

# **ALGORITMOS en Pseudocódigos**

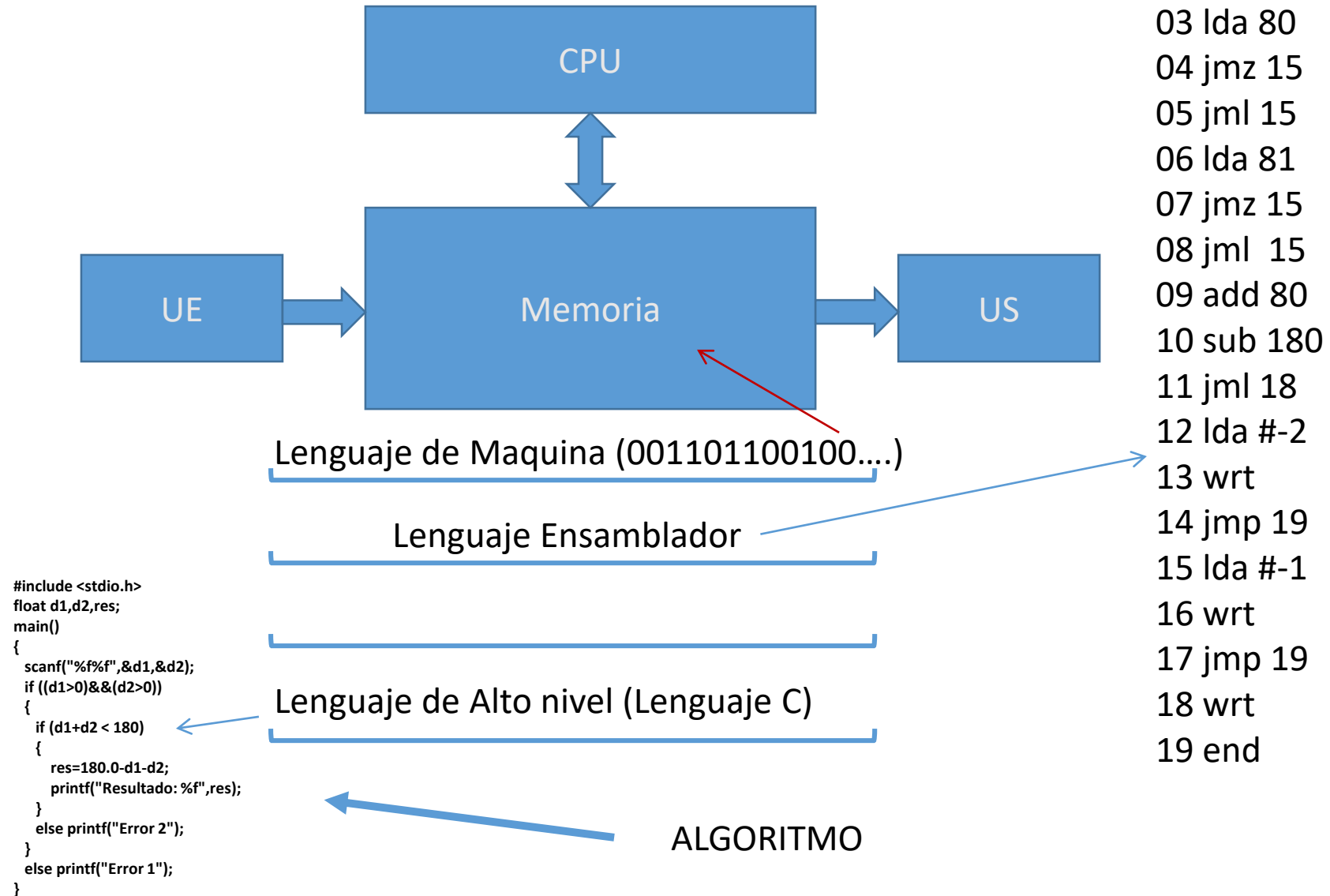
## **Algoritmo y Bases de la Programación**

**Sección 1: Manuel Crisosto Muñoz [mcrisost@ubiobio.cl](mailto:mcrisost@ubiobio.cl)**

**Sección 2: Christian Vidal Castro [cvidal@ubiobio.cl](mailto:cvidal@ubiobio.cl)**

**Abril, 2022**

## Herramienta: Computador



# Elementos básicos presentes en un Algoritmo

**PROGRAMA**: Conjunto de instrucciones, con una secuencia lógica, escrito en algún Lenguaje de Programación que permite resolver un Problema. El programa recibe datos de entrada, realiza las operaciones de transformación requeridas, y entrega los resultados esperados.

**ALGORITMO**: Una secuencia de pasos (modelo) para realizar una tarea.

# Estructura general de un Algoritmo utilizando Pseudocódigo

**Algoritmo <Nombre\_Algoritmo>**

//Variables

**Definir** variables **Como *Tipo\_Dato***;  Las variables son espacios de memoria reservados para el almacenamiento de datos requeridos por el algoritmo

Acción\_1

Acción\_2

Acción\_3

:

:

:

Acción\_n

**FinAlgoritmo**



Las instrucciones o acciones presentes en un programa o algoritmo se pueden clasificar de la siguiente manera:

- Instrucciones o acciones de transferencia de datos
- Instrucciones o acciones aritméticos / lógicos
- Instrucciones o acciones de transferencia de control (condicional o incondicional)

# Variables

Algoritmo <Nombre\_Algoritmo>

//Variables

Definir variables Como *Tipo\_Dato*;

Acción1

Acción\_2

Acción\_3

:

:

Acción\_n

FinAlgoritmo

Las variables son espacios de memoria reservados para el almacenamiento de datos requeridos por el algoritmo.

Las variables quedan definidas por un **nombre** y el **tipo de dato** que pueda contener. La posición de memoria especificada por una variable solo puede contener un valor a la vez.

**Nombre de Variable:** se recomienda usar nombres “significativo”, asociado al uso que se le dará al dato almacenado.

**Tipo de dato:** Corresponde a un atributo del dato que permite especificar el dominio de valores que puede tomar y las operaciones que se pueden hacer sobre ellos. Los tipos de datos primitivos (básicos) son: **Numérico** (Entero o Decimal), **Carácter** y **Lógicos**.

# Instrucciones

Algoritmo <Nombre\_Algoritmo>

//Variables

Definir variables Como Tipo\_Dato;

Acción\_1

Acción\_2

Acción\_3

:

:

Acción\_n

FinAlgoritmo

Las instrucciones o acciones presentes en un programa o algoritmo se pueden clasificar de la siguiente manera:

- Instrucciones o acciones de transferencia de datos:

- **INGRESO/SALIDA DE DATOS (LECTURA/ESCRITURA)**

Leer lista\_de\_variables; //separadas por coma

Escribir lista\_de\_expresiones; //separadas por coma

lista\_de\_expresiones: pueden ser variables, expresiones aritméticas, 'mensajes entre comillas simples'

- **ASIGNACIÓN**

variable ← <expresión>/<variable>

**expresión:** corresponde a una transformación matemática que usa variables definidas y/o funciones predefinidas o desarrolladas.

# Instrucciones

**Algoritmo** <Nombre\_Algoritmo>

//Variables

**Definir** variables Como Tipo\_Dato;

Acción\_1

Acción\_2

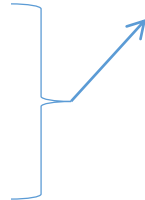
Acción\_3

:

:

Acción\_n

**FinAlgoritmo**



• Ejemplos Instrucciones o acciones de transferencia de datos:

## •INGRESO/SALIDA DE DATOS (LECTURA/ESCRITURA)

Leer a, b, c;

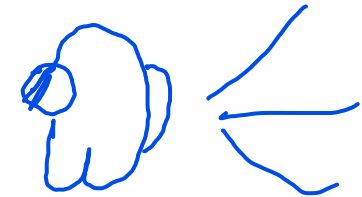
Escribir 'El valor ingresado de a =',a;

Escribir 'La suma de a y b es =', a+b;

Escribir 'La suma de ', a , ' + ', b , 'es =', a+b;

Escribir 'Ingrese un ´numero: ';

Leer n



## •ASIGNACIÓN

a ← 5;

b ← a+z;

c ← (a\*a+b\*b)/(a\*b);

# Instrucciones

**Algoritmo <Nombre\_Algoritmo>**

//Variables

**Definir** variables **Como Tipo\_Dato;**

Acción\_1

Acción\_2

Acción\_3

:

:

Acción\_n

**FinAlgoritmo**

Las instrucciones o acciones presentes en un programa o algoritmo se pueden clasificar de la siguiente manera:

- Instrucciones o acciones aritméticos / lógicos:

- Suma (+), Resta(-), Multiplicación(\*), División (/), *Módulo (%)*, *Raíz cuadrada (RC)*

*(el módulo como la raíz cuadrada pueden ser implementados con las operaciones básicas, pero por comodidad los utilizaremos en forma explícita)*

- y (and o &), o (or o |), no (not o ~)

(los operadores lógicos, al igual que los aritméticos y relacionales se utilizan en las expresiones lógicas presentes en las transferencias de control condicional)



# Ejemplos de Algoritmo básicos utilizando Pseudocódigo

Construir los algoritmos en pseudocódigo que permita resolver los siguientes problemas, mostrando el resultado por pantalla:

- sume dos valores
- multiplica tres números
- divida dos números
- Calcular el cuadrado de un número
- Determine el año de nacimiento de una persona a partir de la edad

# Ejemplos de Algoritmo básicos utilizando Pseudocódigo

## **Sume dos valores**

### **Algoritmo Suma**

//Variables

**Definir** a,b,c **Como Entero**;

Escribir 'Ingrese los datos a sumar';

Leer a,b;

$c \leftarrow a + b$ ;

Escribir 'La suma de los numeros ingresados es = ',c;

**FinAlgoritmo**

**Nota:** //En la herramienta PSeInt no se definen previamente las variables

## **Multiplica tres números**

### **Algoritmo Suma**

//Variables

**Definir** a,b,c,d **Como Entero**;

Leer a,b,c;

$d \leftarrow a * b * c$ ;

Escribir d;

**FinAlgoritmo**

# Ejemplos de Algoritmo básicos utilizando Pseudocódigo

## Divida dos números

Algoritmo División

**Definir**

**a,b,c Como Entero;**

Escribir 'Ingresar el dividendo y divisor';

Leer a,b;

$c \leftarrow a/b$ ;

Escribir 'El resultado es =', c;

FinAlgoritmo

## Calcular el cuadrado de un número

Algoritmo cuadrado

**Definir**

**a,c Como Entero;**

Escribir 'Ingresar un numero';

Leer a;

$c \leftarrow a*a$ ;

Escribir 'El cuadrado es ', c;

FinAlgoritmo

Algoritmo cuadrado

**Definir a Como Entero; //una sola variable**

Escribir 'Ingresar un numero';

Leer a;

$a \leftarrow a*a$ ;

Escribir 'El cuadrado es', a;

FinAlgoritmo

**Determine los años de nacimiento de una persona a partir de la edad**

# Ejemplos de Algoritmo básicos utilizando Pseudo-Código

Construir los algoritmos en pseudocódigo que permita resolver los siguientes problemas, mostrando el resultado por pantalla:

- Determinar la distancia entre dos puntos en el plano cartesiano
- Evaluar la función  $f(x)=3*x^3+4*x^2 - 5$  para un valor específico de  $x$
- Evaluar la ecuación de primer grado
- Evaluar la ecuación de segundo grado

# Instrucciones de transferencia de control

**Algoritmo** <Nombre\_Algoritmo>

Variables

Lista de variables Tipo\_Dato;

Acción\_1

Acción\_2

Acción\_3

:

:

Acción\_n

**FinAlgoritmo**

Las instrucciones o acciones presentes en un programa o algoritmo se pueden clasificar de la siguiente manera:

- Instrucciones o acciones de transferencia de control:

- **Transferencia condicional**

Esta instrucción es necesaria cuando el flujo de ejecución del algoritmo depende de una condición lógica. La instrucción más simple es la **bifurcación**, en la cual la ejecución puede tomar dos cursos de acción dependiendo de una condición lógica.

**Si *expresion\_logica* Entonces**

acciones\_por\_verdadero;

**Sino**

acciones\_por\_falso;

**FinSi**

- Transferencia repetitiva (se vera con posterioridad)
- Transferencia incondicional (se explicara el porque no es aconsejable usar)

# Instrucciones de transferencia de control

**Algoritmo <Nombre\_Algoritmo>**

**Variables**

**Lista de variables Tipo\_Dato;**

Acción\_1

Acción\_2

Si expresion\_logica Entonces  
    acciones\_por\_verdadero;

Sino  
    acciones\_por\_falso;

FinSi

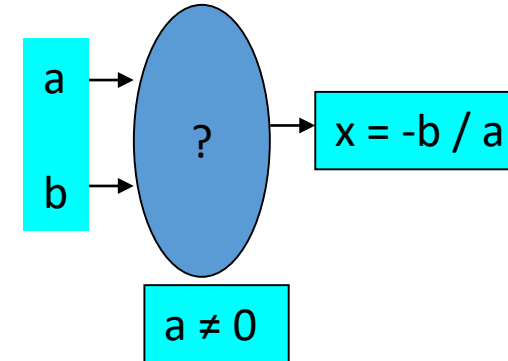
:

Acción\_n

**FinAlgoritmo**

Evaluar la ecuación de primer grado

$$a x + b = 0$$



**Algoritmo Ecuacion\_primer\_grado**

**Variables**

**Definir a,b Como Entero;**

**Definir x Como Real;**

Leer a,b;

Si **a = 0** Entonces

    Escribir 'La ecuacion no es de primer grado';

Sino

$x \leftarrow -b/a$ ;      //probar la solución con PSeInt

    Escribir x;

FinSi

    Escribir 'Fin de Programa';

**FinAlgoritmo**

# Instrucciones de transferencia de control

- Del ejemplo anterior podemos observar que la bifurcación permite seguir el flujo adecuado dada una condición.
- Las condiciones lógicas que se evalúan en las bifurcaciones son siempre verdaderas o falsas.
- También es posible que existan bifurcaciones contiguas.
- No necesariamente deben existir los dos cursos de acción.

# Instrucciones de transferencia de control

En las bifurcaciones, se evalúan proposiciones utilizando:

- **Operadores lógicos**

**y** (and o **&**), **o** (or o **|**), **no** (not o **~**)

- **Operadores aritméticos**

(**\***) multiplicación, (**/**) división, (**+**) suma, (**-**) resta, (**%**) módulo

- **Operadores Relacionales**

**>** (mayor que), **<** (menor que), **=** (igual), **<=** (menor o igual),  
**>=** (mayor o igual), **!=** (distinto)\*

\* No presente en PSeint



# Tablas de verdad asociadas a los operadores lógicos and, or, not

**y, And, &**

Y &	V	F
V	V	F
F	F	F

**O, Or, ||**

O 	V	F
V	V	V
F	V	F

**No, ~**

No   ~	V	F
	F	V

Ejemplos de proposición lógica:

$7 > 3$

$(a+b > c) \text{ y } (b>0)$

$\sim(a = 0)$

$(b*b - 4*a*c) \geq 0$

$((b*b - 4*a*c) < 0) \text{ o } (a=0)$

Deportes concepción es el mejor equipo del año

Cada una de estas expresiones tendrá un valor de verdad dependiendo de los valores de los operandos.

# Ejemplos se evaluación de proposiciones lógicas

## Algoritmo <Nombre\_Algoritmo>

Definir Var1, var2, var3, var4 Como Entero;

var1  $\leftarrow$  2;

var2  $\leftarrow$  0;

var3  $\leftarrow$  4;

var4  $\leftarrow$  (var1\*var1)/var3;

Si (**var4>=1**) y (**var1 !=0**) Entonces

    acciones\_por\_verdadero;

Sino

    acciones\_por\_falso;

FinSi

:

Acción\_n

**FinAlgoritmo**

## Algoritmo <Nombre\_Algoritmo>

Variables

Definir Var1, var2, var3, var4 Como Entero;

var1  $\leftarrow$  2;

var2  $\leftarrow$  0;

var3  $\leftarrow$  4;

var4  $\leftarrow$  (var3%var1)

Si ((**var4!=0**) o (**var2 >0**)) y (**var3 =4**) Entonces

    acciones\_por\_verdadero;

Sino

    acciones\_por\_falso;

FinSi

:

Acción\_n

**FinAlgoritmo**

Es importante notar que existen prioridades entre los operadores. Por lo tanto, deben utilizarse paréntesis en los casos que correspondan

Ejercicios: Construir, utilizando pseudocódigo, un algoritmo para cada uno de los siguientes enunciados

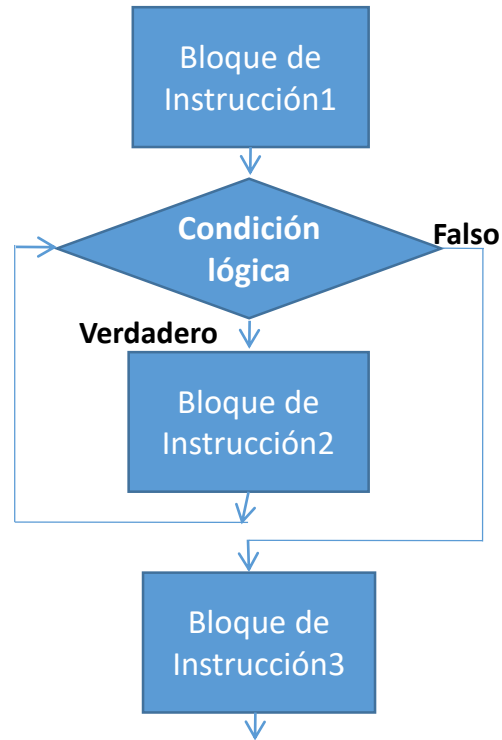
- ❖ Cree un algoritmo que divida dos números y muestre el resultado
- ❖ Crear un algoritmo que muestre en forma ordenada tres números enteros ingresados desde teclado.
- ❖ Cree un algoritmo que resuelva la ecuación de segundo grado, para valores reales, y muestre el resultado
- ❖ Cree un algoritmo que resuelva la ecuación de segundo grado, para valores reales e imaginarios, y muestre el resultado
- ❖ Cree un algoritmo que permita evaluar la siguiente función para un valor de x:
$$f(x)= \begin{cases} (3x^2-5x)/(x-10) & \text{si } x > 10 \\ (5x)/x & \text{si } 0 < x \leq 10 \\ x^2 & \text{si } x \leq 0 \end{cases}$$
- ❖ Cree un algoritmo que permita calcular el valor absoluto de un número
- ❖ Cree un algoritmo que permita calcular el promedio de 20 números
- ❖ Cree un algoritmo que permita sumar n números y muestre el resultado. El valor de n debe ser ingresado por teclado al igual que los números que se sumarán.

# Instrucciones repetitivas

El último ejemplo planteado permite introducir el concepto de **iteración o ciclo**. Las estructuras iterativas permiten la ejecución de un grupo de instrucciones un número conocido o desconocido de veces.

- Concepto de Ciclo
  - Un ciclo es la repetición de un conjunto de instrucciones. Dicho ciclo culmina cuando se cumple una condición lógica de de término.
- Cuándo se aplican los Ciclos
  - Se aplican cuando queremos ejecutar un conjunto de instrucciones varias veces.

# Estructuras Iterativas

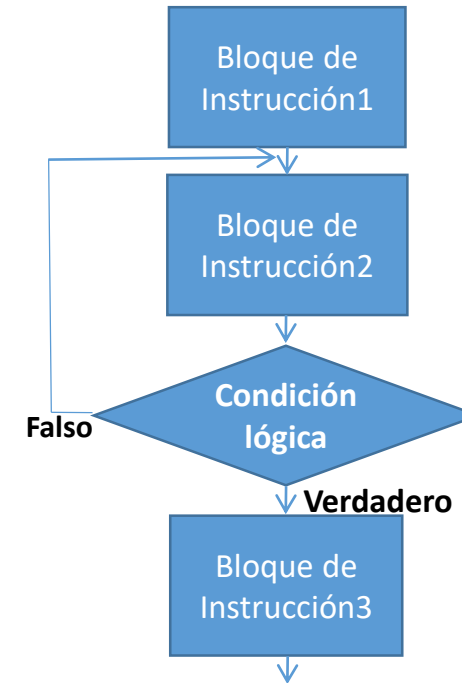


Bloque de Instrucción1;  
**Mientras *Condición\_Logica* Hacer**

Bloque de Instrucción2;

**FinMientras**

Bloque de Instrucción3;



Bloque de Instrucción1;  
**Hasta Que *Condición\_Logica* Repetir**

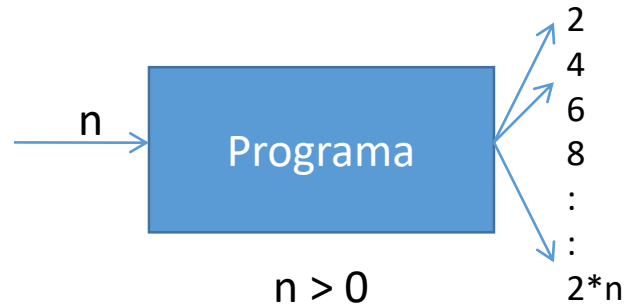
Bloque de Instrucción2;

**Hasta Que *Condición\_Logica***

Bloque de Instrucción3;

Construir un algoritmo que permita imprimir los primeros n números pares

Algoritmo Numeros\_Pares



Definir n,i Como Entero;

Repetir

    Escribir 'Ingrese numeros de pares a imprimir';

    Leer n;

Hasta Que (n > 0)

    i <- -1;

    Mientras (i <= n) Hacer

        Escribir 2\*i;

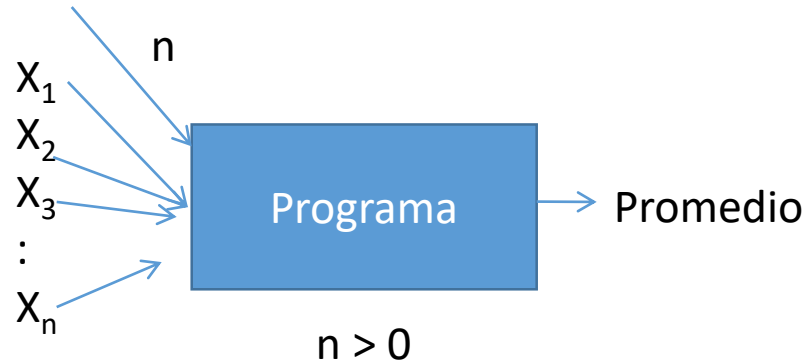
        i <- i+1;

    FinMientras

    Escribir 'Fin del Programa';

FinAlgoritmo

Construir un algoritmo que permita calcular el promedio de n números ingresados por teclado



$$\text{Promedio} = (x_1 + x_2 + x_3 + \dots + x_n) / n$$

$$\left( \sum_{i=1}^n x_i = x_1 + x_2 + x_3 + \dots + x_n \right) / n$$

Algoritmo Promedio

**Variables**

**Definir  $n, i, x, \text{suma}, \text{prom}$  Como Entero;**

Repetir

    Escribir 'Cuántos números sumara?';

    Leer  $n$ ;

Hasta Que ( $n > 0$ )

$i \leftarrow -1$ ;

$\text{suma} \leftarrow 0$ ;

    Mientras ( $i \leq n$ ) Hacer

        Leer  $x$ ;

$\text{suma} \leftarrow \text{suma} + x$ ;

$i \leftarrow i + 1$ ;

    FinMientras

$\text{prom} \leftarrow \text{suma} / n$

    Escribir 'Resultado:',  $\text{prom}$ ;

FinAlgoritmo

Construir un algoritmo que permita sumar números ingresados por teclado. El ingreso de números debe realizarse hasta que se ingrese un 0. Se debe imprimir la suma y la cantidad de números ingresados.



$$\text{suma} = x_1 + x_2 + x_3 + \dots + x_?$$

Algoritmo Sumatoria

**Variables**

**Definir  $n, i, x, \text{suma}$  Como Entero;**

Escribir 'El programa terminara cuando ingrese un cero';

$i < -0$ ;

$\text{suma} < -0$ ;

Repetir

Escribir 'Ingrese un numero';

Leer  $x$ ;

$\text{suma} < -\text{suma} + x$ ;

$i < -i + 1$

Hasta Que  $(x = 0)$

Escribir 'Resultado:',  $\text{suma}$ ;

Escribir 'Numeros ingresados: ',  $i$ ;

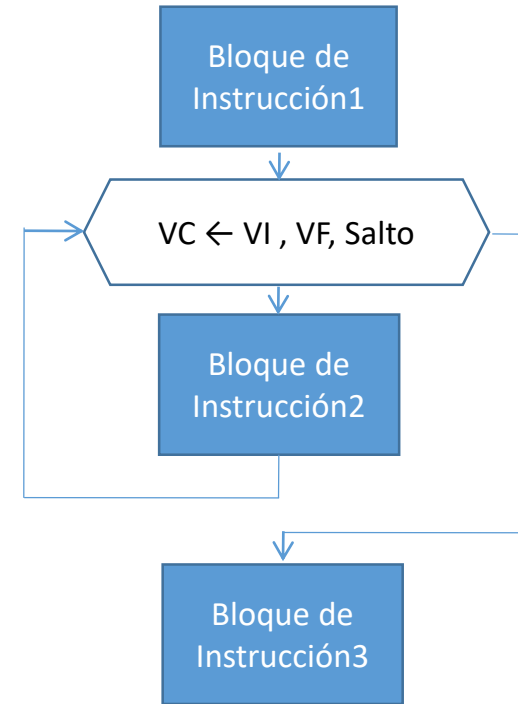
FinAlgoritmo



Construir un algoritmo que permita sumar números ingresados por teclado. El ingreso de números debe realizarse hasta que se ingrese un 0. Se debe imprimir la suma, la cantidad de números ingresados, además el menor y mayor valor ingresado.

# Estructuras Iterativas

Otra estructura iterativa es la siguiente:



Bloque de Instrucción1;

Para ***Variable\_Numerica*** <- ***valor\_inicial*** Hasta ***valor\_final*** Con Paso ***paso*** Hacer

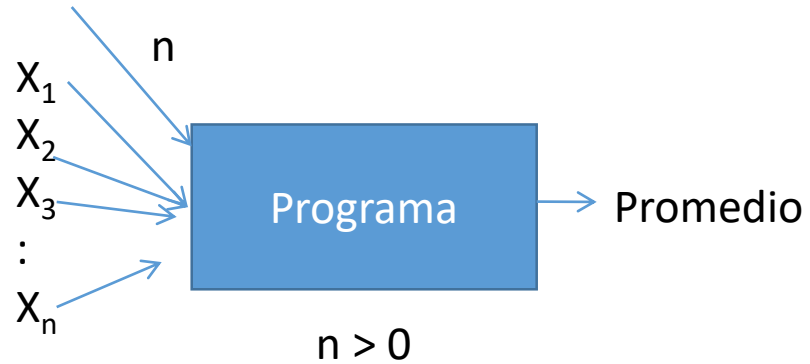
**Bloque de Instrucción2;**

**FinPara**

Bloque de Instrucción3;



Construir un algoritmo que permita calcular el promedio de n números ingresados por teclado



$$\text{Promedio} = (x_1 + x_2 + x_3 + \dots + x_n) / n$$

$$\left( \sum_{i=1}^n x_i = x_1 + x_2 + x_3 + \dots + x_n \right) / n$$

Algoritmo Promedio

**Variables**

**Definir n,i,x,suma, prom Como Entero;**

Repetir

    Escribir 'Cuantos numeros sumara?';

    Leer n;

Hasta Que (n > 0)

    suma<-0;

    Para i<-1 Hasta n Con Paso 1 Hacer

        Leer x;

        suma<-suma+x;

    FinPara

    prom<-suma/n

    Escribir 'Resultado: ',prom;

FinAlgoritmo

¿Preguntas?



# Ejercicios

- Crear un algoritmo que encuentre e imprima el número mayor de N números enteros positivos ingresados por teclado.
- Crear un algoritmo que calcule e imprima el resultado de la siguiente sumatoria

$$\sum_{i=1}^n i$$

- Cree un algoritmo que permita evaluar la siguiente función, para n valores de x:

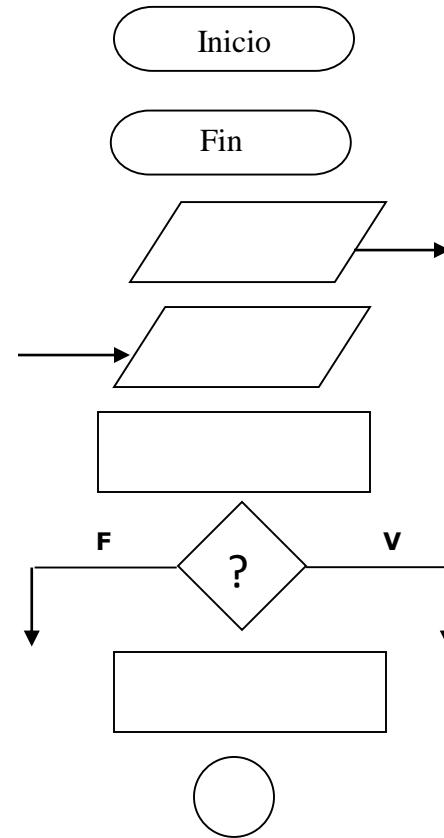
$$f(x) = \begin{cases} (3x^2 - 5x) / (x - 10) & \text{si } x > 10 \\ (5x) / x & \text{si } 0 < x \leq 10 \\ x^2 & \text{si } x \leq 0 \end{cases}$$

- Crear un algoritmo que permita evaluar la función  $f(x) = 3x^3 + 4x^2 - 5$  para todos los valores enteros de x comprendidos en el intervalo [a,b]
- Crear un algoritmo que permita generar los primeros n números de la serie de Fibonacci
- Crear un algoritmo que permita calcular la siguiente sumatoria:


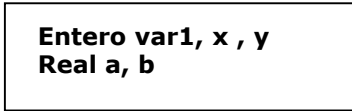
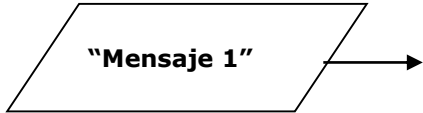
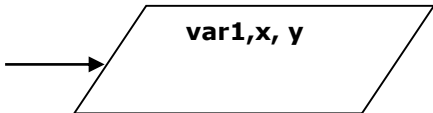
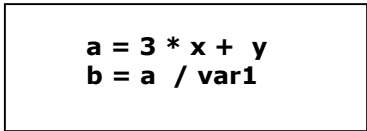
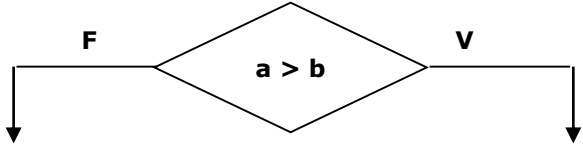


$$S = n + 2*n + 3*n + 4*n + \dots + n*n$$

# Simbología Diagrama de Flujo

- **Nomenclatura:**
  - Inicio del algoritmo
  - Término del algoritmo
  - Impresión de mensajes
  - Receptor de datos
  - Operación de datos
  - Bifurcación (Decisión)
  - Definición de variables
  - Conector



# Diagrama de Flujo y Pseudo-Código

DIAGRAMA DE FLUJO	NOMENCLATURA	PSEUDO-CÓDIGO
	Inicio de Algoritmo	Inicio
	Definición de variables	Variables Entero var1, x, y Real a, b
	Impresión de mensajes	Imprimir ("Mensaje 1")
	Lectura de mensajes	Leer (var1, x, y)
	Transformación de datos Operaciones sobre los datos	$a = 3 * x + y$ $b = a / var1$
	Bifurcación en la ejecución de instrucciones	Si $a > b$ Entonces Inicio Instrucciones Fin Sino Inicio Instrucciones Fin
	Conector	
	Fin de algoritmo	Fin

# Estructura general de un programa en C

## **Algoritmo Prueba**

Definición de Variables

**Inicio**

**bloque de instrucciones**

**Fin**

```
#include<stdio.h>
```

```
#include<otras librerias>
```

```
//Definición de Variables globales
```

```
//Definición prototipo de funciones
```

```
main()
```

```
{
```

```
    //Definición de variables locales
```

```
    bloque de instrucciones
```

```
}
```

```
//desarrollo de funciones
```



# Instrucciones de un programa en C

## **Variables**

**entero a,  
flotante b,**

**Leer a**

**Leer b**

**Escribir "El valor de a es",a**

**Si a > 0 Entonces**

**BLOQUE DE  
INSTRUCCIONES UNO**

**sino**

**BLOQUE DE  
INSTRUCCIONES DOS**

**Fin Si**

**int a;  
float b;**

**scanf("%d",&a);**

**scanf("%f",&b);**

**printf("EL valor de a es %d",a);**

**if (a>0)**

**{**

**BLOQUE DE  
INSTRUCCIONES UNO;**

**}**

**else**

**{**

**BLOQUE DE  
INSTRUCCIONES DOS;**

**}**

# Instrucciones de un programa en C

<b>Repetir</b>  <b>Bloque de instrucciones;</b>  <b>Hasta (Condición de Terminó)</b>   <b>Mientras (Condición de repetición)</b>  <b>Bloque de instrucciones;</b>  <b>Fin Mientras</b>  <b>Para i=Vi, Vf, salto</b>  <b>Bloque de instrucciones;</b>  <b>Fin Para</b>	<b>do</b>  <b>Bloque de instrucciones;</b>  <b>while (condición de repetición);</b>   <b>while (condición de repetición)</b> <b>{</b> <b>    Bloque de instrucciones;</b> <b>}</b>  <b>for (i=vi; condición de terminó, salto)</b> <b>{</b> <b>    Bloque de instrucciones;</b> <b>}</b>
--	---