1

Construya un algoritmo, que dado como datos la categoría y el sueldo de un trabajador, calcule el aumento correspondiente teniendo en cuenta la siguiente

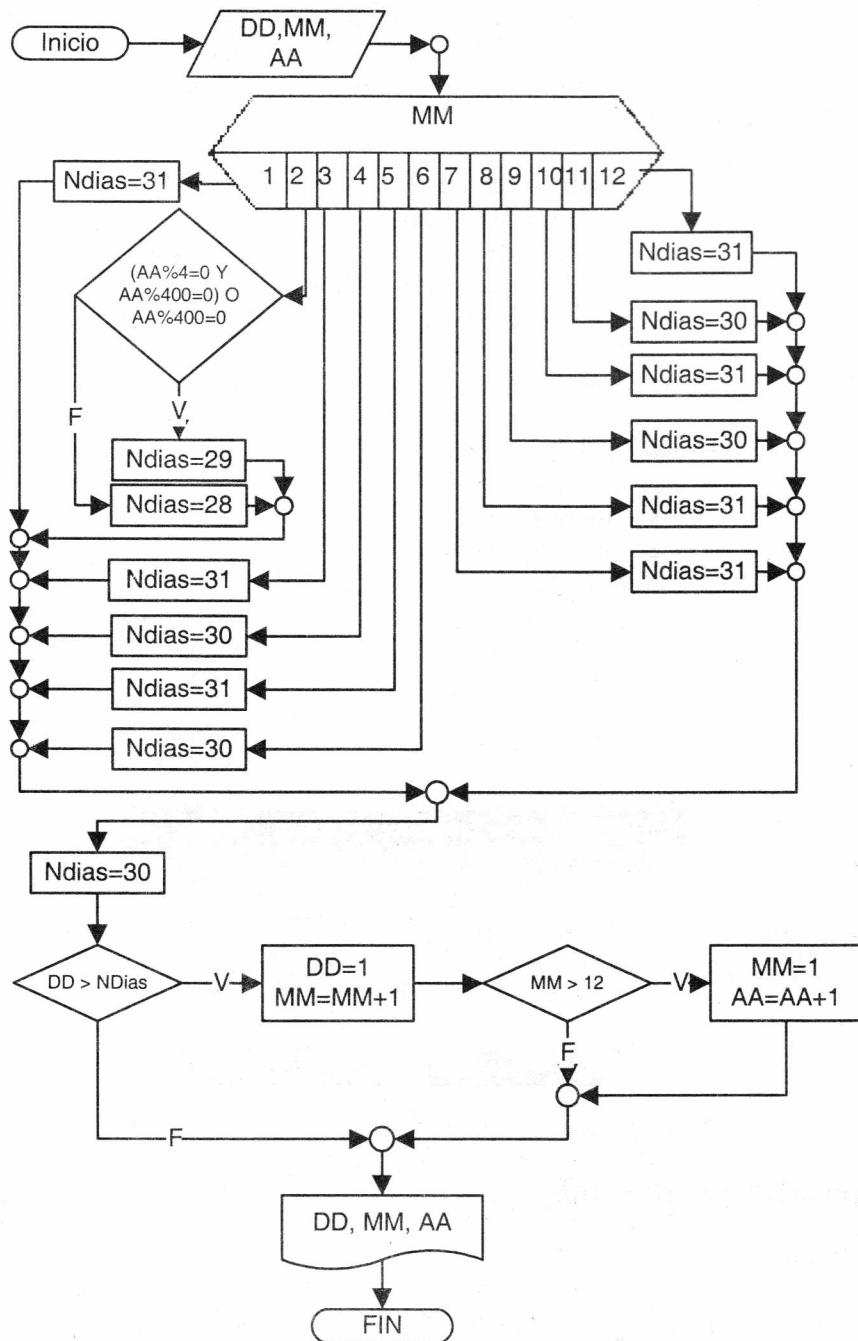


tabla. Imprima la categoría del trabajador y su nuevo sueldo.

Categoría	Aumento
1	15%
2	10%
3	8%
4	5%

### **Ejercicio 2**

El número de sonidos emitidos por un grillo en un minuto, es una función de la temperatura. Como resultado de esto, es posible determinar el nivel de la temperatura haciendo uso de un grillo como termómetro.

La fórmula para la función es:  $T = N / 4 + 40$

Donde: T representa la temperatura en grados Fahrenheit y N el número de sonidos emitidos por minuto.

Construya un pseudocódigo que le permita calcular la temperatura, teniendo en cuenta el número de sonidos emitidos por el grillo.

### **Ejercicio 3**

Construya un pseudocódigo, que dado como datos los valores enteros P y Q, determine si los mismos satisfacen la siguiente expresión:

$$P^3 + Q^4 - 2*P^2 < 680$$

En caso afirmativo debe imprimir los valores P y Q.

Donde P y Q son variables de tipo entero que expresan los datos que se ingresan.

### **Ejercicio 4**

Escribir un programa que lea dos números enteros por teclado y determine cuál es el mayor y cual es el menor. También deberá considerar el caso en el que los dos números sean iguales.

### **Ejercicio 5**

Escribir un programa que lea tres números enteros por teclado y emita un mensaje indicando si están o no ordenados crecientemente.

### **Ejercicio 6**

Escribir un programa que permita introducir por teclado tres letras y responda si existen al menos dos letras iguales.

### **Ejercicio 7**

Construya un pseudocódigo tal, que dados como datos la matrícula y 5 calificaciones de un alumno; imprima la matrícula, el promedio y la palabra "Aprobado" si el alumno tiene un promedio mayor o igual que 10.5, y la palabra "No aprobado" en caso contrario.

El promedio se calculará en base a las 4 notas mayores.

### **Ejercicio 8**

Diseñe un algoritmo que permita analizar la validez de una fecha, considere que el año válido sea mayor a 1800.

### **Ejercicio 9**

Diseñe un algoritmo que dada una fecha, se calcule el número de días que han transcurrido desde el inicio del año. Considere que la fecha ingresada es válida.

# Capítulo III

## Estructuras Lógicas Repetitivas

Este capítulo contiene:

Estructura Hacer Mientras

Estructura Mientras

Estructura Desde

Ejemplos desarrollados

Ejemplos propuestos



**Algoritmos y Diagramas de Flujo aplicados en Java 2**



## **Introducción**

Es muy común encontrar en los algoritmos operaciones que se deben ejecutar un número repetido de veces. Si bien las instrucciones son las mismas, los datos sobre los que se opera varían. El conjunto de instrucciones que se ejecuta repetidamente se llama ciclo.

Todo ciclo debe terminar de ejecutar luego de un número finito de veces, por lo que es necesario en cada iteración del mismo, evaluar las condiciones necesarias para decidir si debe seguir ejecutándose o debe detenerse. En todo ciclo, siempre debe existir una condición de parada o fin de ciclo.

## **Estructura *Hacer Mientras***

La estructura *Hacer Mientras*, es la estructura algorítmica adecuada para utilizar en un ciclo que se ejecutará un número definido de veces. Por ejemplo cuando necesitamos calcular la nómina total de la empresa, tenemos que sumar los sueldos de los N empleados de la misma. Cuando necesitamos obtener el promedio de calificaciones de un curso, debemos sumar las N calificaciones de los alumnos y dividir esa suma entre N. Es decir, sabemos de antemano cuántas veces tenemos que repetir una determinada operación, acción o tarea. El número de repeticiones no depende de las proposiciones dentro del ciclo.

La sintaxis para esta estructura es la siguiente:

```
Hacer
    {Proceso}
    Mientras (Condición)
    {Fin del ciclo}
```

Proceso, es cualquier operación o conjunto de operaciones que deban ejecutarse mientras se repita el ciclo. El ciclo se repetirá siempre que la Condición sea verdadera, es decir se cumplirá hasta que la Condición sea falsa.

Esta estructura garantiza por lo menos una iteración del bloque de proceso, debido a que primero realiza el proceso y al final evalúa la condición.

Muchas veces usaremos esta estructura con el uso de un contador, y la condición estará ligada a dicho contador, obteniendo así la siguiente sintaxis:

```
V = Vi  
Hacer  
    {proceso}  
    Calcular V = V + INC  
    Mientras que (Condición)  
    {Fin del ciclo}
```

**Donde**      *V*      :      es una variable de control  
                  *Vi*      :      es el valor inicial.  
                  *INC*     :      es el incremento

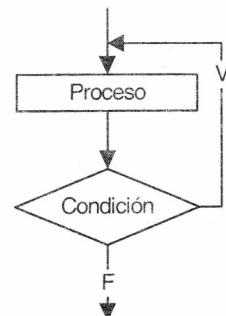
*V* (contador del ciclo, generalmente representado por las letras I, J, K, V) toma un valor inicial (*Vi*) y se compara con el valor esperado en la condición. El ciclo se ejecuta hasta que la condición sea verdadera. El valor de *V* se incrementa en cada iteración.

El incremento de *INC* por lo general es en un unidad, pero eso depende mucho de la naturaleza del problemas, veremos casos en que el incremento será de dos unidades o de tres unidades, etc , en general podremos tener un incremento en *X* unidades, donde *X* es un número entero.

Las variables de control son enteras o son reales o cualquier otro tipo, pero no pueden ser una mezcla de ambas.

Podemos tener tanto incrementos o decrementos, eso dependiendo de la naturaleza del programa.

Veamos a continuación algunos ejemplo de esta estructura.



### Ejemplo 3.1

Construya un algoritmo el cual le permita obtener la tabla de multiplicar de un número, se deberá indicar cuantos términos desea mostrar en el resultado.

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    Num, Can, I : Entero  
    Nomina, SUE : Real
3. Hacer I = 0
4. Hacer
  - 4.1. Escribir (I\*Num)
  - 4.2. I = I + 1
5. Mientras I < Can {Fin del ciclo del paso 4}
6. Terminar el proceso

#### Codificación en Java

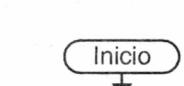
```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
public class Ejemplo3_1 extends Applet implements  
ActionListener{  
    Label lbl1=new Label("Número");  
    Label lbl2=new Label("Cantidad");  
    Label lbl3=new Label("Resultado:");  
    TextField N=new TextField();  
    TextField C=new TextField();  
    TextArea R=new TextArea("");  
    Button Ejecutar=new Button("Ejecutar");  
    Button Limpiar=new Button("Limpiar");  
    public void init() {  
        setLayout(null);  
        setBackground(java.awt.Color.white);  
        resize (400, 250);  
        lbl1.setBounds(40,15,80,20);  
        lbl2.setBounds(40,40,80,20);  
        lbl3.setBounds(40,65,80,20);  
        N.setBounds(150,15,60,20);  
        C.setBounds(150,40,60,20);  
        R.setBounds(150,60,150,150);  
        Ejecutar.setBounds(60,220,80,20);
```

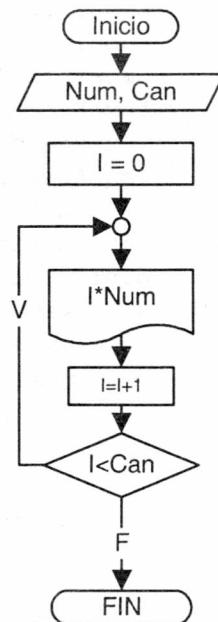
```

Limpiar.setBounds(200,220,80,20);
Ejecutar.addActionListener(this);
Limpiar.addActionListener(this);
add(lbl1);add(lbl2);add(lbl3);add(N);
add(R);add(C);add(Ejecutar);add(Limpiar);
}

public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        int Num,Can,I = 0;
        Num=Integer.parseInt(N.getText());
        Can=Integer.parseInt(C.getText());
        do{
            R.append((I*Num)+"\n");
            I++;
        }while(I<Can);
        Limpiar.requestFocus();
    }
    if(str.equals("Limpiar")){
        R.setText("");
        N.setText("");
        C.setText("");
        N.requestFocus();
    }
}
}

```





## Ejemplo 3.2

Encuentre la cantidad de número pares e impares y muestre estos número de un grupo de 20 número autogenerados

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    x,I,CP,CI : Entero
- 3 Hacer I = 0  
    CP = 0  
    CI = 0
4. Hacer
  - 4.1. Autogenerar x
  - 4.2. Si (Impar(x)) Entonces
    - 4.2.1. Escribir x
    - 4.2.2. CP=CP+1
  - 4.3. Si no
    - 4.3.1. Escribir x
    - 4.3.2. CI=CI+1
  - 4.4. Fin\_Si
  - 4.5. I=I+1
5. Mientras I < 20 {Fin del ciclo}
6. Escribir CI,CP
7. Terminar proceso

Función Impar(x)

1. Iniciar Función
2. Si (x MOD 2!=0)
  - 2.1. Retornar Verdadero
3. Si no
  - 3.1. Retornar Falso
4. Fin Función

Creamos la función impar() el cual recibe por valor el número autogenerado y comprueba si es impar o no y devuelve un valor de tipo booleano. Para mas detalles sobre funciones consulte el apéndice A.

### Código en Java

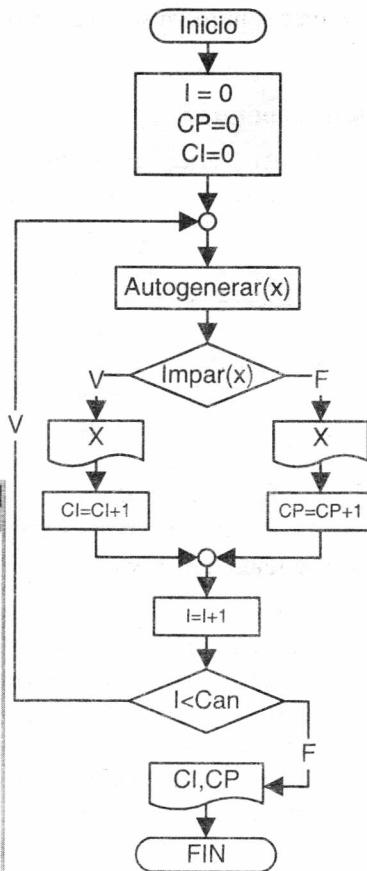
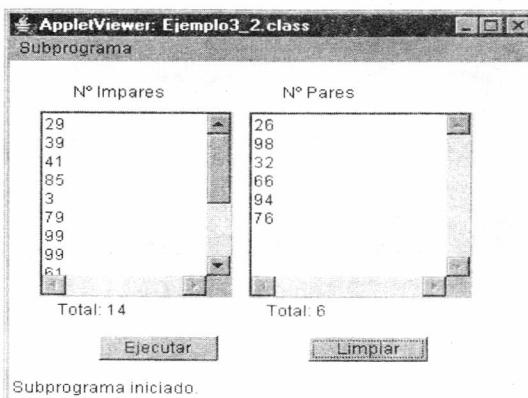
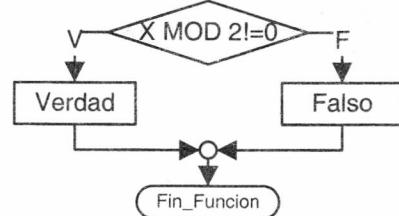
```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;
```

```
public class Ejemplo3_2 extends Applet implements  
ActionListener{  
    Label lbl1=new Label("Nº Impares");  
    Label lbl2=new Label("Nº Pares");  
    Label TP=new Label();  
    Label TI=new Label();  
    TextArea Par=new TextArea("");  
    TextArea Impar=new TextArea("");  
    Button Ejecutar=new Button("Ejecutar");  
    Button Limpiar=new Button("Limpiar");  
    public void init() {  
        setLayout(null);  
        setBackground(java.awt.Color.white);  
        resize (350, 250);  
        lbl1.setBounds(40,15,80,20);  
        lbl2.setBounds(180,15,80,20);  
        TI.setBounds(30,190,80,20);  
        TP.setBounds(170,190,80,20);  
        Impar.setBounds(20,40,130,150);  
        Par.setBounds(160,40,150,150);  
        Ejecutar.setBounds(60,220,80,20);  
        Limpiar.setBounds(200,220,80,20);  
        Ejecutar.addActionListener(this);  
        Limpiar.addActionListener(this);  
        add(lbl1);add(lbl2);add(Par);add(Impar);  
        add(Ejecutar);add(Limpiar);add(TI);add(TP);  
    }  
    public void actionPerformed(ActionEvent ac){  
        String str=ac.getActionCommand();  
        if(str.equals("Ejecutar")){  
            int x,I = 0,CP=0,CI=0;  
            do{  
                x=(int)(100*Math.random())+1;  
                if(Impar(x)){  
                    Impar.append(x+"\n");  
                    CI++;  
                }else{  
                    Par.append(x+"\n");  
                    CP++;  
                }  
                I++;  
            }  
        }  
    }  
}
```

```

        }while(I<20);
        TI.setText("Total: "+CI);
        TP.setText("Total: "+CP);
        Limpiar.requestFocus();
    }
    if(str.equals("Limpiar")){
        Impar.setText("");
        Par.setText("");
        TI.setText("");
        TP.setText("");
        Ejecutar.requestFocus();
    }
}
boolean Impar(int x){
    if(x%2!=0) return true;
    else return false;
}

```



Como observamos en el diagrama de flujo, al hacer un llamado a la función Impar(x) le entregamos el control del flujo a la función, la cual luego de finalizar su trabajo devuelve el control al diagrama de flujo principal o al programa principal.

### Ejemplo 3.3

Calcular la suma siguiente:

$100 + 98 + 96 + 94 + \dots + 0$  en este orden

#### Algoritmo

Declarar variables y constantes

Inicializar variables

Hacer

    Calcular suma y decrementar el número en dos unidades

    Mientras número sea mayor que cero.

    Mostrar suma

#### Pseudocódigo

1. Iniciar proceso

2. Declarar Variables

    N : Entero

    S : doble

3 Hacer S = 0

    N = 100

4. Hacer

    4.1. S = S + N

        N = N - 2

5. Mientras N > 0 {Fin del ciclo}

6. Escribir S

7. Terminar proceso

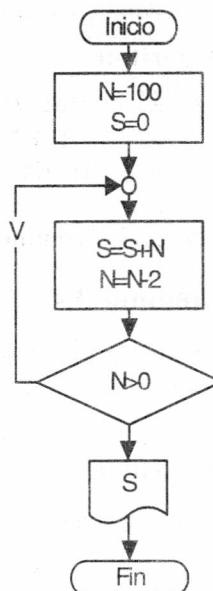
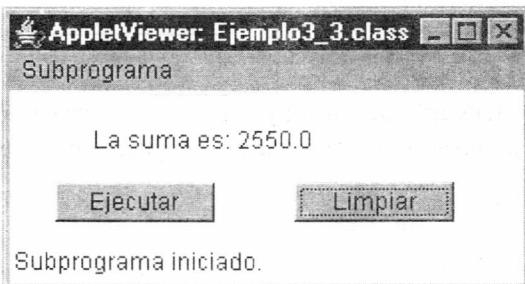
#### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo3_3 extends Applet implements
ActionListener{
    Label lbl1=new Label();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
```

```

public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (300, 100);
    lbl1.setBounds(40,15,150,20);
    Ejecutar.setBounds(40,70,80,20);
    Limpiar.setBounds(160,70,80,20);
    Ejecutar.addActionListener(this);
    Limpiar.addActionListener(this);
    add(lbl1);add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        double S=0;
        int N=100;
        do{
            S=S+N;
            N=N-2;
        }while(N>0);
        lbl1.setText("La suma es: "+S);
        Limpiar.requestFocus();
    }
    if(str.equals("Limpiar")){
        lbl1.setText("");
        Ejecutar.requestFocus();
    }
}

```



## Estructura Mientras

La estructura algorítmica *mientras* es la estructura adecuada para utilizar en un ciclo cuando no sabemos el número de veces que éste se ha de repetir. Dicho número depende de las proposiciones dentro del ciclo. Ejemplos en la vida cotidiana encontramos muchos. Por ejemplo, supongamos que tenemos que obtener el total de una serie de gastos, pero no sabemos exactamente cuántos son; o cuando tenemos que sacar el promedio de calificaciones de un examen, pero no sabemos precisamente cuántos alumnos lo aplicaron. Tenemos que sumar las calificaciones e ir contando el número de alumnos, ésto con el fin de poder obtener posteriormente el promedio. El ciclo se repite mientras tengamos calificaciones de alumnos.

En la estructura *mientras* se distinguen dos partes:

- ❖ *Ciclo*: Conjunto de instrucciones que se ejecutarán repetidamente.
- ❖ *Condición de terminación*: La evaluación de esta condición permite decidir cuando finalizará la ejecución del ciclo. La condición se evalúa al inicio del mismo.

Esta estructura se ejecuta mientras que la condición es verdadera, en caso contrario terminará el ciclo.

Debe existir también un ansiado dentro del ciclo que afecte la condición, para evitar que el ciclo se ejecute indefinidamente.

En lenguaje algorítmico de la estructura *mientras* la expresamos de esta forma:

### Algoritmo

```
Mientras Condición
    { PROCESO }
    Fin_Mientras
```

Veamos a continuación el siguiente ejemplo.

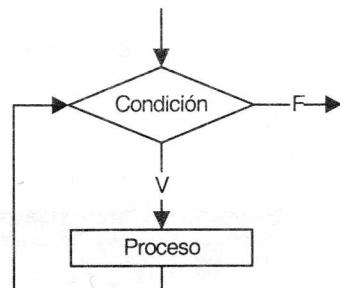
### Ejemplo 3.4

Obtenga el seno, coseno y tangente de un rango de ángulos expresados en grados, dichos ángulos serán ingresados por el usuario. Considere un aumento de 10°.

### Algoritmo

Leer el ángulo de inicio y final

Obtener el seno, coseno y tangente de cada ángulo e ir mostrando los resultados.



### Pseudocódigo

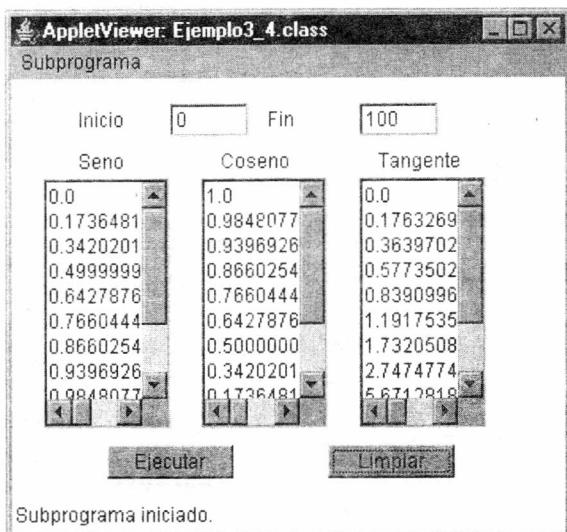
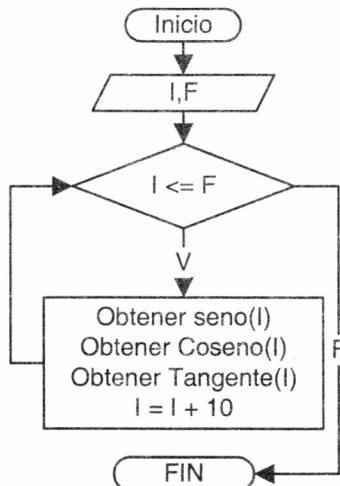
1. Iniciar proceso
2. Declarar Variables  
    I,F : Entero
3. Leer I,F
4. Mientras  $I \leq F$ 
  - 4.1. Obtener Seno(I), Coseno(I) y tangente(I)
  - 4.2.  $I = I + 10$
  - 4.3. Escribir Seno, coseno, tangente
5. Fin mientras {Fin del ciclo}
6. Terminar el proceso

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo3_4 extends Applet implements
ActionListener{
    Label lbl1=new Label("Seno");
    Label lbl2=new Label("Coseno");
    Label lbl3=new Label("Tangente");
    Label lbl4=new Label("Inicio");
    Label lbl5=new Label("Fin");
    TextField A1=new TextField();
    TextField A2=new TextField();
    TextArea Sen=new TextArea();
    TextArea Cos=new TextArea();
    TextArea Tan=new TextArea();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (350, 250);
        lbl1.setBounds(40,40,80,20);
        lbl2.setBounds(130,40,80,20);
        lbl3.setBounds(230,40,80,20);
        lbl4.setBounds(40,15,50,20);
        lbl5.setBounds(160,15,40,20);
        A1.setBounds(100,15,50,20);
        A2.setBounds(220,15,50,20);
```

```
Sen.setBounds(20,60,80,150);
Cos.setBounds(120,60,80,150);
Tan.setBounds(220,60,80,150);
Ejecutar.setBounds(60,220,80,20);
Limpiar.setBounds(200,220,80,20);
Ejecutar.addActionListener(this);
Limpiar.addActionListener(this);
add(lbl1);add(lbl2);add(lbl3);add(lbl4);
add(lbl5);add(Sen);add(Cos);add(Tan);
add(A1);add(A2);add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        int I,F;
        double ang;
        I=Integer.parseInt(A1.getText());
        F=Integer.parseInt(A2.getText());
        while(I<=F){
            ang=I*Math.PI/180;
            Sen.append(Math.sin(ang)+"\n");
            Cos.append(Math.cos(ang)+"\n");
            Tan.append(Math.tan(ang)+"\n");
            I+=10;
        }
        Limpiar.requestFocus();
    }
    if(str.equals("Limpiar")){
        A1.setText("");
        A2.setText("");
        Sen.setText("");
        Cos.setText("");
        Tan.setText("");
        Ejecutar.requestFocus();
    }
}
```

En java tenemos funciones matemáticas que nos permiten obtener cálculos de funciones trigonométricas, pero como dato de entrada requieren un ángulo expresado en radianes, por ese motivo debemos hacer la conversión de grados a radianes para poder obtener el cálculo correcto.



### Ejemplo 3.5

Realice un programa que permita obtener una estructura similar a la siguiente

```

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

```

El número de asteriscos será ingresado por el usuario, tenga en cuenta que si el número ingresado es par, se debe empezar desde el número impar superior al ingresado.

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    int N, aux, x, I;
3. Leer N
4. N = Par(N)
5. Hacer I=0
6. Mientras N>=1
  - 6.1. AumentarBlancos(I)
  - 6.2. I=I+1
  - x=N
  - 6.3. Mientras x>=1

6.3.1. Escribir "\*"  
6.3.2.  $X = X - 1$   
6.4. Fin Mientras  
6.5.  $N = N - 2$   
7. Fin Mientras  
8. Terminar el proceso

Procedimiento AumentarBlancos

1. Iniciar Procedimiento  
2. Leer I  
3. *Mientras*  $I > 0$   
    3.1. Escribir " "  
    3.2.  $I=I-1$   
4. Fin Mientras  
5. Terminar Procedimiento

Función Par

1. Iniciar Función  
2. Leer N  
3. Si  $(N \text{ MOD } 2) = 0$   
    3.1. Retornar  $N+1$   
4. Si no  
    4.1. Retornar N  
5. Fin Si  
6. Terminar Función

### Codificación en Java

```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
public class Ejemplo3_5 extends Applet implements  
ActionListener{  
    Label lbl1=new Label("Repeticiones");  
    TextField NUM=new TextField();  
    TextArea R=new TextArea("");  
    Button Ejecutar=new Button("Ejecutar");  
    Button Limpiar=new Button("Limpiar");  
    public void init() {  
        setLayout(null);  
        setBackground(java.awt.Color.white);  
        resize (350, 250);
```

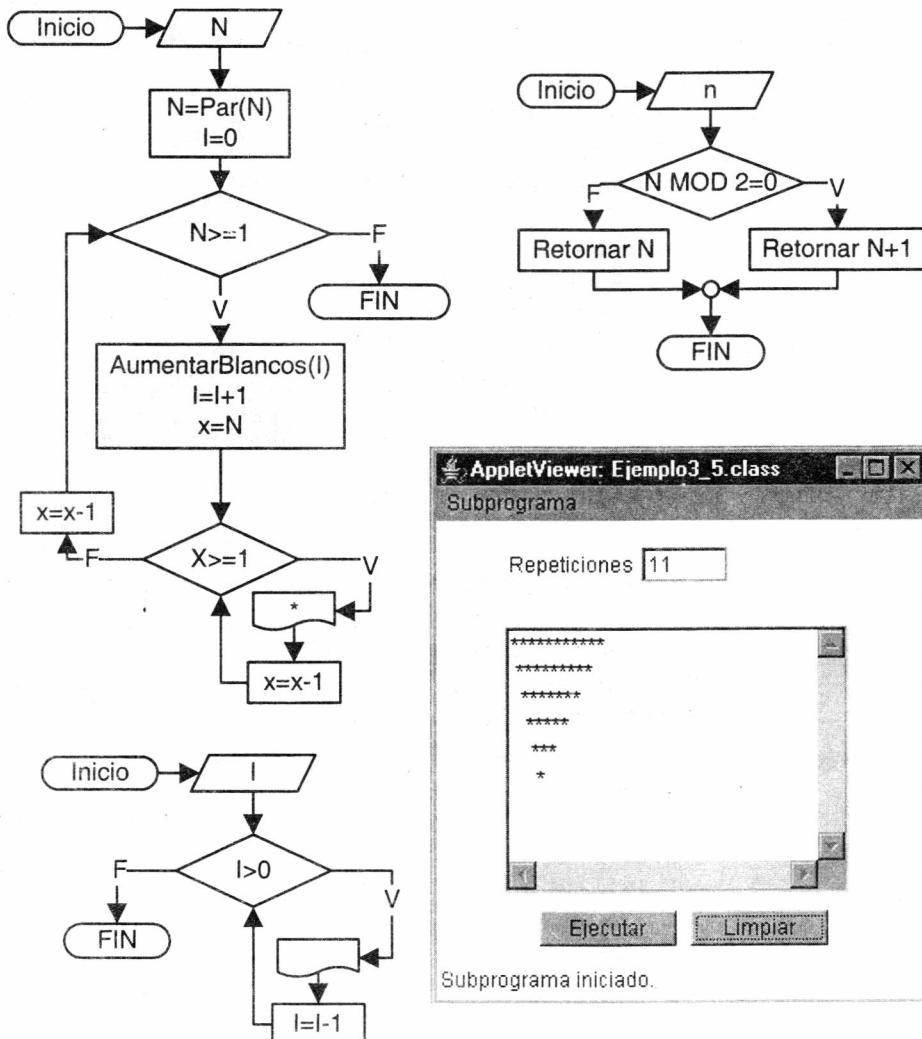
### Capítulo III: Estructuras lógicas repetitivas

```
lbl1.setBounds(40,15,80,20);
NUM.setBounds(120,15,50,20);
R.setBounds(40,60,200,150);
Ejecutar.setBounds(60,220,80,20);
Limpiar.setBounds(200,220,80,20);
Ejecutar.addActionListener(this);
Limpiar.addActionListener(this);
add(lbl1);add(NUM);add(R);
add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        int N,aux,x,I=0;
        N=Integer.parseInt(NUM.getText());
        N=Par(N);
        while(N>=1){
            AumentarBlancos(I++);
            x=N;
            while(x>=1){
                R.append("*");
                x--;
            }
            R.append("\n");
            N-=2;
        }
        Limpiar.requestFocus();
    }
    if(str.equals("Limpiar")){
        NUM.setText("");
        R.setText("");
        NUM.requestFocus();
    }
}
void AumentarBlancos(int I){
    while(I>0){
        R.append(" ");
        I--;
    }
}
```

```

        int Par(int n){
            if (n%2==0) return (n+1);
            else return n;
        }
    }
}
    
```

El procedimiento AumentarBlancos(I) nos permite insertar espacios en blanco antes de empezar a colocar los asteriscos para darle la forma adecuada y obtener una pirámide invertida, por otro lado la función Par(N) comprueba si el número de asteriscos ingresado es par, si es verdad aumenta en uno.



### Ejemplo 3.6

Se tienen las calificaciones de un grupo de alumnos que presentaron un examen. El profesor desea obtener el promedio de estas calificaciones. Escriba un pseudocódigo para resolver lo planteado anteriormente.

### Solución

El único problema de este ejemplo sería ver cuando debe terminar bucle Mientras, podemos usar un pequeño artificio para resolver esto, que al ingresar un número negativo termina el bucle, esto es posible ya que no hay notas negativas.

## Algoritmo

## Leer nota

Mientras Nota sea válida

## Sumar notas

## Leer nueva nota

Calcular promedio

## Reportar Promedio

## Pseudocódigo

1. Iniciar proceso
  2. Declarar Variables I : Enteros  
Nota, SUMCAL : Real
  3. Hacer I = 0  
SUMCAL = 0
  4. Leer Nota
  5. Mientras (Nota >= 0)
    - 5.1. Calcular SUMCAL = SUMCAL + Nota  
I = I + 1
    - 5.2. Leer Nota
  6. Fin\_Mientras
  7. Escribir SUMCAL/I
  8. Terminar proceso

SUMCAL acumula las calificaciones.

**Nota** expresa la calificación del alumno i

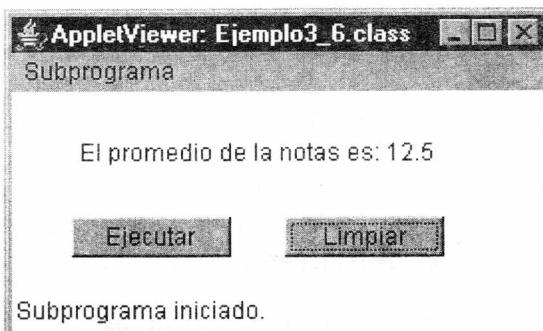
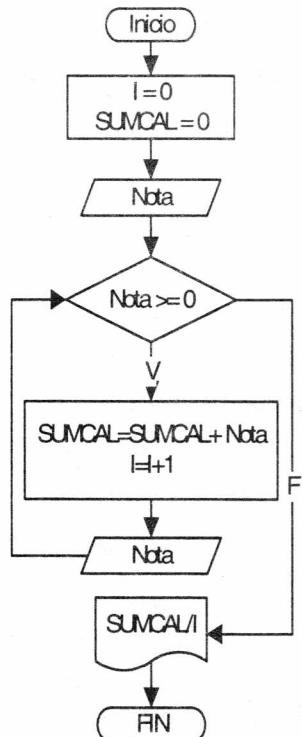
**PRO** almacena el promedio de las calificaciones.

## Codificación en Java

```
import java.awt.*;
```

```
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo3_6 extends Applet implements
ActionListener{
    Label R=new Label();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (350, 120);
        R.setBounds(40,40,200,20);
        Ejecutar.setBounds(60,80,80,20);
        Limpiar.setBounds(200,80,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(R);add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int I=0;
            double Nota=0, SUMCAL=0;
            while(Nota>=0){
                Nota =
Double.parseDouble(JOptionPane.showInputDialog(null,"Ingrese
nota # "+(I+1),"Cálculo de promedio
",JOptionPane.INFORMATION_MESSAGE));
                if (Nota>=0){
                    SUMCAL+= Nota; I++;
                }
            }
            R.setText("El promedio es: "+(SUMCAL/I));
            Limpiar.requestFocus();
        }
        if(str.equals("Limpiar")){
            R.setText(""); Ejecutar.requestFocus();
        }
    }
}
```

Nótese que es necesario llevar la contabilidad del número de notas ingresadas, es por ello que usamos el contador I.



### Ejemplo 3.7

Escriba un algoritmo que permita invertir un número entero positivo.

#### Solución

Hemos visto anteriormente como invertir un número, pero esta vez no nos indican cuantas cifras tendrá el número, motivo por el cual debemos cambiar de estrategia, por otro lado debemos asegurarnos que el número ingresado, es positivo.

#### Algoritmo

Declarar e inicializar variables

Leer y verificar si número ingresado es positivo

Mientras que el número sea diferente de cero

Realizar cálculos necesarios

Mostrar resultado

### Pseudocódigo

```
1. Iniciar proceso
2. Declarar Variables
    N : Entero
    NI, FI, F : Real
3. Hacer FI = 1
    F = 0.1
    NI = 0
4. Hacer
    4.1. Leer N
5. Mientras (N<=0)
6. Mientras N <> 0 Hacer
    6.1. Calcular NI = NI + (N(MOD)10 * F)
        N = N DIV 10
        FI = FI * 10
        F = F * 0.1
7. Fin_Mientras
8. Calcular NI = NI * FI
9. Escribir 'El número invertido es : ', NI
10. Terminar proceso
```

Veamos la ejecución de este algoritmo, para N = 5438

3. FI = 1 Y F=0.1

6. Mientras 5438 <> 0 Hacer (Se cumple la condición)

$$NI = NI + (N(MOD)10 * F) \implies 0 + (5438(MOD)10 * 0.1) \implies NI = 0.8$$

$$N = N \text{ DIV } 10 \implies 5438 \text{ DIV } 10 \implies N = 543$$

$$FI = FI * 10 \implies 1 * 10 \implies F = 10$$

$$F = F * 0.1 \implies 0.1 * 0.1 \implies F = 0.01$$

Evaluamos nuevamente la condición

6. Mientras 543 <> 0 Hacer (Se cumple la condición)

$$NI = NI + (N(MOD)10 * F) \implies 0.8 + (543(MOD)10 * 0.01) \implies NI = 0.83$$

$$N = N \text{ DIV } 10 \implies 543 \text{ DIV } 10 \implies N = 54$$

$$FI = FI * 10 \implies 10 * 10 \implies F = 100$$

$$F = F * 0.1 \implies 0.1 * 0.1 \implies F = 0.001$$

Evaluamos la condición

6. Mientras  $54 <> 0$  Hacer (Se cumple la condición)

$$NI = NI + (N \text{ MOD } 10 * F) \implies 0.83 + (54 \text{ MOD } 10 * 0.001) \implies NI = 0.834$$

$$N = N \text{ DIV } 10 \implies 54 \text{ DIV } 10 \implies N = 5$$

$$FI = FI * 10 \implies 100 * 10 \implies FI = 1000$$

$$F = F * 0.1 \implies 0.1 * 0.1 \implies F = 0.0001$$

Evaluamos la condición

6. Mientras  $5 <> 0$  Hacer (Se cumple la condición)

$$NI = NI + (N \text{ MOD } 10 * F) \implies 0.834 + (5 \text{ MOD } 10 * 0.001) \implies NI = 0.8345$$

$$N = N \text{ DIV } 10 \implies 5 \text{ DIV } 10 \implies N = 0$$

$$FI = FI * 10 \implies 1000 * 10 \implies FI = 10000$$

$$F = F * 0.1 \implies 0.1 * 0.1 \implies F = 0.00001$$

6. Mientras  $0 <> 0$  Hacer (No se cumple condición fin Mientras)

Finalmente calculamos el número

8.  $NI = NI * FI = 0.8345 * 10000 = 8345$

Note que la lectura de N se ejecutará siempre que ingresemos un valor válido, en este caso un número mayor que cero, en caso contrario el bucle Hacer...Mientras seguirá ejecutándose indefinidamente. Esta es una forma de obligar al usuario a ingresar un valor válido.

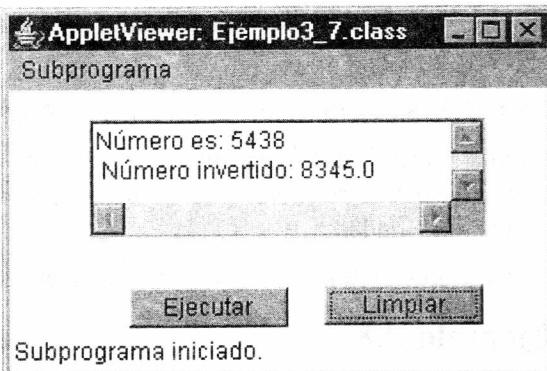
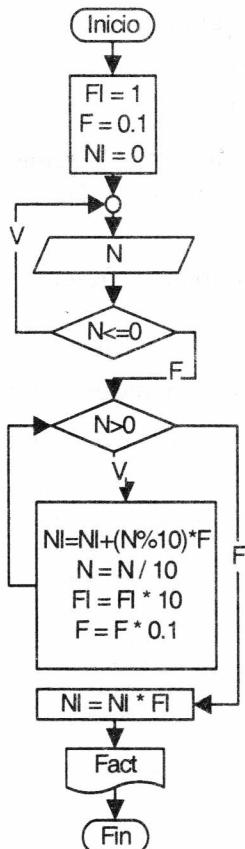
### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo3_7 extends Applet implements
ActionListener{
    TextArea R=new TextArea();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
       setLayout(null);
        setBackground(java.awt.Color.white);
```

```
        resize (350, 120);
        R.setBounds(40,15,250,60);
        Ejecutar.setBounds(60,100,80,20);
        Limpiar.setBounds(200,100,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(R);add(Ejecutar);add(Limpiar);
    }

    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            {
                long N;
                double NI=0, FI=0.1, F=1;
                do{
                    N =
Long.parseLong(JOptionPane.showInputDialog(null,"Ingrese
número","Invertir número
",JOptionPane.INFORMATION_MESSAGE));
                }while(N<=0);

                R.setText("Número es: "+N);
                while (N>0)
                {
                    NI = NI+(N%10)*F;
                    N=N/10;
                    FI=FI*10;
                    F=F*0.1;
                }
                NI*=FI;
                R.append("\n Número invertido: "+NI);
                Limpiar.requestFocus();
            }
            if(str.equals("Limpiar"))
            {
                R.setText("");
                Ejecutar.requestFocus();
            }
        }
    }
}
```



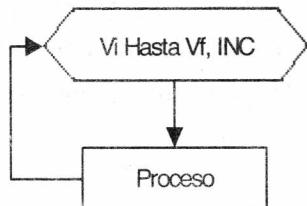
## Estructura Desde

Cuando se sabe el número de iteraciones que tendrán que ejecutarse cierta cantidad de acciones, es recomendable usar la estructura Desde, esta estructura no solo ejecuta un número de acciones, además cuenta internamente el número de iteraciones, esto elimina la necesidad de un contador, lo que no sucede con las estructuras Mientras y Hacer Mientras.

A continuación veremos el pseudocódigo de flujo con su respectivo algoritmo para la estructura:

```

    Desde I = Vi Hasta Vf INC
    Hacer
        Acciones
    Fin_Desde
  
```



I: Variable de tipo entero, representa la variable de control del ciclo.

Vi: Variable de tipo entero, representa el inicio del ciclo.

Vf: Variable de tipo entero, representa el final del ciclo.

INC: Expresión de tipo entero. Representa el incremento/decremento de la variable de control, cuando se omite esta expresión se supone que el incremento es en una unidad.

Veamos un ejemplo sencillo para entender mejor la estructura Desde.

### Ejemplo 3.8

Haga un pseudocódigo para obtener la suma de los 20 primeros números naturales enteros positivos.

#### Algoritmo

Iniciarizar acumulador

Acumular el valor de los números naturales

Reporta sumatoria

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    I : Entero  
    S : Real
3. Hacer S = 0
4. Desde I = 1 Hasta 20 INC I = I + 1 Hacer  
    4.1. S = S + I
5. Fin\_Desde {Fin del ciclo del paso 4}
6. Imprimir S
7. Terminar proceso

I: Variable de tipo entero. Representa la variable de control del ciclo.

S: Variable de tipo real, en ella se almacena la suma.

Nótese en la línea 4 que el contador está siendo inicializado en 1 y finaliza cuando llegue a 20, el contador irá avanzando en una unidad según  $I=I+1$ . Por otro lado en este ejemplo no se necesita ningún dato de entrada.

Hagamos un seguimiento de este ejemplo para ver como actúa la estructura Desde.

3. S = 0

#### **4. Desde I = 1 Hasta 20 INC I = I + 1 Hacer**

Primera iteración, I = 1

4.1.  $S = S + I \implies S = 0 + 1 \implies S = 1$

Segunda iteración, I = 2

4.1.  $S = S + I \implies S = 1 + 2 \implies S = 3$

Tercera iteración, I = 3

4.1.  $S = S + I \implies S = 3 + 3 \implies S = 6$

Cuarta iteración, I = 4

4.1.  $S = S + I \implies S = 6 + 4 \implies S = 10$

Quinta iteración, I = 5

4.1.  $S = S + I \implies S = 10 + 5 \implies S = 15$

Sexta iteración, I = 6

4.1.  $S = S + I \implies S = 15 + 6 \implies S = 21$

Las iteraciones terminan cuando I=20

4.1.  $S = S + I \implies S = 190 + 20 \implies S = 210$

#### **5. Fin\_Desde**

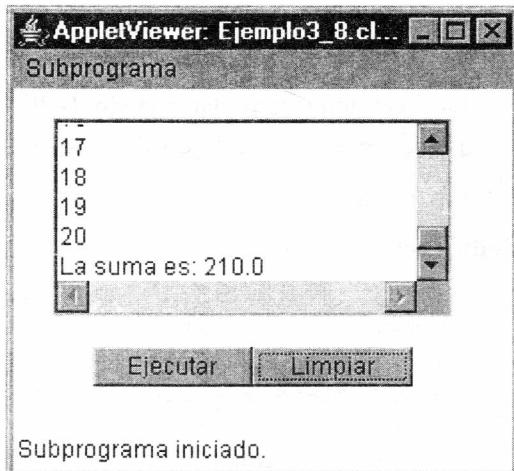
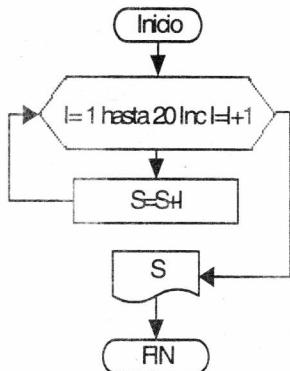
#### **Codificación en Java**

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo3_8 extends Applet implements
ActionListener{
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 170);
        R.setBounds(20,15,200,100);
        Ejecutar.setBounds(40,130,80,20);
        Limpiar.setBounds(120,130,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
```

```

        add(R);add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            float S=0;
            for (int I= 1;I<=20;I++){
                S+=I;R.append(I+"\n");
            }
            R.append("La suma es: "+S);
            Limpiar.requestFocus();
        }
        if(str.equals("Limpiar")){
            R.setText("");Ejecutar.requestFocus();
        }
    }
}

```



Veamos la sintaxis de la estructura for (Desde) en Java

```
for (int I=1;I<=20;I++)
```

Primero tenemos el valor de inicio  $I=1$ , seguido por la expresión que al ser falsa indicará el término del ciclo, por lo tanto tenemos que si  $I \leq 20$  es falso termina el bucle for. Por otro lado Java permite la declaración de la variable a usar dentro de la sintaxis de la estructura for.

### Ejemplo 3.9

Realice un pseudocódigo para identificar si un número ingresado es primo o no

## Solución

Un número es primo cuando sólo puede ser dividido entre la unidad y el mismo número, teniendo en cuenta esto podemos decir que si dividimos el número entre valores consecutivos empezando en la unidad y terminando en el número en mención, sólo debe de haber dos valores que dividan al número, si hay más de dos entonces el número no es primo.

## Algoritmo

Iniciar contador

Leer número

Comprobar si la división entre número e I tiene un residuo cero. Si es cierto entonces incrementar contador

Si el contador es menor que dos, entonces el número es primo

## Pseudocódigo

1. Iniciar proceso
2. Declarar Variables N, C, I : Enteros
3. Leer N
4. Hacer C = 0
5. Desde I = 1 Hasta N Hacer
  - 5.1. Si (N Mod I) = 0 Entonces
    - 5.1.1. Calcular C = C + 1
  - 5.2. Fin\_Si
6. Fin\_Desde{Fin del ciclo}
7. Si c <= 2 Entonces
  - 7.1. Escribir 'Es primo'
8. Si no
  - 8.1. Escribir 'No es primo'
9. Fin\_Si
10. Terminar proceso

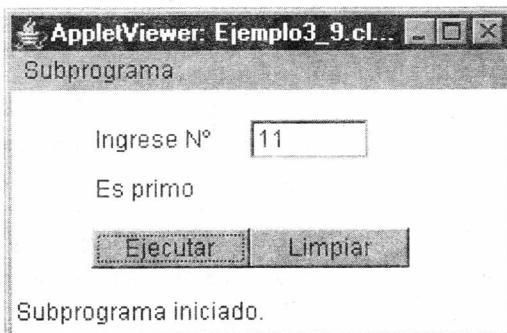
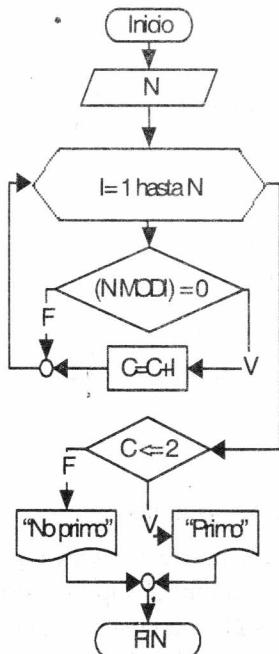
**C:** Variable de tipo entero. Usado como contador, para determinar cuantas veces el número N es dividido exactamente.

Nótese que en la estructura Desde no hemos colocado el incremento, esto es cuando el incremento es en una unidad, podemos omitirla, quedando sobre entendido.

## Codificación en Java

```
import java.awt.*;
```

```
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo3_9 extends Applet implements
ActionListener{
    Label lbl=new Label("Ingrese N°");
    Label R=new Label("");
    TextField Num=new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 100);
        lbl.setBounds(40,15,60,20);
        R.setBounds(40,40,100,20);
        Num.setBounds(120,15,60,20);
        Ejecutar.setBounds(40,70,80,20);
        Limpiar.setBounds(120,70,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl); add(R);add(Num);
        add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int N,C = 0;
            N=Integer.parseInt(Num.getText());
            for (int I= 1;I<=N;I++)
                if ((N%I) == 0) C++;
            if (C <= 2) R.setText("Es primo");
            else R.setText("No es primo");
        }
        if(str.equals("Limpiar")){
            R.setText("");
            Num.setText("");
            Ejecutar.requestFocus();
        }
    }
}
```



### Ejemplo 3.10

Dado 4 notas de un alumno, eliminar la menor y calcular el promedio de las tres notas restantes. Realice el algoritmo para aplicarlo a N alumnos.

### Solución

Un alumno tendrá cuatro notas, eso nos indica el uso de una estructura Desde de 1 a 4, pero por otro lado son N alumnos, eso quiere decir que necesitaremos otra estructura Desde, de 1 hasta N, ambas estructuras deben ir anidadas.

### Algoritmo

Leer cantidad de alumnos

Para cada alumno: Colocar como nota menor 21 e inicializar el acumulador de notas a cero. Hallar la menor nota y calcular el promedio

Mostrar resultado

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables      Nota, S, Menor : Real  
N, I, J : Entero

3. Leer N
4. Desde I = 1 Hasta N Hacer
  - 4.1. Hacer Menor = 21
  - S = 0
  - 4.2. Desde J = 1 Hasta 4
    - 4.2.1. Leer Nota
    - 4.2.2. Hacer S = S + Nota
    - 4.2.3. Si Nota < Menor Entonces
      - 4.2.3.1. Hacer Menor = Nota
    - 4.2.4. Fin\_Si
  - 4.3. Desde
  - 4.4 Imprimir "El promedio es", (S - Menor)/3
5. Fin\_Desde
6. Terminar proceso

**Menor:** Variable de tipo real. Almacena la menor de las notas

Se coloca como la nota menor inicial 21, este artificio puede usarse, ya que las notas son en el sistema vigesimal, y el máximo es 20; entonces suponiendo que todas las notas de un alumno sean 20, se cumple que  $20 < 21$ , entonces se actualizará la nota menor a 20.

Luego de terminar el bucle **desde**, con la variable de control J, se devuelve el control al Desde principal con la variable I, entonces I = 2; nuevamente tendremos Menor = 21 y S = 0, ya que se trata de un nuevo alumno y por lo tanto se trata de un nuevo promedio.

Veamos como se ejecuta este pseudocódigo.

3. N = 3, suponiendo que son tres alumnos

4. Desde I = 1 Hasta 3 Hacer

Para I = 1 tenemos

4.1. Hacer Menor = 21

S = 0

Obtendremos el promedio del alumno I=1

4.2. Desde J = 1 Hasta 4 Hacer

Para J = 1 tenemos S=0 y Menor = 21

4.2.1. Leer Nota = 16

4.2.2. S = S + Nota ==> 0 + 16 ==> S = 16

4.2.3. Si  $16 < 21$  Entonces

    4.2.3.1. Menor = 16

Para J = 2 tenemos, S=16 y Menor = 16

    4.2.1. Leer Nota = 18

    4.2.2.  $S = S + Nota \implies 16 + 18 \implies S = 34$

    4.2.3. Si  $18 < 16$  (Falso, Menor se queda en 16

Para J = 3 tenemos, S=34 y Menor = 16

    4.2.1. Leer Nota = 10

    4.2.2.  $S = S + Nota \implies 34 + 10 \implies S = 44$

    4.2.3. Si  $10 < 16$  Entonces Menor = 10

        4.2.3.1. Menor = 16

Para J = 4 tenemos S=44 y Menor 10

    4.2.1. Leer Nota = 15

    4.2.2.  $S = S + Nota \implies 44 + 15 \implies S = 59$

    4.2.3. Si  $15 < 16$  (Falso)

4.3. Fin\_Desde

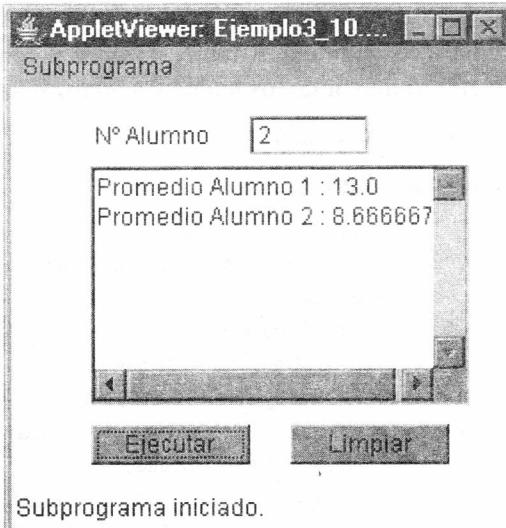
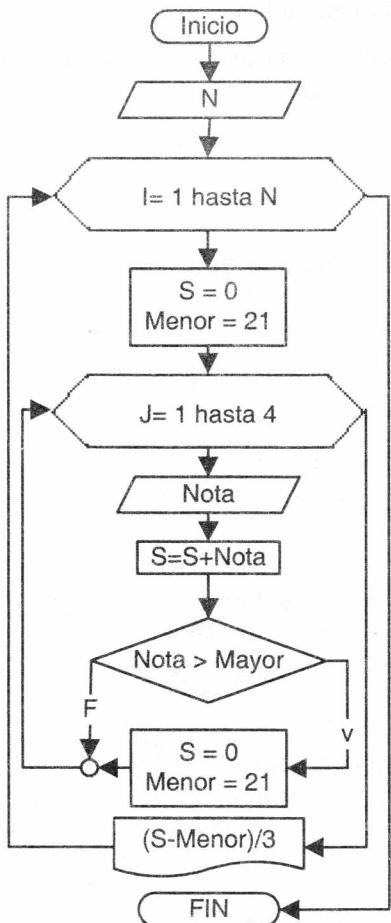
4.4. Imprimir "El promedio es:",  $(S - Menor)/3 \implies (59-10)/3 \implies 16.33$

5. Fin\_Desde

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo3_10 extends Applet implements
ActionListener{
    Label lbl=new Label("Nº Alumnos");
    TextField Num=new TextField();
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
```

```
        resize (250, 200);
        lbl.setBounds(40,15,60,20);
        Num.setBounds(120,15,60,20);
        R.setBounds(40,40,190,120);
        Ejecutar.setBounds(40,170,80,20);
        Limpiar.setBounds(140,170,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl);add(Num);add(R);
        add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            float Nota, S, Menor;
            int N;
            N=Integer.parseInt(Num.getText());
            for (int I = 1;I<=N;I++){
                Menor = 21;
                S = 0;
                for (int J=1;J<=4;J++){
                    Nota =
                    Float.parseFloat(JOptionPane.showInputDialog(null,"Ingrese
nota "+J,"Cálculo de Promedio
",JOptionPane.INFORMATION_MESSAGE));
                    S+= Nota;
                    if(Nota < Menor)
                        Menor = Nota;
                }
                R.append("Promedio Alumno "+I+" :
"+( (S-Menor)/3)+"\n");
            }
            if(str.equals("Limpiar")){
                R.setText("");
                Num.setText("");
                Ejecutar.requestFocus();
            }
        }
    }
}
```



### Ejemplo 3.11

Calcular el menor y el mayor de una lista de N números enteros

#### Solución

Resultaría imposible usar el artificio anterior para hallar el menor número si no se sabe entre qué límites se encuentra, en este caso podemos hacer que el primer número leído sea el menor y el mayor a la vez, este sería el mejor artificio para encontrar el menor y el mayor de una lista.

#### Algoritmo

Leer primer número

Hacer que primer número sea el mayor y el menor a la vez

Para los siguientes números: Leer y comprobar si el leído anteriormente es menor o mayor de los leídos y actualizar si es necesario.

Mostrar resultados

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables Numero, I, J,N : Entero
3. Leer N
4. Leer Numero
5. Hacer Menor = Numero  
    Mayor = Numero
6. Desde I = 2 Hasta N Hacer
  - 6.1. Leer Numero
  - 6.2. Si Numero < Menor Entonces
    - 6.1.1. Hacer Menor = Numero
  - 6.3. Si no
    - 6.3.1. Si Numero > Mayor Entonces
      - 6.3.1.1. Hacer Mayor = Numero
    - 6.3.2. Fin\_Si
  - 6.4. Fin\_Si
7. Fin\_Desde
8. Escribir 'El número mayor es: ', Mayor  
    'El número menor es: ', Menor
9. Terminar proceso

N: Numero de tipo entero. Representa la cantidad de números.

Numero: Número de tipo entero. Representa el número leído.

Nótese que el bucle Desde empieza en 2, a diferencia de los anteriores vistos, esto se debe a que en la línea 4 ya leemos el primer valor de la lista, por lo tanto en el bucle Desde debemos empezar a leer los números de la lista a partir del segundo.

Se obtiene el mismo resultado reemplazando la línea 6 por la siguiente

6. Desde I = 1 Hasta N-1 Hacer

recuerde que la estructura Desde sólo la usamos para contabilizar los números que se van ingresando, por este motivo es indiferente, en este caso, usar cualquier forma, de las ya presentadas, para la estructura Desde.

### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo3_11 extends Applet implements
ActionListener{
    Label lbl=new Label("Cantidad de números");
    TextField Num=new TextField();
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 150);
        lbl.setBounds(10,15,125,20);
        Num.setBounds(145,15,60,20);
        R.setBounds(20,40,190,60);
        Ejecutar.setBounds(20,120,80,20);
        Limpiar.setBounds(110,120,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl);add(Num);add(R);
        add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int Numero, N, Menor,Mayor;
            N=Integer.parseInt(Num.getText());
            Numero =
Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
Nº 1","Cálculo de Mayor y Menor
",JOptionPane.INFORMATION_MESSAGE));
            Menor = Mayor = Numero;
            for(int I=2;I<=N;I++){
                Numero =
Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
Nº "+I,"Cálculo de Mayor y Menor
```

```

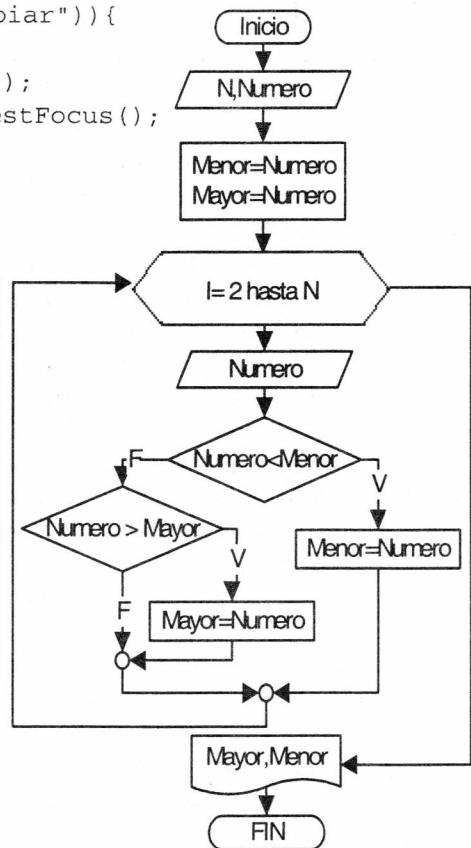
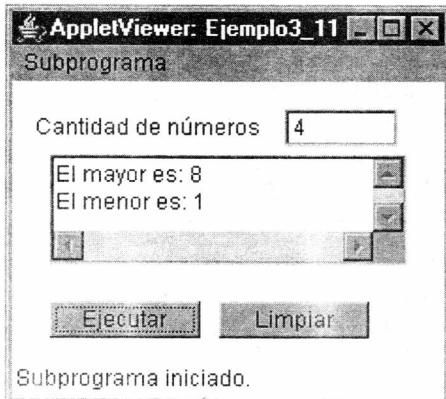
        , JOptionPane.INFORMATION_MESSAGE));
        if (Numero<Menor) Menor = Numero;
        else if(Numero>Mayor) Mayor=Numero;
    }
    R.append("El mayor es: "+Mayor+"\n");
    R.append("El menor es: "+Menor);
}
if(str.equals("Limpiar")){
    R.setText("");
    Num.setText("");
    Ejecutar.requestFocus();
}
}

```

```

graph TD
    Inicio([Inicio]) --> N[/N,Numero/]
    N --> Menor[Menor=Numero  
Mayor=Numero]

```



### Ejemplo 3.12

Hallar cuantos términos hay en la progresión aritmética mostrada a continuación, También halle la suma de los términos.

$$5, 8, 11, 14, 17, 20, 23, \dots, n$$

## Algoritmo

Inicializa acumuladores y contadores

Leer último término posible

Para cada valor de la progresión, acumular el valor

Aumentar contador de la cantidad de número sumados.

Mostrar resultado

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
Suma, N, I, C : Entero
3. Hacer C = 0  
Suma = 0
4. Leer N
5. Desde I = 5 Hasta N INC I = I + 3
  - 5.1. Suma = Suma + I
  - 5.2. C = C + 1
6. Fin\_Desde
7. Escribir 'El número de términos es : ', N  
'La suma de los términos es: ', Suma
8. Terminar proceso

Véase el incremento de la estructura Desde, el cual es en 3 unidades, por cada corrida, esto es debido a que la razón de la progresión es de 3 unidades.

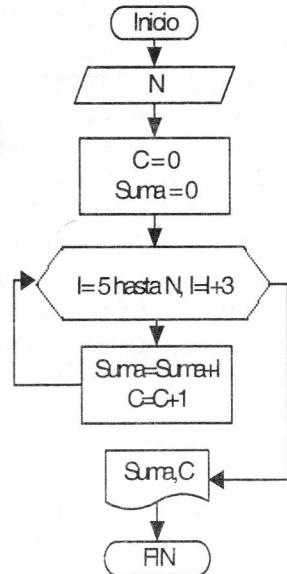
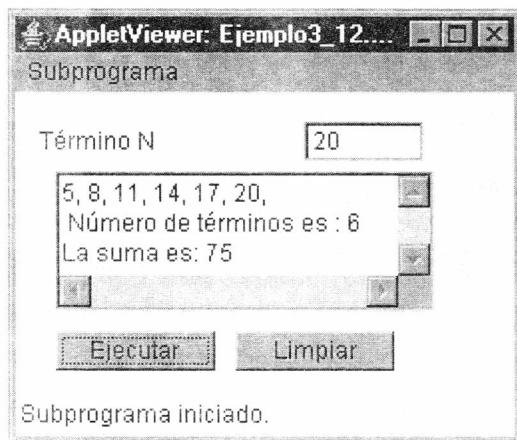
#### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo3_12 extends Applet implements
ActionListener{
    Label lbl=new Label("Término N");
    TextField Num=new TextField();
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 150);
        lbl.setBounds(10,15,125,20);
        Num.setBounds(145,15,60,20);
        R.setBounds(20,40,190,70);
```

```

        Ejecutar.setBounds(20,120,80,20);
        Limpiar.setBounds(110,120,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl);add(Num);add(R);
        add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int Suma=0, N, C=0;
            N=Integer.parseInt(Num.getText());
            for (int I=5;I<=N;I+=3) {
                Suma+= I; C++; R.append(I+", ");
            }
            R.append("\n Número de términos es : "+C);
            R.append("\nLa suma es: "+Suma);
        }
        if(str.equals("Limpiar")){
            R.setText(""); Num.setText("");
            Ejecutar.requestFocus();
        }
    }
}

```



### Ejemplo 3.13

Escriba un algoritmo que nos permita hallar el factorial de un número

#### Solución

Dado un número N el factorial de dicho número es

$$N * (N-1) * (N-2) - \dots * 1$$

Por ejemplo si el número es 4, su factorial es

$$4 * 3 * 2 * 1$$

Por otro lado hay un factorial especial que tenemos que considerar, es el factorial de cero, cuyo resultado es 1, entonces:

$$!0 = 1$$

#### Algoritmo

Leer número

Comprobar si número es cero, entonces su factorial es 1

Si número es diferente de cero hallar factorial.

Mostrar resultado

#### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    I, N : Entero  
    Fact : Real
3. Fact = 1
4. Leer N
5. Si N <> 0 Entonces  
        5.1. Desde I = N Hasta 1 con I = I-1 Hacer  
            5.1.1. Fact = Fact \* I  
        5.2. Fin\_Desde
6. Fin\_Si
7. Escribir 'El factorial es : ', Fact
8. Terminar proceso

Nótese en la línea 5.1 que I empieza en N y luego vamos bajando en una unidad hasta llegar a uno. También estamos asumiendo que el número ingresado es un número positivo o cero. A manera de ejercicio puede

implementar la validación del número ingresado.

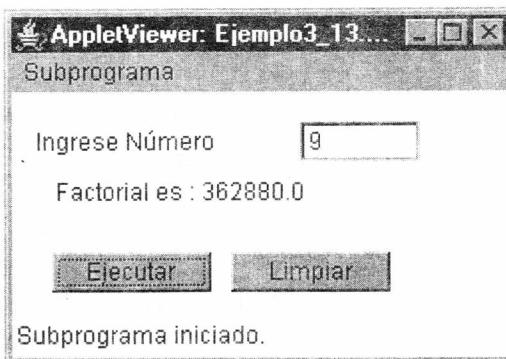
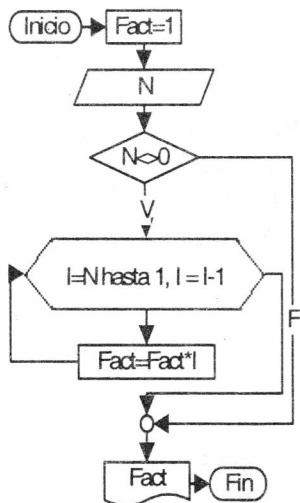
### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
public class Ejemplo3_13 extends Applet implements
ActionListener{
    Label lbl=new Label("Ingrese Número");
    Label R=new Label();
    TextField Num=new TextField();
    Button Ejecutar=new Button("Ejecutar");
    Button Limpieza=new Button("Limpieza");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 110);
        lbl.setBounds(10,15,125,20);
        Num.setBounds(145,15,60,20);
        R.setBounds(20,40,150,20);
        Ejecutar.setBounds(20,80,80,20);
        Limpieza.setBounds(110,80,80,20);
        Ejecutar.addActionListener(this);
        Limpieza.addActionListener(this);
        add(lbl); add(Num);add(R);
        add(Ejecutar);add(Limpieza);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int N;
            float Fact=1;
            N=Integer.parseInt(Num.getText());
            if (N!=0)
                for (int I=N;I>=1;I--)
                    Fact*= I;
            R.setText("Factorial es : "+Fact);
        }
    }
}
```

```

        }
        if(str.equals("Limpiar")){
            R.setText(" "); Num.setText(" ");
            Ejecutar.requestFocus();
        }
    }
}

```



### Ejemplo 3.14

Diseñar un algoritmo que me permita calcular las 5 primeras parejas de números primos gemelos.

#### Solución

Dos números son primos gemelos si además de ser números primos, la diferencia entre ellos es exactamente dos. Entonces primero debemos de ver si ambos números son primos, y luego comprobar si son primos gemelos.

#### Algoritmo

Iniciar contador

Repetir

    Leer números válidos

    Comprobar si números son primos

        Si son primos comprobar si son primos gemelos

Hasta encontrar 5 pares de números gemelos

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables C, N1, N2, I : Entero
3. Hacer C = 0
4. Hacer
  - 4.1. Hacer
    - 4.1.1. Leer N1, N2
  - 4.2. Mientras (N1 <= 0) o (N2 <= 0)
  - 4.3. Si Primo(N1) && Primo(N2) Entonces
    - 4.3.1. Si (N1-N2==2) || (N2-N1==2)
      - 4.3.1.1. Escribir 'primos gemelos'
      - 4.3.1.2. C=C+1
    - 4.3.2. Si no
      - 4.3.2.1. Escribir 'primos no gemelos'
    - 4.3.3. Fin Si
  - 4.4. Si no
    - 4.4.1. Escribir 'No son números primos'
- 4.4. Fin Si
5. Mientras (C<5)
6. Terminar proceso

Función Primo, retorna un tipo boolean

1. Iniciar Función
2. Declarar Variables G : Entero
3. Hacer G = 0
4. Leer N
5. Desde I=1 hasta N hacer
  - 5.1. Si (N MOD I = 0) Entonces
    - 5.1.1. G = G + 1
  - 5.2. Fin Si
6. Fin Desde
7. Si G<=2 Entonces
  - 7.1. Retornar Verdad
8. Si no
  - 8.1. Retornar Falso

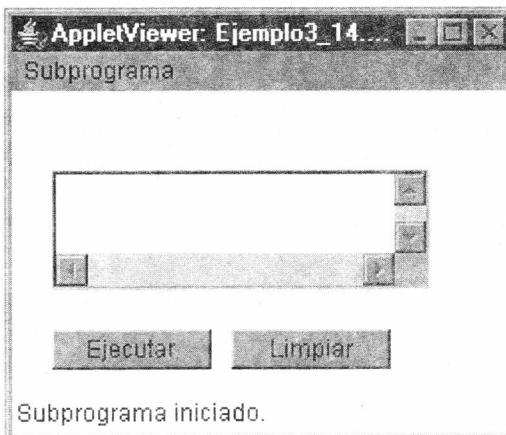
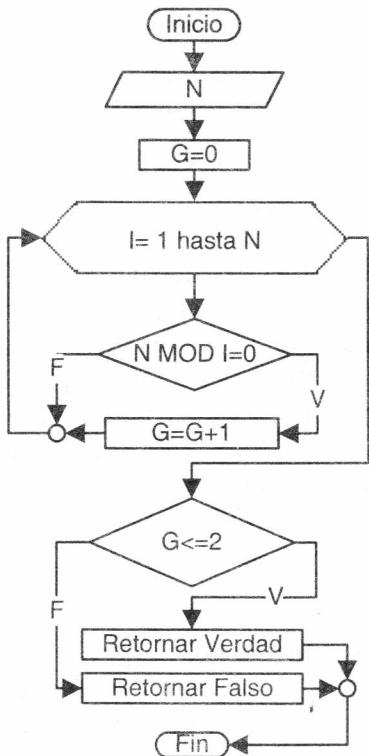
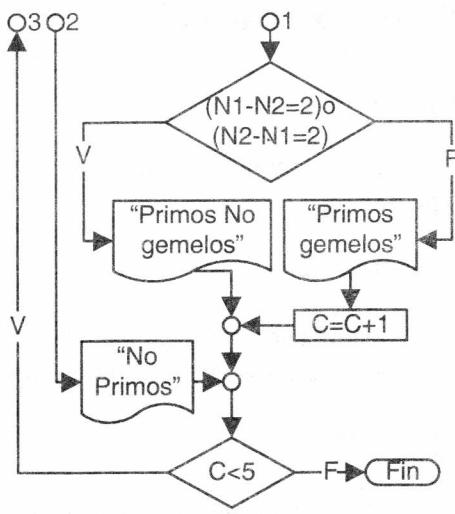
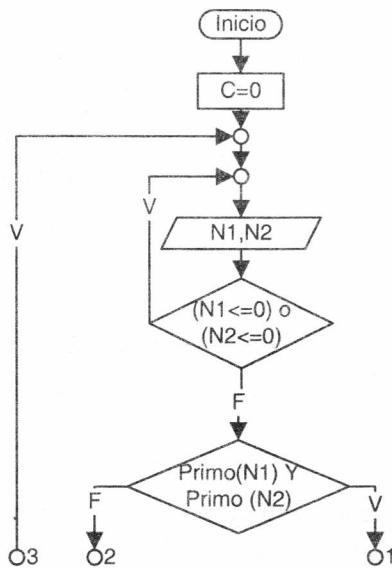
9. Fin\_Si
10. Fin Función

#### Codificación en Java

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo3_14 extends Applet implements
ActionListener{
    Label lbl=new Label("Cantidad de números");
    TextField Num=new TextField();
    TextArea R=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");
    Button Limpiar=new Button("Limpiar");
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (250, 150);
        lbl.setBounds(10,15,125,20);
        Num.setBounds(145,15,60,20);
        R.setBounds(20,40,190,60);
        Ejecutar.setBounds(20,120,80,20);
        Limpiar.setBounds(110,120,80,20);
        Ejecutar.addActionListener(this);
        Limpiar.addActionListener(this);
        add(lbl);add(R);add(Ejecutar);add(Limpiar);
    }
    public void actionPerformed(ActionEvent ac){
        String str=ac.getActionCommand();
        if(str.equals("Ejecutar")){
            int C=0, N1, N2;
            do{
                do{
                    N1 =
                    Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
Nº 1","Números
```

```
perfectos", JOptionPane.INFORMATION_MESSAGE));
N2 =
Integer.parseInt(JOptionPane.showInputDialog(null, "Ingrese
Nº 2", "Números
perfectos", JOptionPane.INFORMATION_MESSAGE));
}while((N1 <= 0) || (N2<=0));
if (Primo(N1) && Primo(N2))
    if ((N1-N2==2) || (N2-N1==2)){
        R.append("Primos gemelos \n");
        C++;
    }else
        R.append("Son primos no
                gemelos \n");
    else
        R.append("No primos \n");
}while (C<5);
}
if(str.equals("Limpiar")){
    R.setText(" "); Num.setText(" ");
    Ejecutar.requestFocus();
}
}

boolean Primo(int N){
    int G=0;
    for (int I=1;I<=N;I++)
        if (N%I==0)
            G++;
    if (G<=2)
        return true;
    else
        return false;
}
}
```



El algoritmo lee pares de números y los valida, en caso de no ser números válidos vuelve a pedir un par de números. Al ser los números válidos verifica si son primos, al ser primos comprueba si con primos gemelos.

El algoritmo terminará cuando se encuentre cinco pares de números primos gemelos.

## Ejercicios propuestos

### Ejercicio 1

Calcule la nomina de un grupo de trabajadores usando las tres estructuras repetitivas

### Ejercicio 2

Escriba un pseudocódigo, que dados como datos N números enteros, obtenga el número de ceros que hay entre estos números.

### Ejercicio 3

Supongamos que debemos obtener la suma de los gastos que hicimos en nuestro último viaje, pero no sabemos exactamente cuántos fueron. Realice un algoritmo para resolver este problema.

### Ejercicio 4

Escriba un pseudocódigo que dado un grupo de números naturales positivos, calcule e imprima el cubo de estos números.

### Ejercicio 5

En un supermercado una ama de casa pone en su carrito los artículos que va tomando de los estantes. La señora quiere asegurarse de que el cajero le cobre bien lo que ella ha comprado, por lo que cada vez que toma un artículo anota su precio junto con la cantidad de artículos iguales que ha tomado y determina cuánto dinero gastará en ese artículo; a esto le suma lo que irá gastando en los demás artículos, hasta que decide que ya tomó todo lo que necesitaba. Ayúdale a esta señora a obtener el total de sus compras.

### Ejercicio 6

Determinar la cantidad semanal de dinero que recibirá cada uno de los  $n$  obreros de una empresa. Se sabe que cuando las horas que trabaja un obrero exceden de 40, el resto se convierte en horas extras que se pagan al doble de una hora

normal, cuando no exceden de 8; cuando las horas extras exceden de 8 se pagan las primeras 8 al doble de lo que se paga por una hora normal y el resto al triple. Considere que todos los obreros ganan el mismo sueldo (\$200)

### **Ejercicio 7**

Construya un pseudocódigo que calcule e imprima la suma de los N primeros números naturales.

### **Ejercicio 8**

Escriba un pseudocódigo, que dados como datos 270 números enteros, obtenga la suma de los números pares y el promedio de los números impares. Además indique cuantos ceros se ingresaron.

### **Ejercicio 9**

Hacer un pseudocódigo para obtener la tabla de multiplicar de un número entero K, comenzando desde 1.

Halle el número de término para la progresión siguiente:

1, 2, 4, 8, 16, 32, ...., 10000



# Capítulo IV

# Vectores y

# Matrices

Este capítulo contiene:

Lectura y escritura de un vector

Métodos de ordenamiento de un vector

    Método de la burbuja

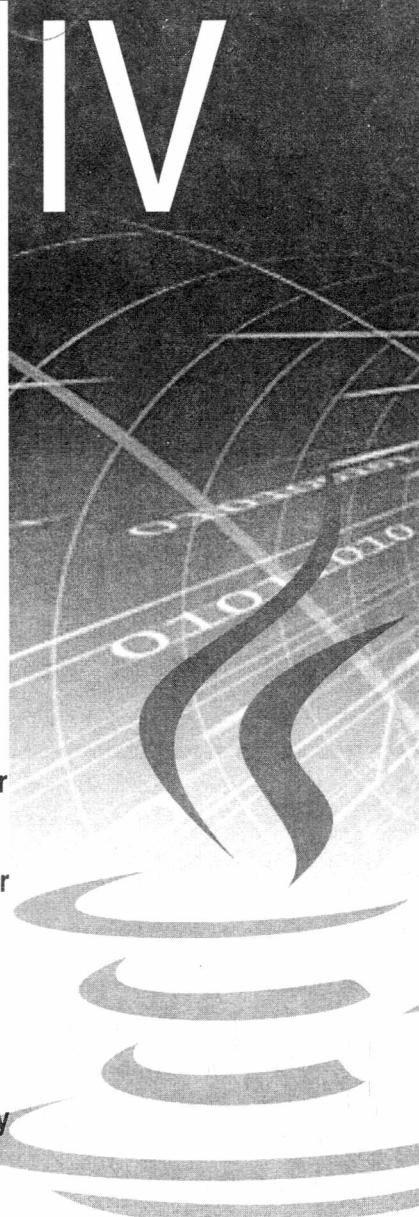
    Método de ordenamiento por  
    Inserción

    Método de Selección

    Método de Shell

Búsqueda en un vector

Ejemplos desarrollados de vectores y  
matrices



Algoritmos y Diagramas de Flujo aplicados en Java 2



## **Vectores**

Empezaremos con los tipos de datos estructurados, y con el más sencillo, los arreglos o vectores.

Los arreglos o arrays, permiten agrupar datos usando un mismo identificador. Todos los elementos de un array son del mismo tipo, y para acceder a cada elemento se usan subíndices.

Los valores para el número de elementos deben ser constantes, y se pueden usar tantas dimensiones como queramos, limitado sólo por la memoria disponible, al momento de codificar el algoritmo.

Cuando sólo se usa una dimensión se suele hablar de listas o vectores, cuando se usan dos o más, de tablas o matrices.

Luego veremos las cadenas de caracteres que son un tipo especial de arreglos. Se trata en realidad de arreglos de una dimensión de tipo char.

Los subíndices son enteros, y pueden tomar valores desde 1 hasta "número de elementos". Esto es muy importante, y hay que tener mucho cuidado, por ejemplo:

Vector : arreglo[1..10] de enteros

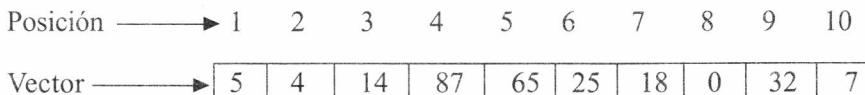
En java la declaración es

```
int V[] = new int[10];
```

Creará un arreglo llamado Vector con 10 enteros a los que accederemos como

Vector[1], Vector[2]...Vector[10]. Como subíndice podremos usar cualquier expresión entera.

En la ilustración siguiente podemos ver la representación gráfica de un vector, como podemos observar la primera posición de un vector es 1, aunque en ciertos lenguajes de codificación como el Java, un vector empieza en la posición 0 y no en la posición 1, en esta ocasión, usaremos el 1 como principio de un vector solo para el caso del algoritmo.



## Lectura y escritura de un vector

Un vector es una colección de datos y no un dato en particular, por lo tanto, si por ejemplo tenemos el siguiente vector:

Vector : Arreglo [1..10] de enteros

el cual, como podemos ver, es un arreglo de enteros, el cual posee como máximo 10 números. Por tanto sería incorrecto decir:

Vector = 7

esto implica que el proceso de lectura y escritura de un vector no lo podemos realizar en una simple línea de código, esto no quiere decir que no sea fácil. Lo más práctico es usar un bucle repetitivo para leer todos los datos del vector, puede usar cualquiera de las estructuras vistos en el capítulo anterior.

Entonces tenemos el siguiente ejemplo de lectura de un arreglo:

Desde I = 1 Hasta 5 Hacer

    Leer (Vector[I])

Fin\_Desde

Por ejemplo, si queremos ingresar los siguientes datos al vector

5, 98, 47, 12, 34

La ejecución del algoritmo sería de la siguiente manera

Primero al declarar el vector este se encuentra vacío



Desde I = 1 Hasta 5 Hacer

Para I = 1, es decir se leerá el valor y se almacenará en la posición 1

Leer (5)

3				
---	--	--	--	--

Para I = 2, almacenará en valor leido en la posición 2

Leer (98)

5	98			
---	----	--	--	--

Para I = 3, almacenará en valor leido en la posición 3

Leer (47)

5	98	47		
---	----	----	--	--

Para I = 4, almacenará en valor leido en la posición 4

Leer (12)

5	98	47	12	
---	----	----	----	--

Para I = 5, almacenará en valor leido en la posición 5

Leer (34)

5	98	47	12	34
---	----	----	----	----

De esta manera hemos visto como se van ingresando los datos dentro de un vector. De manera análoga se realiza la escritura, de la siguiente manera:

Desde I = 1 Hasta 5 Hacer

Escribir (Vector[I])

Fin\_desde

Veamos un ejemplo clásico y sencillo de arreglos.

### **Ejemplo 4.1**

Realice un pseudocódigo que calcule la suma de los elementos de un arreglo de 10 elementos.

#### **Solución**

Esta es la forma más sencilla de leer un vector, muchas veces al adaptar el algoritmo a un lenguaje de programación específico, se deben de realizar cambios mínimos pero la lógica central es la misma.

#### **Algoritmo**

Iniciar acumulador

Leer vector

Hallar suma de los componentes del vector

Reportar resultado

#### **Pseudocódigo**

1. Iniciar proceso

2. Declarar Variables  
I, Sum : Entero  
V : Arreglo [1..10] de Enteros
3. Hacer Sum = 0
4. Desde I = 1 Hasta 10 Hacer  
    4.1. Leer V[I]
5. Fin\_Desde
6. Desde I = 1 Hasta 10 Hacer  
    6.1. Sum = Sum + V[I]
7. Fin\_Desde
8. Imprimir 'La suma es:', Sum
9. Terminar proceso

Como podemos observar antes de realizar cualquier operación con los arreglos primero debemos de ingresar el vector completo, luego podemos usar el vector en cualquier parte del algoritmo.

### Codificación en Java

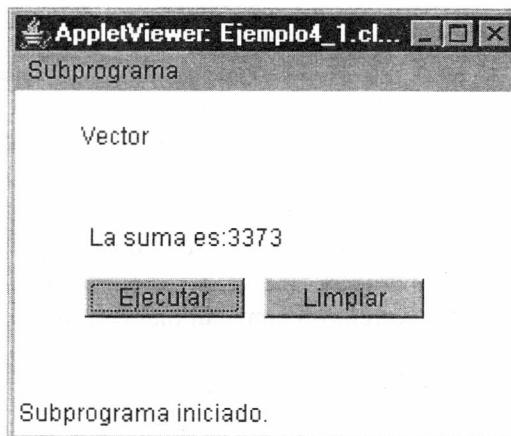
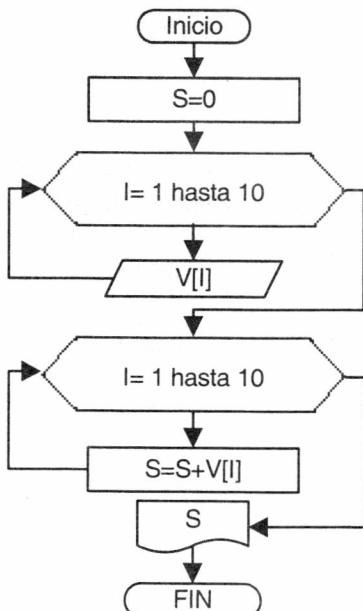
```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
import javax.swing.JOptionPane;  
public class Ejemplo4_1 extends Applet implements  
ActionListener{  
    Label lbl=new Label("Vector");  
    Label Suma=new Label();  
    Button Ejecutar=new Button("Ejecutar");  
    Button Limpiar=new Button("Limpiar");  
    public void init() {  
        setLayout(null);  
        setBackground(java.awt.Color.white);  
        resize (250, 150);  
        lbl.setBounds(10,15,125,20);  
        Suma.setBounds(20,90,190,20);  
        Ejecutar.setBounds(20,120,80,20);  
        Limpiar.setBounds(110,120,80,20);  
        Ejecutar.addActionListener(this);  
        Limpiar.addActionListener(this);  
        add(lbl); add(Suma);add(Ejecutar);add(Limpiar);  
    }  
    public void actionPerformed(ActionEvent ac){  
        String str=ac.getActionCommand();
```

```

if(str.equals("Ejecutar")){
    int Sum;
    int V[] = new int[10];
    Sum = 0;
    for (int I=0;I<10;I++)
        V[I]=Integer.parseInt(JOptionPane.showInputDialog(null, "Ingrese
dato # "+(I+1), "Operación con
vectores", JOptionPane.INFORMATION_MESSAGE));
        for (int I=0;I<10;I++) Sum+=V[I];
        Suma.setText("La suma es:"+Sum);
    }
    if(str.equals("Limpiar")){
        Suma.setText("");
        Ejecutar.requestFocus();
    }
}
}

```

Como mencionamos antes en Java todos los vectores siempre comienzan en la posición cero debido a eso el contador empieza en  $I=0$  y va hasta  $I=9$ , en este ejemplo en particular.



## Ejemplo 4.2

Realice un pseudocódigo que calcule la media aritmética de N valores, además reporte los números ingresados en orden inverso.

### Algoritmo

Iniciar acumulador

Leer y sumar elementos del vector

Reportar resultado y reportar vector inverso

### Pseudocódigo

1. Iniciar proceso
2. Declarar Variables  
    N, I, Sum : Entero  
    V : Arreglo [1..100] de Enteros
3. Hacer Sum = 0
4. Leer N
5. Desde I = 1 Hasta N Hacer  
        5.1. Leer V[I]  
        5.2. Sum = Sum + V[I]
6. Fin\_Desde
7. Desde I = N Hasta 1 con [I=I-1] Hacer  
        7.1. Escribir V[I]
8. Fin\_Desde
9. Imprimir 'La media Aritmética es:', (Sum/N)
10. Terminar proceso

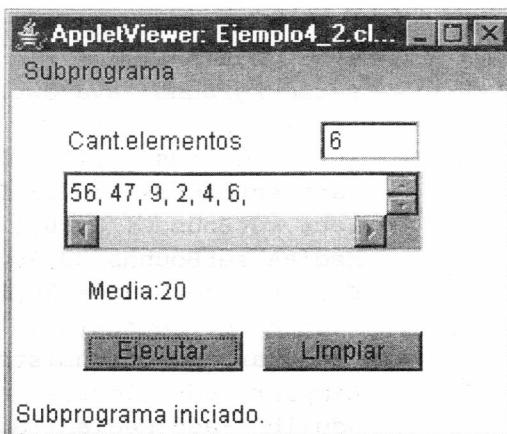
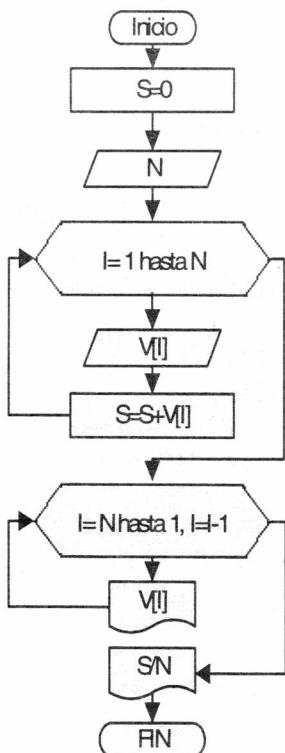
Como se puede apreciar luego de ingresado el número a la posición I del arreglo, este es acumulado en la variable Sum, este procedimiento es válido ya que primero leemos el número, por otro lado estamos usando un bucle inverso, para mostrar el arreglo al revés, en verdad lo único que se hace es mostrar el arreglo en orden inverso pero los valores del vector siguen en sus respectivas posiciones, más adelante veremos como cambiar de posiciones los valores de un arreglo.

### Codificación en Java

```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
import javax.swing.JOptionPane;  
public class Ejemplo4_2 extends Applet implements  
ActionListener{  
    Label lbl=new Label("Cant.elementos");
```

```
Label MediaA=new Label();
TextField Cant=new TextField();
TextArea Vect=new TextArea("");
Button Ejecutar=new Button("Ejecutar");
Button Limpiar=new Button("Limpiar");
public void init() {
    setLayout(null);
    setBackground(java.awt.Color.white);
    resize (250, 150);
    lbl.setBounds(10,15,125,20);
    Cant.setBounds(150,15,50,20);
    Vect.setBounds(10,40,180,40);
    MediaA.setBounds(20,90,150,20);
    Ejecutar.setBounds(20,120,80,20);
    Limpiar.setBounds(110,120,80,20);
    Ejecutar.addActionListener(this);
    Limpiar.addActionListener(this);
    add(lbl);add(MediaA);add(Vect);add(Cant);
    add(Ejecutar);add(Limpiar);
}
public void actionPerformed(ActionEvent ac){
    String str=ac.getActionCommand();
    if(str.equals("Ejecutar")){
        int Sum=0,N;
        N=Integer.parseInt(Cant.getText());
        int V[]={};
        for (int I=0;I<N;I++){
            V[I]=Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
dato # "+(I+1),"Operación con
vectores",JOptionPane.INFORMATION_MESSAGE));
            Sum = Sum + V[I];
        }
        for (int I=N-1;I>=0;I--)
            Vect.append(V[I]+", ");
        MediaA.setText("Media:"+ (Sum/N));
    }
    if(str.equals("Limpiar")){
        Cant.setText("");MediaA.setText("");
        Vect.setText("");Cant.requestFocus();
    }
}
```

Nótese en la programación en java que primero leemos la cantidad de elementos que tendrá el vector y luego creamos dicho vector, esa una de las bondades de java, de esa manera ocupamos solo el espacio suficiente en memoria.



## Ordenamiento de un vector

Una operación que se hace muy a menudo con los arrays, sobre todo con los de una dimensión, es ordenar sus elementos.

### Método de la burbuja

Este es uno de los métodos de ordenamiento más usados, aunque no de los más eficaces, se trata del método de la burbuja.

Consiste en recorrer la lista de valores a ordenar y compararlos dos a dos. Si los elementos están bien ordenados, pasamos al siguiente par, si no lo están los intercambiamos, y pasamos al siguiente, hasta llegar al final de la lista. El proceso completo se repite hasta que la lista está ordenada.

El algoritmo de ordenación es el siguiente:

```

1. Desde I = .1 hasta N-1 hacer
  1.1. Desde J = 1 hasta N-I hacer
    1.1.1. Si V[J] > V[J+1] entonces
      1.1.1.1. Calcular Aux = V[J]
      V[J] = V[J+1]
      V[J+1] = Aux
    1.1.2. Fin_Si
  1.2. Fin_Desde
2. Fin_Desde

```

En donde:

I, J : Variables de tipo entero usados para el control de los bucles.

N representa el tamaño del vector, el cual está dado por el número de elementos que posee.

V : arreglo a ordenar

Aux : Variable de tipo del arreglo, usada para intercambiar los valores.

### Codificación en Java

```

int Aux;
for (int I=0; I<N-1; I++)
  for (int J=0; J<(N-1)-I; J++)
    if (V[J]>V[J+1]){
      Aux=V[J];
      V[J] = V[J+1];
      V[J+1] = Aux;
    }
}

```

Veamos un ejemplo, ordenando el siguiente vector de menor a mayor:

15	3	8	6	18	1
----	---	---	---	----	---

Desde I = 1 hasta N-1 , es decir hasta 5

Para i = 1

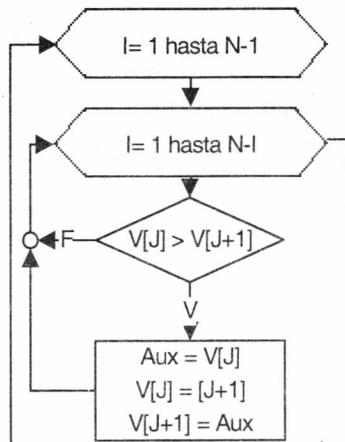
Desde J = 1 hasta N-I, es decir desde 1 hasta 5

Para J=1 Si V[J] > V[J+1] ==> V[1] > V[2] ==> 15>3 (Verdadero)

Aux = V[J] ==> Aux = 15

V[J] = V[J+1] ==> V[1]=3

V[J+1] = Aux ==>V[2]=15



El vector ahora es de la siguiente manera

3	15	8	6	18	1
---	----	---	---	----	---

Para J=2 Si  $V[J] > V[J+1]$   $\Rightarrow V[2] > V[3] \Rightarrow 15 > 8$  (Verdadero)

Aux =  $V[J] \Rightarrow Aux = 15$

$V[J] = V[J+1] \Rightarrow V[2] = 8$

$V[J+1] = Aux \Rightarrow V[3] = 15$

El vector ahora es de la siguiente manera

3	8	15	6	18	1
---	---	----	---	----	---

Para J=3 Si  $V[J] > V[J+1]$   $\Rightarrow V[3] > V[4] \Rightarrow 15 > 6$  (Verdadero)

Aux =  $V[J] \Rightarrow Aux = 15$

$V[J] = V[J+1] \Rightarrow V[3] = 6$

$V[J+1] = Aux \Rightarrow V[4] = 15$

3	8	6	15	18	1
---	---	---	----	----	---

Para J=4 Si  $V[J] > V[J+1]$   $\Rightarrow V[4] > V[5] \Rightarrow 15 > 18$  (Falso)

Para J=5 Si  $V[J] > V[J+1]$   $\Rightarrow V[5] > V[6] \Rightarrow 18 > 1$  (Verdadero)

Aux =  $V[J] \Rightarrow Aux = 18$

$V[J] = V[J+1] \Rightarrow V[5] = 1$

$V[J+1] = Aux \Rightarrow V[6] = 18$

3	8	6	15	1	18
---	---	---	----	---	----

Fin del bucle con el indice J, entonces I aumenta en una unidad, por lo tanto I=2 y vuelve a iniciar J = 1 Hasta 4

Para J=1 Si  $V[J] > V[J+1]$   $\Rightarrow V[1] > V[2] \Rightarrow 3 > 8$  (Falso)

Para J=2 Si  $V[J] > V[J+1]$   $\Rightarrow V[2] > V[3] \Rightarrow 8 > 6$  (Verdadero)

Aux =  $V[J] \Rightarrow Aux = 8$

$V[J] = V[J+1] \Rightarrow V[2] = 6$

$V[J+1] = Aux \Rightarrow V[3] = 8$

3	6	8	15	1	18
---	---	---	----	---	----

Para J=3 Si  $V[J] > V[J+1]$   $\implies V[3] > V[4] \implies 8 > 15$  (Falso)

Para J=4 Si  $V[J] > V[J+1]$   $\implies V[4] > V[5] \implies 15 > 1$  (Verdadero)

Aux =  $V[J] \implies \text{Aux} = 15$

$V[J] = V[J+1] \implies V[4] = 1$

$V[J+1] = \text{Aux} \implies V[5] = 15$

3	6	8	1	15	18
---	---	---	---	----	----

Ahora I = 3 y J va desde 1 hasta 3

Para J=1 Si  $V[J] > V[J+1]$   $\implies V[1] > V[2] \implies 3 > 6$  (Falso)

Para J=2 Si  $V[J] > V[J+1]$   $\implies V[2] > V[3] \implies 6 > 8$  (Falso)

Para J=3 Si  $V[J] > V[J+1]$   $\implies V[3] > V[4] \implies 8 > 1$  (Verdadero)

Aux =  $V[J] \implies \text{Aux} = 8$

$V[J] = V[J+1] \implies V[3] = 1$

$V[J+1] = \text{Aux} \implies V[4] = 15$

3	6	1	8	15	18
---	---	---	---	----	----

Ahora I = 4 y J va desde 1 hasta 2

Para J=1 Si  $V[J] > V[J+1]$   $\implies V[1] > V[2] \implies 3 > 6$  (Falso)

Para J=2 Si  $V[J] > V[J+1]$   $\implies V[2] > V[3] \implies 6 > 1$  (Verdadero)

Aux =  $V[J] \implies \text{Aux} = 6$

$V[J] = V[J+1] \implies V[2] = 1$

$V[J+1] = \text{Aux} \implies V[3] = 6$

3	1	6	8	15	18
---	---	---	---	----	----

Ahora I = 5 y J va desde 1 hasta 1

Para J=1 Si  $V[J] > V[J+1]$   $\implies V[1] > V[1] \implies 3 > 1$  (Verdadero)

Aux =  $V[J] \implies \text{Aux} = 3$

$V[J] = V[J+1] \implies V[2] = 1$

$V[J+1] = \text{Aux} \implies V[3] = 3$

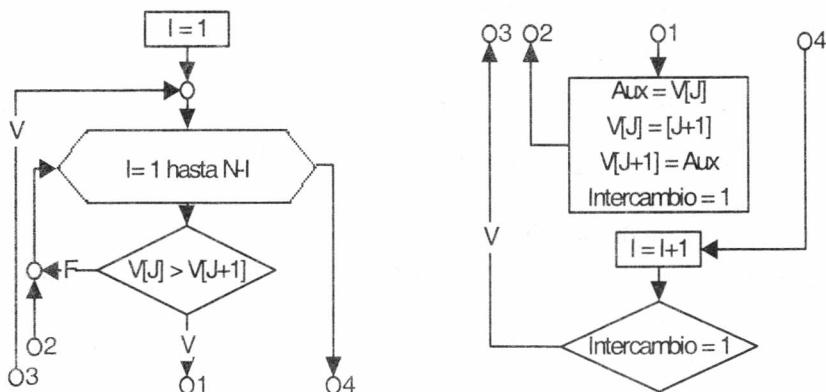
1	3	6	8	15	18
---	---	---	---	----	----

Al finalizar ambos bucles `Desde`, el vector ya queda ordenado en forma creciente.

## Método de ordenamiento de la burbuja mejorada

Como hemos visto anteriormente, el método de ordenación de la burbuja consiste en la comparación de elementos consecutivos y en donde encuentra dos elementos que cumplan la condición se hace el intercambio, eso quiere decir que si en una corrida no hace ningún intercambio significaría que el vector ya está ordenado, se podría controlar este evento mediante una variable booleana y según el cambio de ésta terminar el proceso de ordenamiento, esto se resume en ahorro de tiempo al momento de ordenar. El algoritmo para esta burbuja sería el siguiente:

1. Hacer  $I = 1$
2. Hacer
  - 2.1. Intercambio = 0
  - 2.2. Desde  $J = 1$  hasta  $N-I$  hacer
    - 2.2.1. Si  $V[J] > V[J+1]$  entonces
      - Aux =  $V[J]$
      - $V[J] = V[J+1]$
      - $V[J+1] = Aux$
      - Intercambio = 1
    - 2.2.2. Fin\_Si {Fin del paso 2.2.1}
  - 2.3. Fin\_Desde {Fin del paso 2.2}
  - 2.4. Hacer  $I = I + 1$
3. Mientras Intercambio = 1



Como podemos observar, estamos usando la variable **Intercambio** como una variable booleana, el bucle `Hacer..` Mientras se ejecutará mientras que esta variable

tenga el valor de 1, si esta variable llegara al final del bucle con un valor 0, significaría que no hay nada que comparar y fin del bucle.

### Codificación en Java

```
I = 1;  
Do {  
    Intercambio = 0;  
    for(int J=0;J<N-I;J++)  
        if (V[J]>V[J+1])  
        {  
            Aux = V[J];  
            V[J] = V[J+1];  
            V[J+1] = Aux;  
            Intercambio = 1;  
        }  
    I++;  
}while(Intercambio == 1);
```

### Método de ordenamiento por Inserción

Este método toma cada elemento del arreglo para ser ordenado y lo compara con los que se encuentran en posiciones anteriores a la de él dentro del arreglo. Si resulta que el elemento con el que se está comparando es mayor que el elemento a ordenar, se recorre hacia la siguiente posición superior. Si por el contrario, resulta que el elemento con el que se está comparando es menor que el elemento a ordenar, se detiene el proceso de comparación pues se encontró que el elemento ya está ordenado y se coloca en su posición (que es la siguiente a la del último número con el que se comparó).

El algoritmo en el cual se basa este método es el siguiente:

1. Desde I = 2 Hasta N hacer
  - 1.1. Hacer Aux = V[I]
    - j = I-1
    - SW = Verdadero
  - 1.2. Mientras (SW) Y (J>=1) Hacer
    - 1.2.1. Si Aux < V[j] Hacer
      - V[J+1] = V[J]
      - j = j -1
    - 1.2.2. Si no
      - SW = Falso
    - 1.1.3 Fin si

- 1.3. Fin\_Mientras
- 1.4. Hacer  $V[j+1] = Aux$
2. Fin\_desde

Nótese que en la línea 1.2 se debe cumplir ambas condiciones en caso contrario, se terminará el bucle Mientras. Por otro lado vemos que una de las condiciones solo es (SW), en este caso se debe asumir que es verdadero, también es válido si se pone (SW=Verdadero).

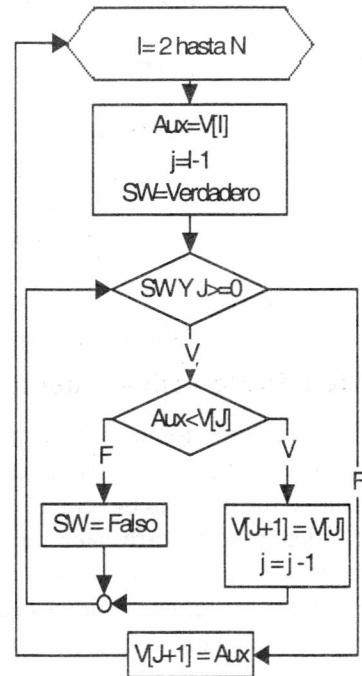
### Codificación en Java

```
boolean SW;
int J,Aux;
for(int I=1;I<N;I++) {
    Aux = V[I];
    J = I-1;
    SW = true;
    while (SW && J>=0) {
        if (Aux<V[J]){
            V[J+1] = V[J];
            J--;
        }else
            SW = false;
    }
    V[J+1] = Aux;
}
```

### Método de Selección

El método de ordenamiento por selección consiste en encontrar el menor de todos los elementos del arreglo e intercambiarlo con el que está en la primera posición. Luego el segundo más pequeño, y así sucesivamente hasta ordenar todo el arreglo. Su algoritmo es el siguiente:

```
// Buscamos la posición del mínimo en  $V[I]$ ,  $V[I+1]$ , ...,  $V[N-1]$ 
1. Desde i= 0 hasta N-1 Hacer
    1.1. Hacer K = I
    1.2. Desde J = I+1 Hasta N Hacer
        1.2.1. Si  $V[J] < V[K]$ 
            1.2.1.1. Hacer K = J
        1.2.2. Fin_Si
    1.3. Fin_Desde
// intercambiamos  $V[I]$  con  $V[K]$ 
```

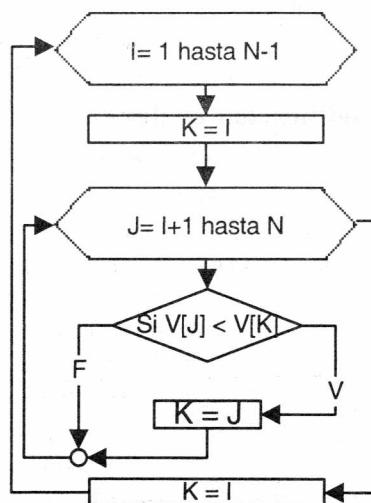


```
1.4. Hacer Aux = V[I]
      V[I] = V[K]
      V[K] = Aux
```

2. Fin\_Desde

### Codificación en Java

```
int K,Aux;
for(int I=0;I<N-1;I++){
    K = I;
    for (int J = I+1;J<N;J++)
        if(V[J]<V[K])
            K = J;
    Aux = V[I];
    V[I]= V[K];
    V[K]= Aux;
}
```



### Método de Shell

A diferencia del algoritmo de ordenación por inserción, este algoritmo intercambia elementos distantes. Es por esto que puede deshacer más de una inversión en cada intercambio, hecho del cual nos aprovechamos para ganar velocidad.

La velocidad del algoritmo dependerá de una secuencia de valores (llamados incrementos) con los cuales trabaja utilizándolos como distancias entre elementos a intercambiar.

Se considera la ordenación de Shell como el algoritmo más adecuado para ordenar entradas de datos moderadamente grandes. A continuación damos el algoritmo formal de la ordenación de Shell, en el cual utilizaremos los incrementos propuestos por Shell por simplicidad en el código:

### Algoritmo

```
Inc = n DIV 2
Hacer
  Desde i = Inc + 1 hasta n hacer
    tmp = v[i]
    J = i-Inc
    Mientras (J >= 0) Y (tmp < v[J])
      v[J+Inc] = v[J]
      J = J - Inc
    Fin_mientras
    v[J+Inc] = tmp
```

```

Fin_Desde
Inc = Inc DIV 2
Mientras (Inc<>0)

```

### Codificación en Java

```

int Inc,J,tmp;
Inc = N/2;
do{
    for(int I=Inc;I<N;I++)
    {
        tmp = V[I];
        J = I-Inc;
        while((J>=0)&&(tmp<V[J]))
        {
            V[J+Inc] = V[J];
            J-=Inc;
        }
        V[J+Inc] = tmp;
    }
    Inc/=2;
}while(Inc!=0);

```

### Búsqueda binaria en un vector

Esta es otra operación muy usada con los arreglos, y consiste en la búsqueda en un conjunto de datos de un elemento específico y la recuperación de alguna información asociada al mismo. La forma mas idónea de realizar una búsqueda es a través del método binario

Este método es muy útil en arreglos muy grandes, el único problema es que se requiere de un vector ordenado. Su algoritmo es el siguiente:

#### Algoritmo

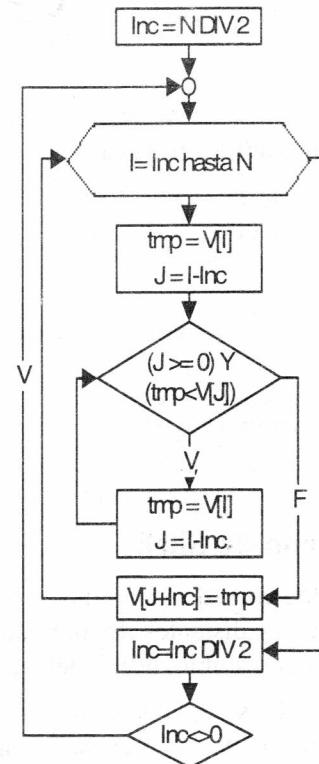
Se compara el dato buscado con el elemento central del vector

Si coinciden hemos encontrado el dato buscado

si el dato es mayor que el elemento central del vector, buscar el dato en la parte superior del vector

Por el contrario si el dato es menor, busque en la parte inferior del vector

La idea en que se basa el algoritmo es bien conocida: si el vector V está ordenado, la localización del valor X puede hacerse de manera que en cada paso se vaya



reduciendo el intervalo de búsqueda. Para ello, en lugar de comenzar por un extremo, se considera en primer lugar el valor central V [Cen], de modo que se tienen las tres posibilidades siguientes:

V [Cen] = Dato. Es decir, ya se ha encontrado el elemento buscado; no es por tanto necesario seguir buscando. Esto resulta bastante improbable, al menos en la primera iteración.

V [Cen] > Dato. Al estar el vector ordenado, que el punto medio quede por encima implica que X, si se encuentra en el vector, ha de estar en la mitad inferior. Por tanto, se restringe la búsqueda al intervalo [Inf; Cen -1].

V [med] < Dato. De modo análogo, puede ubicarse ahora al valor X en la mitad superior del vector. Por tanto, la búsqueda se restringe al intervalo [Cen + 1; Sup].

Así pues, comenzando con el intervalo inicial i..f , se aplica esta técnica de manera sucesiva; en cada paso cambian los límites del intervalo, que se irá haciendo cada vez más pequeño, hasta que el valor X se encuentra (o se determina que no está en el vector).

Dado que la variante mostrada está desestructurada, es fundamental lograr una versión estructurada. De hecho, no hay una, sino varias propuestas en este sentido: la propia importancia del algoritmo ha propiciado esta diversificación. En definitiva, todas las variantes parten de la misma idea, y son igualmente válidas, dado que las diferencias entre unas y otras son mínimas.

### **Pseudocódigo**

```
1. Hacer Inf = 1
    Sup = N
    NoEncontrado = Verdadero
2. Mientras (Inf <= Sup) Y (NoEncontrado) hacer
    2.1. Hacer Cen = (Inf + Sup) / 2
    2.2. Si V[Cen] = Dato entonces
        2.2.1. NoEncontrado = Falso
    2.3. Si no
        2.3.1. Si V[Cen] < Dato entonces
            2.3.1.1. Hacer Inf = Cen + 1
        2.3.2. Si no
            2.3.2.1. Hacer Sup = Cen - 1
        2.3.3. Fin_Si
    2.4. Fin_Si
3. Fin_Mientras
```

```
4. Si NoEncontrado hacer
    4.1. Hacer Escribir "Dato no encontrado"
5. Si no
    5.1. Hacer Escribir "Dato encontrado en posición", Cen
6. Fin_Si
```

En donde:

P, U, Cen : Variables de tipo entero, que indican la primera posición, la última posición y la posición central respectivamente

### Codificación en Java

```
int Inf = 0, Sup, Cen=0;
boolean NoEncontrado = true;
Sup = N-1;
while(Inf<=Sup && NoEncontrado) {
    Cen = (Inf + Sup) / 2;
    if(V[Cen]==Dato) NoEncontrado = false;
    else
        if(V[Cen]<Dato) Inf = Cen + 1;
        else Sup = Cen - 1;
}
if (NoEncontrado)
    R.setText("No encontrado");
else
    R.setText("Encontrado en posición:"+Cen);
```

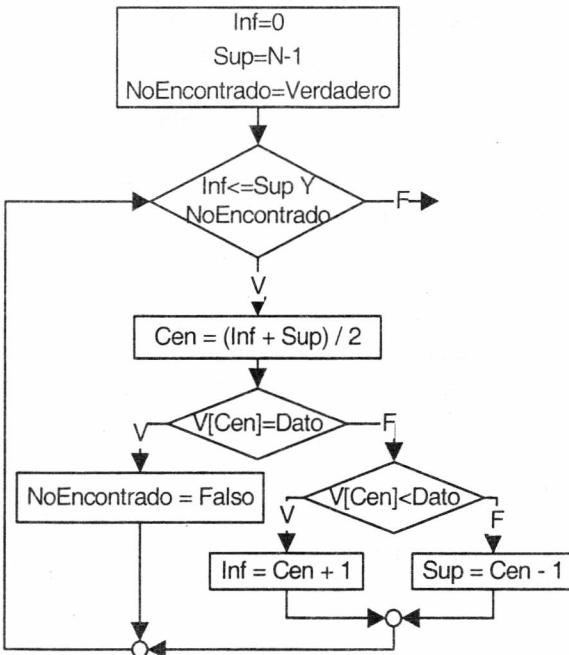
## Ejemplos desarrollados

### Ejemplo 4.3

Construya un programa que realice las siguientes acciones. Considere un vector ordenado en forma ascendente

- ❖ Genere un vector de 100 números aleatorios entre 0 y 100
- ❖ Dado un número ingresado por el usuario, de como resultado la posición de la primera ocurrencia.
- ❖ Dado un número ingresado ver cuantas veces se repite en el vector.
- ❖ Muestre el vector las veces que el usuario lo requiera.

### Solución



Como es obvio este ejercicio tienes que ser desarrollado por medio de un pequeño menú de opciones. La generación de un vector de número aleatorios es muy fácil y por lo general todos los programadores tienen comandos especiales q nos ayudan en esta tarea. Para el caso del Pseudocódigo y el diagrama de flujo solo quedará indicado como GenerarNúmero(), veremos en la programación el uso del comando Random() para la generación de número aleatorios

### Algoritmo

Construir procedimiento que genere números aleatorios

Construir procedimiento para escribir vector

Implementar función de búsqueda de un elemento en el vector.

Implementar función q cuente el número de repeticiones

Implementar procedimiento de número pares e impares

### Pseudocódigo

```

import java.awt.*;
import java.applet.*;
import java.awt.event.*;
  
```

```
import javax.swing.JOptionPane;
public class Ejemplo4_3 extends Applet implements
ActionListener{
    CheckboxGroup Menul=new CheckboxGroup();
    Checkbox G=new Checkbox("Generar",Menul,true);
    Checkbox M=new Checkbox("Mostrar",Menul,false);
    Checkbox B=new Checkbox("Búsqueda",Menul,false);
    Checkbox R=new Checkbox("Repeticiones",Menul,false);
    Label lbl=new Label("Operación con vectores");
    Label Resul=new Label();
    TextArea Vect=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");

    int V[]={};
    int Max;

    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (270, 200);
        lbl.setBounds(60,15,200,20);
        G.setBounds(10,40,70,20);
        M.setBounds(90,65,70,20);
        B.setBounds(180,40,90,20);
        R.setBounds(80,40,90,20);
        Vect.setBounds(10,90,220,50);
        Resul.setBounds(10,140,200,20);
        Ejecutar.setBounds(90,175,70,20);
        Ejecutar.addActionListener(this);
        add(G);add(M);
        add(B);add(R);
        add(lbl); add(Vect);
        add(Resul);
        add(Ejecutar);
    }

    public void actionPerformed(ActionEvent ac){
        if (Menul.getSelectedCheckbox() == G)
            Generar();
        else
            if (Menul.getSelectedCheckbox() == M)
```

```
        MostrarVector();
    else
        if (Menul.getSelectedCheckbox() == B)
            Busqueda();
        else
            if (Menul.getSelectedCheckbox() == R)
    Resul.setText("Número de repeticiones:" +Repeticiones());
}
void Generar(){
    Resul.setText("");
    Vect.setText("");
Max=Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
número máximo de
elementos","Vectores",JOptionPane.INFORMATION_MESSAGE));
    for(int i=0;i<Max;i++)
        V[i]=(int)(100*Math.random()+1);
    OrdenarVector();
    Resul.setText("Vector generado y ordenado");
}

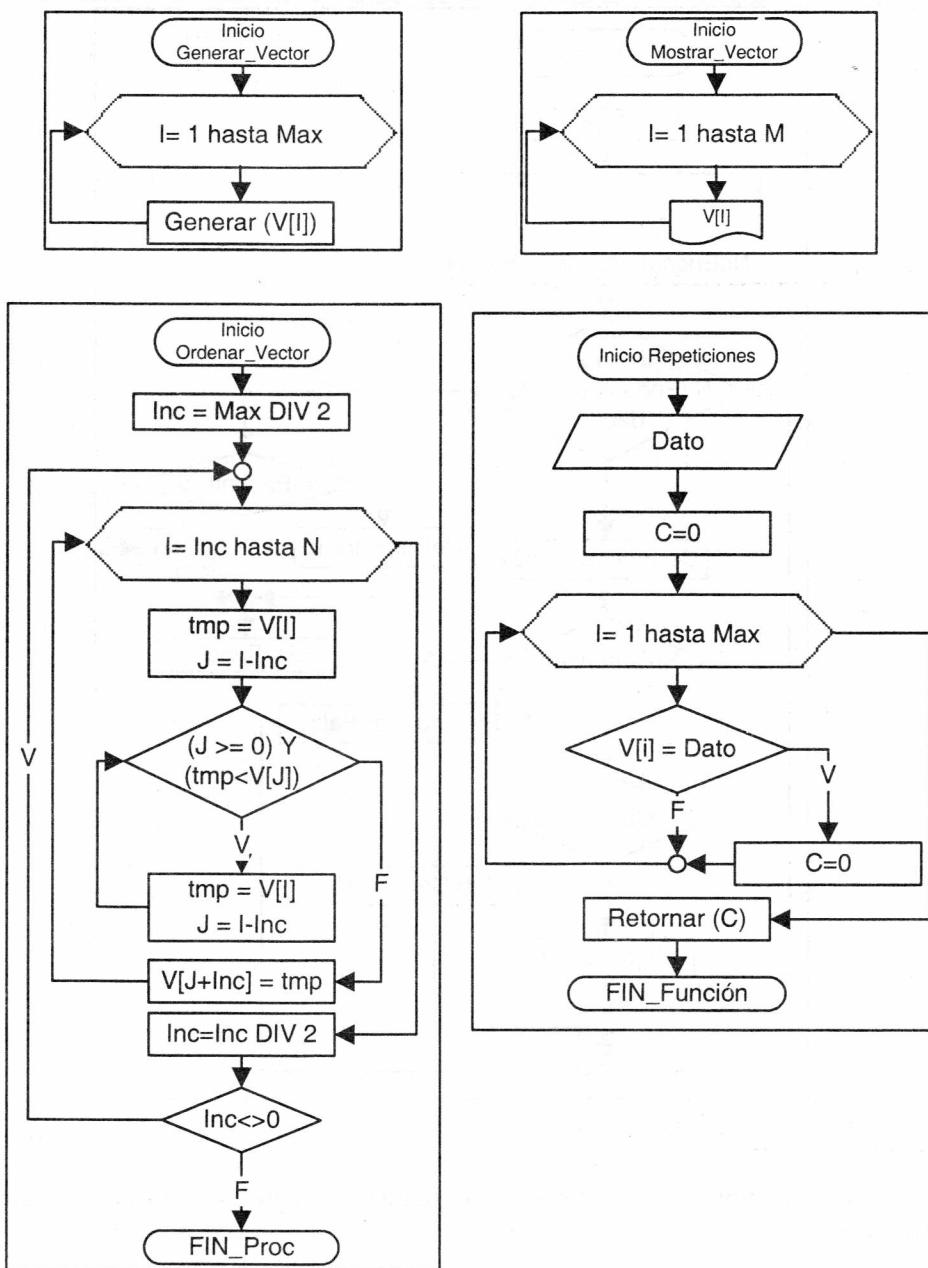
void OrdenarVector(){
    int Inc,tmp,j;
    Inc=Max/2;
    do{
        for(int i=Inc;i<Max;i++){
            tmp=V[i];
            j=i-Inc;
            while((j>=0)&&(tmp<V[j])){
                V[j+Inc]=V[j];
                j-=Inc;
            }
            V[j+Inc]=tmp;
        }
        Inc/=2;
    }while(Inc!=0);
}

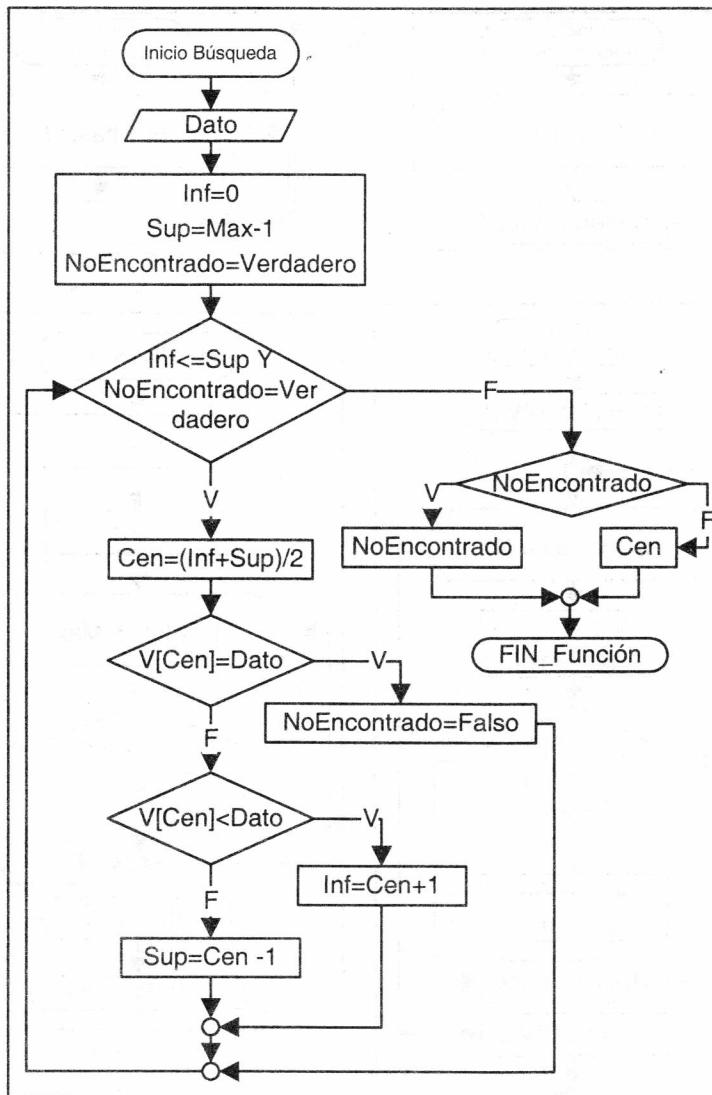
void Busqueda(){
    int Inf = 0,Sup,Cen=0,Dato;
    boolean NoEncontrado = true;
    Sup = Max-1;
```

```
Dato=Integer.parseInt(JOptionPane.showInputDialog(null, "Ingrese
número a
buscar", "Vectores", JOptionPane.INFORMATION_MESSAGE));
while(Inf<=Sup && NoEncontrado) {
    Cen = (Inf + Sup) / 2;
    if(V[Cen]==Dato)
        NoEncontrado = false;
    else
        if(V[Cen]<Dato)     Inf = Cen + 1;
        else Sup = Cen - 1;
}
if (NoEncontrado)Resul.setText("No encontrado");
else
    Resul.setText("Encontrado en posición:"+Cen);
}

void MostrarVector(){
    Resul.setText("");
    Vect.setText("");
    for(int i=0;i<Max;i++)
        Vect.append(V[i]+", ");
}

int Repeticiones (){
    int Dato,c=0;
Dato=Integer.parseInt(JOptionPane.showInputDialog(null, "Ingrese
número a buscar
repeticiones", "Vectores", JOptionPane.INFORMATION_MESSAGE));
    for(int i=0;i<Max;i++)
        if(V[i]==Dato) c++;
    return c;
}
}
```





#### Ejemplo 4.4

Construya un programa que realice las siguientes acciones. Considere una matriz cuadrada.

- ❖ Genere dos matrices del mismo orden, con valores entre 1 y 50
- ❖ Muestre las matrices a petición de usuario

- ❖ Sume ambas matrices
- ❖ Realice la multiplicación de las dos matrices
- ❖ Obtenga la determinante de cada matriz
- ❖ obtenga la matriz transpuesta de cada una de las matrices generadas.

### Algoritmo

Construir procedimiento que genere una matriz aleatoria

Construir procedimiento para mostrar la matriz

Implementar procedimiento para hallar la suma de las matrices

Implementar procedimiento que obtenga el producto de ambas matrices

Implementar procedimiento para hallar la determinante de una matriz

Implementar procedimiento para hallar la transpuesta de una matriz

### Pseudocódigo

```
import java.awt.*;
import java.applet.*;
import java.awt.event.*;
import javax.swing.JOptionPane;
public class Ejemplo4_4 extends Applet implements
ActionListener{
    CheckboxGroup Menul=new CheckboxGroup();
    Checkbox G=new Checkbox("Generar",Menul,true);
    Checkbox V=new Checkbox("Mostrar",Menul,false);
    Checkbox S=new Checkbox("Sumar Matriz",Menul,false);
    Checkbox P=new Checkbox("Prod. Matriz",Menul,false);
    Checkbox T=new Checkbox("Traspuesta",Menul,false);
    Checkbox D=new Checkbox("Determinante",Menul,false);
    Label lbl=new Label("Operación con Matrices");
    Label Resul=new Label();
    TextArea Mat=new TextArea("");
    Button Ejecutar=new Button("Ejecutar");
    int M[][]=new int[10][10];
    int M1[][]=new int[10][10];
    int Max;
    public void init() {
        setLayout(null);
        setBackground(java.awt.Color.white);
        resize (350, 360);
```

```
        lbl.setBounds(60,15,200,20);
        G.setBounds(10,40,70,20);
        V.setBounds(90,40,70,20);
        S.setBounds(180,40,90,20);
        P.setBounds(10,65,70,20);
        T.setBounds(90,65,70,20);
        D.setBounds(180,65,90,20);
        Mat.setBounds(10,90,270,200);
        Resul.setBounds(10,320,200,20);
        Ejecutar.setBounds(90,340,70,20);
        Ejecutar.addActionListener(this);
        add(G);add(V);add(S);add(P);add(T);add(D);
        add(lbl);add(Mat);add(Resul);add(Ejecutar);
    }
    public void actionPerformed(ActionEvent ac) {
        int OP;
        if (Menu1.getSelectedCheckbox() == G){
            Max=Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
número máximo de
elementos","Matrices",JOptionPane.INFORMATION_MESSAGE));
            Generar(M);Generar(M1);
        }else
            if (Menu1.getSelectedCheckbox() == V){
                OP=Integer.parseInt(JOptionPane.showInputDialog(null,"Matriz1[1]
o Matriz2[2]","Mostrar
Matriz",JOptionPane.INFORMATION_MESSAGE));
                if (OP==1) MostrarMatriz(M);
                else if (OP==2) MostrarMatriz(M1);
            }else
                if (Menu1.getSelectedCheckbox() == S)
                    MatrizSuma(M,M1);
                else
                    if (Menu1.getSelectedCheckbox() == P)
                        MatrizProducto(M,M1);
                else
                    if (Menu1.getSelectedCheckbox() == T){
                        OP=Integer.parseInt(JOptionPane.showInputDialog(null,"Matriz1[1]
o Matriz2[2]","Matriz
Traspuesta",JOptionPane.INFORMATION_MESSAGE));
                        if (OP==1) TransponerMatriz(M);
```

```
        else if (OP==2) TransponerMatriz(M1);
        Resul.setText("Tranpuesta de matriz "+OP);
    }else
        if (Menu1.getSelectedCheckbox() == D){
    OP=Integer.parseInt(JOptionPane.showInputDialog(null,"Matriz1[1]
o
Matriz2[2])","Determinate",JOptionPane.INFORMATION_MESSAGE));
        if (OP==1) Determinante(M);
        else if (OP==2) Determinante(M1);
    }
}
void Generar(int M[][]){
    Resul.setText("");
    Mat.setText("");
    for(int i=0;i<Max;i++)
        for(int j=0;j<Max;j++)
            M[i][j]=(int)(50*Math.random()+1);
    Resul.setText("Matriz generada");
}
void TransponerMatriz (int M[][]){
    int i,j, aux;
    for (i=0; i< Max; i++)
        for (j = i; j < Max; j++){
            aux = M[i][j];
            M[i][j] = M[j][i];
            M[j][i] = aux;
        }
    MostrarMatriz(M);
}
void MatrizProducto(int M[][], int M1[][]){
    int Pro[][]=new int[Max][Max];
    for(int i=0; i<Max; i++){
        for(int j=0; j<Max; j++){
            for(int k=0; k<Max; k++){
                Pro[i][j]+=M[i][k]*M1[k][j];
            }
        }
    }
    MostrarMatriz(Pro);
}
```

```
void Determinante(int M[][]){
    int MAux[][] =new int[Max] [Max];
    for(int i=0; i<Max; i++)
        for(int j=0; j<Max; j++)
            MAux[i][j]=M[i][j];
    for(int k=0; k<Max-1; k++){
        for(int i=k+1; i<Max; i++){
            for(int j=k+1; j<Max; j++){
                MAux[i][j]-=MAux[i][k]*MAux[k][j]/
                    MAux[k][k];
            }
        }
    }
    double deter=1.0;
    for(int i=0; i<Max; i++){
        deter*=MAux[i][i];
    }
    MostrarMatriz(M);
    Resul.setText("La determinante es: "+deter);
}
void MatrizSuma(int M[][], int M1[][]){
    int MAux[][] =new int[Max] [Max];
    for(int i=0; i<Max; i++){
        for(int j=0; j<Max; j++){
            MAux[i][j]=M[i][j]+M1[i][j];
        }
    }
    MostrarMatriz(MAux);
}
void MostrarMatriz(int MM[][]){
    Resul.setText("");
    Mat.setText("");
    for(int i=0;i<Max;i++){
        for(int j=0;j<Max;j++){
            Mat.append(String.valueOf(MM[i][j]));
            LlenarEspacio(MM[i][j]);
        }
        Mat.append("\n");
    }
}
```

```
void LlenarEspacio(int x){  
    if (x>10) Mat.append("      ");  
    else    Mat.append("    ");  
}  
}
```

### Ejemplo 4.5

Construya un programa que sea capaz de manipular una array de cadenas y realice las siguientes acciones.

- ❖Lea el array de cadenas
- ❖Muestre el array de cadenas
- ❖Halle el número de repeticiones de una letra en cada una de las cadenas ingresadas.
- ❖Ordene la cadena

### Pseudocódigo

```
import java.awt.*;  
import java.applet.*;  
import java.awt.event.*;  
import javax.swing.JOptionPane;  
import java.lang.String;  
public class Ejemplo4_5 extends Applet implements  
ActionListener{  
    CheckboxGroup Menul=new CheckboxGroup();  
    Checkbox I=new Checkbox("Ingresar",Menul,true);  
    Checkbox M=new Checkbox("Mostrar",Menul,false);  
    Checkbox O=new Checkbox("Ocurrencias",Menul,false);  
    Checkbox R=new Checkbox("Ordenar",Menul,false);  
    Label lbl=new Label("Manejo con cadenas");  
    TextArea CAD=new TextArea("");  
    Button Ejecutar=new Button("Ejecutar");  
    String[] AC = new String[20];  
    int Max;  
    public void init() {  
        setLayout(null);  
        setBackground(java.awt.Color.white);  
        resize (350, 360);  
        lbl.setBounds(60,15,200,20);
```

```
I.setBounds(50,40,70,20);
M.setBounds(150,40,70,20);
O.setBounds(50,65,90,20);
R.setBounds(150,65,90,20);
CAD.setBounds(10,90,270,200);
Ejecutar.setBounds(90,340,70,20);
Ejecutar.addActionListener(this);
add(I);add(M);add(O);
add(lbl);add(CAD);add(R);
add(Ejecutar);
}
public void actionPerformed(ActionEvent ac){
    int OP;
    if (Menul.getSelectedCheckbox() == I){
        Max=Integer.parseInt(JOptionPane.showInputDialog(null,"Ingrese
número de cadenas","Operaciones con
cadenas",JOptionPane.INFORMATION_MESSAGE));
        Leer();
    }
    else
        if (Menul.getSelectedCheckbox() == M) Mostrar();
        else
            if (Menul.getSelectedCheckbox() == O)
                Ocurrencias();
            else
                if (Menul.getSelectedCheckbox() == R)
                    BurbujaM();
    }
    void Leer(){
        CAD.setText("");
        for(int i=0;i<Max;i++)
            AC[i]=JOptionPane.showInputDialog(null,"Ingrese
palabra
"+(i+1),"Cadenas",JOptionPane.INFORMATION_MESSAGE);
    }
    void Mostrar(){
        CAD.setText("");
        for(int i=0;i<Max;i++)
            CAD.append(AC[i]+"\n");
    }
    void BurbujaM(){
```

```
int I = 0, Intercambio;
String Aux;
do{
    Intercambio = 0;
    for(int J=I+1;J<Max;J++)
        if (AC[I].compareTo(AC[J]) > 0) {
            Aux = AC[J];
            AC[J] = AC[I];
            AC[I] = Aux;
            Intercambio = 1;
        }
    I++;
}while(Intercambio == 1);
Mostrar();
}
void Ocurrencias(){
    int x;
    CAD.setText(" ");
    String
C= JOptionPane.showInputDialog(null,"Ingrese letra a
buscar ","Cadenas",JOptionPane.INFORMATION_MESSAGE);
    for(int i=0; i<Max; i++){
x=0;
CAD.append(AC[i]+"  ");
        for(int j=0; j<AC[i].length(); j++)
if (AC[i].charAt(j)==C.charAt(0)) x++;
        CAD.append(" ("+x+") \n");
    }
}
}
```

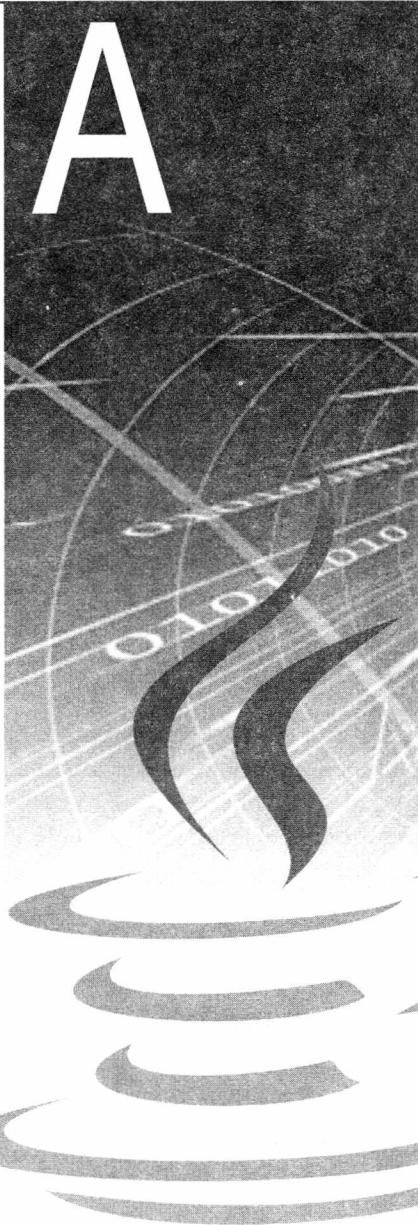


# Apéndice

## Uso de funciones para desarrollar programas

Este apéndice contiene:

- ➔ Funciones
- ➔ Procedimientos
- ➔ Semejanzas y diferencias entre funciones y procedimientos
- ➔ Ámbito de las variables
- ➔ Paso de parámetros
- ➔ Funciones y Procedimientos como parámetros





## **Introducción**

La resolución de problemas complejos se facilita considerablemente si se divide en pequeños problemas (subproblemas).

La solución de estos subproblemas se realiza con **subprogramas**. Su uso permite al programador desarrollar programas más complejos utilizando un método de diseño **descendente**. Se denomina descendente ya que se inicia en la parte superior con un problema general y luego abajo el diseño específico de la soluciones en los subproblemas. Normalmente las partes en que se divide un programa deben poder desarrollarse independientemente entre sí, planteando los requerimientos de variables, el cuerpo del programa principal y luego el código de los subprogramas.

El subprograma, por ser un algoritmo, debe cumplir con las misma características de éste y hacer tareas similares como aceptar datos, escribir dato y hacer cálculos; sin embargo, es utilizado para un propósito específico. El subprograma recibe datos del algoritmo o subalgoritmo que lo invoca y éste le devuelve resultados.

## **Funciones**

Una función va a ser un subprograma que nos devuelve un dato de tipo estándar es un subprograma que puede tomar uno o varios parámetros como entrada ;

devuelve a la salida un único resultado. Su sintaxis es la siguiente:

### Sintaxis:

```
Función nomrefunción (P1: tipo, P2: tipo, ...): Tipo  
Declaraciones locales  
Inicio  
    Cuerpo de la función  
    Retornar (valor)  
Fin_Función
```

Donde P1, P2,... son parámetros, los parámetros son la información que se le tiene que pasar a la función. Los parámetros luego dentro de la función los podemos utilizar igual que si fueran variables locales definidas en la función y para cada parámetro hay que poner su nombre y tipo.

El nombre de la función lo da el usuario y tiene que ser significativo.

En las variables locales se declaran las variables que se pueden usar dentro de la función.

La línea:

```
    retornar valor
```

indica que la función devuelve el valor del proceso final por lo tanto se debe de colocar cuando finalice la función, por otro lado **valor** debe ser del mismo tipo de la función.

Una función es llamada por medio de su nombre, en una sentencia de asignación o en una sentencia de salida.

Se puede llamar a una función en cualquiera de las siguientes formas:

```
nomrefunción o nomrefuncion(x)
```

La primera es cuando no se necesita ningún parámetro y en la segunda obligatoriamente se necesita un parámetro.

Como las funciones devuelven un valor específico la forma más usual de utilizarlas es por medio de asignaciones de una variable a la función. Por ejemplo:

```
    idVar = nomrefunción
```

No se permiten funciones que no devuelvan nada.

### Ejemplo de función

Una función que calcule el cuadrado de un valor que le pasa parámetro. Suponemos que es un valor entero.

```
Función Cuadrado (n: entero): real
```

```
Declarar variables
    C : real
Inicio
    C = n*n
    Retornar (C)
Fin_Función_Cuadrado
```

## Procedimientos

Un procedimiento es un subprograma o un subalgoritmo que ejecuta una determinada tarea, pero que tras ejecutar esa tarea no tienen ningún valor asociado a su nombre como en las funciones, sino que si devuelve información, lo hace a través de parámetros.

Al llamar a un procedimiento, se le cede el control, comienza a ejecutarse y cuando termina devuelve el control a la siguiente instrucción a la de llamada. Su sintaxis es la siguiente:

### Sintaxis:

```
Procedimiento Nombre_Proc (P1: tipo, P2: tipo, ...)
    Declaraciones locales
    Inicio
        Cuerpo del procedimiento
    Fin_Procedimiento
```

La cabecera va a estar formada por el nombre del procedimiento que será un identificador y que debe de ser significativo, y luego entre paréntesis los parámetros o la información que se le pasa al procedimiento. Para cada parámetro hay que indicar el tipo de paso de parámetro. Veamos un ejemplo de procedimientos.

```
Procedimiento Cuadrado (num:entero,M:real)
    Inicio
        M = num/2
    Fin_Procedimiento_Cuadrado
```

## Ambito de las variables

La parte del programa principal o función en que una variable se define y se puede utilizar o alterar su contenido se conoce como **Ambito**.

Las variables utilizadas en los programas principales y subprogramas se clasifican en dos tipos:

- ❖ variables locales
- ❖ variables globales

## Variable local

Una **variable local** es aquella que está declarada y definida dentro de un subprograma, en el sentido de que está dentro de ese subprograma y es distinta de las variables con el mismo nombre declaradas en cualquier parte del programa principal. El significado o visibilidad de una variable se confina a la función en la que está declarada.

Esto quiere decir que la variable no tiene ningún significado, no se conoce y no se puede acceder a ella desde fuera del subprograma y que tiene una posición de memoria distinta a la de cualquier otra, incluso si es de una variable que tiene el mismo nombre pero que está definida fuera del subprograma.

Las variables locales a un subprograma se definen en la parte de la definición de variables del mismo. Los parámetros formales que se le ponen a un subprograma se comportan dentro de él como si fueran también variables locales a él.

Las variables locales al finalizar la función o el procedimiento, desaparecen de la memoria. Si dos variables, una global y una local, tienen el mismo nombre, la local prevalecerá sobre la global dentro del módulo en que ha sido declarada. Dos variables locales pueden tener el mismo nombre siempre que estén declaradas en funciones o procedimientos diferentes.

## Variable global

Una **variable global** es aquella que está declarada para el programa principal.

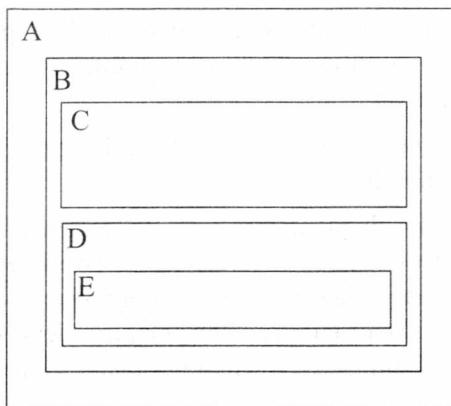
A esta variable podemos acceder desde cualquiera de los subprogramas y el programa principal, salvo que alguno de esos subprogramas tenga definida una variable local con el mismo nombre que la variable global, en este caso si utilizo el nombre de esa variable me referiré a la local, nunca a la global (ya que tienen 2 zonas de memoria distintas).

Las variables permanecen activas durante todo el programa. Se crean al iniciarse éste y se destruyen de la memoria al finalizar.

Las variables globales tienen la ventaja de compartir información de diferentes subprogramas sin una correspondiente entrada en la lista de parámetros.

La figura siguiente muestra un esquema de un programa con diferentes subprogramas con variables locales y otras globales, aquí se muestra el ámbito de cada definición.

En donde:



Variables definidas en:

- A
- B
- C
- D
- E

Son accesibles desde:

- A, B, C, D, E
- B, C, D, E
- C
- D, E
- E

## Paso de parámetros

Existen diferentes métodos para la transmisión o *paso de los parámetros* a subprogramas. Es preciso conocer el método adoptado por cada lenguaje, ya que no siempre son los mismos. Dicho de otro modo, un mismo programa puede producir diferentes resultados bajo diferentes **sistemas de paso de parámetros**.

Cuando llamamos a una función o procedimiento, le pasamos a través de los parámetros la información que necesita, y en el caso de un procedimiento también devolvemos a través de sus parámetros los resultados. Para ello definiremos el tipo del parámetro a principio del subprograma, que es lo que conocemos como parámetros formales, y al hacer la llamada pasamos la información a través de los parámetros reales.

Los parámetros pueden ser clasificados como:

- ❖ Entradas: Solo proporcionan valores desde el programa que los llama y que se utilizan dentro del subprograma. En el caso de una función, todos sus

parámetros son de este tipo.

Como solo sirven como entrada, solo pueden ser leídos, pero no modificados, y aunque se modifiquesen dentro de un subprograma, fuera no va a tener efecto esa modificación.

- ❖ Salidas: las salidas producen los resultados del subprograma, este devuelve un valor calculado por dicha función.
- ❖ Entradas/Salidas: un solo parámetro se utiliza para mandar argumentos a un programa y para devolver resultados. El valor del parámetro tiene importancia tanto a la entrada como a la salida del subprograma, nos aporta información cuando llamamos al subprograma y por otra parte devolvemos a través de él resultados cuando terminamos el subprograma, en este caso, tiene sentido tanto leer como actualizar el parámetro.

Teniendo en cuenta los tipos de parámetros, los métodos más empleados para realizar el paso de parámetros son:

### Paso de parámetros por valor

Son los parámetros que pueden recibir valores pero que no pueden devolverlos. Es una variable global que se conecta con una variable local mediante el envío de su valor, después de lo cual ya no hay relación. Lo que le sucede a la variable local no afectará a la global. Cuando un parámetro actual se pasa por valor, el subprograma hace una copia del valor de éste en una posición de memoria idéntica en tamaño pero distinta en ubicación a la del parámetro actual y la asigna al parámetro formal correspondiente. Como el subprograma trabaja a partir de sus parámetros formales, si durante la ejecución se modifica el valor de un parámetro formal correspondiente a un paso por valor, el contenido de la posición de memoria del parámetro actual no se verá alterado.

A continuación mostramos el mecanismo de paso por valor de un procedimiento con dos parámetros.

1. A = 3
2. B = 9
3. Llamar a procedimiento Pro1 (A, 4\*B)

Procedimiento Pro1 (X, Y)

Aquí podemos ver que se realiza una **copia** de los valores de los argumentos en la llamada al procedimiento en las variables X e Y , pero en ningún caso los

valores de A y B se verán modificados.

Por lo tanto X toma el valor de A es decir X = 3, y Y toma el valor de la expresión 4\*B, es decir Y=36

Por ejemplo:

Función F (M : Entero) : Entero

Declarar variables

X : entero

Inicio

X = M

X = X \* X

M = X

Retornar (M)

Fin\_Función\_F

Programa principal

Inicio

1. Declarar variables

X, Y : Entero

2. Leer(X)

3. X = X + 1

4. Y = F(X)

5. Escribe (X,Y)

Fin\_Programa\_Principal

Si en la ejecución de este programa damos a X el valor 2, la siguiente línea lo incrementa en 1 (línea 3) y llamamos a la función con f(3). En la llamada a la función se hace una **copia** del valor de la variable **X** del programa principal en la variable **M** de la función. Una vez dentro de la función se declara otra variable **X** que es distinta a la del programa principal, siendo después asignada al valor de M, luego hacemos el cuadrado, lo asignamos a M y retornamos el valor.

Aquí finaliza la llamada a la función. En el programa principal escribimos los valores de X e Y, y dando como resultado X=3 e Y=9. Es muy importante comprender este mecanismo.

### **Paso por referencia (dirección) o variable**

Son los que pueden recibir y devolver valores. Son variables globales que se conectan con una local a través de su contenido; al establecerse dicha conexión las variables se convierten en sinónimos, lo que afecte a la variable local le sucederá a la variable global.

Se utiliza el paso por variable cuando el subprograma debe modificar el contenido de una variable del programa principal o devolver algún valor más, recordemos que una función solo devuelve un valor directamente.

Por ejemplo si la función anterior la modificamos así:

```
Función F (M : Entero) : Entero
    Declarar variables
        X : entero
    Inicio
        X = M
        X = X * X
        M = X
        Retornar (2X)
    Fin_Función_F

    Programa principal
    Inicio
        1. Declarar variables
            X, Y : Entero
        2. Leer (X)
        3. X = X + 1
        4. Y = F(X)
        5. Escribe (X,Y)
    Fin_Programa_Principal
```

Si llegamos a la llamada con  $f(3)$ , al ser un paso por variable o dirección, significa que la variable **M** de la función  $f$  tiene la dirección de memoria de la variable **X** lo que significa que cualquier cambio del valor de **M** se verá reflejado en **X**. Por tanto dentro de la función hacemos  $X=3$ ,  $X$  al cuadrado (9),  $M=9$  y devolvemos  $2x$ , con lo que en la programación principal escribiremos  $X=9$  e  $Y=18$ .

## Funciones y Procedimientos como parámetros

En la mayor parte de los lenguajes se permite también que una función o procedimiento pueda ser pasado como parámetro de otro subprograma. En este caso, es decir, cuando el parámetro formal es de tipo función o procedimiento, pasaremos como parámetro real funciones o procedimientos que tengan la misma definición que el que hemos puesto como parámetro formal, y en nuestro caso para indicar que se pasa como parámetro real una función o procedimiento.

# Apéndice

## Uso básico de

### JCreator

Este apéndice contiene:

El modo básico de como crear un applet de java mediante el uso de JCreator, de esa manera podrá realizar y compilar los ejemplo aqui descritos.

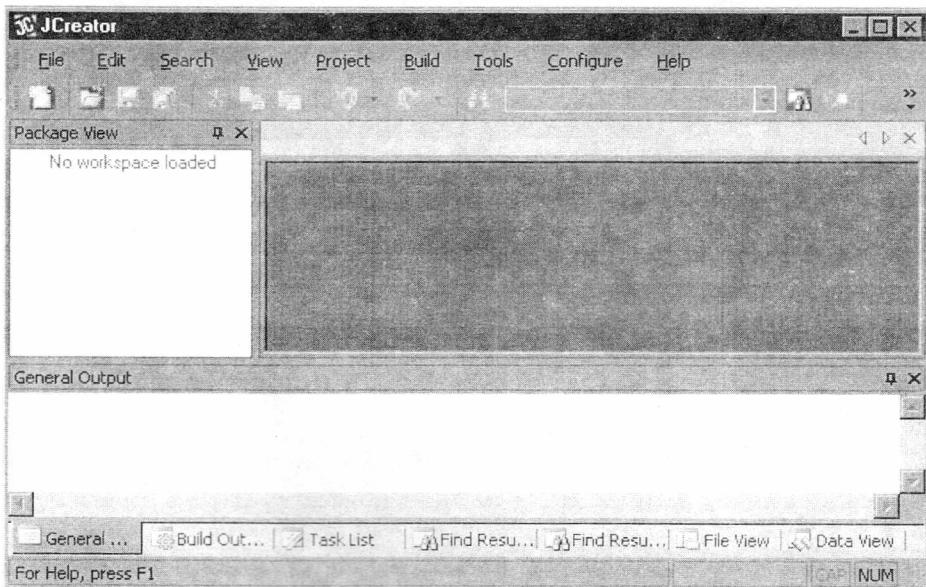




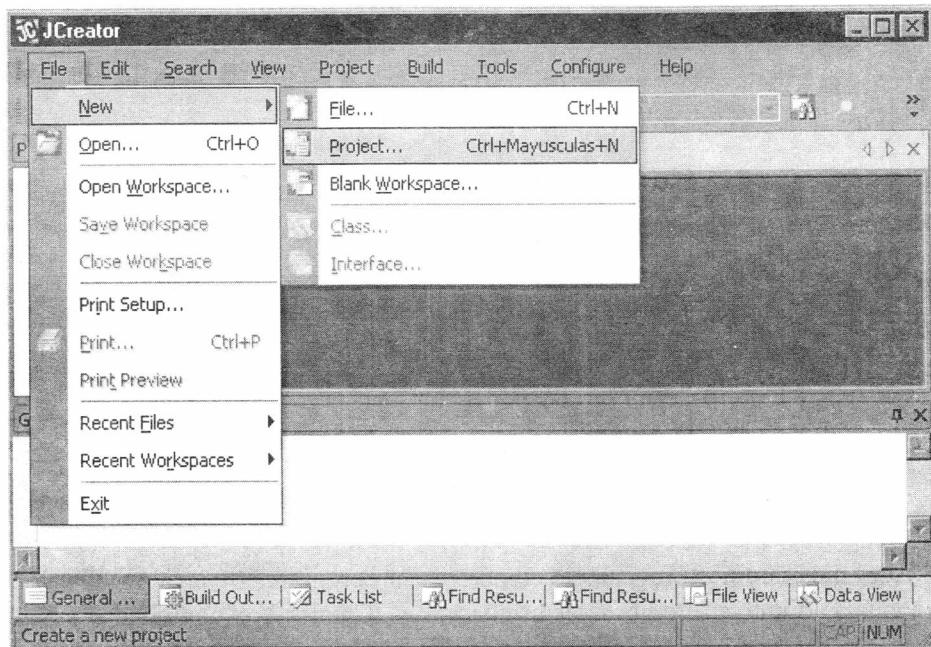


Aquí mostraremos brevemente como crear un programa de java en JCreator, en el CD adjunto a este libro, les brindamos el JCreator Y el compilador 1.5 de Java, de tal forma que pueda compilar y revisar todos los ejercicios mostrados en el desarrollo de libro.

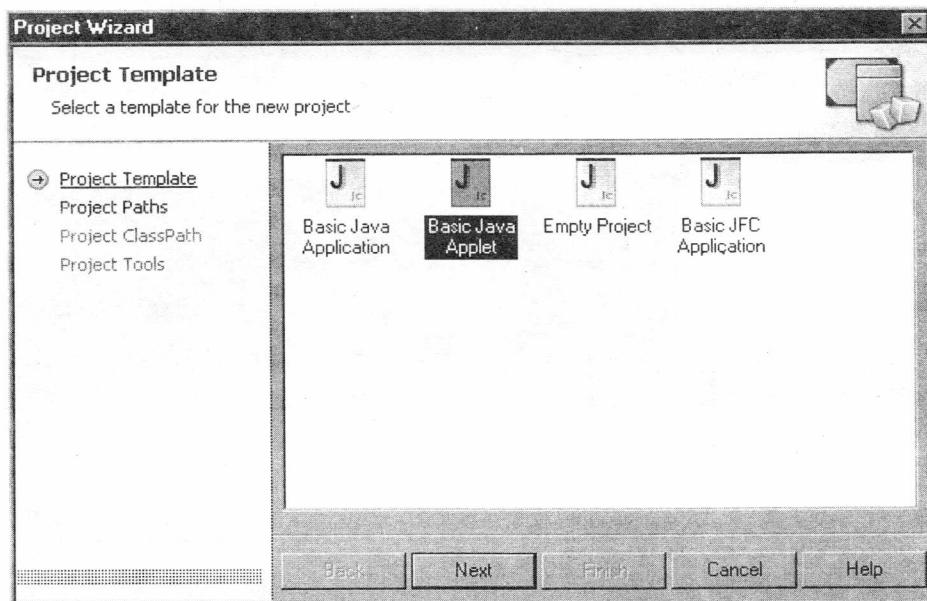
Tras ejecutar el JCreator obtendrá una pantalla similar a la siguiente



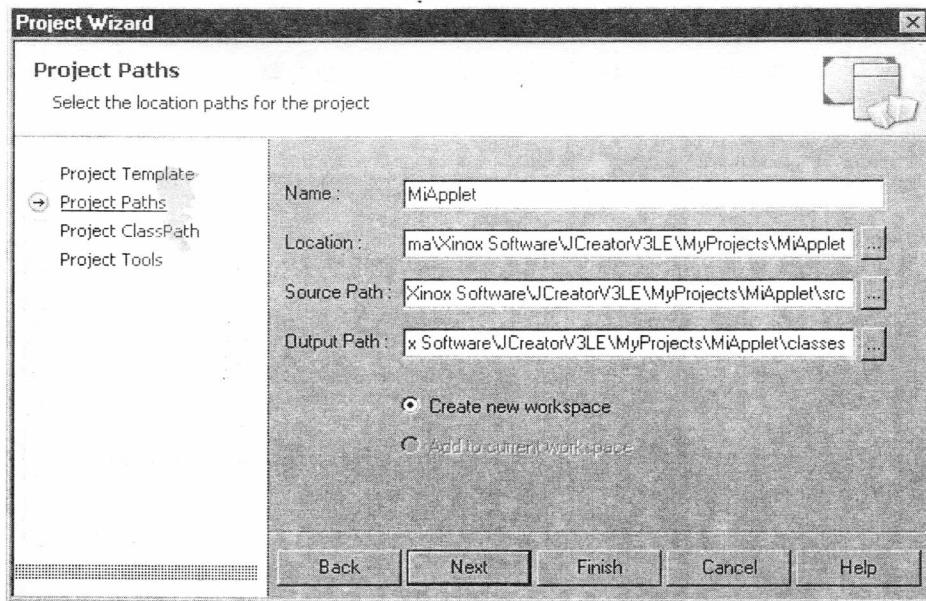
Desde el menú File seleccione la opción New y luego la opción Project.



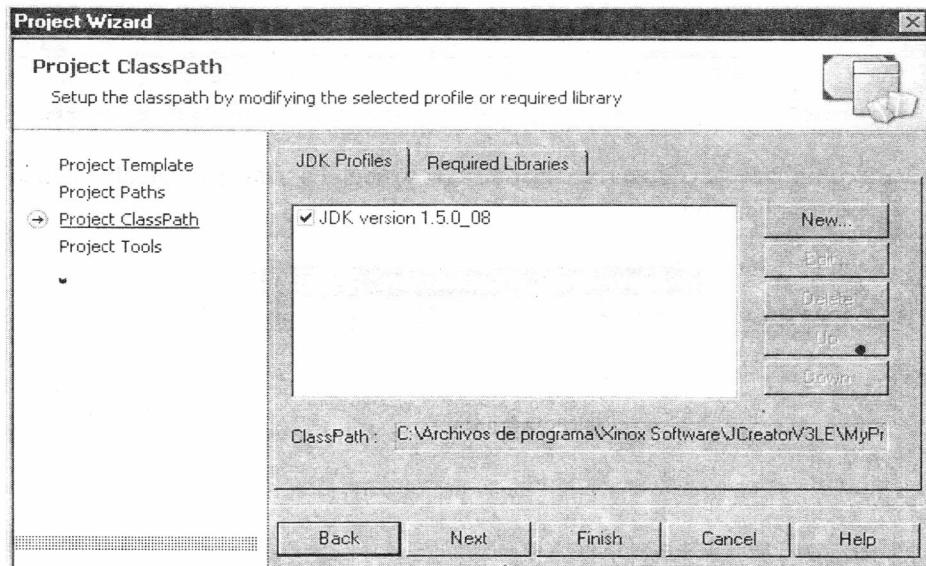
Luego seleccione la opción Basic Java Applet y pulse el botón Next



Luego coloque el nombre del applet y elija la ubicación donde este se almacenará y compilara. Pulse el botón Next.



Ahora vemos la versión del compilador que estamos usando, en nuestro caso la versión 1.5. Pulse el botón Finish.



Se nos mostrará el código inicial generado por java las cuales contiene init() y paint() con un pequeño mensaje inicial.

The screenshot shows the JCreator IDE interface. The title bar reads "MiApplet - JCreator - [MiApplet.java]". The menu bar includes File, Edit, Search, View, Project, Build, Tools, Configure, Window, Help. The toolbar has icons for New, Open, Save, Run, Stop, and Exit. The left pane is the "Package View" showing a package named "MiApplet" with a "MiApplet" class containing "init ()" and "paint (Graphics)". The right pane is the code editor for "MiApplet.java":

```
1  /**
2  * @(#)MiApplet.java
3  *
4  * Sample Applet application
5  *
6  * @author
7  * @version 1.00 06/09/27
8  */
9
10 import java.awt.*;
11 import java.applet.*;
12
13 public class MiApplet extends Applet {
14
15     public void init() {
16
17
18     public void paint(Graphics g) {
19         g.drawString("Welcome to Java!!", 50,
20
21 }
```

Below the code editor is a "General Output" window. At the bottom of the interface are tabs for General, Build Out..., Task List, Find Resu..., Find Resu..., File View, Data View, and a numeric keypad.

Para compilar este código reciente pulse F7 y luego F5 para ejecutar, obteniendo la siguiente aplicación.



# FREELIBROS.ORG

G R U P O  
EDITORIAL



Megabyte

Esta impresión se realizó  
en los talleres gráficos de  
Grupo editorial Megabyte  
LIMA - PERÚ

# ALGORITMOS Y DIAGRAMAS DE FLUJO APLICADOS EN



Este libro esta orientado a las personas que empiezan en el mundo de la programación. Los algoritmos son la base de todo sistema de cómputo, por lo que se ha querido brindarle los conceptos básicos apoyados en una solución de cómputo, motivo por el cual todos los ejemplos están desarrollados en Java 2.

Al final del desarrollo del libro el lector estará en capacidad de diseñar algoritmos mediante ejercicios prácticos, con el objetivo de adquirir una mentalidad de programador.

Ya hemos mencionado que para el desarrollo de los algoritmos usaremos el Java 2, el cual es un lenguaje de programación orientado a objetos, además de ser versátil y muy potente. En java podemos realizar tanto aplicaciones, es decir programas para PC, como applet los cuales nos permite realizar tareas desde un browser como Internet Explorer o FireFox, entre otros.

El realizar un algoritmo y luego codificarlo en un lenguaje de programación, en nuestro caso Java 2, no es más que una simple traducción y en algunos casos una adaptación del algoritmo al lenguaje específico. Mientras que el algoritmo es mas parecido al lenguaje normal que diariamente hablamos, un programa ya codificado procesa y ejecuta las ideas plasmadas en el algoritmo.

Estoy seguro que este libro les ayuda en el ingreso al mundo de la programación, el cual, si bien es cierto, es muy grande y complejo, teniendo muy en claro los conceptos básicos, podremos resolver cualquier problema sin importar su complejidad.

Básico



Intermedio

Avanzado

Colección: Algoritmia Megabyte



**Megabyte®**  
GRUPO EDITORIAL



Jr. Rufino Torrico 889- OF. 208- Cercado de Lima  
[www.grupomegabyte.com](http://www.grupomegabyte.com) [www.editorialmegabyte.com](http://www.editorialmegabyte.com)

LIMA-PERÚ

Telf.: 332-4110 Nextel.: 407\*4515