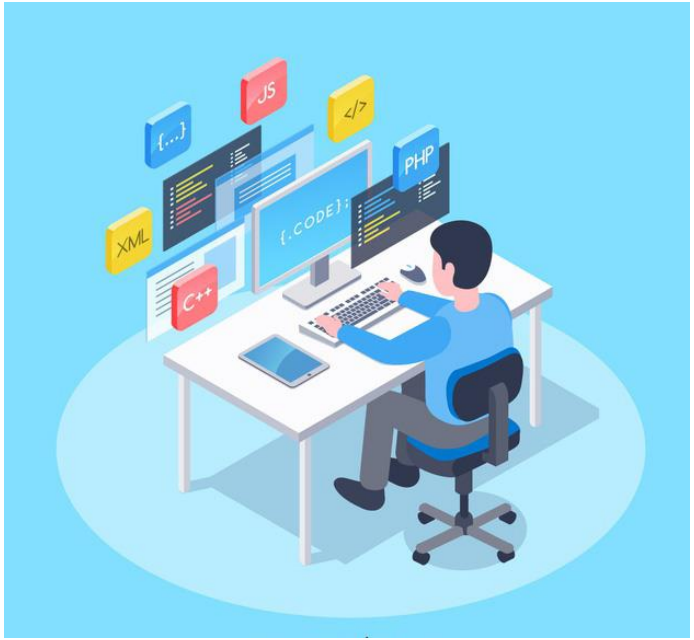
# INTRODUCCIÓN A LA *programación*

polotic
misiones

# INDUSTRIA 4.0

La Cuarta Revolución Industrial (HOY), que ya está entre nosotros, se asocia con la informatización y digitalización de la producción, y con la generación, integración y análisis de una gran cantidad de datos a lo largo del proceso productivo y del ciclo de vida de los productos, facilitados fundamentalmente por Internet.

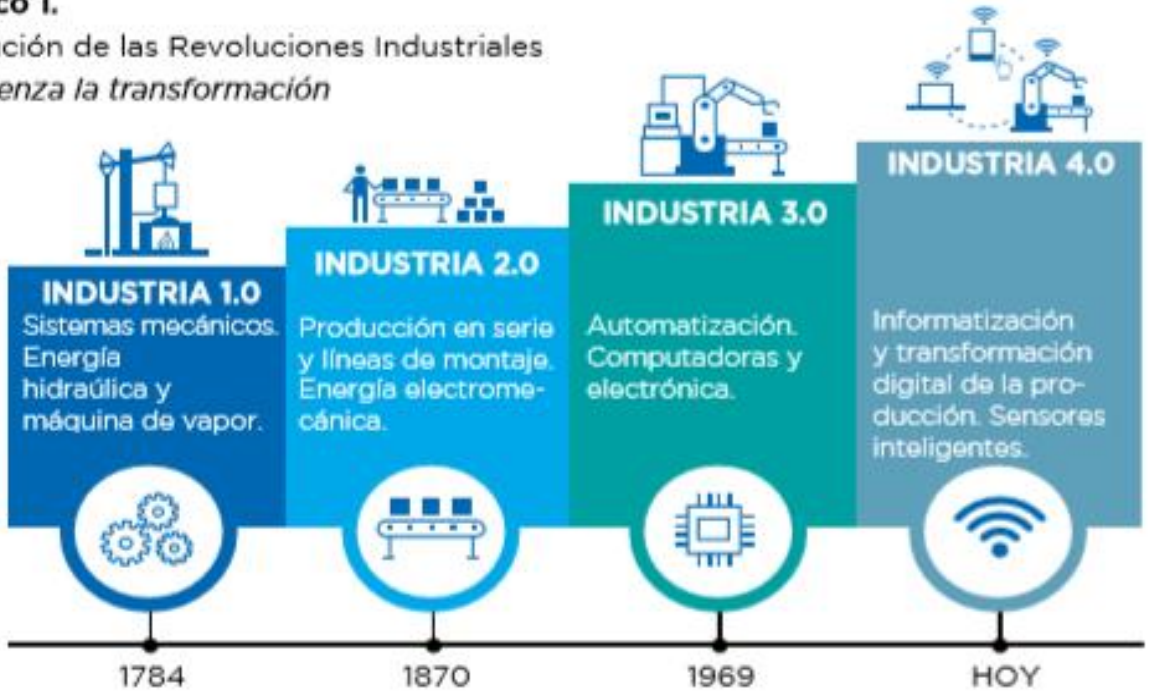


# REVOLUCION INDUSTRIAL

**Gráfico 1.**

Evolución de las Revoluciones Industriales

*Comienza la transformación*



# PILARES 4.0

**Gráfico 2.**
Pilares Tecnológicos de la Industria 4.0.

# Como estamos en Misiones?

polotic misiones

marandu COMUNICACIONES

DATA CENTER

DESARROLLO GEEK

CONECTIVIDAD FIBER

# Como estamos en Misiones?

Silicon Misiones EMPRENDEDORES

polotic misiones

Silicon Misiones AGRO TECNOLOGÍA

Silicon Misiones

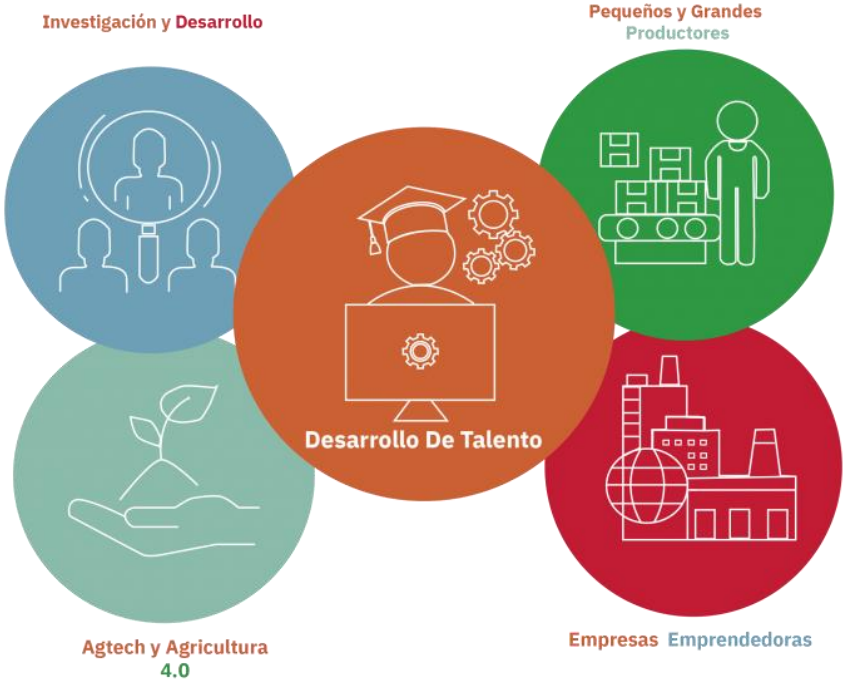
Silicon Misiones CONOCIMIENTO

Es un espacio de encuentro y sinergia entre estudiantes, emprendedores, institutos públicos de investigación y centros dedicados a la innovación y el desarrollo

Silicon Misiones LABORATORIO

# Como estamos en Misiones?

Silicon Misiones



# Como estamos en Misiones?

ESCUELA DE ROBÓTICA Misiones

La escuela de Robótica, es un espacio educativo de gestión estatal no arancelado que ofrece una propuesta pedagógica entorno a la ciencia y a la tecnología, orientadas a la programación y la robótica educativa en el marco de las transformaciones culturales del Siglo XXI.

# Como estamos en Misiones?

FANIOT
FABRICA ARGENTINA DE NANO SENSORES IOT

polotic misiones

PARQUE INDUSTRIAL Y DE LA INNOVACIÓN POSADAS

YECTO

Es un proyecto provincial de innovación industrial que desarrolla en el Parque Industrial y de la Innovación Posadas. Con la finalidad de cambiar la matriz productiva de la provincia orientada a las nuevas tecnologías.

www.aerobotsmisiones.com
AEROBOTS
www.aerobotsmisiones.com
MISIONES

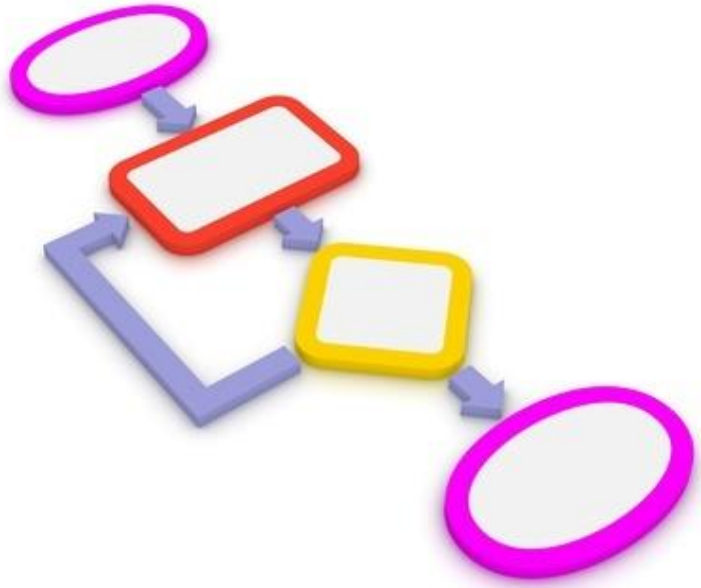
eControl
SMART HOME SYSTEMS

biem!

VyM
INGENIERÍA

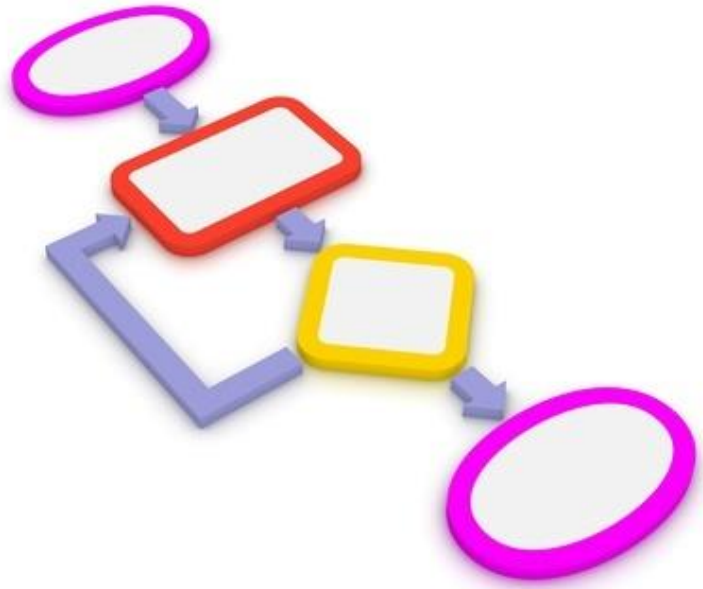
# METODOLOGIA DE ENSEÑANZA

- ENCUENTROS ASINCRONICOS
- ENCUENTROS SINCRONICOS
- CLASES TEORICAS
- CLASES PRACTICAS



# REQUERIMIENTOS DEL CURSO

- COMPUTADORA
- PSEINT O CUALQUIER IDE
- GANAS DE APRENDER
- MUCHA PRACTICA



# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

Gobierno de Misiones

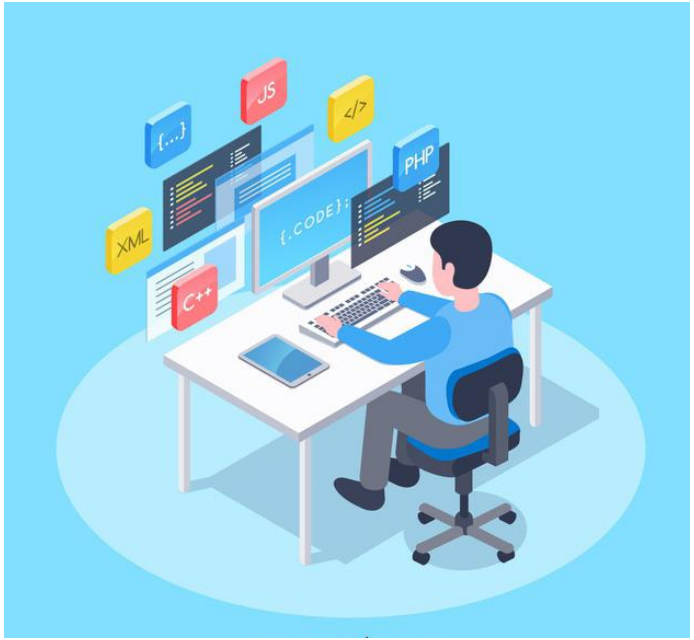
# INTRODUCCIÓN A LA *programación*

polotic
misiones

# FUNDAMENTACIÓN

La informática y las ciencias de la computación en los primeros años del siglo xxi vienen marcadas por los avances tecnológicos. El rápido crecimiento del mundo de las redes y, en consecuencia, la World Wide Web y las aplicaciones Móviles hacen revolucionarios a estos cambios y afectan al cuerpo de conocimiento de los procesos educativos y profesionales.

En consecuencia es importante aprender técnicas de análisis, diseño y construcción de algoritmos y objetos, así como reglas para la escritura de programas, eficientes tanto estructurados como orientados a objetos, se busca también enseñar al alumno técnicas de abstracción que le permitan resolver los problemas de programación del modo más sencillo y racional, aprender a pensar para conseguir la resolución del problema en cuestión de forma clara, eficaz y fácil de implementar en cualquier lenguaje de programación.



# DESCRIPCIÓN DEL CURSO

El curso proporcionará al alumno las herramientas (DFD y pseudocódigos) para desarrollar programas correctos, eficientes, bien estructurados y con estilo, que sirvan de base para la construcción de unos fundamentos teóricos y prácticos que le permitan continuar con éxito sus estudios de los cursos superiores. En consecuencia se pretende enseñar técnicas de análisis, diseño y construcción de algoritmos, así como reglas para la escritura de programas eficientes.

DATOS

INFORMACIÓN

**CONOCIMIENTO**

TOMA DE DECISIONES

**SABIDURÍA**

# ¿Qué es la Informática?

La informática, también llamada computación, es la rama de la ciencia que se encarga de estudiar la administración de métodos, técnicas y procesos con el fin de almacenar, procesar y transmitir información y datos en formato digital.

De esta manera, la informática se refiere al procesamiento automático de información mediante dispositivos electrónicos y sistemas computacionales. Los sistemas informáticos deben contar con la capacidad de cumplir tres tareas básicas: entrada (captación de la información), procesamiento y salida (transmisión de los resultados). El conjunto de estas tres tareas se conoce como algoritmo.

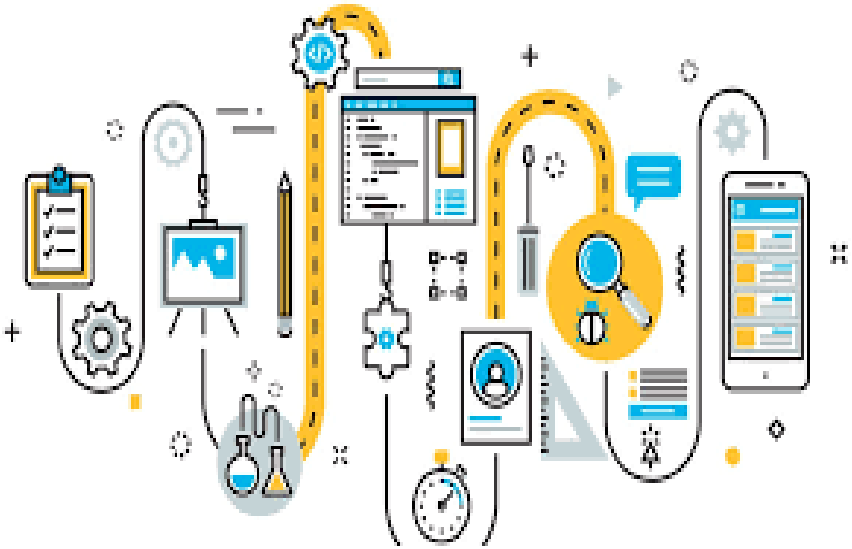

ENTRADA
PROCESO
SALIDA
DATOS
PROCESO
INFORMACIÓN

# ¿Qué es un Algoritmo?

Conjunto ordenado de operaciones sistemáticas que permite hacer un cálculo y hallar la solución de un tipo de problemas.

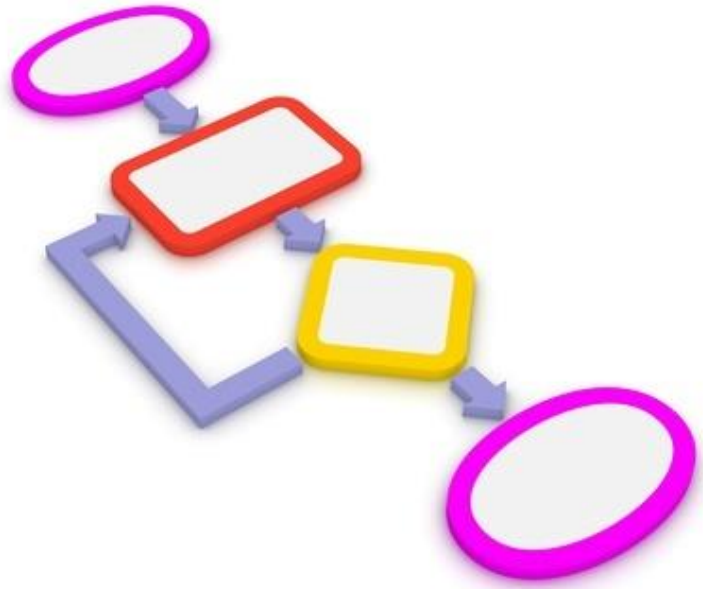
En informática o programación, el algoritmo es la secuencia de instrucciones mediante la cual podemos resolver un problema o cuestión.

La palabra algoritmo proviene del sobrenombre de un matemático árabe del siglo IX, Al-Khwarizmi, que fue reconocido por enunciar paso a paso las reglas para las operaciones matemáticas básicas con decimales (suma, resta, multiplicación y división).



# ASPECTOS FUNDAMENTALES EN EL DESARROLLO DEL ALGORITMO

- DECLARAR VARIABLES
- INICIALIAZAR VARIABLES
- DEFINIR CONDICION DE SALIDA
- RESOLUCION DEL ALGORITMO



# HISTORIA DE LOS PROGRAMAS

- LENGUAJES SECUENCIALES

La estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

```
 10 INPUT "Cuál es su nombre:"; NN$
 20 PRINT "Bienvenido al 'asterisquero' ";NN$
 25 PRINT
 30 INPUT "con cuántos asteriscos inicia [Cero sale]:"; N
 40 IF N<=0 THEN GOTO 200
 50 AS$=""
 60 FOR I=1 TO N
 70   AS$=AS$+"*"
 80 NEXT I
 90 PRINT "AQUI ESTAN:"; AS$
100 INPUT "Desea más asteriscos:";SN$S
110 IF SN$="" THEN GOTO 100
120 IF SN$<>"S" OR N$<>"s" THEN GOTO 200
130 INPUT "CUANTAS VECES DESEA REPETIRLOS [Cero sale]:"; VECES
140 IF VECES<=0 THEN GOTO 200
150 FOR I=1 TO VECES
160   PRINT AS$;
170 NEXT I
180 PRINT
185 REM A repetir todo el ciclo (comentario)
190 GOTO 25
200 END
```

# HISTORIA DE LOS PROGRAMAS

- LENGUAJES ESTRUCTURADOS

La programación estructurada es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa de computadora recurriendo únicamente a subrutinas y tres estructuras básicas: secuencia, selección (if y switch) e iteración (bucles for y while)

```
DEFINT I-N          'Declara entera toda variable que comience con letras
iTrue = -1          'Flag en Verdadero
INPUT "¿Cuál es su nombre"; NombreUsuario$
PRINT "Bievenido al 'asterisquero',"; NombreUsuario$
DO
   PRINT ""
   INPUT "¿Con cuántos asteriscos inicia [Cero sale]:"; NroAsteriscos
   IF NroAsteriscos<=0 THEN EXIT DO
   Asteriscos$ = ""
   FOR I=1 TO NroAsteriscos
      Asteriscos$=Asteriscos$ + "*"
   NEXT I
   PRINT "AQUI ESTAN: "; Asteriscos$
   DO
      INPUT "Desea más asteriscos:";SN$
   LOOP UNTIL SN$<>""
   IF SN$<>"S" OR SN$<>"s" THEN EXIT DO       'Salida
   INPUT "CUANTAS VECES DESEA REPETIRLOS [Cero sale]:";iVeces
   IF iVeces<=0 THEN EXIT DO     'Salida
   FOR I = 1 TO iVeces
      PRINT Asteriscos$;
   NEXT I
   PRINT
LOOP WHILE iTrue
END
```

# HISTORIA DE LOS PROGRAMAS

- LENGUAJES ORIENTADOS A OBJETOS

La Programación Orientada a Objetos es un paradigma de programación que viene a innovar la forma de obtener resultados. Los objetos se utilizan como metáfora para emular las entidades reales del negocio a modelar.

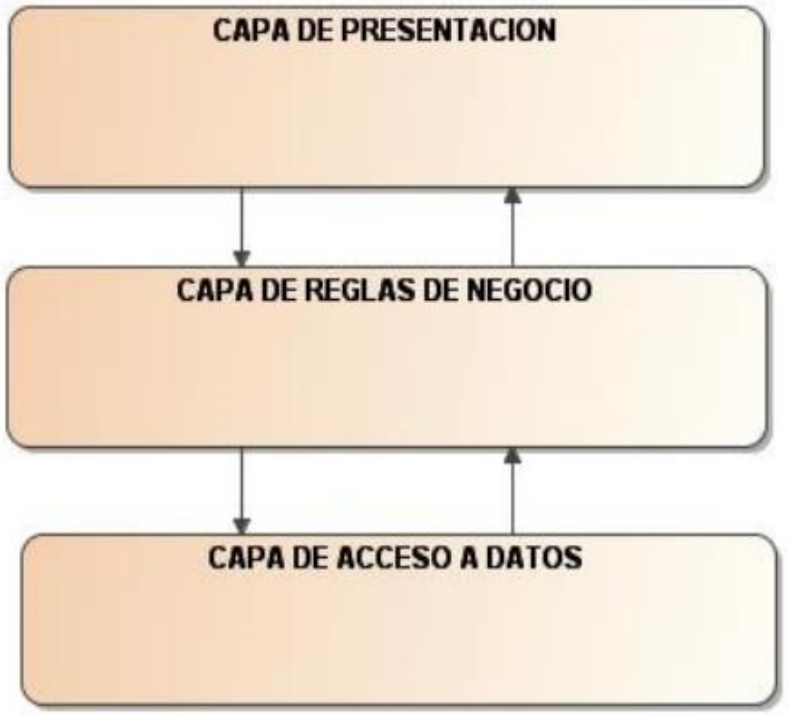
Muchos de los objetos prediseñados de los lenguajes de programación actuales permiten la agrupación en bibliotecas o librerías, sin embargo, muchos de estos lenguajes permiten al usuario la creación de sus propias bibliotecas.

```java
package poo;

/**
 *
 * @author JoseLuis
 */
public class Furgoneta extends Coche {
    private int capacidad_carga;
    private int plazas_extra;

    public Furgoneta(int plazas_extra, int capacidad_carga){
        super(); //llama constructor de la clase padre
        this.plazas_extra = plazas_extra;
        this.capacidad_carga = capacidad_carga;
    }

    public String dimeDatosFurgoneta(){
        return "Capacidad:"+capacidad_carga +" Plazas:"+plazas_extra;
    }

}
```

# HISTORIA DE LOS PROGRAMAS

- PATRONES - FRAMEWORKS

El patrón busca objetos similares que ya existen para reutilizarlo en lugar de crear otros nuevos que sean similares. Se utiliza para crear objetos que pueden representar funciones de otras clases u objetos y la interfaz se utiliza para acceder a estas funcionalidades.



# FORMAS DE APRENDIZAJE

- PSEUDOCÓDIGO

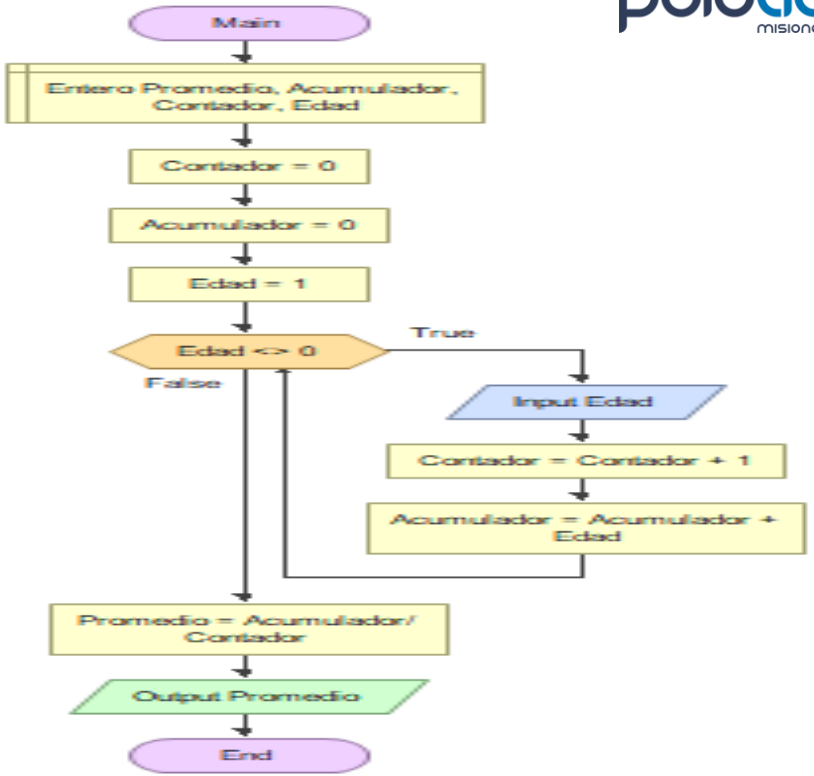
El pseudocódigo es una forma de escribir los pasos que va a realizar un programa de la forma más cercana al lenguaje de programación que vamos a utilizar posteriormente

```
Function Main
    Declare Entero Promedio, Acumulador, Contador, Edad

    Assign Contador = 0
    Assign Acumulador = 0
    Assign Edad = 1
    While Edad <> 0
        Input Edad
        Assign Contador = Contador + 1
        Assign Acumulador = Acumulador + Edad
    End
    Assign Promedio = Acumulador/Contador
    Output Promedio
End
```

# FORMAS DE APRENDIZAJE

- FLUJOGRAMAS

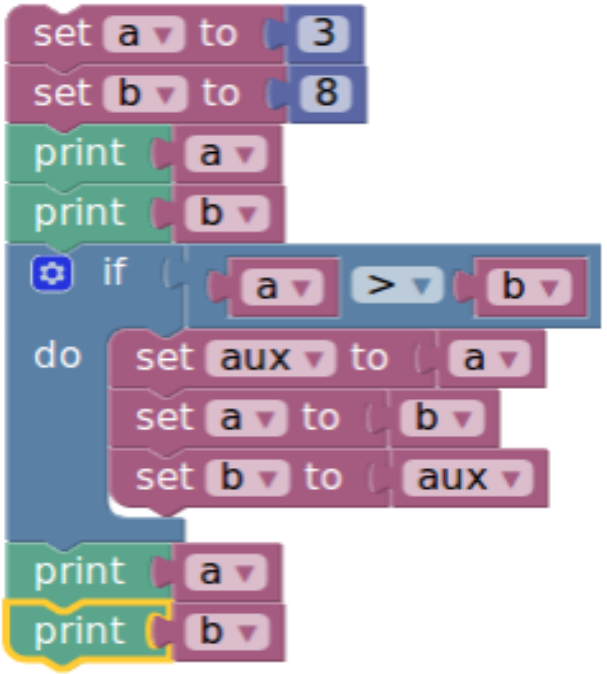
El diagrama de flujo o flujograma o diagrama de actividades es la representación gráfica de un algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.



# FORMAS DE APRENDIZAJE

- BLOQUES

El diagrama de bloques es la representación del funcionamiento interno de un sistema, que se hace mediante bloques y sus relaciones, y que, además, definen la organización de todo el proceso interno, sus entradas y sus salidas.



# FORMAS DE APRENDIZAJE

- LENGUAJES

Los lenguajes de programación están formados por un conjunto de símbolos (llamado alfabeto), reglas gramaticales (léxico/morfológicas y sintácticas) y semánticas, que en conjunto definen las estructuras válidas del lenguaje y su significado.

```java
package poo;

/**
 *
 * @author JoseLuis
 */
public class Furgoneta extends Coche {
    private int capacidad_carga;
    private int plazas_extra;

    public Furgoneta(int plazas_extra, int capacidad_carga){
        super(); //llama constructor de la clase padre
        this.plazas_extra = plazas_extra;
        this.capacidad_carga = capacidad_carga;
    }

    public String dimeDatosFurgoneta(){
        return "Capacidad:"+capacidad_carga +" Plazas:"+plazas_extra;
    }

}
```

# PRUEBA DE ESCRITORIO

Una prueba de escritorio es un tipo de prueba algorítmica, que consiste en la validación y verificación del algoritmo a través de la ejecución de las sentencias que lo componen (proceso) para determinar sus resultados (salida) a partir de un conjunto determinado de elementos (entrada)

| nombre | precioInicial | descuento | precioFinal |
|---|---|---|---|
| - | - | - | - |
| - | 0 | - | - |
| - | 0 | 0 | - |
| - | 0 | 0 | 0 |
| | 0 | 0 | 0 |
| María | 0 | 0 | 0 |
| María | 750 | 0 | 0 |
| María | 750 | 112.5 | 0 |

```
Algoritmo Venta
Var
    Cadena: nombre
    Real: precioInicial, descuento, precioFinal
Inicio
    precioInicial ← 0
    descuento ← 0
    precioFinal ← 0
    nombre ← " "
    Mostrar "Digite su nombre "
    Leer nombre
    Mostrar "Digite precio inicial $"
    Leer precioInicial
    descuento ← precioInicial*(15/100)
    precioFinal ← precioInicial-descuento
    Mostrar "Cliente: ", nombre
    Mostrar "Descuento: $", descuento
    Mostrar "Valor a pagar: $", precioFinal
Fin
```

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

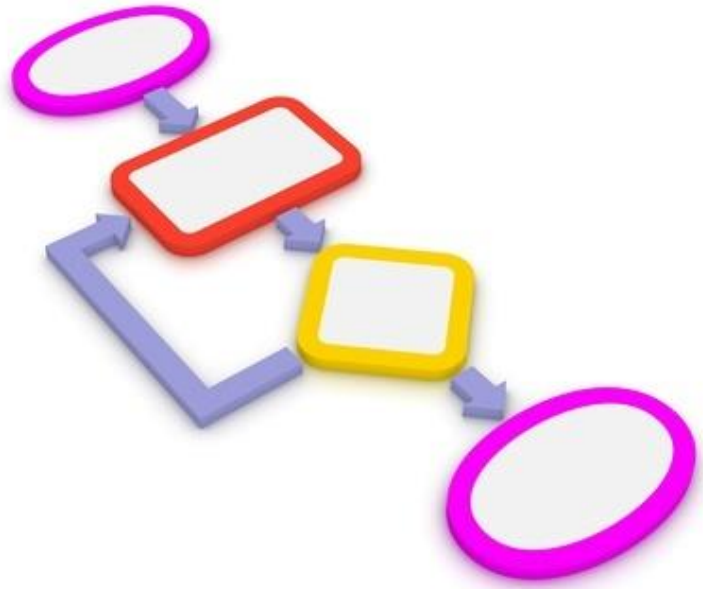
Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# ASPECTOS FUNDAMENTALES EN EL DESARROLLO DEL ALGORITMO

- DECLARAR VARIABLES
- INICIALIAZAR VARIABLES
- DEFINIR CONDICION DE SALIDA
- RESOLUCION DEL ALGORITMO



# DFD (Diagramas de Flujo)

Sentencias

Entrada/Salida

Entrada

Salida

Variables

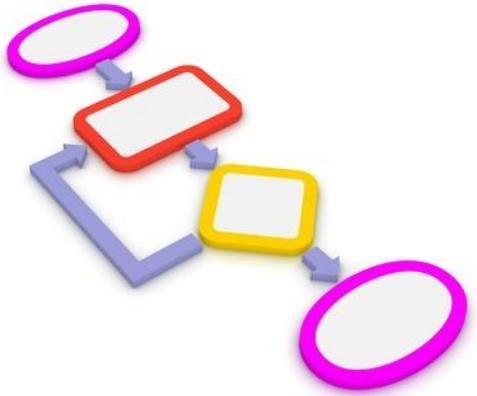
Declaración

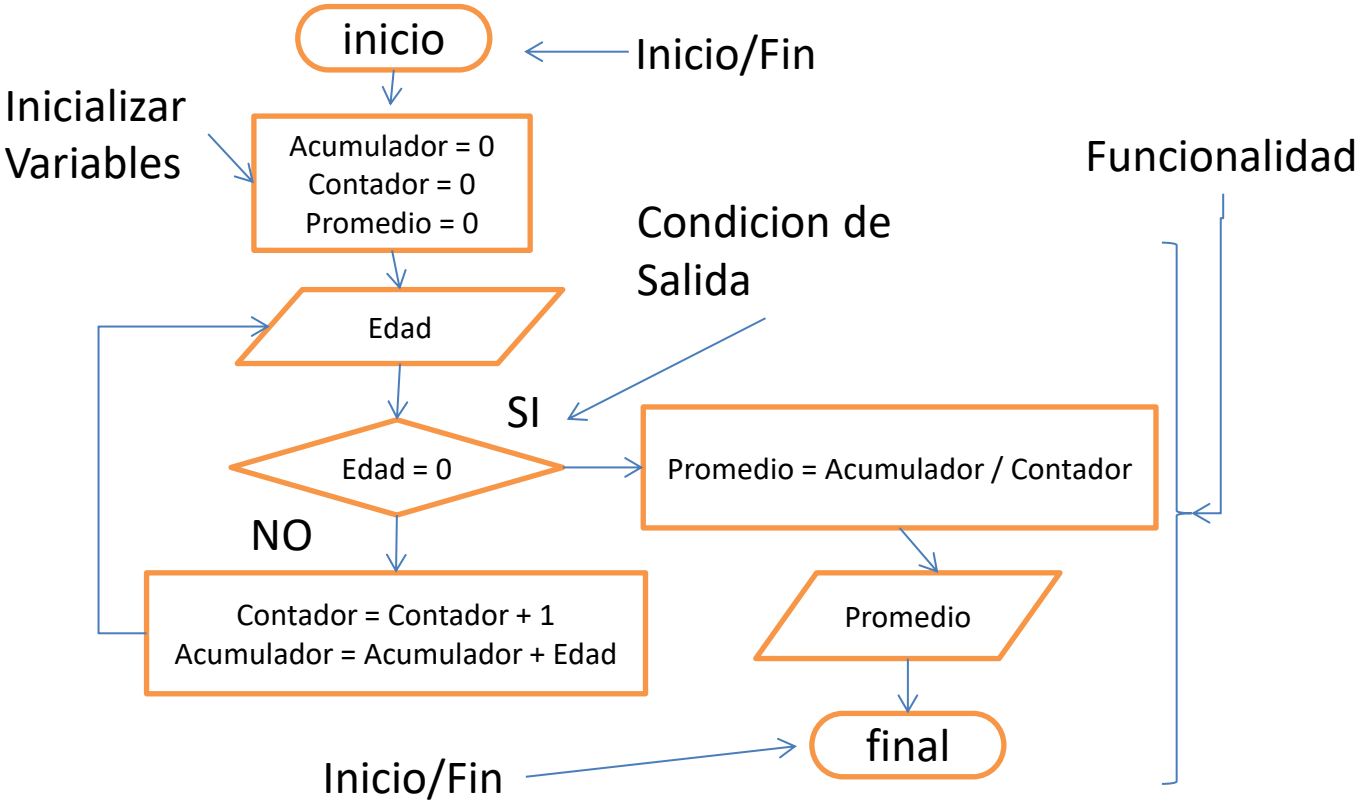
Asignación

Control

Si

Llamada




inicio
Inicio/Fin
Inicializar Variables
Acumulador = 0
Contador = 0
Promedio = 0
Funcionalidad
Condicion de Salida
Edad
SI
Edad = 0
Promedio = Acumulador / Contador
NO
Contador = Contador + 1
Acumulador = Acumulador + Edad
Promedio
final
Inicio/Fin

# PSEUDO-CODIGO

```
Iniciar
    Declarar
        entero Promedio
        entero Acumulador
        entero Contador
        entero Edad
    Asignar
        Contador = 0
        Acumulador = 0
        Edad = 1

    Mientras Edad <> 0
        Ingresar Edad
        Contador = Contador + 1
        Acumulador = Acumulador + Edad
    Fin Mientras

    Promedio = Acumulador/Contador
    Escribir Promedio
Fin
```

Declaración Variables

Inicializacion Variables

Condicion Salida

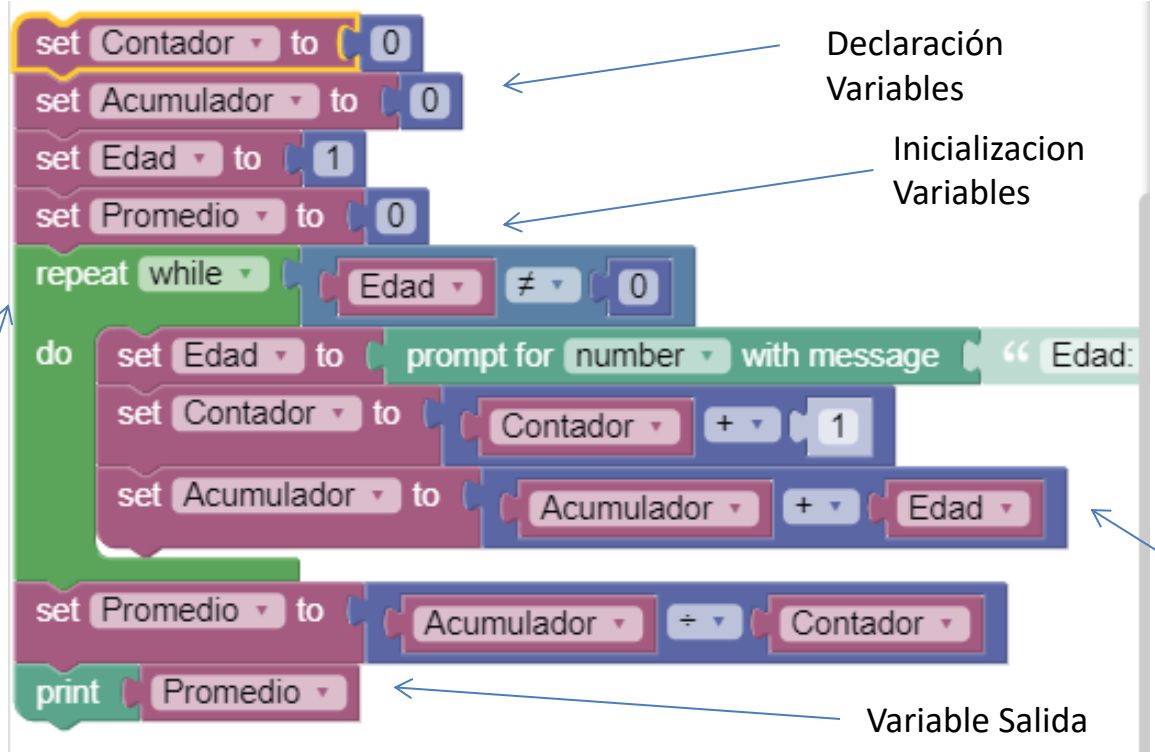
Acumulador

Variable Entrada

Contador

Funcionalidad

# PROGRAMACION EN BLOQUES



# JavaScript

```
var Contador, Acumulador, Edad, Promedio;


Contador = 0;
Acumulador = 0;
Edad = 1;
Promedio = 0;
while (Edad != 0) {
  Edad = Number(window.prompt('Edad:'));
  Contador = Contador + 1;
  Acumulador = Acumulador + Edad;
}
Promedio = Acumulador / Contador;
window.alert(Promedio);
```

Declaración Variables
Inicialicacion Variables

Condicion
Salida

Variable Entrada

Variables
Proceso

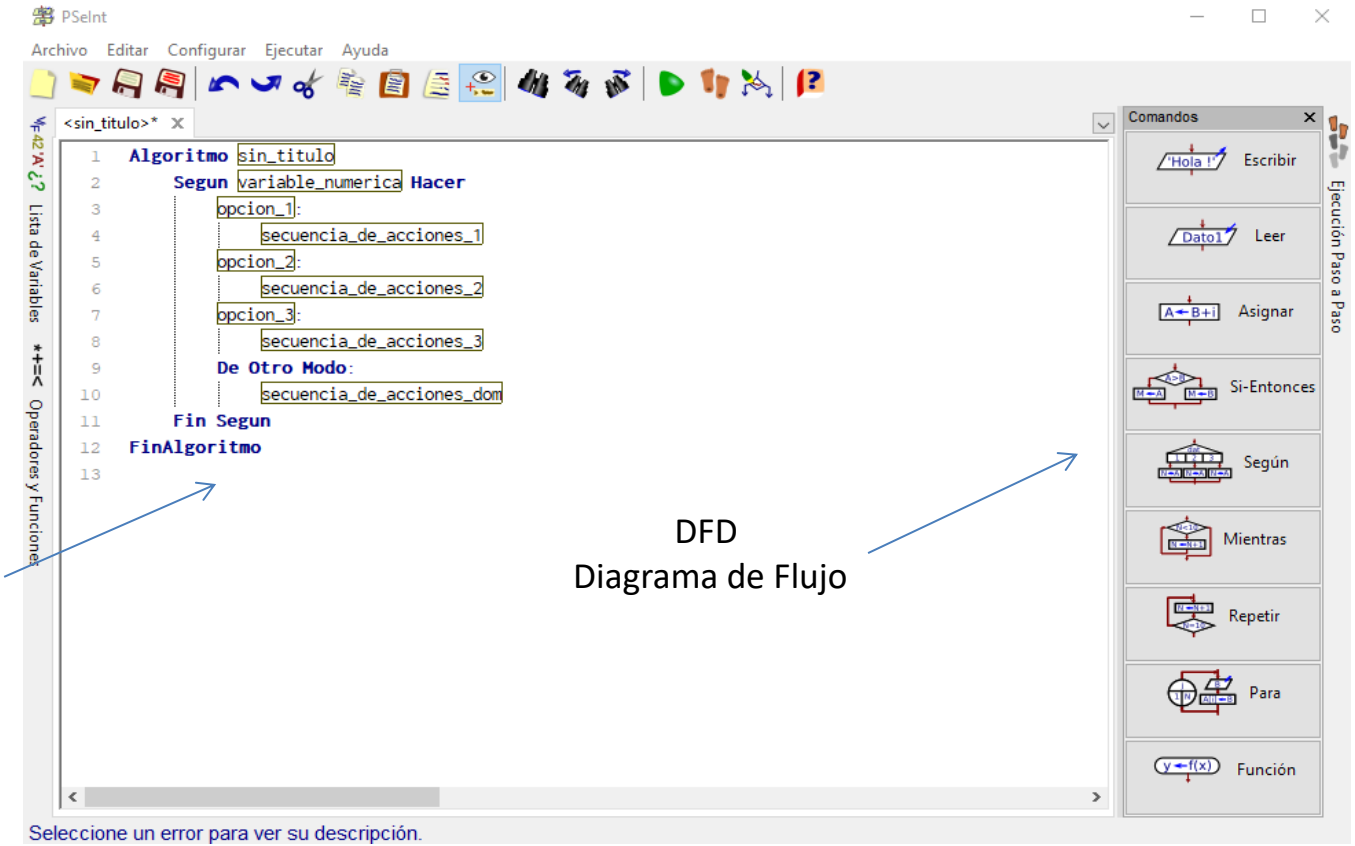
Variable Salida

JavaScript

PHP

SQL

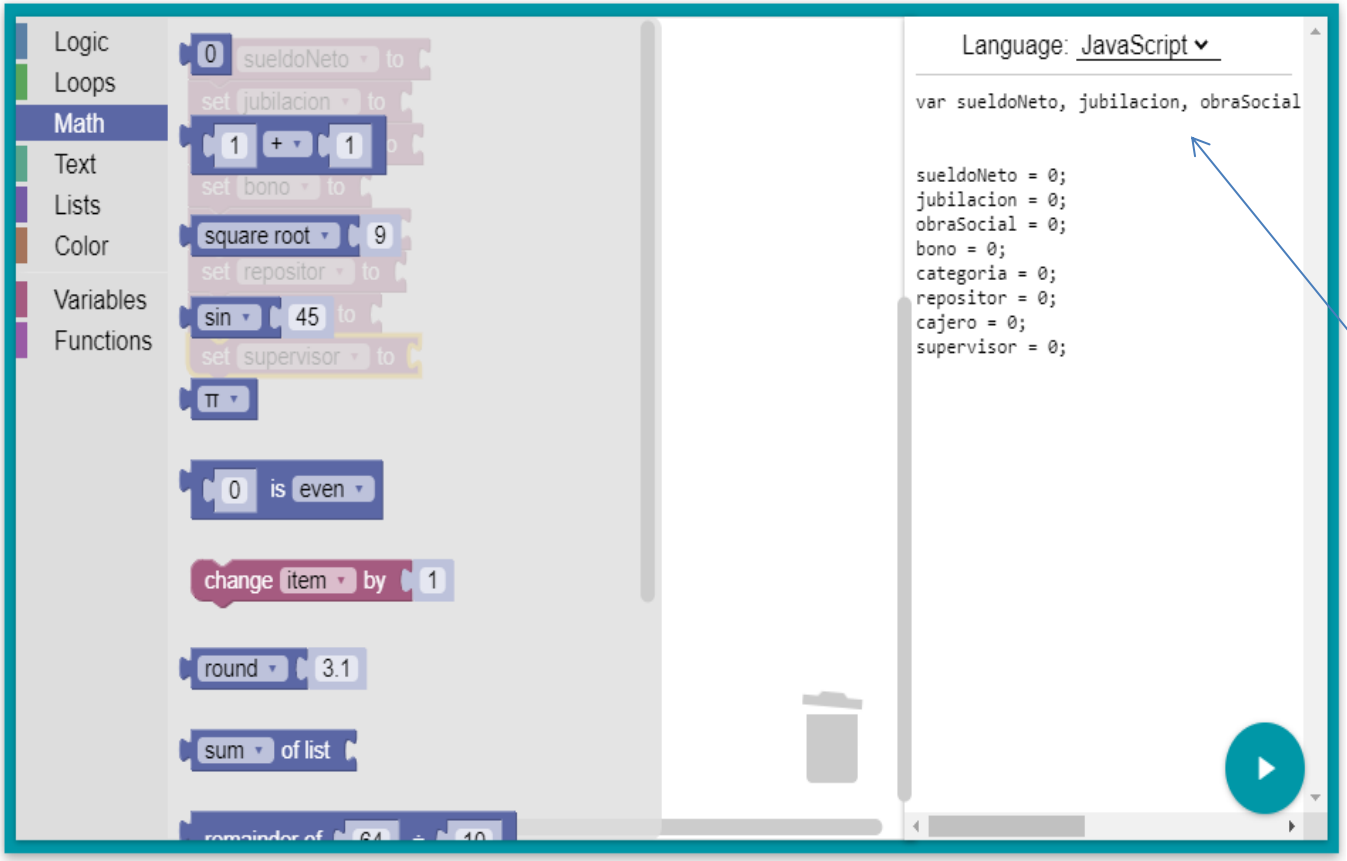
# PSeInt




Pseudo-Codigo

DFD
Diagrama de Flujo

# Blocky




Tipos de sentencias

Equivalente en lenguajes de programacion

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

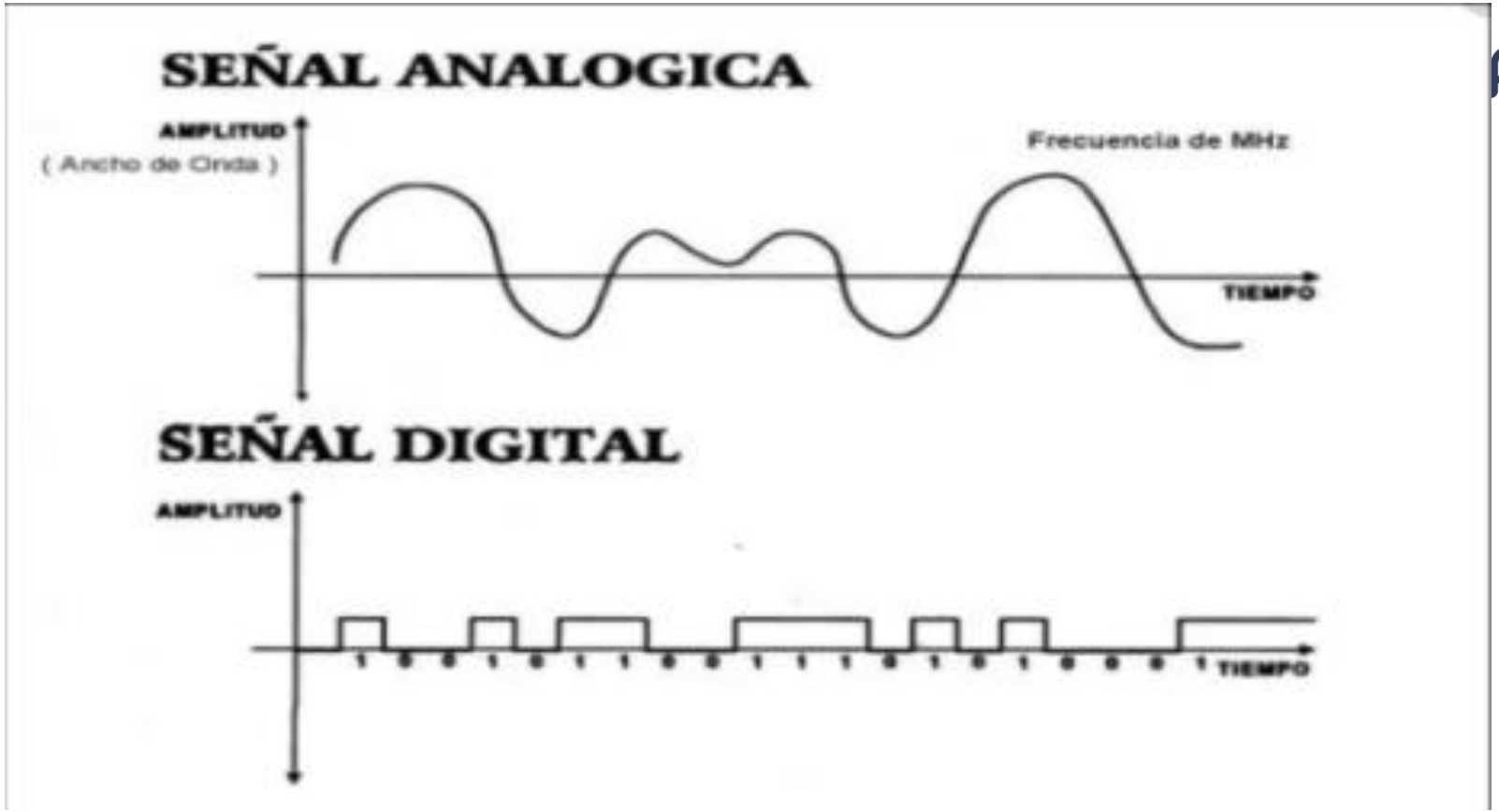
/poloticmisiones

polotic misiones

Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones


SEÑAL ANALOGICA
AMPLITUD
( Ancho de Onda )
Frecuencia de MHz
TIEMPO
SEÑAL DIGITAL
AMPLITUD
TIEMPO

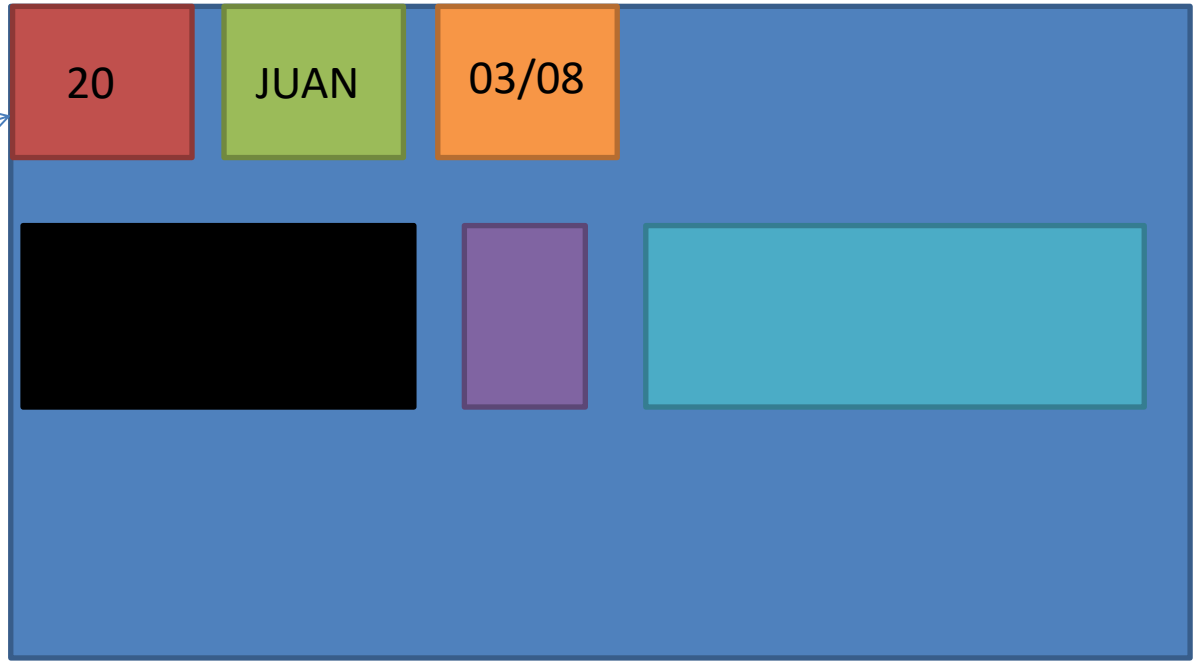
| Medida | Simbología | Equivalencia | Equivalente en Bytes |
|---|---|---|---|
| byte | b | 8 bits | 1 byte |
| kilobyte | Kb | 1024 bytes | 1 024 bytes |
| megabyte | MB | 1024 KB | 1 048 576 bytes |
| gigabyte | GB | 1024 MB | 1 073 741 824 bytes |
| terabyte | TB | 1024 GB | 1 099 511 627 776 bytes |
| Petabyte | PB | 1024 TB | 1 125 899 906 842 624 bytes |
| Exabyte | EB | 1024 PB | 1 152 921 504 606 846 976 bytes |
| Zetabyte | ZB | 1024 EB | 1 180 591 620 717 411 303 424 bytes |
| Yottabyte | YB | 1024 ZB | 1 208 925 819 614 629 174 706 176 bytes |
| Brontobyte | BB | 1024 YB | 1 237 940 039 285 380 274 899 124 224 bytes |
| Geopbyte | GB | 1024 BB | 1 267 650 600 228 229 401 496 703 205 376 bytes |

| Tipos de Datos | Memoria que ocupa | Rango de valores |
|---|---|---|
| boolean | 1 byte | 0 o 1 (True o False) |
| byte / unsigned char | 1 byte | 0 – 255 |
| char | 1 byte | -128 – 127 |
| int | 2 bytes | -32.768 – 32.767 |
| word / unsigned int | 2 bytes | 0 – 65.535 |
| long | 2 bytes | -2.147.483.648 – 2.147.483.647 |
| unsigned long | 4 bytes | 0 – 4.294.967.295 |
| float / double | 4 bytes | -3,4028235E+38 - 3,4028235E+38 |
| string | 1 byte + x | Array de caracteres |
| array | 1 byte + x | Colección de variables |

# VARIABLES

VARIABLES

EJEMPLO: EDAD
NOMBRE
FECHA




MEMORIA

TIPO DE VARIABLE:
-ENTERO
-DECIMAL
-FECHA
-CADENA
-IMAGEN
-BOOLEAN

# COMPORTAMIENTO VARIABLES

VARIABLES: SU VALOR PUEDE CAMBIAR A LO LARGO DE LA VIDA DEL ALGORITMO.

CONSTANTES: SU VALOR **NO** CAMBIA A LO LARGO DE LA VIDA DEL ALGORITMO.

ACUMULADOR: ACUMULA EN UNA VARIABLE EL RESULTADO DE ALGUNA OPERACIÓN MATEMATICA.

CONTADOR:CUENTA LAS ITERACIONES EN UNA VARIABLE UTILIZANDO UNA CONSTANTE.

IMPORTANTE: SE PUEDEN REALIZAR TODAS LAS OPERACIONES MATEMATICAS (SUMA, RESTA, MULTIPLICACION Y DIVISION), ADEMAS DE PODER REPRESENTAR TODO TIPO DE FUNCIONES ARITMETICAS, RELACIONALES O LOGICAS.

# EJEMPLO

SE DESEA CALCULAR EL PROMEDIO DE LOS ALUMNOS DE FUNDAMENTOS DE PROGRAMACION DE LA COMISION “A”.

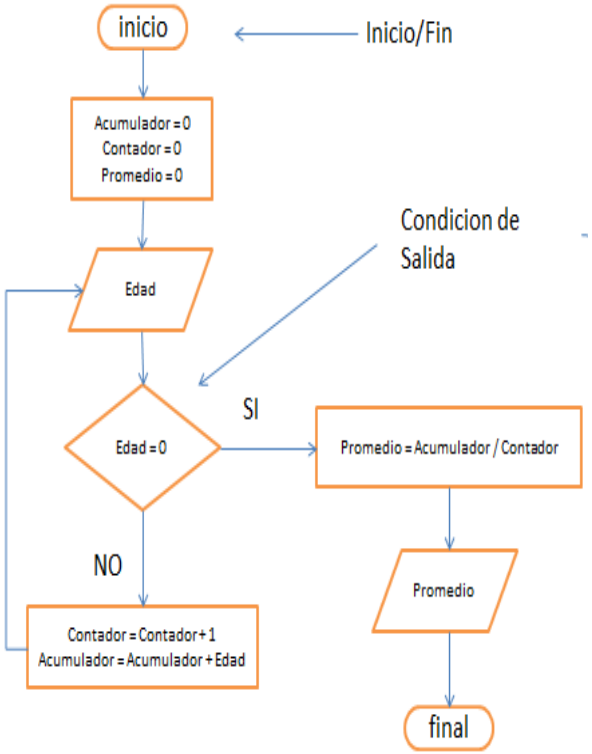
PARA CALCULAR EL PROMEDIO SE DEBE DIVIDIR LA CANTIDAD ACUMULADA DE EDADES SOBRE LA CANTIDAD DE ALUMNOS.

VARIABLES DE ENTRADA: EDAD

VARIABLES DE PROCESO: ACUMULADOR CONTADOR

VARIABLES DE SALIDA: PROMEDIO

# PRUEBA DE ESCRITORIO




| Edad | Promedio | Acumulador | Contador |
|---|---|---|---|
| ~~20~~ | 0 | 0 | 0 |
| ~~30~~ | 24 | 20 | 1 |
| ~~25~~ | | 50 | 2 |
| ~~28~~ | | 75 | 3 |
| ~~17~~ | | 103 | 4 |
| 0 | | 120 | 5 |
| | | | |

# PSEUDO CODIGO

```
1  Algoritmo Ejercicio1
2      Definir Acumulador Como Real;
3      Definir Contador Como Entero;
4      Definir Promedio Como Real;
5      Definir Edad Como Entero;
6
7      Acumulador = 0;
8      Contador = 0;
9      Repetir
10         Escribir"Ingresar Edad:";
11         Leer Edad;
12         Contador = Contador + 1;
13         Acumulador = Acumulador + Edad;
14     Hasta Que Edad = 0;
15     Promedio = Acumulador/Contador;
16     Escribir "El Promedio es:",Promedio;
17 FinAlgoritmo
18
```

Declaración de Variables

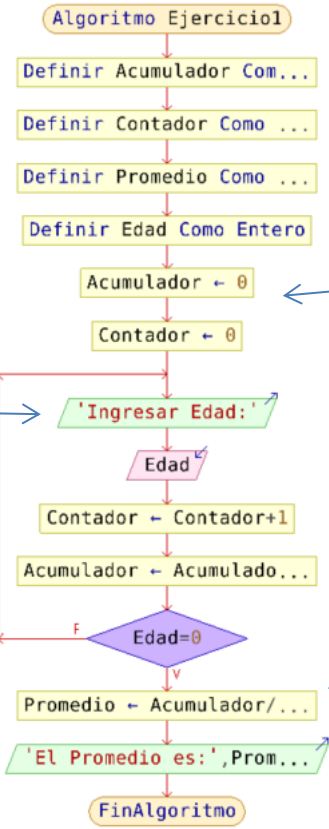
Inicialización de Variables

Inicialización durante la Lectura

Condición de Salida

Inicialización en la Asignación

# DIAGRAMAS DE FLUJO

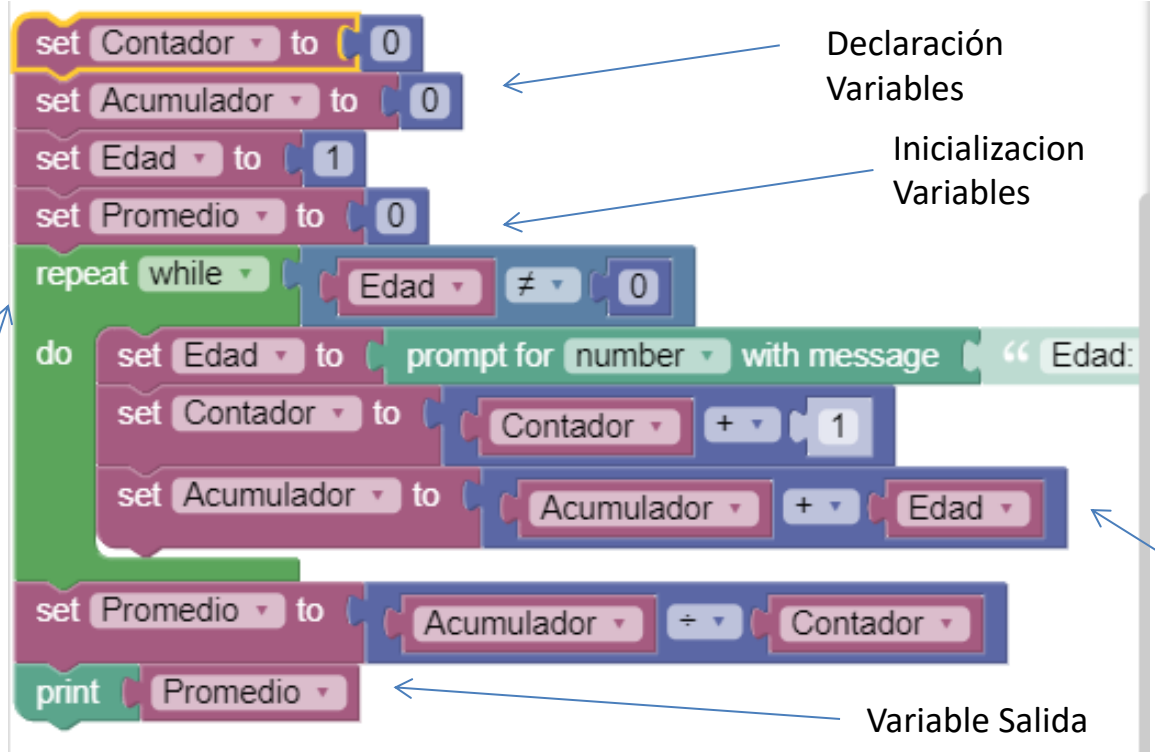



Variables de Proceso

Variables de Entrada

Variables de Salida

# PROGRAMACION EN BLOQUES



# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

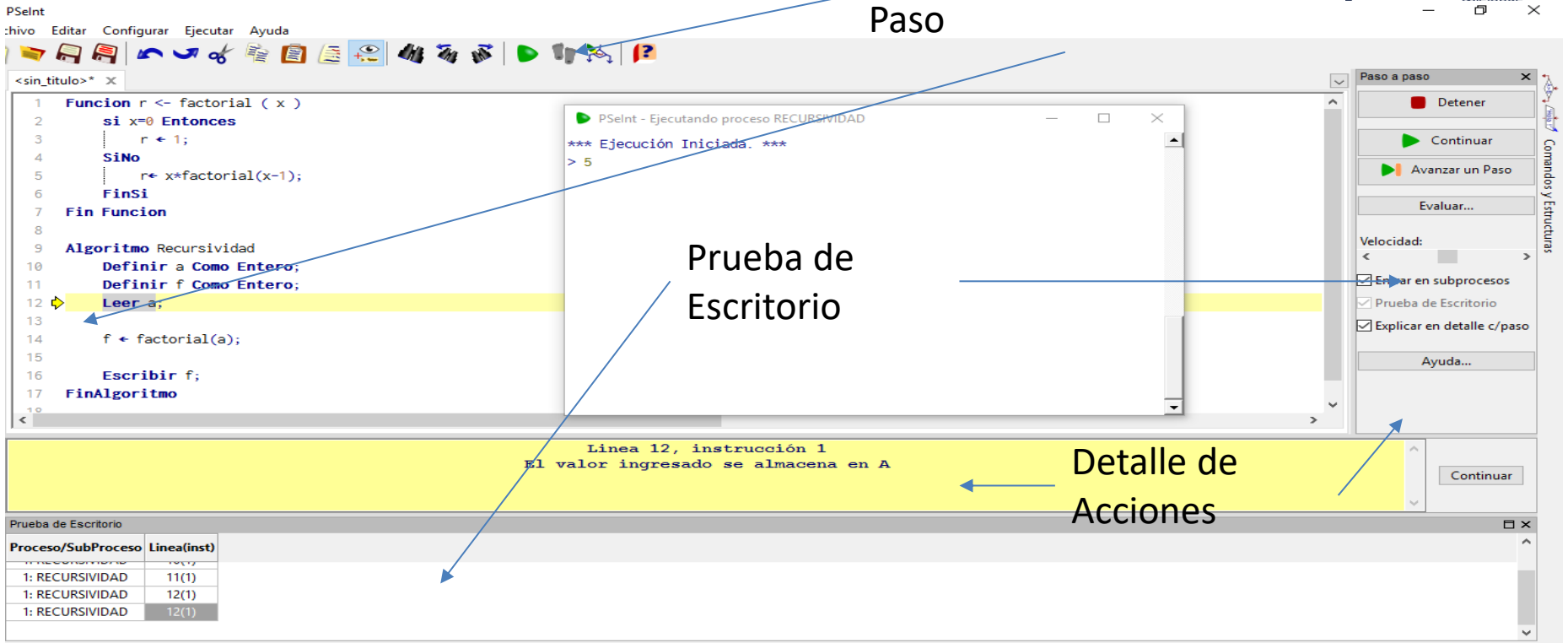
Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# IDE: PSeInt

Ejecución Paso por Paso




Prueba de Escritorio

Detalle de Acciones

## Ejercicio N.º 1: Tipos de datos

1- Determinar qué tipo de dato podría ser utilizado para los siguientes datos:

| | |
|---|---|
| a. 5,45 | REAL |
| b. 10 | ENTERO |
| c. 358 | ENTERO |
| d. Leonardo | CARACTERES |
| e. Verdadero | LOGICO |
| f. 78,3 | REAL |

2- Brindar al menos 2 ejemplos de cada uno de los siguientes tipos de datos

| | |
|---|---|
| a. Entero | 10 - 358 |
| b. Real | 5,45 |
| c. Lógico | VERDADERO - FALSO |
| d. Carácter | JOAQUIN - MARTINA |

## Ejercicio N.º 2: Variables

1- Determinar ¿Cuáles de los siguientes nombres de variables son válidos?

| | |
|---|---|
| a. Fecha de Nacimiento | NO |
| b. @pellido | SI (DEPENDE DEL LENGUAJE) |
| c. nombre | SI |
| d. cant_hijos | SI |
| e. tiene-pc | SI |
| f. edad | SI |
| g. DNI | SI |
| h. nombre.persona | NO |
| i. nombre&apellido | NO |
| j. em@ail | NO |
| k. dirección | SI |

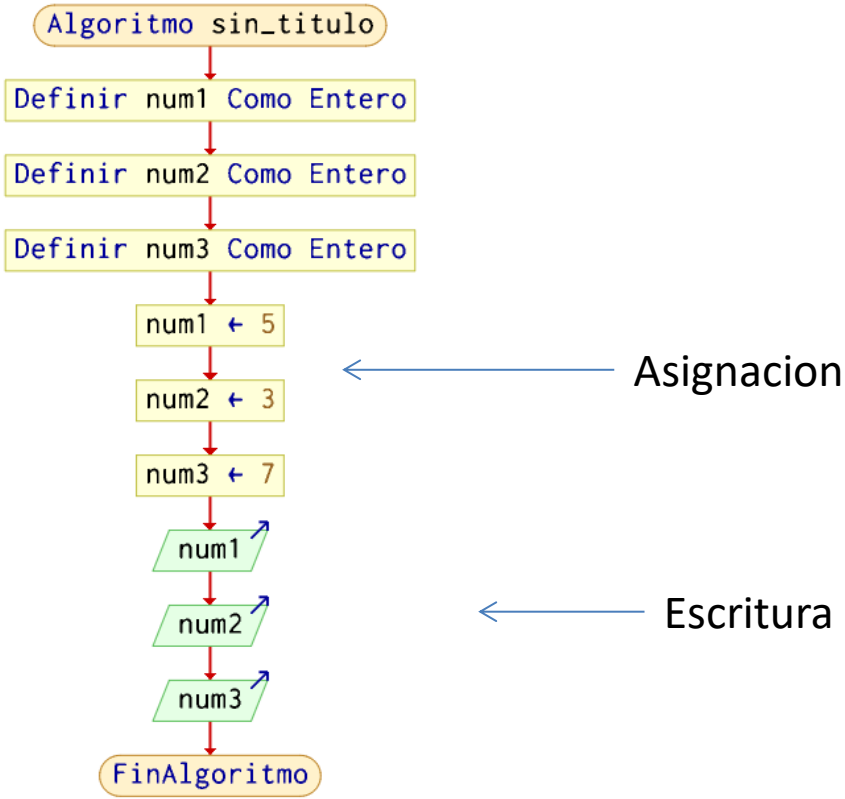
# Asignación de Variables y Escritura de Resultados

Una persona decidió realizar un algoritmo para mostrar 3 números por pantalla. Decidió llamar a las variables num1, num2 y num3 y colocarles los valores 5, 3 y 7. Sin embargo, no sabe de qué tipos de datos deberían ser sus tres variables ni tampoco como asignar dichos valores. Realizar un algoritmo que declare las variables, les asigne los valores que se necesitan y mostrar por pantalla.

```
1   Algoritmo sin_titulo
2       Definir num1 Como Entero;
3       Definir num2 Como Entero;
4       Definir num3 Como Entero;
5
6       num1 ← 5;
7       num2 ← 3;
8       num3 ← 7;
9
10      Escribir  num1;
11      Escribir  num2;
12      Escribir  num3;
13  FinAlgoritmo
```

Asignacion

Escritura




Asignacion

Escritura

# Asignación de Variables, Lectura y Escritura de Resultados

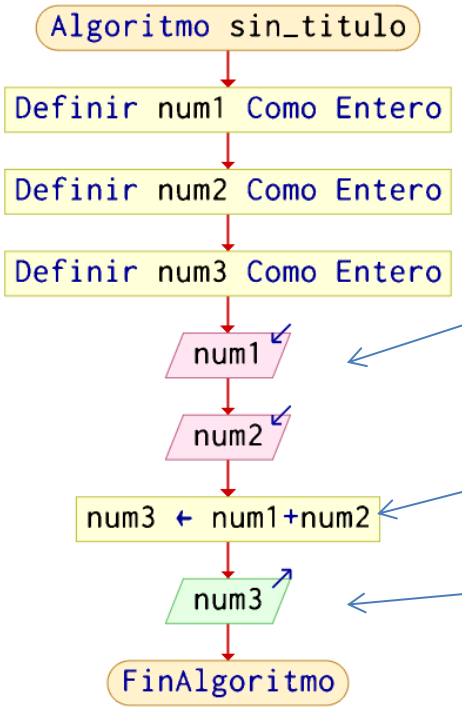
Escribir un algoritmo que permita ingresar por teclado dos números e imprima su suma.

```
1   Algoritmo sin_titulo
2       Definir num1 Como Entero;
3       Definir num2 Como Entero;
4       Definir num3 Como Entero;
5
6       Leer num1;
7       Leer num2
8
9       num3 ← num1 + num2;
10
11      Escribir  num3;
12  FinAlgoritmo
```

Lectura

Asignacion

Escritura




Lectura

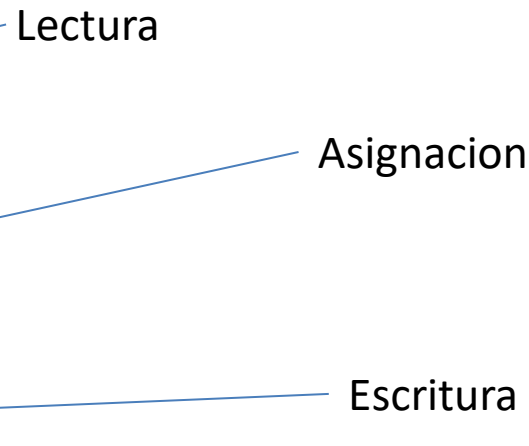
Asignacion

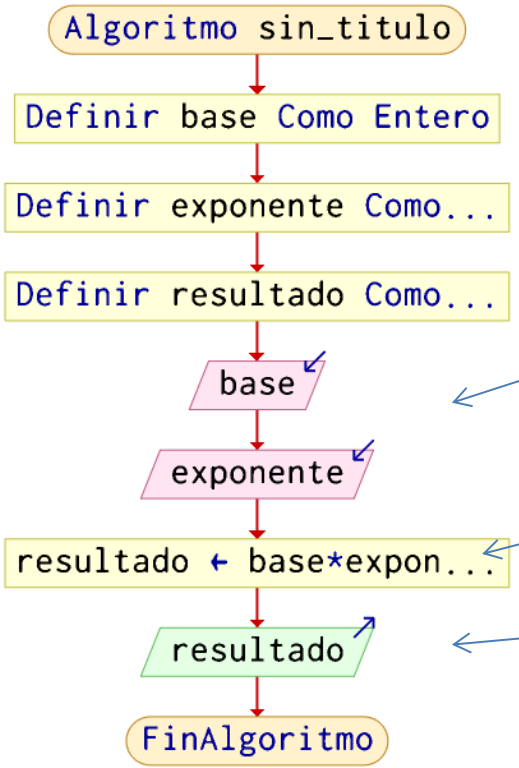
Escritura

# Asignación de Variables, Lectura y Escritura de Resultados

Realizar un algoritmo que permita a un usuario ingresar por teclado la BASE y el EXPONENTE de una potencia y que el resultado sea mostrado por pantalla.

```
1   Algoritmo sin_titulo
2       Definir base Como Entero;
3       Definir exponente Como Entero;
4       Definir resultado Como Entero;
5
6       Leer base;
7       Leer exponente;
8
9       resultado ← base * exponente;
10
11      Escribir  resultado;
12  FinAlgoritmo
```






Lectura

Asignacion

Escritura

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

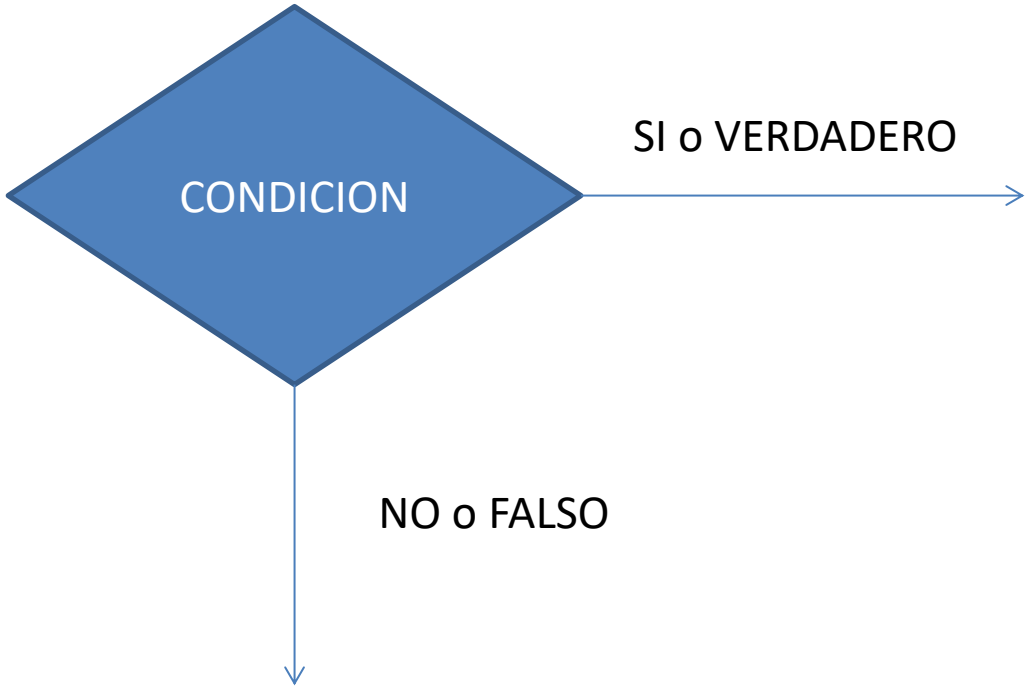
polotic misiones

Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# BIFURCACIONES CONDICIONALES





```
If (condicion) {
   VERDADERO
}
Else{
      FALSO
}
```

## CONDICIONES – PREGUNTAS - COMPARACIONES

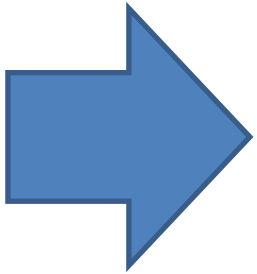
# COMPARACIONES

VARIABLES vs CONSTANTES

CONSTANTES vs VARIABLES

VARIABLES vs VARIABLES

CONSTANTES vs CONSTANTES



## TIPO DE COMPARACIONES

> MAYOR

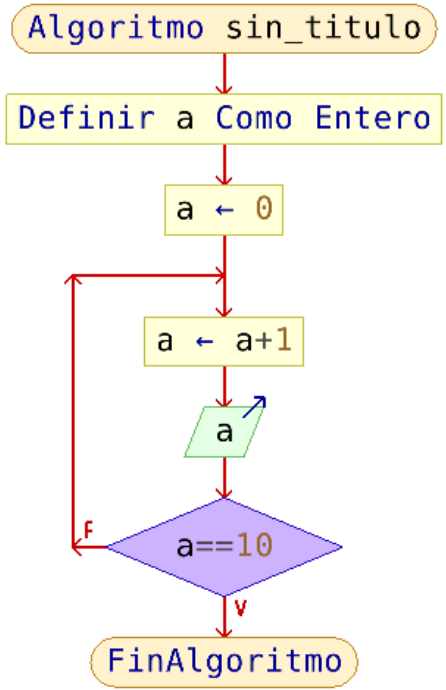
< MENOR

= IGUAL

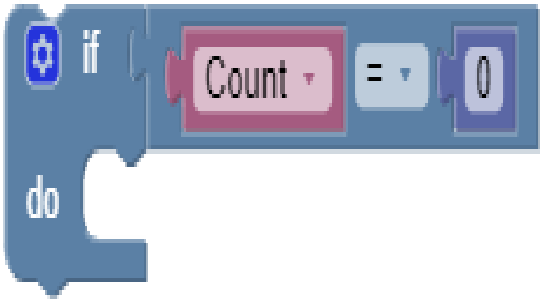
>= MAYOR IGUAL

<= MENOR IGUAL

<> != DISTINTOS




| EJEMPLO 1 | EJEMPLO 2 | EJEMPLO 3 |
|---|---|---|
| a= 0 | 10 = 0 | Edad = Variable |
| a> 0 | 10 > 0 | Edad > Variable |
| a< 0 | 10 < 0 | Edad < Variable |
| a>= 0 | 10 >= 0 | Edad >= Variable |
| a<= 0 | 10 <= 0 | Edad <= Variable |
| a<> 0 | 10 <> 0 | Edad <> Variable |
| a!= 0 | 10 != 0 | Edad != Variable |




| EJEMPLO 1 | EJEMPLO 2 | EJEMPLO 3 |
|---|---|---|
| Edad = 0 | 18 = 0 | Edad = Variable |
| Edad > 0 | 18 > 0 | Edad > Variable |
| Edad < 0 | 18 < 0 | Edad < Variable |
| Edad >= 0 | 18 >= 0 | Edad >= Variable |
| Edad <= 0 | 18 <= 0 | Edad <= Variable |
| Edad <> 0 | 18 <> 0 | Edad <> Variable |
| Edad != 0 | 18 != 0 | Edad != Variable |

```
If (Edad  = 0) {
  VERDADERO
}
Else
{
   FALSO
}
```

| EJEMPLO 1 | EJEMPLO 2 | EJEMPLO 3 |
|---|---|---|
| Edad = 0 | 18 = 0 | Edad = Variable |
| Edad > 0 | 18 > 0 | Edad > Variable |
| Edad < 0 | 18 < 0 | Edad < Variable |
| Edad >= 0 | 18 >= 0 | Edad >= Variable |
| Edad <= 0 | 18 <= 0 | Edad <= Variable |
| Edad <> 0 | 18 <> 0 | Edad <> Variable |
| Edad != 0 | 18 != 0 | Edad != Variable |

# OPERADORES BOOLEANOS

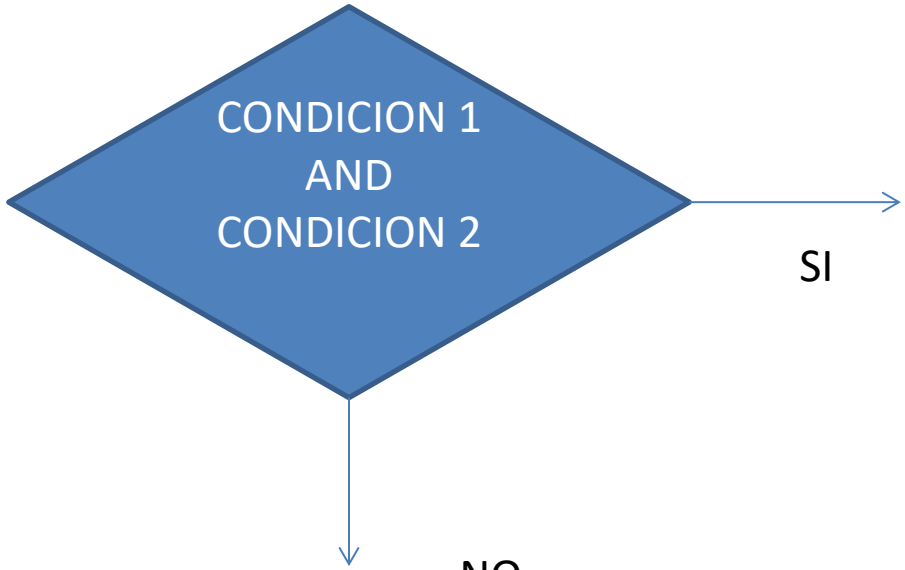
Los operadores booleanos permiten la concatenación de los criterios

## AND

CONDICION 1 AND CONDICION 2

| | | |
|---|---|---|
| V | V | V |
| V | F | F |
| F | F | V |
| F | F | F |

```
If (CONDICION 1 AND CONDICION 2
) {
  VERDADERO
}
Else
{
   FALSO
}
```



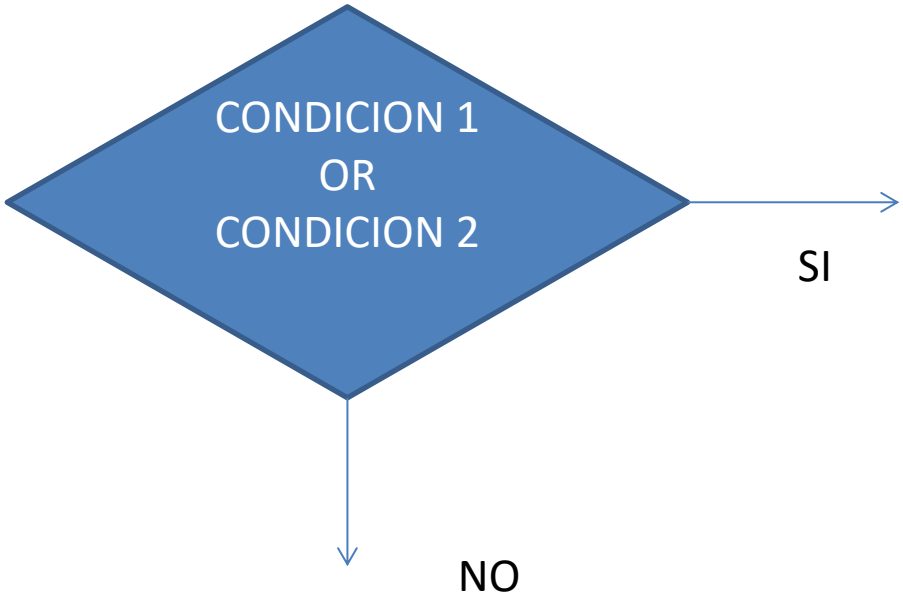
# OPERADORES BOOLEANOS

Los operadores booleanos permiten la concatenación de los criterios

## OR

| CONDICION 1 | OR | CONDICION 2 |
|---|---|---|
| V | V | V |
| V | V | F |
| F | V | V |
| F | F | F |

```
If (CONDICION 1 OR CONDICION 2
) {
  VERDADERO
}
Else
{
   FALSO
}
```



# OPERADORES BOOLEANOS

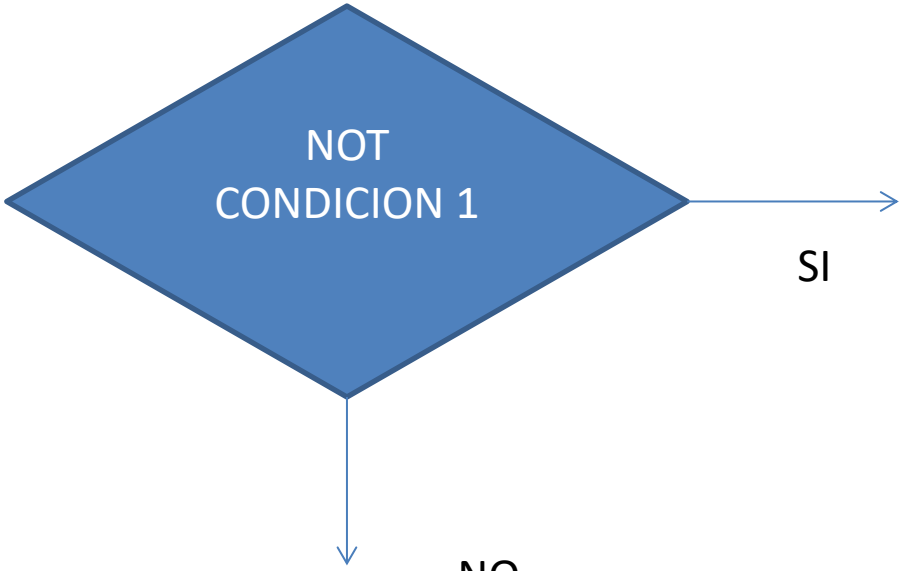
Los operadores booleanos permiten la concatenación de los criterios

## NOT

| NOT CONDICION 1 |
| --- |
| V |
| F |

| |
| --- |
| F |
| V |

```
If ( NOT CONDICION 1) {
  VERDADERO
}
Else
{
   FALSO
}
```

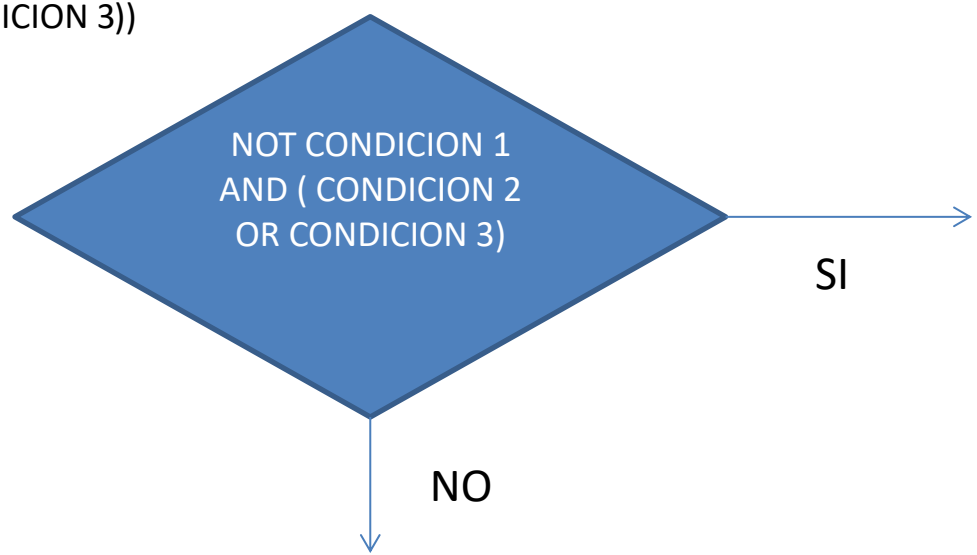


# OPERADORES BOOLEANOS

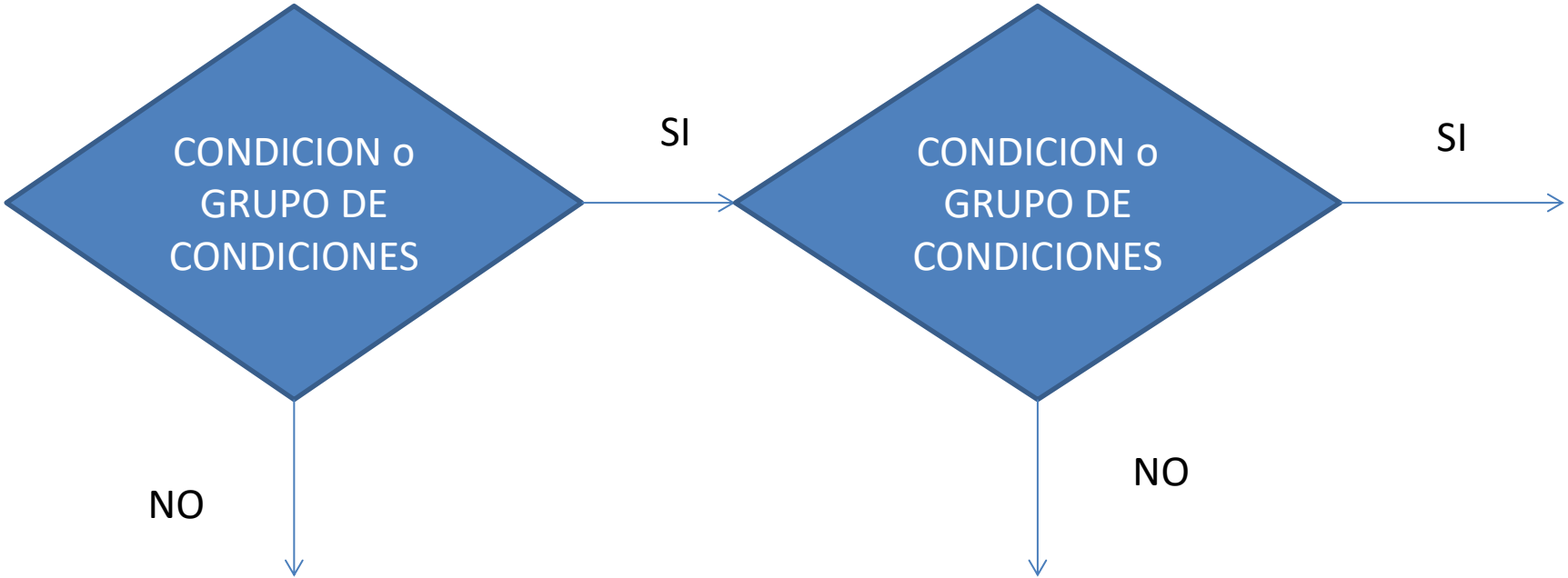
Los operadores booleanos permiten la concatenación de los criterios

NOT CONDICION 1 AND ( CONDICION 2 OR CONDICION 3)

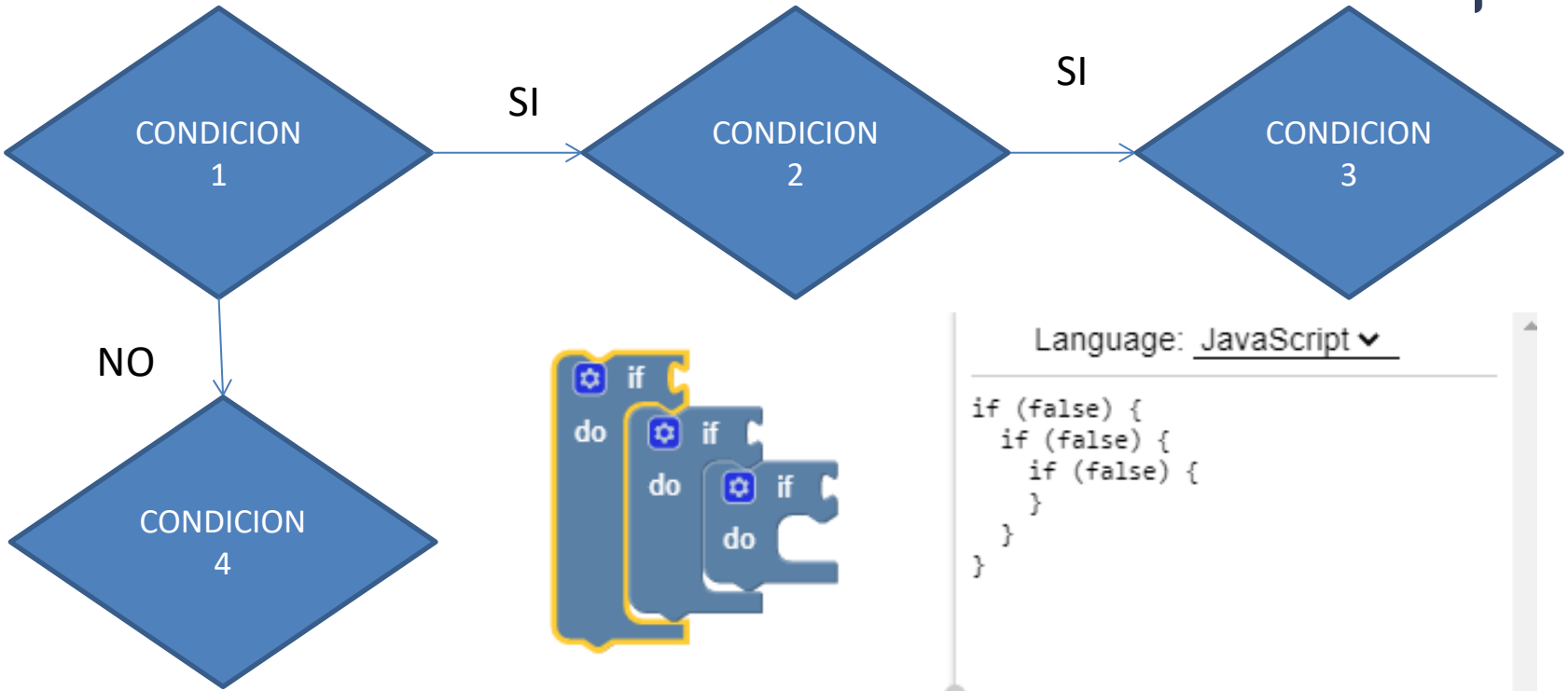
```
If (NOT CONDICION 1 AND ( CONDICION 2 OR CONDICION 3))
{
  VERDADERO
}
Else
{
   FALSO
}
```



# SENTENCIAS ANIDADAS



# SENTENCIAS ANIDADAS



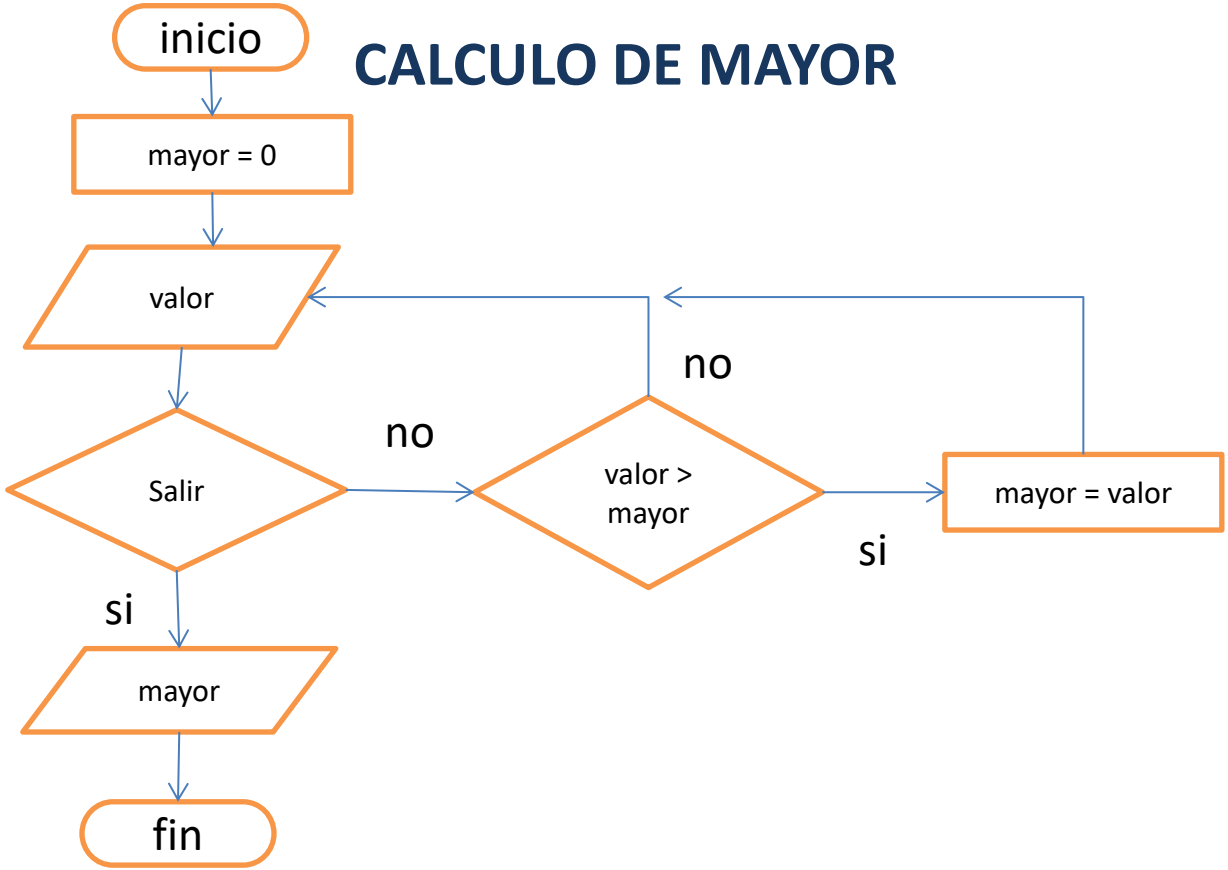




Language: JavaScript

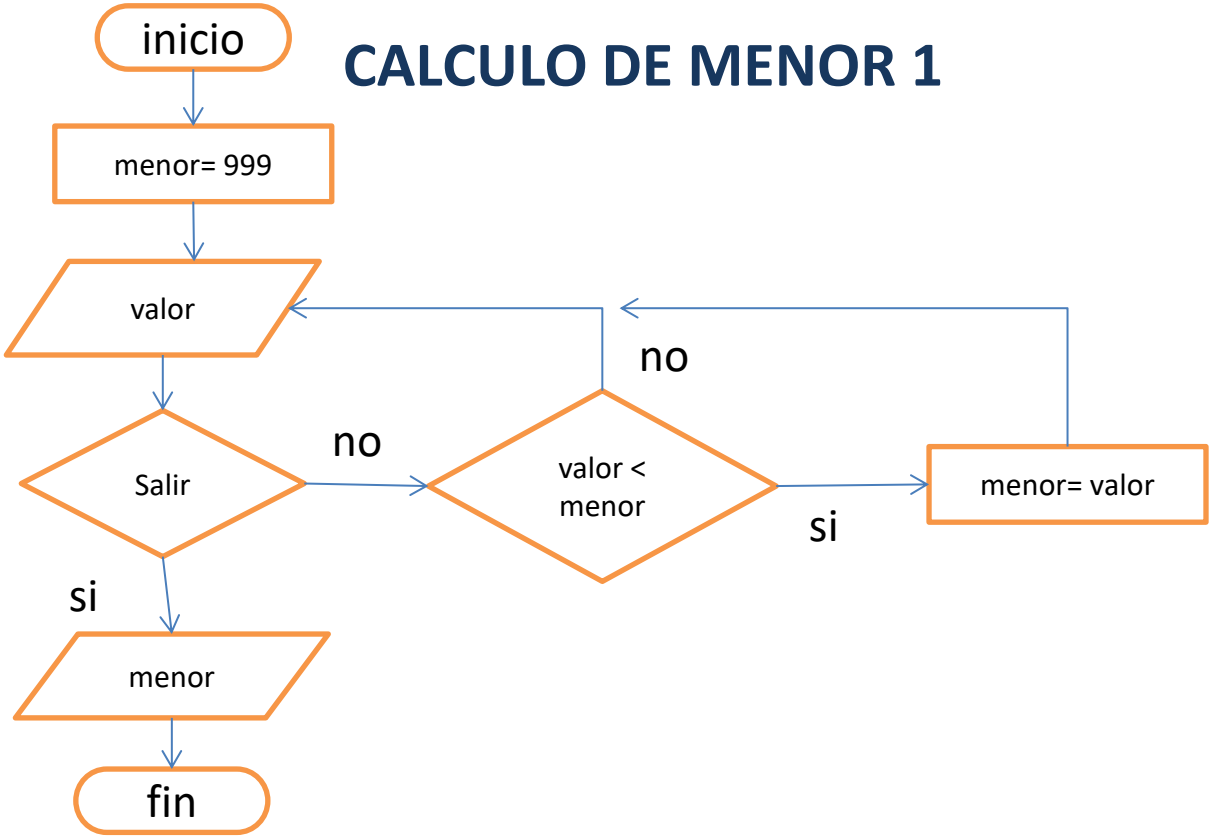
```
if (false) {
  if (false) {
    if (false) {
    }
  }
}
```

# CALCULO DE MAYOR



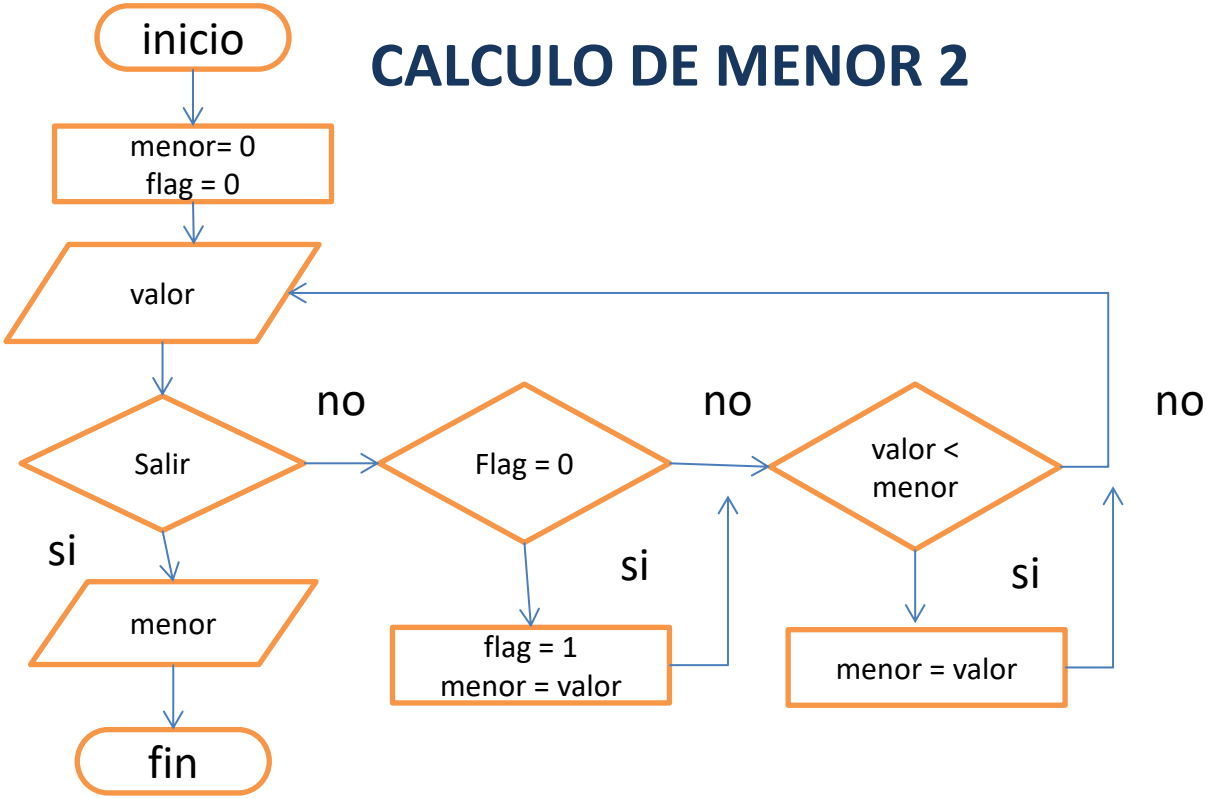

| valor | mayor |
|---|---|
| ~~3~~ | 0 |
| ~~7~~ | 3 |
| ~~6~~ | 7 |
| ~~2~~ | |

# CALCULO DE MENOR 1




| valor | mayor |
|---|---|
| ~~3~~ | 999 |
| ~~7~~ | 3 |
| ~~6~~ | 2 |
| ~~2~~ | |

# CALCULO DE MENOR 2




| valor | menor | flag |
|---|---|---|
| ~~3~~ | 0 | 0 |
| ~~7~~ | 3 | 1 |
| ~~6~~ | 2 | |
| 2 | | |

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# ESTRUCTURAS REPETITIVAS

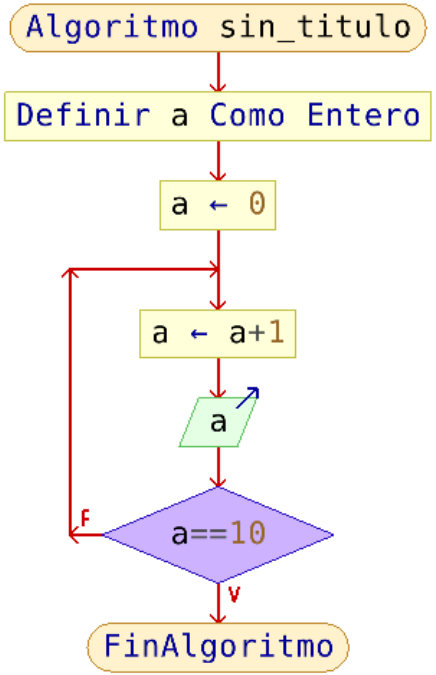
REPETIR

```
Algoritmo sin_titulo
    Repetir
        secuencia_de_acciones
    Hasta Que expresion_logica
FinAlgoritmo
```

```
Algoritmo sin_titulo
    definir a como entero;
    a<-0;
    Repetir
        a <- a+1;
        escribir a;
    Hasta Que a==10;
FinAlgoritmo
```

# ESTRUCTURAS REPETITIVAS

REPETIR




```java
public class DoWhileExample {
    public static void main(String[] args) {
        int a=1;
        do{
            System.out.println(a);
            a++;
        }while(a<=10);
    }
}
```

# ESTRUCTURAS REPETITIVAS

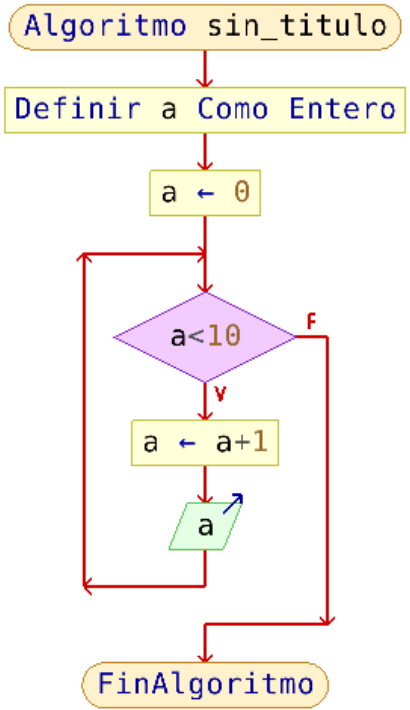
MIENTRAS

```
Algoritmo sin_titulo
    Mientras expresion_logica Hacer
        secuencia_de_acciones
    Fin Mientras
FinAlgoritmo
```

```
Algoritmo sin_titulo
    Definir a Como Entero;
    a <- 0;
    Mientras a < 10 Hacer
        a <- a + 1;
        escribir a;
    Fin Mientras
FinAlgoritmo
```

# ESTRUCTURAS REPETITIVAS

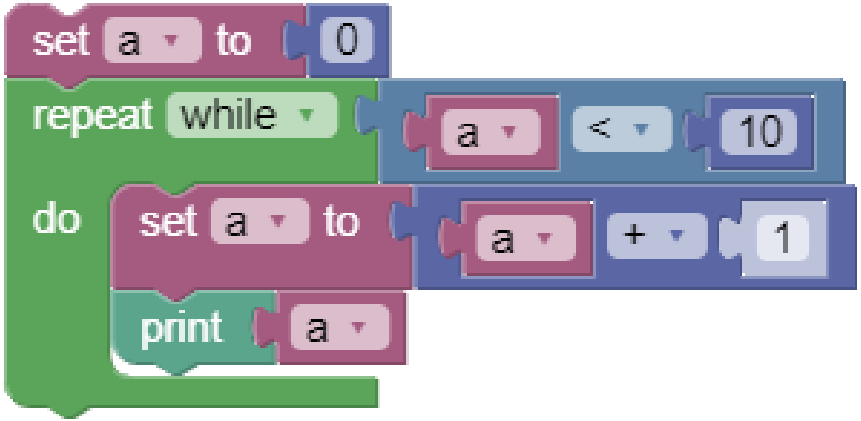
MIENTRAS




```java
public class WhileExample {
    public static void main(String[] args) {
        int a=0;
        while (a<10) {
            a++;
            System.out.println(a);
        }
    }
}
```

# ESTRUCTURAS REPETITIVAS

MIENTRAS



# ESTRUCTURAS REPETITIVAS

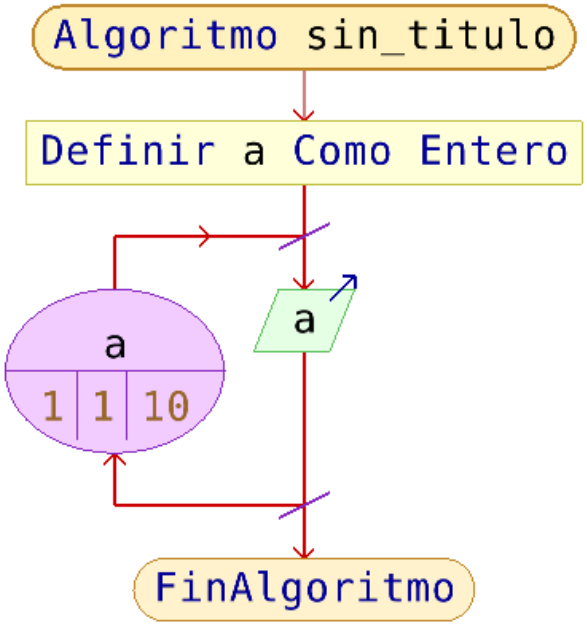
PARA

```
Algoritmo sin_titulo
    Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
        secuencia_de_acciones
    Fin Para
FinAlgoritmo
```

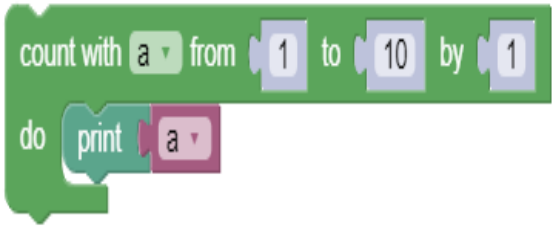
```
Algoritmo sin_titulo
    definir a Como Entero;
    Para a<-1 Hasta 10 Con Paso 1 Hacer
        Escribir a;
    Fin Para
FinAlgoritmo
```

# ESTRUCTURAS REPETITIVAS

PARA




```java
public class ForExample {
    public static void main(String[] args) {
        for (int a = 0; a < 10; a++) {
            System.out.println(a);
        }
    }
}
```



# ESTRUCTURAS REPETITIVAS

CASOS ESPECIALES

foreach

```
var fibNumbers = new List<int> { 0, 1, 1, 2, 3, 5, 8, 13 };
int count = 0;
foreach (int element in fibNumbers) {
        count++;
        Console.WriteLine($"Element #{count}:{element}");
}
Console.WriteLine($"Number of elements: {count}");
```

# ESTRUCTURAS REPETITIVAS

ANIDADOS

```
Algoritmo sin_titulo
    Mientras expresion_logica Hacer
        Mientras expresion_logica Hacer
            Mientras expresion_logica Hacer
                secuencia_de_acciones
            Fin Mientras
        Fin Mientras
    Fin Mientras
FinAlgoritmo
```

```
Algoritmo sin_titulo
    Repetir
        Repetir
            Repetir
                secuencia_de_acciones
            Hasta Que expresion_logica
        Hasta Que expresion_logica
    Hasta Que expresion_logica
FinAlgoritmo
```

# ESTRUCTURAS REPETITIVAS

ANIDADOS

```
Algoritmo sin_titulo
    Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
        Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
            Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
                secuencia_de_acciones
            Fin Para
        Fin Para
    Fin Para
FinAlgoritmo
```

# ESTRUCTURAS REPETITIVAS

ANIDADOS

```
Algoritmo sin_titulo
    Mientras expresion_logica Hacer
        Repetir
            Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
                Si expresion_logica Entonces
                    acciones_por_verdadero
                SiNo
                    acciones_por_falso
                Fin Si
            Fin Para
        Hasta Que expresion_logica
    Fin Mientras
FinAlgoritmo
```

# EJEMPLO

SE DESEA CALCULAR EL PROMEDIO DE LOS ALUMNOS DE FUNDAMENTOS DE PROGRAMACION DE LA COMISION “A”.

PARA CALCULAR EL PROMEDIO SE DEBE DIVIDIR LA CANTIDAD ACUMULADA DE EDADES SOBRE LA CANTIDAD DE ALUMNOS.

VARIABLES DE ENTRADA: EDAD

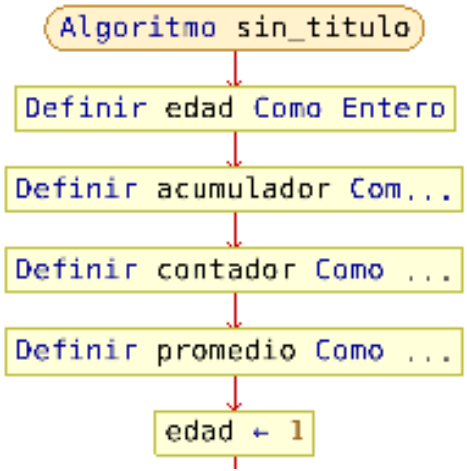
VARIABLES DE PROCESO: ACUMULADOR CONTADOR

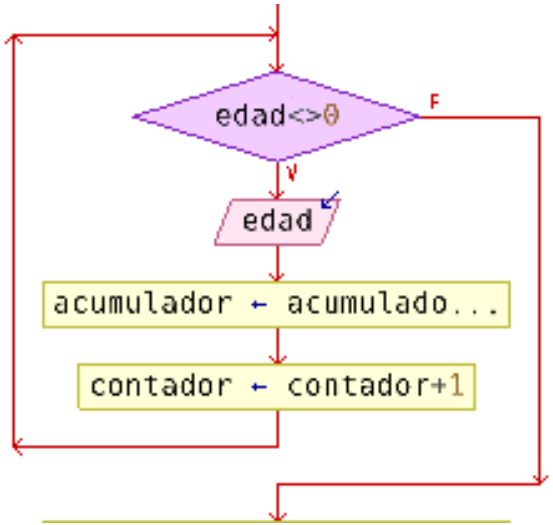
VARIABLES DE SALIDA: PROMEDIO

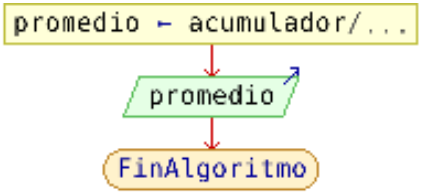
## EJEMPLO

```
Algoritmo sin_titulo
    Definir edad Como Entero;
    Definir acumulador Como Entero;
    Definir contador Como Entero;
    Definir promedio Como Real;
    edad = 1;
    Mientras edad <> 0 Hacer
        Leer edad;
        acumulador = acumulador + edad;
        contador = contador + 1;
    Fin Mientras
    promedio = acumulador/contador;
    Escribir  promedio;
FinAlgoritmo
```

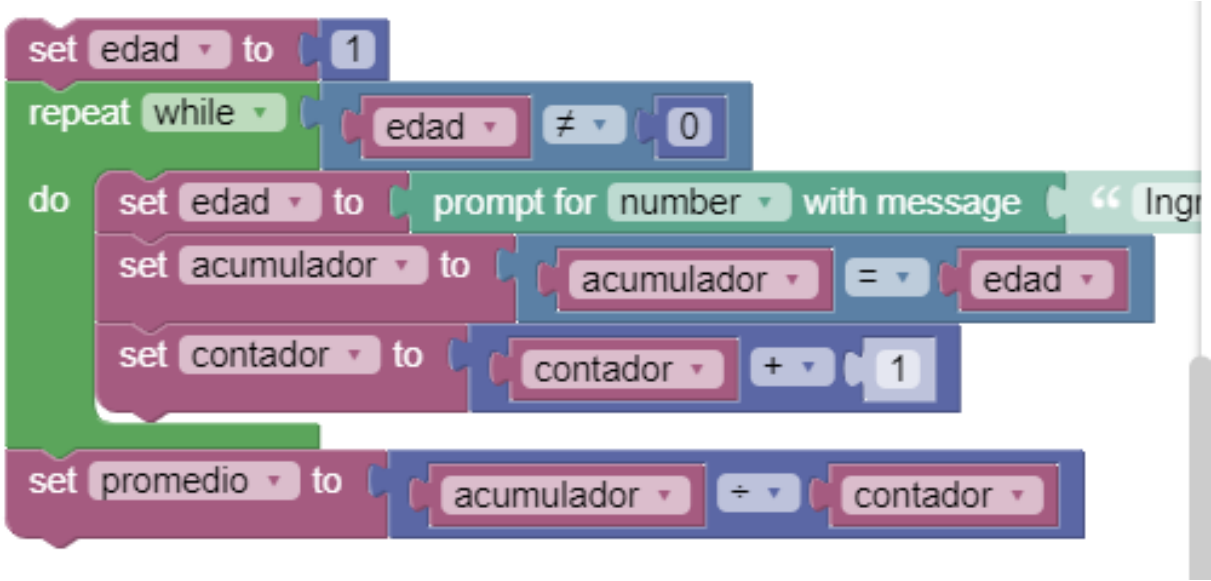
# EJEMPLO


Algoritmo sin_titulo
Definir edad Como Entero
Definir acumulador Com...
Definir contador Como ...
Definir promedio Como ...
edad ← 1


edad<>0
F
V
edad
acumulador ← acumulado...
contador ← contador+1


promedio ← acumulador/...
promedio
FinAlgoritmo

# EJEMPLO


set edad to 1
repeat while edad ≠ 0
do
set edad to prompt for number with message "Ingr
set acumulador to acumulador = edad
set contador to contador + 1
set promedio to acumulador ÷ contador

# EJEMPLO

```
var edad, promedio, acumulador, contador;


edad = 1;
while (edad != 0) {
  edad = Number(window.prompt('Ingresar Edad'));
  acumulador = acumulador == edad;
  contador = contador + 1;
}
promedio = acumulador / contador;
```

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# VECTORES

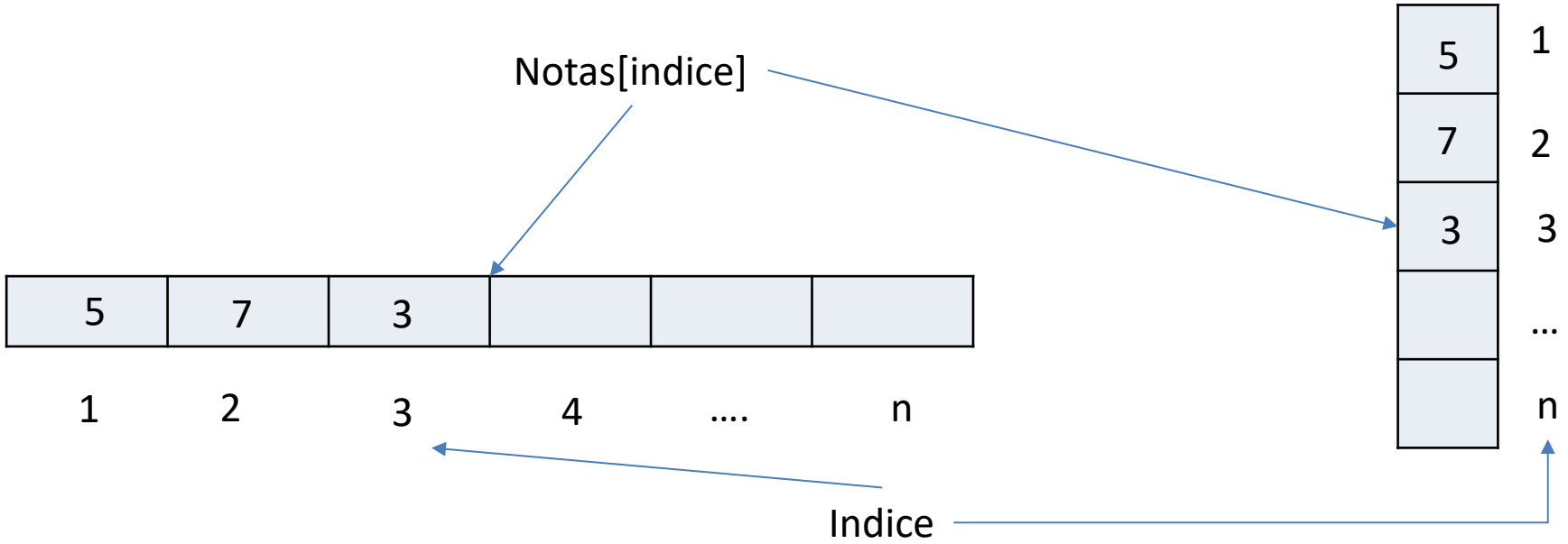
Un vector es una agrupación de variables en un contenedor que posee un nombre genérico y se recorre por medio de un índice

Supongamos que tenemos que ingresar en una aplicación las notas de 25 alumnos y mantenerlos en memoria con el fin de realizar distintos tipos de operaciones

Entonces tendríamos: nota1, nota2, nota3,…..nota25

Con el fin de no utilizar muchas variables, se las agrupa en un concepto denominado VECTORES

# VECTORES



# VECTORES

Los índices se pueden representar como:

Constante : Vector[5]

Variable : Vector[i]

Formula : Vector[5+i]

```
Algoritmo Vectores
    Dimension  vector(5);

    Para i<- 1 Hasta 5 Con Paso 1 Hacer
        Leer vector(i);
    Fin Para

    Para i<-1 Hasta 5 Con Paso 5 Hacer
        Escribir i;
        Escribir vector(i);
    Fin Para
FinAlgoritmo
```

Tamaño

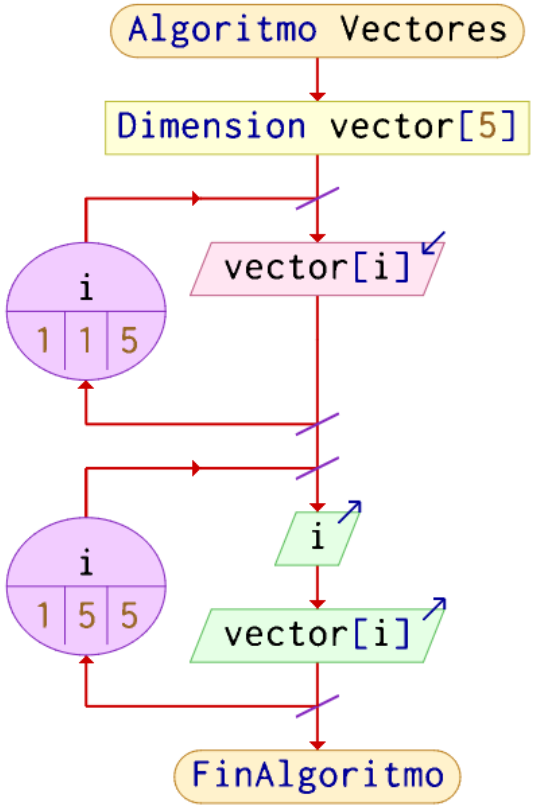
Definición del Vector

Nombre

Ingresar valores al Vector

Índice

Visualizar contenido del Vector

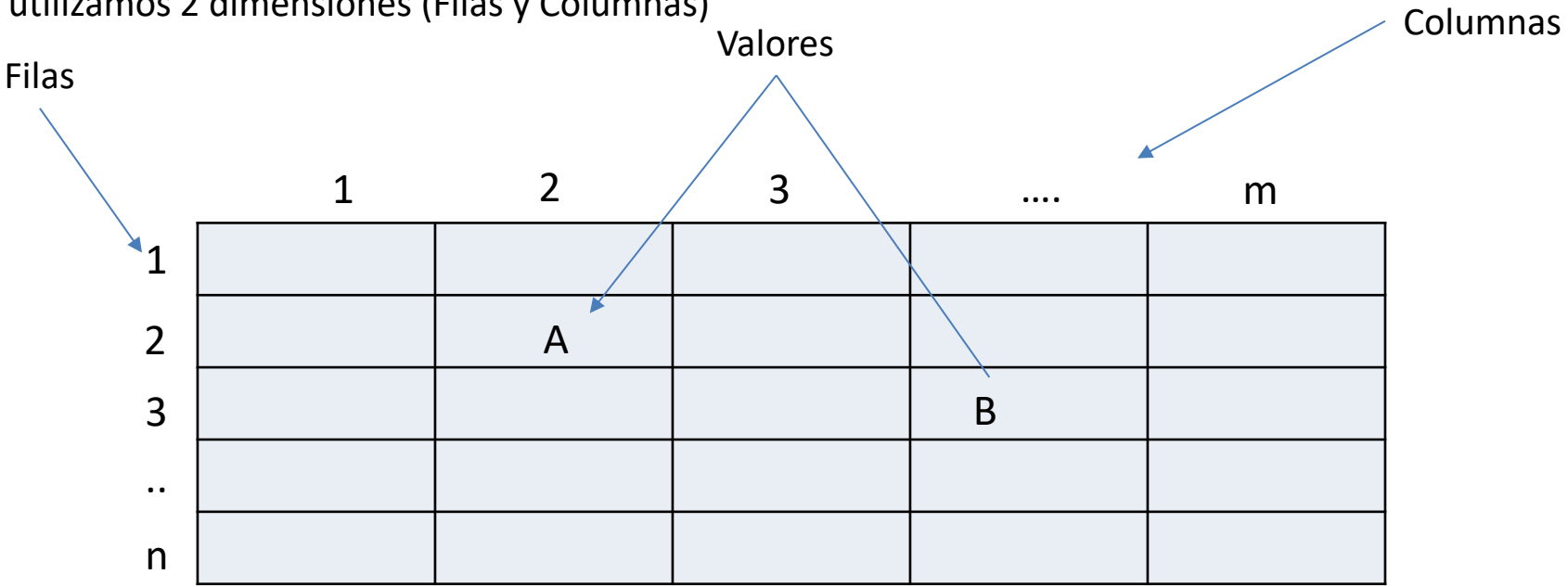

Algoritmo Vectores
Dimension vector[5]
i
1 1 5
vector[i]
i
1 5 5
i
vector[i]
FinAlgoritmo

```
Algoritmo Vectores
    Dimension  vector(5);

    Para i<- 1 Hasta 5 Con Paso 1 Hacer
        Leer vector(i);
    Fin Para

    Para i<-1 Hasta 5 Con Paso 5 Hacer
        Escribir i;
        Escribir vector(i);
    Fin Para
FinAlgoritmo
```

| Vector i | i |
|---|---|
| 7 | 1 |
| 8 | 2 |
| 6 | 3 |
| 5 | 4 |
| 7 | 5 |

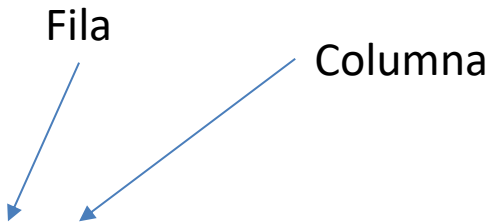
# MATRICES

Se repite en concepto visto en VECTORES, en la cual agrupamos variables en un con tenedor, pero esta vez utilizamos 2 dimensiones (Filas y Columnas)



# MATRICES

Los índices se pueden representar como:




Constante : Matriz[5,6]

Variable : Matriz[i,j]

Formula : Matriz[5+i,j+k]

# MATRICES

Bucle Columnas

Bucle Filas

```
//Recorrido por Fila
Para j← 1 Hasta 6 Con Paso 1 Hacer
    Para i← 1 Hasta 5 Con Paso 1 Hacer
        Escribir i;
        Escribir j;
        Escribir Matriz[i,j];
    Fin Para
Fin Para
```

# MATRICES

Bucle Filas

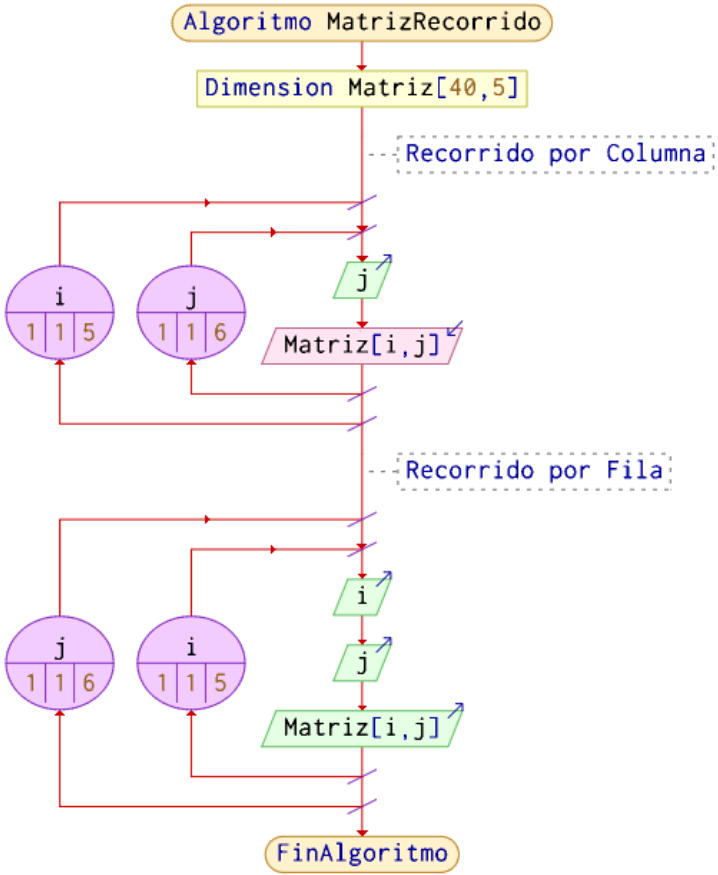
Bucle Columnas

```
//Recorrido por Columna
Para i← 1 Hasta 5 Con Paso 1 Hacer
    Para j← 1 Hasta 6 Con Paso 1 Hacer
        Escribir i;
        Escribir j;
        Escribir Matriz[i,j];
    Fin Para
Fin Para
```

```
Algoritmo MatrizRecorrido
    Dimension Matriz[5,6];
    Matriz[1,1] ← 11;Matriz[1,2] ← 12;Matriz[1,3] ← 13;Matriz[1,4] ← 14;Matriz[1,5] ← 15;Matriz[1,6] ← 16;
    Matriz[2,1] ← 21;Matriz[2,2] ← 22;Matriz[2,3] ← 23;Matriz[2,4] ← 24;Matriz[2,5] ← 25;Matriz[2,6] ← 26;
    Matriz[3,1] ← 31;Matriz[3,2] ← 32;Matriz[3,3] ← 33;Matriz[3,4] ← 34;Matriz[3,5] ← 35;Matriz[3,6] ← 36;
    Matriz[4,1] ← 41;Matriz[4,2] ← 42;Matriz[4,3] ← 43;Matriz[4,4] ← 44;Matriz[4,5] ← 45;Matriz[4,6] ← 46;
    Matriz[5,1] ← 51;Matriz[5,2] ← 52;Matriz[5,3] ← 53;Matriz[5,4] ← 54;Matriz[5,5] ← 55;Matriz[5,6] ← 56;

    //Recorrido por Columna
    Para i← 1 Hasta 5 Con Paso 1 Hacer
        Para j← 1 Hasta 6 Con Paso 1 Hacer
            Escribir i;
            Escribir j;
            Escribir Matriz[i,j];
        Fin Para
    Fin Para

    //Recorrido por Fila
    Para j← 1 Hasta 6 Con Paso 1 Hacer
        Para i← 1 Hasta 5 Con Paso 1 Hacer
            Escribir i;
            Escribir j;
            Escribir Matriz[i,j];
        Fin Para
    Fin Para
FinAlgoritmo
```

Declaración Matriz

Inicialización Matriz


Algoritmo MatrizRecorrido
Dimension Matriz[40,5]
Recorrido por Columna
i
1 1 5
j
1 1 6
j
Matriz[i,j]
Recorrido por Fila
j
1 1 6
i
1 1 5
i
j
Matriz[i,j]
FinAlgoritmo

```
Matriz[1,1] ← 11;Matriz[1,2] ← 12;Matriz[1,3] ← 13;Matriz[1,4] ← 14;Matriz[1,5] ← 15;Matriz[1,6] ← 16;
Matriz[2,1] ← 21;Matriz[2,2] ← 22;Matriz[2,3] ← 23;Matriz[2,4] ← 24;Matriz[2,5] ← 25;Matriz[2,6] ← 26;
Matriz[3,1] ← 31;Matriz[3,2] ← 32;Matriz[3,3] ← 33;Matriz[3,4] ← 34;Matriz[3,5] ← 35;Matriz[3,6] ← 36;
Matriz[4,1] ← 41;Matriz[4,2] ← 42;Matriz[4,3] ← 43;Matriz[4,4] ← 44;Matriz[4,5] ← 45;Matriz[4,6] ← 46;
Matriz[5,1] ← 51;Matriz[5,2] ← 52;Matriz[5,3] ← 53;Matriz[5,4] ← 54;Matriz[5,5] ← 55;Matriz[5,6] ← 56;
```

```
//Recorrido por Columna
Para i← 1 Hasta 5 Con Paso 1 Hacer
    Para j← 1 Hasta 6 Con Paso 1 Hacer
        Escribir i;
        Escribir j;
        Escribir Matriz[i,j];
    Fin Para
Fin Para
```

| i | j | Matriz[i,j] |
|---|---|---|
| 1 | 1 | 11 |
| | 2 | 12 |
| | 3 | 13 |
| | 4 | 14 |
| | 5 | 15 |
| | 6 | 16 |
| 2 | 1 | 21 |

```
Matriz[1,1] ← 11;Matriz[1,2] ← 12;Matriz[1,3] ← 13;Matriz[1,4] ← 14;Matriz[1,5] ← 15;Matriz[1,6] ← 16;
Matriz[2,1] ← 21;Matriz[2,2] ← 22;Matriz[2,3] ← 23;Matriz[2,4] ← 24;Matriz[2,5] ← 25;Matriz[2,6] ← 26;
Matriz[3,1] ← 31;Matriz[3,2] ← 32;Matriz[3,3] ← 33;Matriz[3,4] ← 34;Matriz[3,5] ← 35;Matriz[3,6] ← 36;
Matriz[4,1] ← 41;Matriz[4,2] ← 42;Matriz[4,3] ← 43;Matriz[4,4] ← 44;Matriz[4,5] ← 45;Matriz[4,6] ← 46;
Matriz[5,1] ← 51;Matriz[5,2] ← 52;Matriz[5,3] ← 53;Matriz[5,4] ← 54;Matriz[5,5] ← 55;Matriz[5,6] ← 56;
```

```
//Recorrido por Fila
Para j← 1 Hasta 6 Con Paso 1 Hacer
    Para i← 1 Hasta 5 Con Paso 1 Hacer
        Escribir i;
        Escribir j;
        Escribir Matriz[i,j];
    Fin Para
Fin Para
```

| i | j | Matriz[i,j] |
|---|---|---|
| 1 | 1 | 11 |
| 2 | | 21 |
| 3 | | 31 |
| 4 | | 41 |
| 5 | | 51 |
| 1 | 2 | 12 |

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

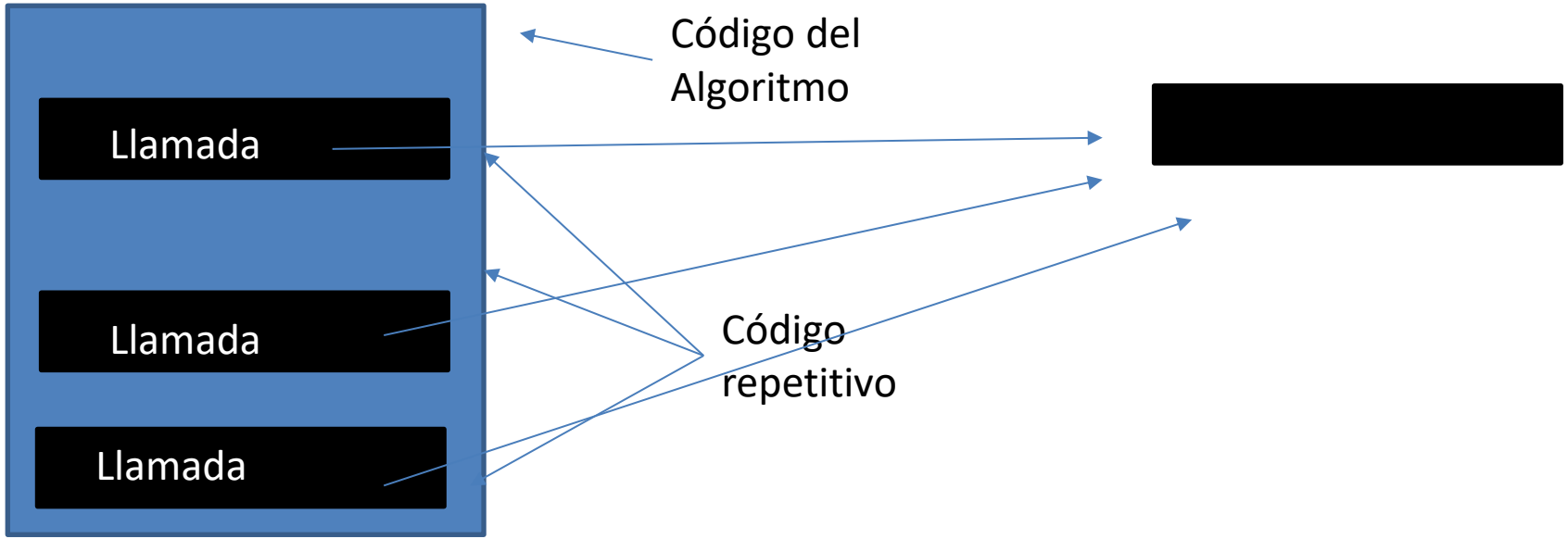
Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# SUB PROGRAMAS

El concepto de Sub Programas, es aplicable también a los conocidos como Funciones, Procedimientos, Store Procedures, Web Services, etc



# SUB PROGRAMAS

Reducir la complejidad del programa (“divide y vencerás”).

Eliminar código duplicado.

Limitar los efectos de los cambios (aislar aspectos concretos).

Ocultar detalles de implementación (p.ej. algoritmos complejos).

Promover la reutilización de código

Mejorar la legibilidad del código.

Facilitar la portabilidad del código.

```
Funcion r <- suma ( n1, n2 )
     r ← n1 + n2;
Fin Funcion

Algoritmo Subprogramas
     Definir a Como Entero;
     Definir b Como Entero;
     Definir c Como Entero;

     Leer a;
     Leer b;

     c ← suma(a,b);

     Escribir  c;
FinAlgoritmo
```

Nombre del Subprograma

Parámetro de salida

Parámetros de entrada

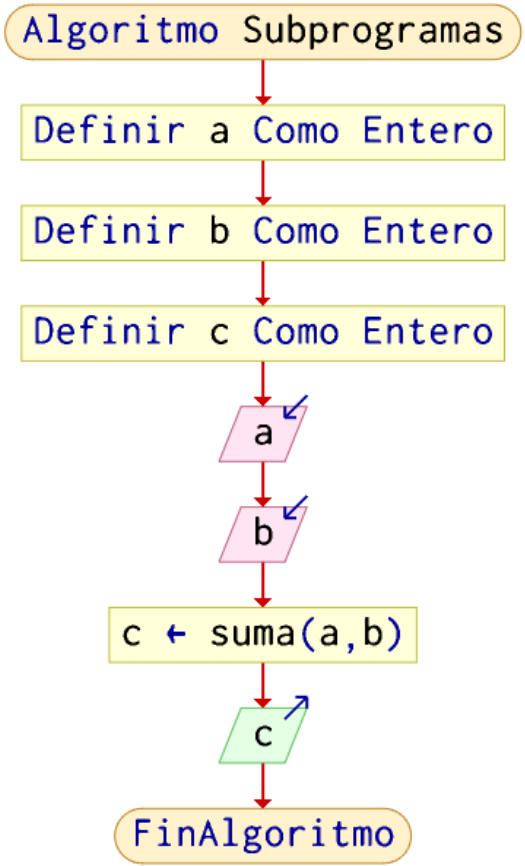
Cuerpo del Subprograma

Programa Principal

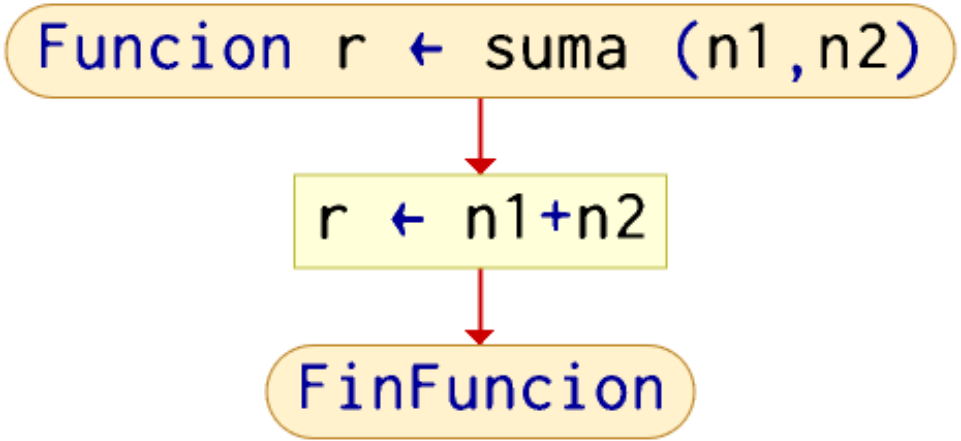
Llama al Subprograma

```
1  Funcion r <- suma ( n1,n2 )
2      r ← n1 + n2;
3  Fin Funcion
4
5  Algoritmo Recu
6      Definir a Como Entero;
7      Definir b Como Entero;
8      Definir c Como Entero;
9
10     Leer a;
11     Leer b;
12
13     c ← suma(a,b);
14
15     Escribir c;
16 FinAlgoritmo
17
```

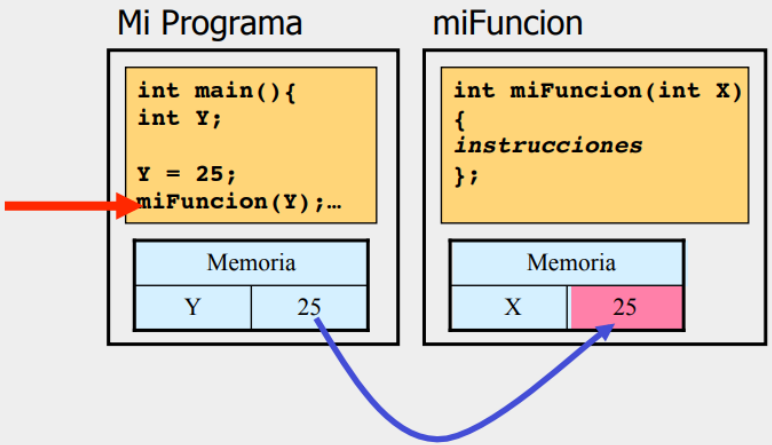
| a | b | c | n1 | n2 | r |
|---|---|---|---|---|---|
| 5 | 10 | 15 | 5 | 10 | 15 |


Algoritmo Subprogramas
Definir a Como Entero
Definir b Como Entero
Definir c Como Entero
a
b
c ← suma(a,b)
c
FinAlgoritmo

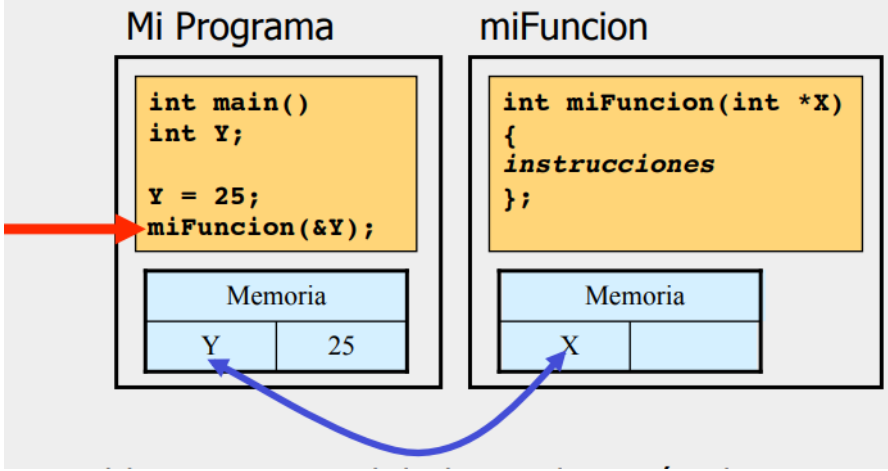
c <- suma(a,b)


Funcion r ← suma (n1,n2)
r ← n1+n2
FinFuncion

# Parámetros por Valor



# Parámetros por Referencia



```
    public Personas(int dni,String nombre,String apellido){
        this.dni = dni;
        this.nombre = nombre;
        this.apellido = apellido;
    }

    public String getApellido(){ //getter
        return this.apellido;
    }

    public String getNombre(){ //getter
        return this.nombre;
    }

    public void setNombre(String nombre){ //setter
        this.nombre = nombre;
    }

    public void setApellido(String apellido){ //setter
        this.nombre = apellido;
    }

    public int CalculoSuma(int a, int b){
        return a+b;
    }
}
```

Tipos de Metodos

Constructores

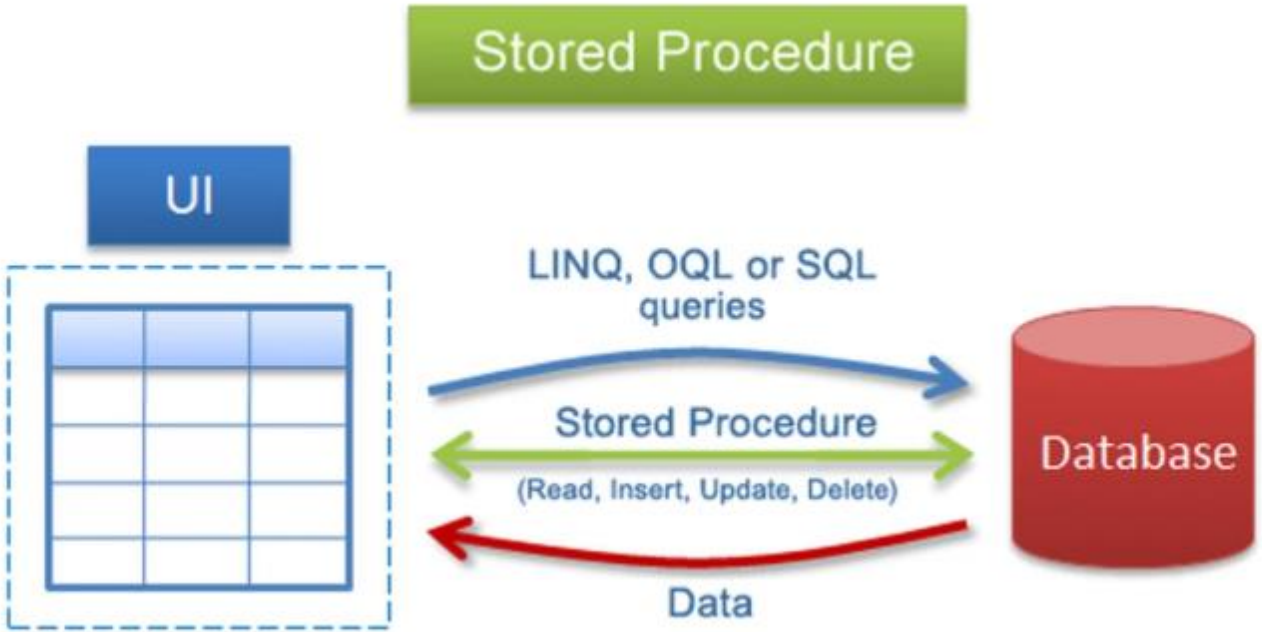
Parametros de Entrada

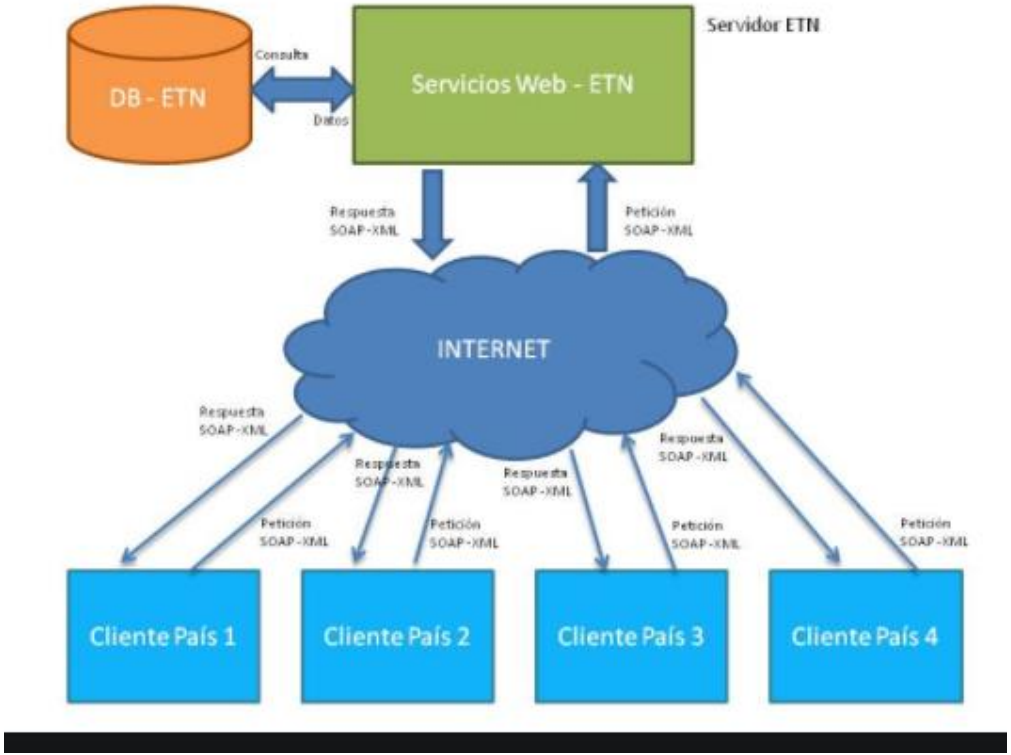
Getter

Parametro de Salida

Setter

Tradicional


Stored Procedure
UI
LINQ, OQL or SQL queries
Stored Procedure
(Read, Insert, Update, Delete)
Data
Database


Servidor ETN
DB - ETN
Consulta
Datos
Servicios Web - ETN
Respuesta SOAP-XML
Petición SOAP-XML
INTERNET
Respuesta SOAP-XML
Respuesta SOAP-XML
Respuesta SOAP-XML
Respuesta SOAP-XML
Petición SOAP-XML
Petición SOAP-XML
Petición SOAP-XML
Petición SOAP-XML
Cliente País 1
Cliente País 2
Cliente País 3
Cliente País 4

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

Gobierno de Misiones

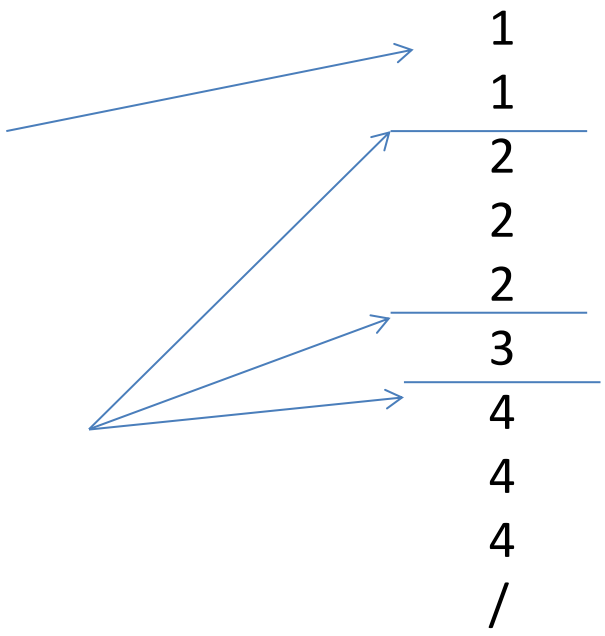
# INTRODUCCIÓN A LA *programación*

polotic
misiones

# CORTE DE CONTROL

Dado una lista de valores
que se encuentran ordenados

El algotitmo debe detectar cuando
en la secuencia ordenada tienen un
cambio en la variable que identifica
A un conjunto de registros



# CORTE DE CONTROL

Supongamos que tenemos los registros de alumnos de una institucion y debemos calcular el promedio de notas de cada alumno.

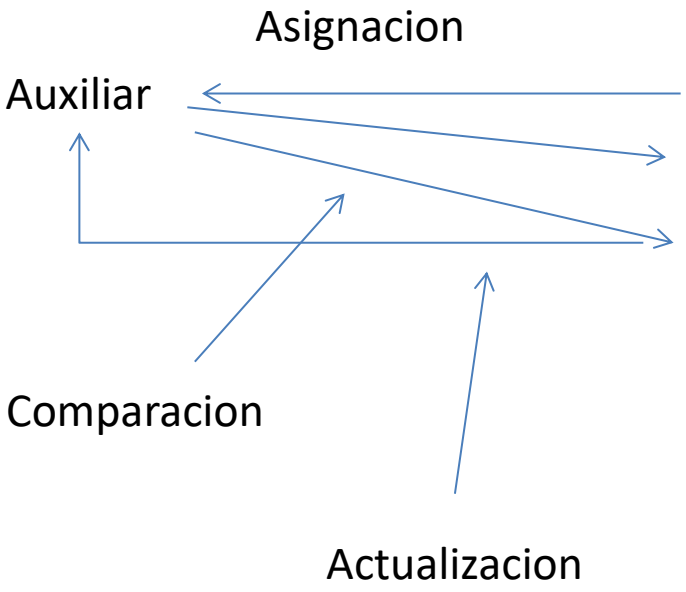
Para poder desarrollar el algoritmo Necesitamos:

- Una variable Auxiliar que nos permita comparar las variables codigo de todos los registros de los alumnos.
- Y utilizamos el conceptos visto en Clase conocido como centinela o Flag o Bandera

| Codigo | Nota |
|---|---|
| 1 | 7 |
| 1 | 8 |
| 2 | 10 |
| 2 | 5 |
| 2 | 9 |
| 3 | 7 |
| 4 | 6 |
| 4 | 10 |
| / | / |

# CORTE DE CONTROL

La variable auxiliar toma el primer valor y lo compara con el segundo, cuando el corte se detecta se actualiza el auxiliar




| Codigo | Nota |
|---|---|
| 1 | 7 |
| 1 | 8 |
| 2 | 10 |
| 2 | 5 |
| 2 | 9 |
| 3 | 7 |
| 4 | 6 |
| 4 | 10 |
| / | / |

# CORTE DE CONTROL

| | | | | |
|---|---|---|---|---|
| Variables de Entrada | Codigo Alumno | Nota | | |
| Variables de Proceso | Flag | Auxiliar | Contador Nota | Acumulador Nota |
| Variables de Salida | Promedio Alumno | | | |

# CORTE DE CONTROL

```
Algoritmo CorteControl
    Definir codigoAlumno Como Entero;
    Definir notaAlumno Como Entero;
    Definir flag Como Logico;
    Definir auxiliar Como Entero;
    Definir contadorNota Como Entero;
    Definir acumuladorNota Como Entero;
    Definir promedioNota Como Real;
```

Primer Paso: Definimos el tipo de las variables autilizar

-1

```
    codigoAlumno <-0;
    flag <- Verdadero;
    contadorNota <-0;
    acumuladorNota <-0;
```

Segundo Paso: Inicializamos las variables que no tienen una inicalizacion implicita en una asignacion

```
    Mientras codigoAlumno <> 0 hacer

    FinMientras
```

Tercer Paso: Elegimos la estructura repetitiva y colocamos la condicion de salida

# CORTE DE CONTROL

```
Mientras codigoAlumno <> 0 hacer
    leer codigoAlumno;
    leer notaAlumno;
    si flag = verdadero entonces
        flag <- falso;
        auxiliar <- codigoAlumno;
    FinSi
```

Cuarto Paso: Flujo de la consigna del algoritmo

Leer las variables de entrada

Aplicar el concepto de Centinela, Flag o Swich

# CORTE DE CONTROL

```
    si codigoAlumno = auxiliar Entonces
        contadorNota <- contadorNota + 1;
        acumuladorNota <- acumuladorNota + notaAlumno;
    SiNo
        promedioNota <- acumuladorNota/contadorNota;
        Escribir auxiliar;
        Escribir promedioNota;
        promedioNota <- 0;
        acumuladorNota <- notaAlumno;
        contadorNota <- 1;
        auxiliar = codigoAlumno;
    FinSi
FinMientras
```

Corte de Control

Proceso

Fin del Corte

Calculo del Promedio

Variables de Salida

Inicializacion de variables para el proximo Alumno

# CORTE DE CONTROL

```
FinMientras
promedioNota <- acumuladorNota/contadorNota;
Escribir auxiliar;
Escribir promedioNota;
```

Salida del Algoritmo

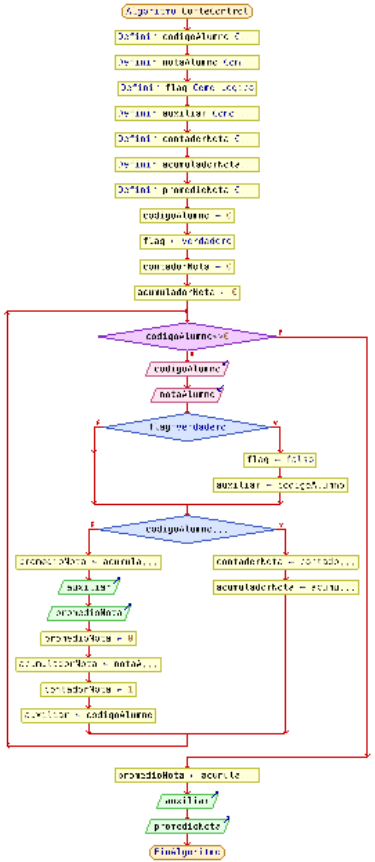
Informe del promedio del ultimo Alumno

```
codigoAlumno <-0;
flag <- Verdadero;
contadorNota <-0;
acumuladorNota <-0;
Mientras codigoAlumno <> 0 hacer
    leer codigoAlumno;
    leer notaAlumno;
    si flag = verdadero entonces
        flag <- falso;
        auxiliar <- codigoAlumno;
    FinSi
    si codigoAlumno = auxiliar Entonces
        contadorNota <- contadorNota + 1;
        acumuladorNota <- acumuladorNota + notaAlumno;
    SiNo
        promedioNota <- acumuladorNota/contadorNota;
        Escribir auxiliar;
        Escribir promedioNota;
        promedioNota <- 0;
        acumuladorNota <- notaAlumno;
        contadorNota <- 1;
        auxiliar = codigoAlumno;
    FinSi
FinMientras
promedioNota <- acumuladorNota/contadorNota;
Escribir auxiliar;
Escribir promedioNota;
```

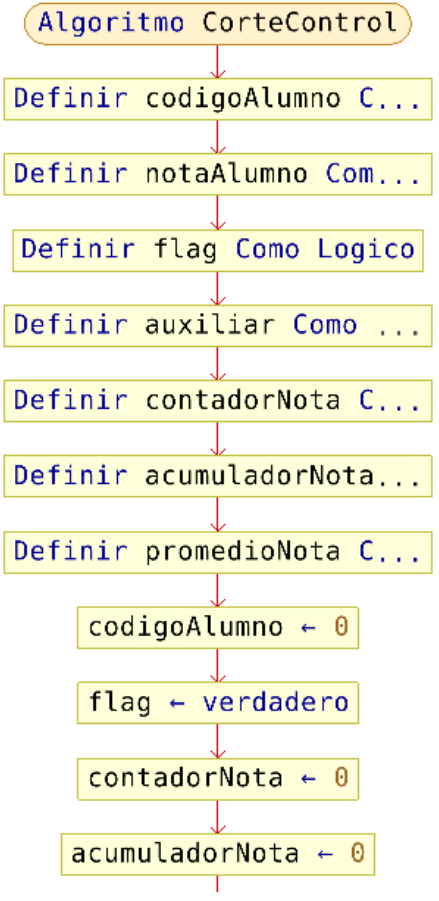
-1

| Codigo | Nota | Flag | Aux | Ctador | Acum | Prom |
|---|---|---|---|---|---|---|
| -1 | - | V | 1 | 0 | 0 | 7,5 |
| 1 | 7 | F | 2 | 1 | 7 | 0 |
| 1 | 8 | | 3 | 2 | 15 | 8 |
| 2 | 10 | | 4 | 1 | 10 | 0 |
| 2 | 5 | | | 2 | 15 | 7 |
| 2 | 9 | | | 3 | 24 | 0 |
| 3 | 7 | | | 1 | 7 | 8 |
| 4 | 6 | | | 1 | 6 | |
| 4 | 10 | | | 2 | 16 | |
| 0 | | | | | | |

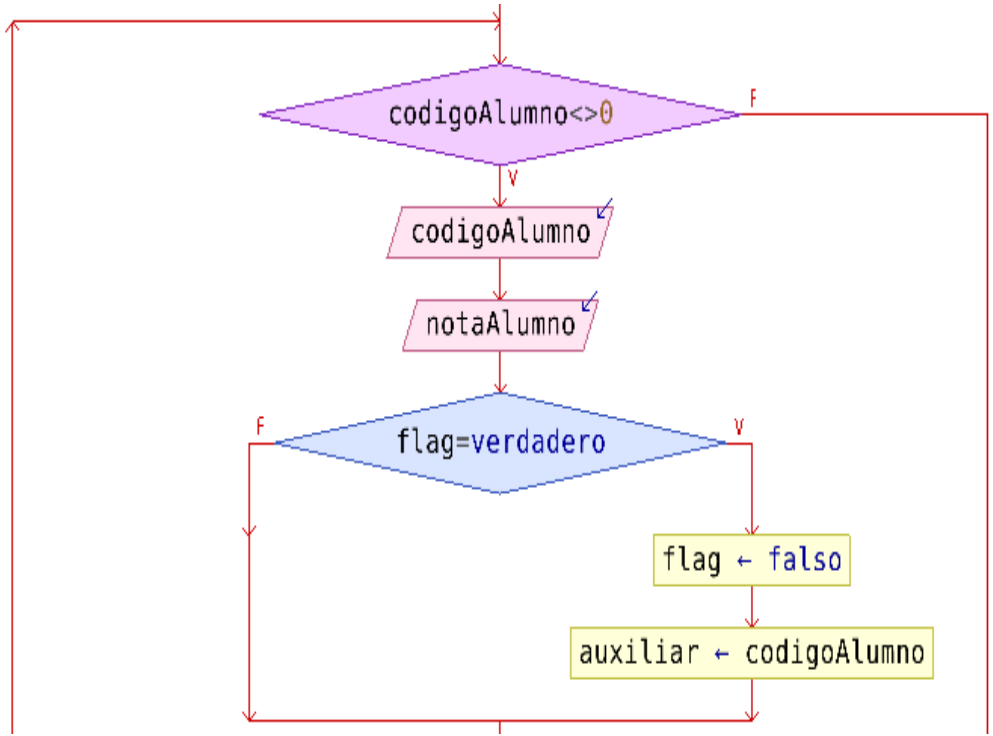
# CORTE DE CONTROL



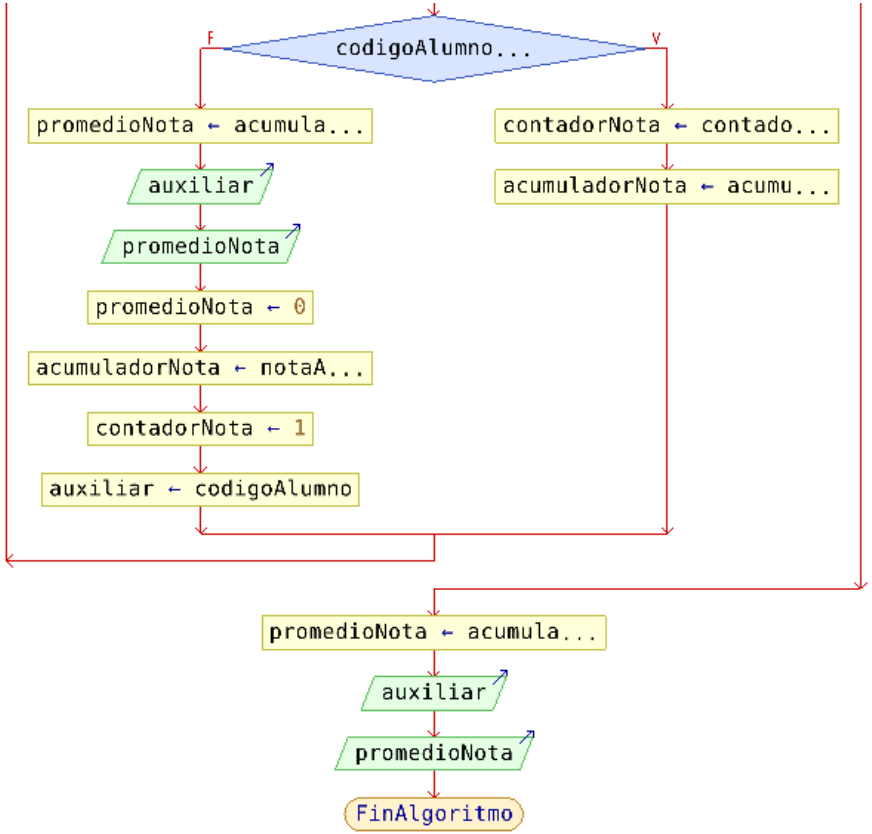
# CORTE DE CONTROL



# CORTE DE CONTROL



# CORTE DE CONTROL



# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# BURBUJA

Metodo creado para ordenar vectores, de mayor a menor O menor a mayor

Antes del Metodo Burbuja

| 7 |
|---|
| 1 |
| 3 |
| 6 |

Despues del Metodo Burbuja

| 1 |
|---|
| 3 |
| 6 |
| 7 |

```
Algoritmo Vectores
    Dimension  vector[5];
    Definir aux Como Entero;

    Para i← 1 Hasta 5 Con Paso 1 Hacer
        Leer Vector[i];
    Fin Para

    Para i←1 Hasta 5 Con Paso 1 Hacer
        Para j←1 Hasta 4 Con Paso 1 Hacer
            si Vector[j] > Vector[j+1] Entonces
                aux ← Vector[j];
                Vector[j] ← Vector[j+1];
                Vector[j+1] ← aux;
            FinSi
        Fin Para
    Fin Para

    Para i← 1 Hasta 5 Con Paso 1 Hacer
        Escribir  Vector[i];
    Fin Para

FinAlgoritmo
```

# BURBUJA

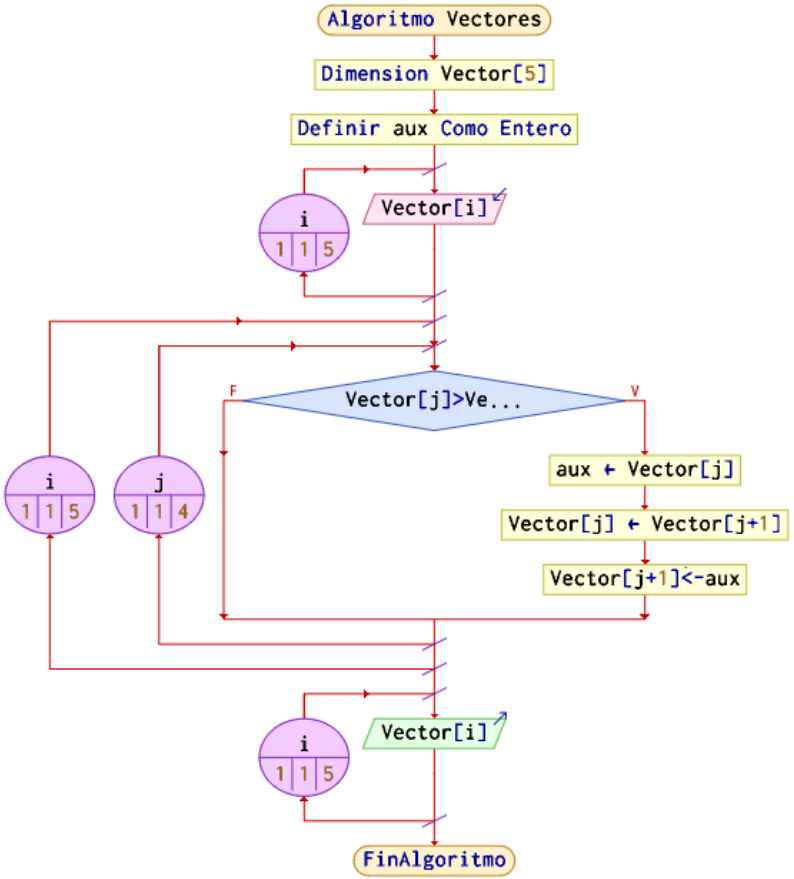
Se utilizan 2 estructuras repetitivas, las cuales recorren el vector N cantidad de veces comparando el n con el n+1 para ordenarlos de mayor a menor o viceversa, según el criterio

```
Algoritmo Vectores
    Dimension  vector[5];
    Definir aux Como Entero;

    Para i← 1 Hasta 5 Con Paso 1 Hacer
        Leer Vector[i];
    Fin Para

    Para i←1 Hasta 5 Con Paso 1 Hacer
        Para j←1 Hasta 4 Con Paso 1 Hacer
            si Vector[j] > Vector[j+1] Entonces
                aux ← Vector[j];
                Vector[j] ← Vector[j+1];
                Vector[j+1] ← aux;
            FinSi
        Fin Para
    Fin Para

    Para i← 1 Hasta 5 Con Paso 1 Hacer
        Escribir  Vector[i];
    Fin Para

FinAlgoritmo
```

| i | j | Vector j | Vector j+1 | aux |
|---|---|---|---|---|
| 1 | 1 | 5 | 4 | 5 |
| | 2 | 4 | 5 | 5 |
| | | 5 | 3 | |
| | | 3 | 5 | |
| | | | | |

| Vector | Vector | Vector |
|---|---|---|
| 5 | 4 | 4 |
| 4 | 5 | 3 |
| 3 | 3 | 5 |
| 2 | 2 | 2 |
| 1 | 1 | 1 |

# BURBUJA


Algoritmo Vectores
Dimension Vector[5]
Definir aux Como Entero
i
1 1 5
Vector[i]
F
Vector[j]>Ve...
V
i
1 1 5
j
1 1 4
aux ← Vector[j]
Vector[j] ← Vector[j+1]
Vector[j+1]<-aux
i
1 1 5
Vector[i]
FinAlgoritmo

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

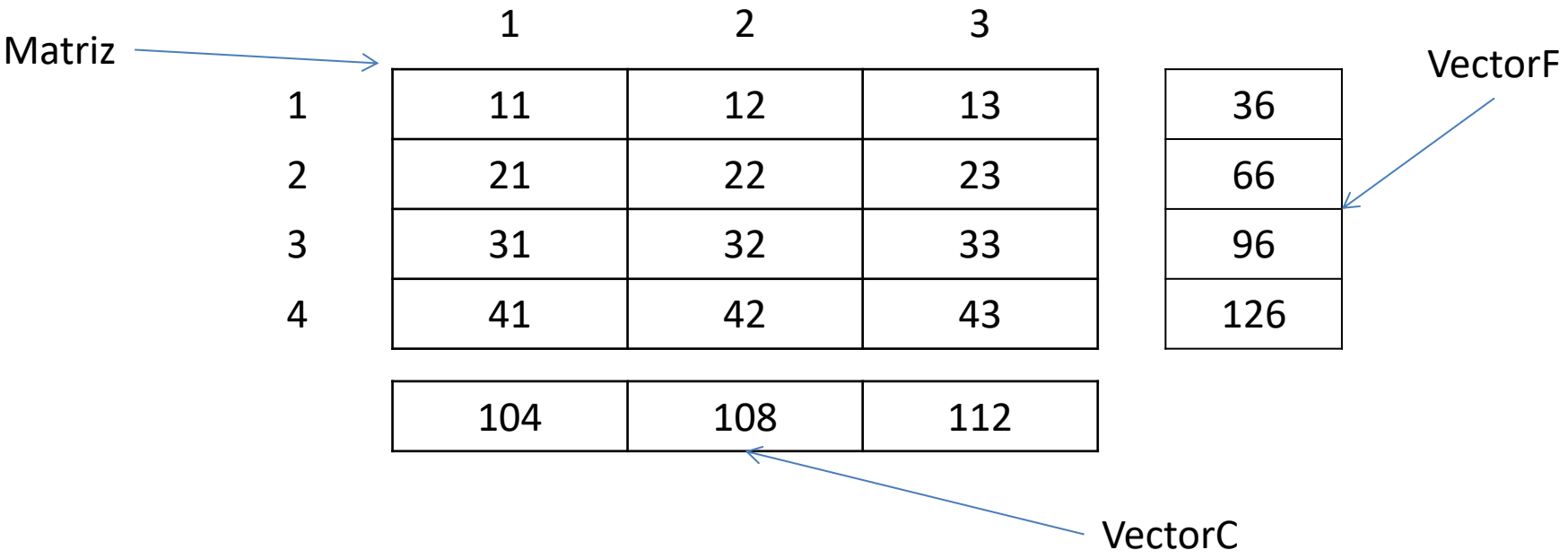
polotic misiones

Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# MATRICES Y VECTORES



# MATRICES Y VECTORES

```
1   Algoritmo sin_titulo
2       Dimension Matriz[4,3];
3       Dimension VectorF[4];
4       Dimension VectorC[3];
5
6       Definir i como Entero;
7       Definir j como Entero;
8
9       Para i<-1 Hasta 4 Con Paso 1 Hacer
10          Para j<-1 Hasta 3 Con Paso 1 Hacer
11              Leer Matriz[i,j];
12          Fin Para
13      Fin Para
```

Inicializacion y Declaracion

Lectura de Matriz

# MATRICES Y VECTORES

```
15      Para i<-1 Hasta 4 Con Paso 1 Hacer
16          Para j<-1 Hasta 3 Con Paso 1 Hacer
17              VectorC[j] = VectorC[j] + Matriz[i,j];
18          Fin Para
19      Fin Para
20
21      Para j<-1 Hasta 3 Con Paso 1 Hacer
22          Para i<-1 Hasta 4 Con Paso 1 Hacer
23              VectorF[i] = VectorF[i] + Matriz[i,j];
24          Fin Para
25      Fin Para
26
27      Para i<-1 Hasta 4 Con Paso 1 Hacer
28          Escribir VectorF[i];
29      Fin Para
30
31      Para j<-1 Hasta 3 Con Paso 1 Hacer
32          Escribir VectorC[j];
33      Fin Para
34
35  FinAlgoritmo
```

Calculo de Sumatorias
En Vectores

Informe de Vectores

# MATRICES Y VECTORES

```
14
15      Para i<-1 Hasta 4 Con Paso 1 Hacer
16          Para j<-1 Hasta 3 Con Paso 1 Hacer
17              VectorC[j] = VectorC[j] + Matriz[i,j];
18          Fin Para
19      Fin Para
20
```

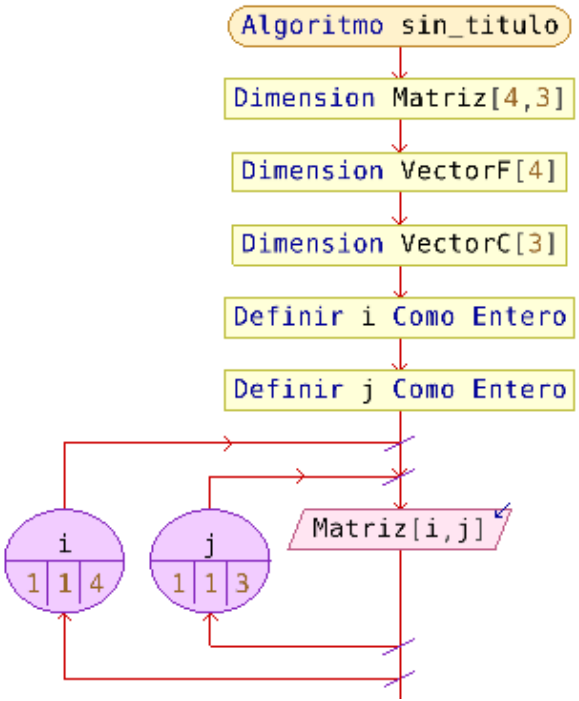
| I | J | VectorC | Matriz |
|---|---|---|---|
| 1 | 1 | 11 | 11 |
| | 2 | 23 | 12 |
| | 3 | 36 | 13 |
| 2 | 1 | 21 | 21 |
| | 2 | 43 | 22 |
| | 3 | 66 | 23 |

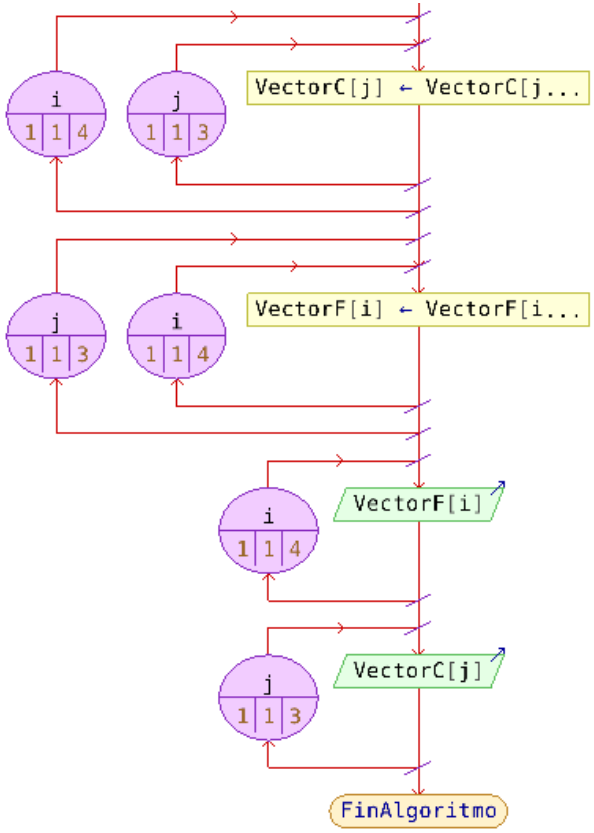
# MATRICES Y VECTORES

```
21      Para j<-1 Hasta 3 Con Paso 1 Hacer
22          Para i<-1 Hasta 4 Con Paso 1 Hacer
23              VectorF[i] = VectorF[i] + Matriz[i,j];
24          Fin Para
25      Fin Para
```

| I | J | VectorF | Matriz |
|---|---|---|---|
| 1 | 1 | 11 | 11 |
| 2 | | 32 | 21 |
| 3 | | 63 | 31 |
| 4 | | 104 | 41 |
| 1 | 2 | | |
| | | | |

# MATRICES Y VECTORES



# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

Gobierno de Misiones

# INTRODUCCIÓN A LA *programación*

polotic
misiones

# APAREO DE ARCHIVOS

Maestro

| | |
|---|---|
| 1 | 15 |
| 3 | 30 |
| 5 | 50 |

| | |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 6 | 60 |

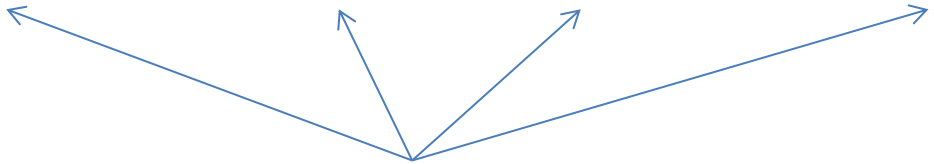
Novedad

| | |
|---|---|
| 1 | 25 |
| 2 | 20 |
| 3 | 30 |
| 4 | 0 |
| 5 | 50 |
| 6 | 60 |

Apareo

# APAREO DE ARCHIVOS

```
Funcion EscribirApareo (cApa por Referencia,vApa Por Referencia,iA Por Referencia,Apareo Por Referencia )
    iA = iA + 1;
    Apareo[iA,1] = cApa;
    Apareo[iA,2] = vApa;
Fin Funcion
```



Valores Por Referencia

# APAREO DE ARCHIVOS

```
Funcion LeerMaestro (cMae por Referencia,vMae Por Referencia,iM Por Referencia,Maestro Por Referencia,FlagM Por Referencia)
    iM = iM + 1;
    Si (iM = 3 y FlagM = Verdadero) Entonces
        cMae = Maestro[iM,1];
        vMae = Maestro[iM,2];
        FlagM = Falso;
    SiNo
        Si (iM < 3) Entonces
            cMae = Maestro[iM,1];
            vMae = Maestro[iM,2];
        FinSi
    FinSi
Fin Funcion
```

Control del Ultimo Registro

# APAREO DE ARCHIVOS

```
Funcion LeerNovedad (cNov por Referencia,vNov Por Referencia,iN Por Referencia,Novedad Por Referencia,FlagN Por Referencia)
    iN = iN + 1;
    Si (iN = 3 y FlagN = Verdadero) Entonces
        cNov = Novedad[iN,1];
        vNov = Novedad[iN,2];
        FlagN = Falso;
    SiNo
        Si (iN < 3) Entonces
            cNov = Novedad[iN,1];
            vNov = Novedad[iN,2];
        FinSi
    FinSi
Fin Funcion
```

# APAREO DE ARCHIVOS

```
Algoritmo ApareoArchivos
    Dimension Maestro[3,2],Novedad[3,2],Apareo[6,2];
    Definir i,j,cMae,vMae,cNov,vNov,cApa,vApa,iM,iN,iA como Entero;
    Definir FlagM, FlagN como Logico;
    iM = 0;
    iN = 0;
    iA = 0;
    FlagN = Verdadero;
    FlagM = Verdadero;
    Para i<-1 Hasta 3 Con Paso 1 Hacer
        Para j<-1 Hasta 2 Con Paso 1 Hacer
            Escribir "Maestro ", i,j;
            Leer Maestro[i,j];
            Escribir "Novedad ", i,j;
            Leer Novedad[i,j];
        Fin Para
    Fin Para
    LeerMaestro(cMae,vMae,iM,Maestro,FlagM);
    LeerNovedad(cNov,vNov,iN,Novedad,FlagN);
```

Lectura de Valores

Primer Lectura de Datos Almacenados

# APAREO DE ARCHIVOS

```
Repetir
    Si(cMae=cNov) Entonces
        cApa = cMae;
        vApa = vMae + vNov;
        EscribirApareo(cApa,vApa,iA,Apareo);
        LeerMaestro(cMae,vMae,iM,Maestro,FlagM);
        LeerNovedad(cNov,vNov,iN,Novedad,FlagN);
    SiNo
        Si(cMae < cNov y FlagM = Verdadero) Entonces
            cApa = cMae;
            vApa = vMae;
            EscribirApareo(cApa,vApa,iA,Apareo);
            LeerMaestro(cMae,vMae,iM,Maestro,FlagM);
        SiNo
            Si(cMae > cNov y FlagN = Verdadero) Entonces
                cApa = cNov;
                vApa = vNov;
                EscribirApareo(cApa,vApa,iA,Apareo);
                LeerNovedad(cNov,vNov,iN,Novedad,FlagN);
            FinSi
        FinSi
    FinSi
Hasta Que FlagM = Falso y FlagN = Falso;
```

Bucle del Proceso de Apareo

Analisis de Alternativas

# APAREO DE ARCHIVOS

```
    Si(cMae < cNov) Entonces
        cApa = cMae;
        vApa = vMae;
        EscribirApareo(cApa,vApa,iA,Apareo);
        cApa = cNov;
        vApa = vNov;
        EscribirApareo(cApa,vApa,iA,Apareo);
    SiNo
        cApa = cNov;
        vApa = vNov;
        EscribirApareo(cApa,vApa,iA,Apareo);
        cApa = cMae;
        vApa = vMae;
        EscribirApareo(cApa,vApa,iA,Apareo);
    FinSi

    Para i<-1 Hasta 6 Con Paso 1 Hacer
        Escribir Apareo[i,1]," ",Apareo[i,2];
    Fin Para
FinAlgoritmo
```

← Ultimos Registros

# APAREO DE ARCHIVOS

```
Repetir
    Si(cMae=cNov) Entonces
        cApa = cMae;
        vApa = vMae + vNov;
        EscribirApareo(cApa,vApa,iA,Apareo);
        LeerMaestro(cMae,vMae,iM,Maestro,FlagM);
        LeerNovedad(cNov,vNov,iN,Novedad,FlagN);
    SiNo
        Si(cMae < cNov y FlagM = Verdadero) Entonces
            cApa = cMae;
            vApa = vMae;
            EscribirApareo(cApa,vApa,iA,Apareo);
            LeerMaestro(cMae,vMae,iM,Maestro,FlagM);
        SiNo
            Si(cMae > cNov y FlagN = Verdadero) Entonces
                cApa = cNov;
                vApa = vNov;
                EscribirApareo(cApa,vApa,iA,Apareo);
                LeerNovedad(cNov,vNov,iN,Novedad,FlagN);
            FinSi
        FinSi
    FinSi
Hasta Que FlagM = Falso y FlagN = Falso;
```

| | |
|---|---|
| 1 | 15 |
| 3 | 30 |
| 5 | 50 |

| | |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 6 | 60 |

| | |
|---|---|
| 1 | 25 |
| 2 | 20 |
| 3 | 30 |
| 5 | 50 |
| 6 | 60 |
| | |

# APAREO DE ARCHIVOS

```
    Si(cMae < cNov) Entonces
        cApa = cMae;
        vApa = vMae;
        EscribirApareo(cApa,vApa,iA,Apareo);
        cApa = cNov;
        vApa = vNov;
        EscribirApareo(cApa,vApa,iA,Apareo);
    SiNo
        cApa = cNov;
        vApa = vNov;
        EscribirApareo(cApa,vApa,iA,Apareo);
        cApa = cMae;
        vApa = vMae;
        EscribirApareo(cApa,vApa,iA,Apareo);
    FinSi

    Para i<-1 Hasta 6 Con Paso 1 Hacer
        Escribir Apareo[i,1]," ",Apareo[i,2];
    Fin Para
FinAlgoritmo
```

| | |
|---|---|
| 1 | 15 |
| 3 | 30 |
| 5 | 50 |

| | |
|---|---|
| 1 | 10 |
| 2 | 20 |
| 6 | 60 |

| | |
|---|---|
| 1 | 25 |
| 2 | 20 |
| 3 | 30 |
| 5 | 50 |
| 6 | 60 |
| | |

# ¡HASTA LA PRÓXIMA!

www.polotic.misiones.gob.ar

/poloticmisiones

polotic misiones

Gobierno de Misiones