IES DR LLUÍS SIMARRO

ENSENYAMENT PÚBLIC

XÀTIVA

CFGS Desarrollo de aplicaciones multiplataforma

Módulo profesional: Programación

GENERALITAT VALENCIANA

Conselleria d'Educació, Investigació, Cultura i Esport

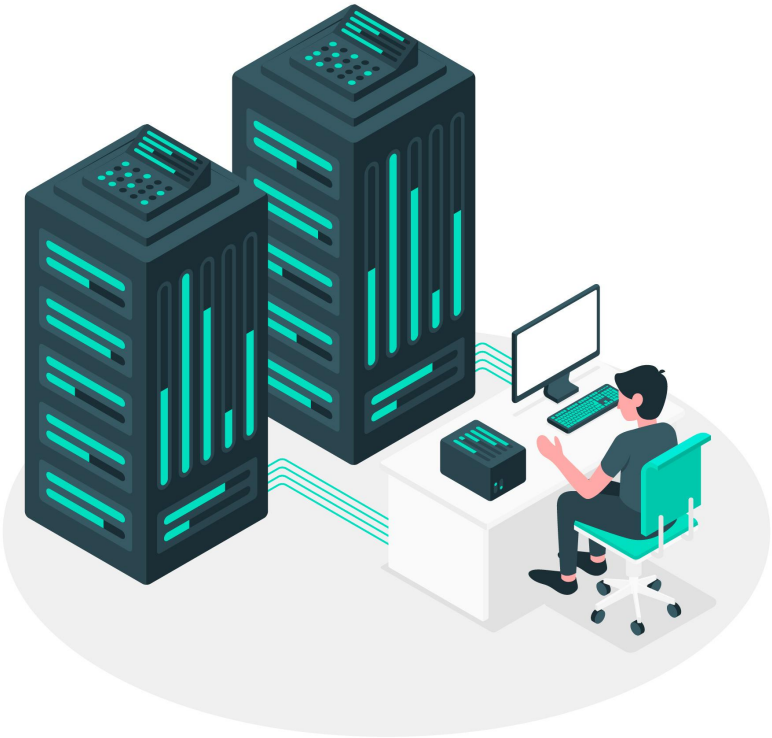
GOBIERNO DE ESPAÑA

MINISTERIO DE EDUCACIÓN Y FORMACIÓN PROFESIONAL

Unió Europea

Fons Social Europeu

L'FSE inverteix en el teu futur

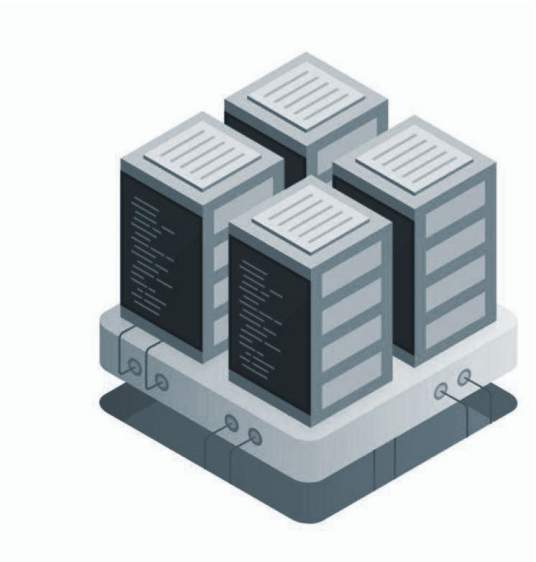


Material elaborado por:

Edu Torregrosa Llácer

(aulaenlanube.com)

Introducción a la programación



1. Introducción a la programación.
    a. Problemas, algoritmos y programas.
    b. Programas y la actividad de programación.
    c. Lenguajes y modelos de programación.
    d. Compiladores e intérpretes.
2. Representación de algoritmos.
    a. Estructura general de un programa.
    b. Representación de algoritmos.
        i. Diagramas de flujo.
        ii. Pseudocódigo.
    c. Elementos de un algoritmo.

Problemas, algoritmos y programas

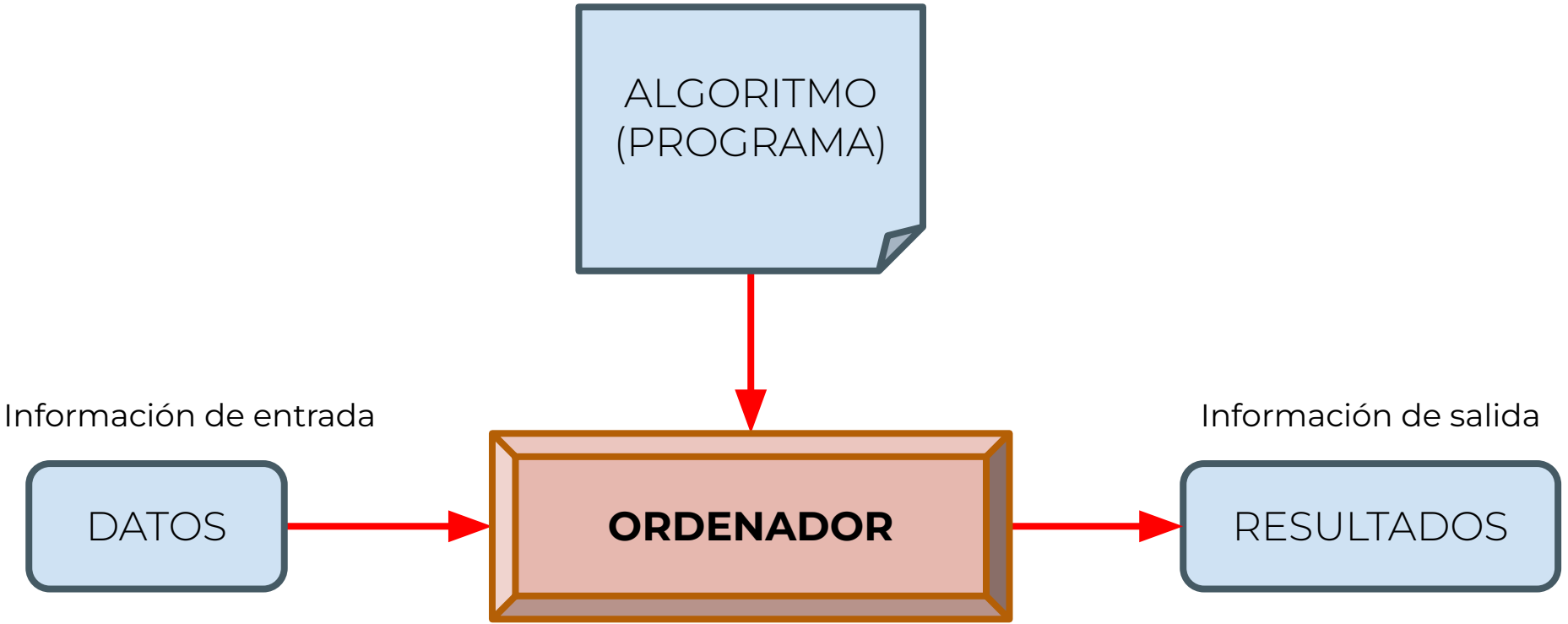
- Un **algoritmo** es una secuencia de acciones que se realizan para resolver un problema. La tarea de programar en realidad se trata de resolver **problemas**.
- La **programación** se encarga de resolver problemas a través de algoritmos
- El algoritmo debe especificar las acciones que se deben ejecutar y en qué orden deben ejecutarse.
- Un algoritmo es independiente del lenguaje de programación. Es decir, una vez tengamos un algoritmo que resuelve un problema, podemos aplicarlo sobre cualquier lenguaje de programación.
- Un **programa** es un algoritmo escrito con las sentencias específicas de un lenguaje de programación concreto.

Problemas, algoritmos y programas

- Un algoritmo es:
  - → **Finito**: debe completarse en un tiempo finito, es decir, el algoritmo debe terminar después de un número finito de pasos.
  - → **Preciso**: debe definirse de forma exacta y precisa, sin ambigüedades.
  - → **Efectivo**: las instrucciones pueden ser ejecutadas por una persona haciendo uso de un papel y un lápiz.
  - → **General**: un algoritmo debe resolver toda una clase de problemas y no un problema aislado particular.

Problemas, algoritmos y programas

Un algoritmo recibe información de entrada, la procesa, y genera información de salida.



Ejemplo de algoritmo

A continuación vamos a ver un ejemplo de algoritmo para elaborar una tortilla de patatas. Tendremos unos datos de entrada, luego el algoritmo que procesa los datos de entrada, y finalmente unos datos de salida o resultado.

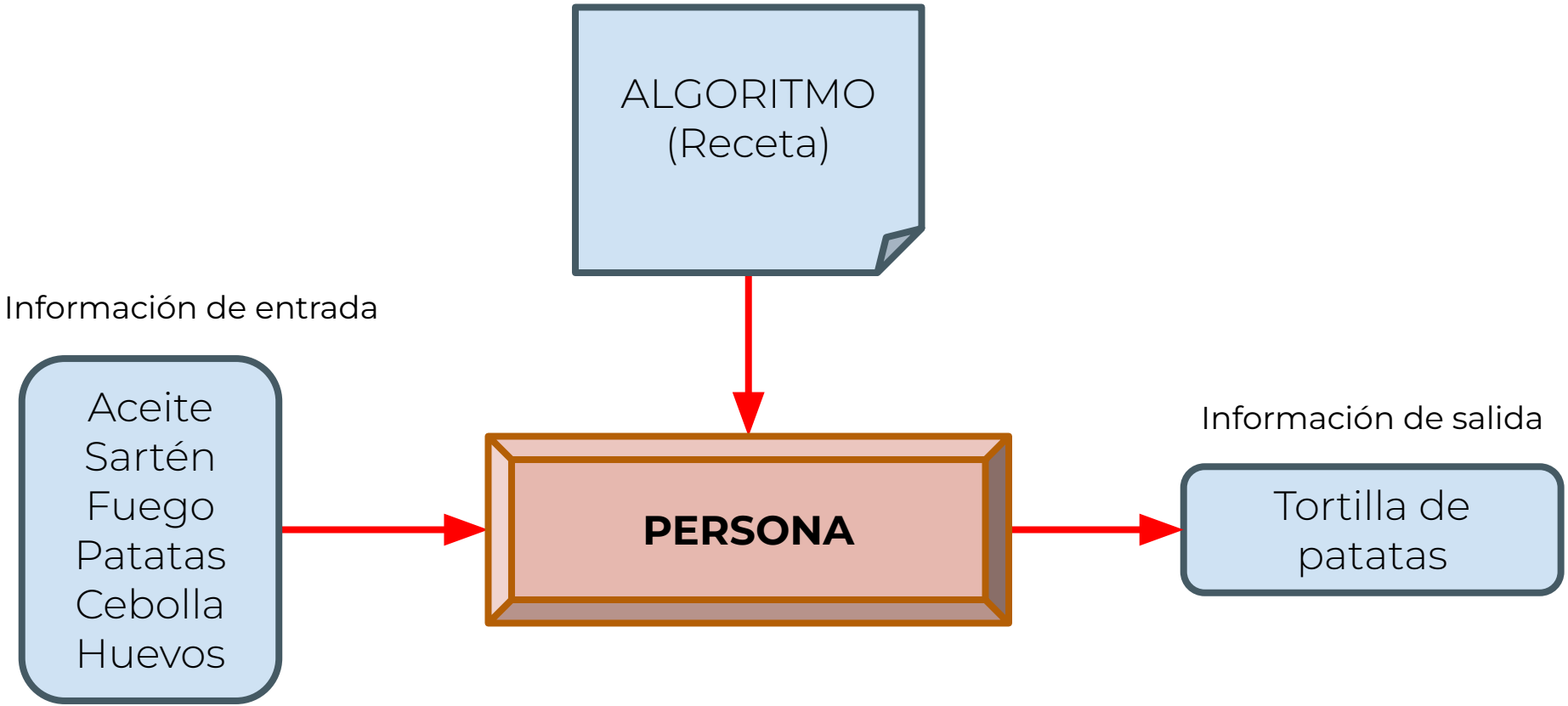
Datos de entrada: Huevo, aceite, patatas, cebolla, sartén, fuego.

Datos de salida: Tortilla de patatas.

Algoritmo:

1. *Poner el aceite en la sartén.*
2. *Poner la sartén al fuego.*
3. *Cuando el aceite esté caliente, freír las patatas y la cebolla.*
4. *Batir los huevos y añadir las patatas y la cebolla cuando estén al gusto.*
5. *Añadir los huevos a la sartén y darle la vuelta varias veces.*

Problemas, algoritmos y programas



Problemas, algoritmos y programas

- Los problemas deben definirse de forma general y precisa, evitando ambigüedades.
- Por ejemplo, el siguiente enunciado: “Determinar la raíz cuadrada positiva de un **número N**”, puede no ser bastante preciso.
- “Determinar la raíz cuadrada positiva de un **número N, entero no negativo** cualquiera”, es más preciso.

> Para un problema P, un **algoritmo** A que resuelva P es un conjunto de reglas/instrucciones que definen cómo resolver P en un tiempo finito.

Problemas, algoritmos y programas

- No todos los problemas pueden resolverse utilizando un ordenador. Solo se podrán resolver aquellos problemas que tengan una solución mecánica. Es decir, aquellos que se resuelvan por medio de una secuencia de instrucciones determinada.
- Son los denominados problemas computacionales o algorítmicos. Como por ejemplo, los relacionados con el cálculo numérico, el tratamiento de palabras, o la representación gráfica.
- Un ejemplo de problema computacional podría ser obtener el área de un cuadrado a partir de su lado. En cambio, un problema no computacional podría ser averiguar el color favorito de una persona a partir de su nombre.

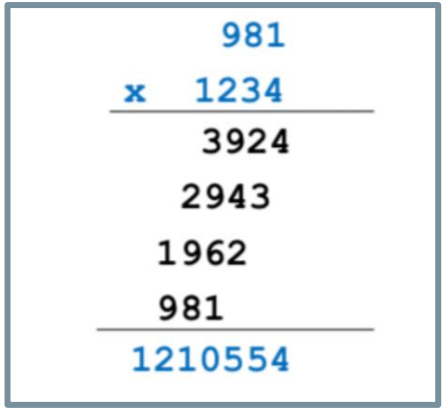
Problemas, algoritmos y programas

- **Ejemplos** de problemas computacionales:
  - → Determinar el producto de dos números A y B.
  - → Determinar la raíz cuadrada de un número N.
  - → Determinar si un número es primo.
  - → A partir de una lista de palabras, determinar las palabras repetidas.
  - → Ordenar alfabéticamente un conjunto de palabras.
  - → Dibujar en pantalla un círculo de radio R.

Problemas, algoritmos y programas

Ejemplos de algoritmos son las secuencias de reglas que utilizamos para realizar operaciones aritméticas: sumas, restas, productos y divisiones. Son algoritmos porque definen de manera precisa los pasos que se deben seguir para encontrar una solución en un tiempo finito.

A continuación podemos ver un ejemplo de algoritmo que nos permite realizar multiplicaciones de varias cifras: El algoritmo que define como hacerlo sería el siguiente:



1. *Se multiplica sucesivamente el **multiplicando** (**981**) por cada una de las cifras del **multiplicador** (**1234**), obtenidas de derecha a izquierda.*
2. *Escribimos los resultados intermedios uno bajo del otro, desplazando cada línea un lugar a la izquierda,*
3. *Se suman todas las filas para obtener el producto*

Conceptos básicos en programación

- Un **procesador** es cualquier entidad capaz de interpretar y ejecutar un cierto repertorio de instrucciones.
- Un **programa** está formado por uno o más algoritmos escritos con una notación precisa para que puedan ser ejecutados por un procesador en concreto (ordenador).
- Un **lenguaje de programación** es una notación, conjunto de reglas y definiciones que determinan aquello que se puede escribir para desarrollar un programa.
- La **programación** es la actividad de resolución de problemas a través de un ordenador.

Conceptos básicos en programación

- Un **proceso** es un algoritmo en ejecución que se caracteriza por una sucesión de estados.
- El **estado de un programa** es un conjunto de valores en un momento determinado. El estado indica el grado de progreso en un proceso.
- El **cómputo** es la transformación del estado de un programa al ejecutarse una o más instrucciones.
- Una **instrucción** es una expresión que indica, dentro de un programa, qué operación se debe realizar y qué datos se deben utilizar.
- Los **datos** son cualquier tipo de información dispuesta de forma adecuada para su tratamiento por un ordenador .

Programas y la actividad de la programación

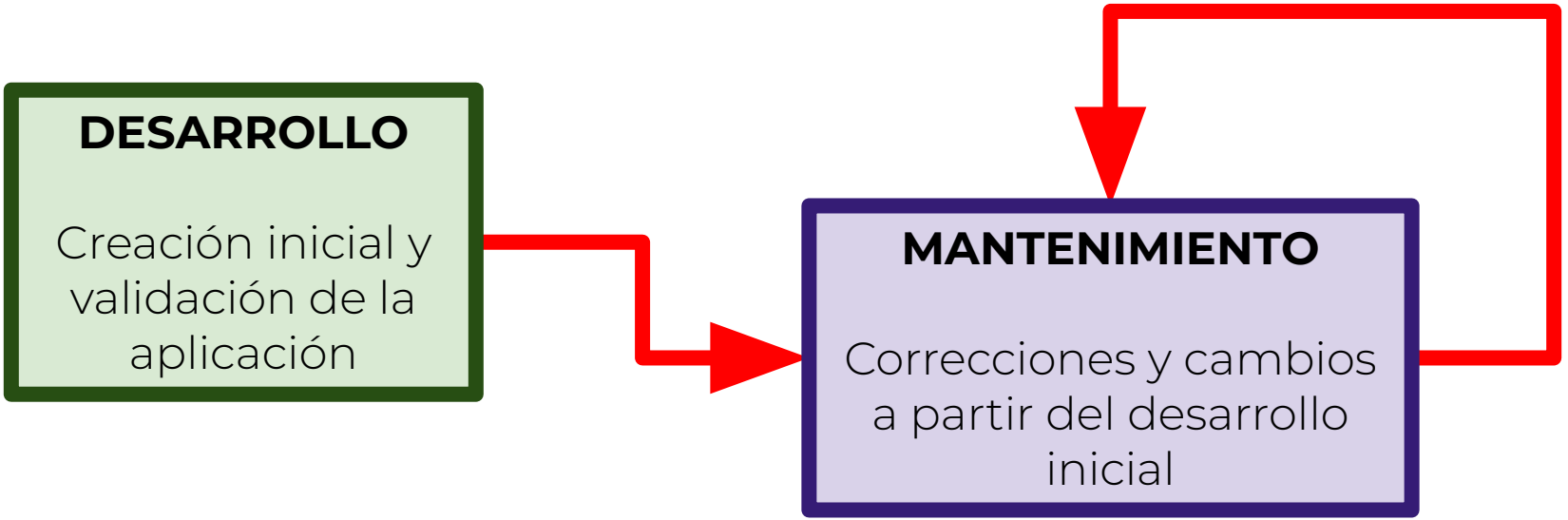
- La tarea de la programación en aplicaciones reales de una cierta envergadura es bastante compleja.
- Según la complejidad del problema a resolver podemos hablar de:

Programación a escala reducida: número reducido de líneas de programa, intervención de una sola persona. Por ejemplo, un programa para ordenar números.

Programación a gran escala: muchas líneas de programa, equipo de programadores. Por ejemplo, el desarrollo de una sistema operativo.

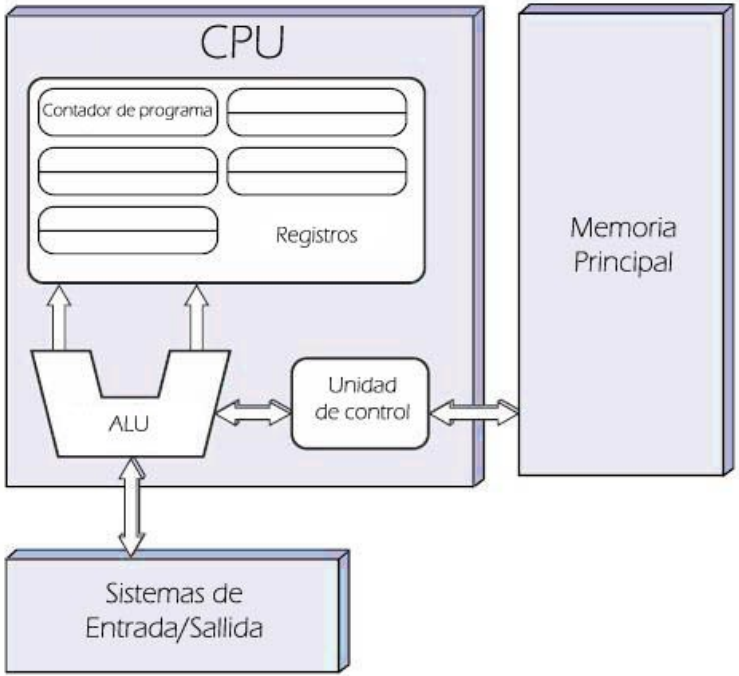
Programas y la actividad de la programación

Los programas tienen un ciclo de vida en el que , de manera muy simplificada, se pueden distinguir dos etapas:



Lenguajes y modelos de programación

- Los orígenes de los lenguajes de programación se encuentran en las máquinas de cálculo.
- Charles Babbage junto con Ada Lovelace son los que diseñan la primera máquina analógica para computar.
- Pero es la arquitectura de **John Von Neumann** la que supone un cambio definitivo. Dicha arquitectura divide a un computador digital en varias partes que se encargan de hacer tareas distintas.



Lenguajes y modelos de programación

- A nivel máquina, un programa es una sucesión de palabras expresadas en código binario (secuencia de 0s y 1s), en posiciones consecutivas de memoria que representan instrucciones o datos. Es el **lenguaje máquina**.
- Es evidente que los programas escritos en lenguaje máquina resultan ilegibles.
- Los **lenguajes ensambladores** hacen uso de mnemotécnicos e identificadores para instrucciones y datos.
- Ambos son lenguajes tan próximos a la máquina que se denominan **lenguajes de bajo nivel**.

Lenguajes y modelos de programación

Ejemplo código ensamblador

```
L0:  MOV   R1, #a       ;
     MOV   R2, #b       ;

L1:  LD    R3, (R1)     ;
     CMP   R3, #0       ;
     BNE   L3           ;

L2:  MOV   R4, #1       ;
     JMP   L4           ;

L3:  MOV   R4, #0       ;

L4:  ST    (R2), R4     ;
     JMP   L1           ;
```

Ejemplo código binario

```
01010100 01101000 01101001 01110011
00100000 01101001 01110011 00100000
01110100 01101000 01100101 00100000
01110100 01110101 01110100 01101111
01110010 01101001 01100001 01101100
00100000 01110100 01101111 00100000
01101100 01100101 01100001 01110010
01101110 00100000 01100010 01101001
01101110 01100001 01110010 01111001
00101110 00100000 01001001 00100000
01101000 01101111 01110000 01100101
00100000 01111001 01101111 01110101
00100000 01100101 01101110 01101010
01101111 01111001 00100000 01101001
01110100 00100001
```

Lenguajes y modelos de programación

- Frente a los lenguajes de bajo nivel, se encuentran los **lenguajes de alto nivel**:
  - → Disponen de operadores y estructuras más próximas al lenguaje humano, esto permite al programador construir instrucciones de forma más sencilla.
  - → Son más seguros que el código máquina y ayudan a no cometer errores evidentes.
  - → El código es transportable (independiente de la máquina).
  - → El código es más legible.

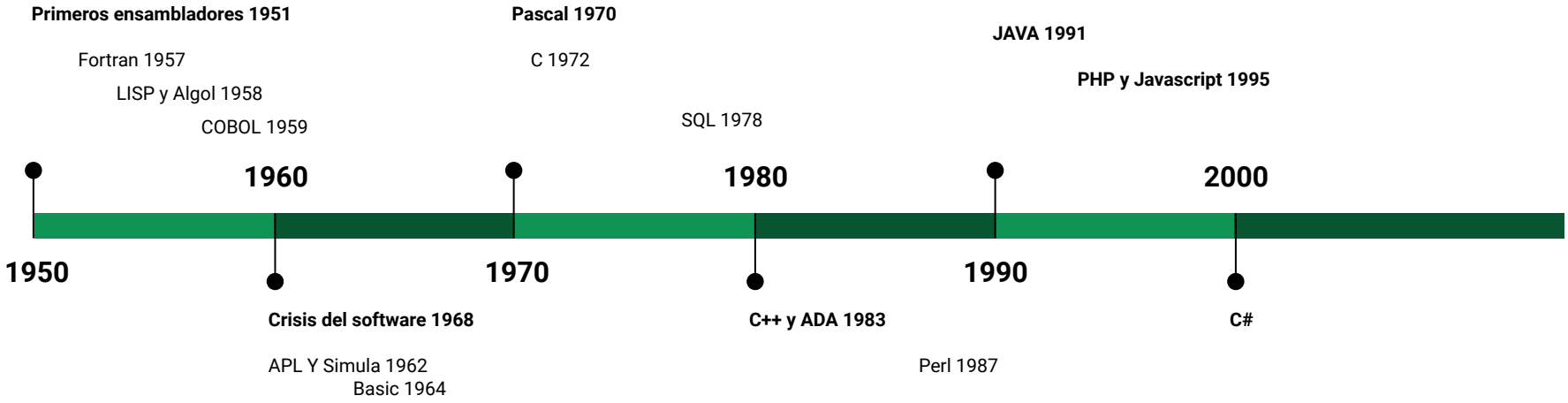
Lenguajes y modelos de programación

Ejemplos lenguaje JAVA

```
int cantidad = 6;
int precio = 10;
int total = cantidad * precio;
System.out.println (“El precio total es : “+total);
```

```
int base = 3;
int altura = 10;
int area = base * altura;
System.out.println (“El área del rectángulo es : “+area);
```

Lenguajes y modelos de programación



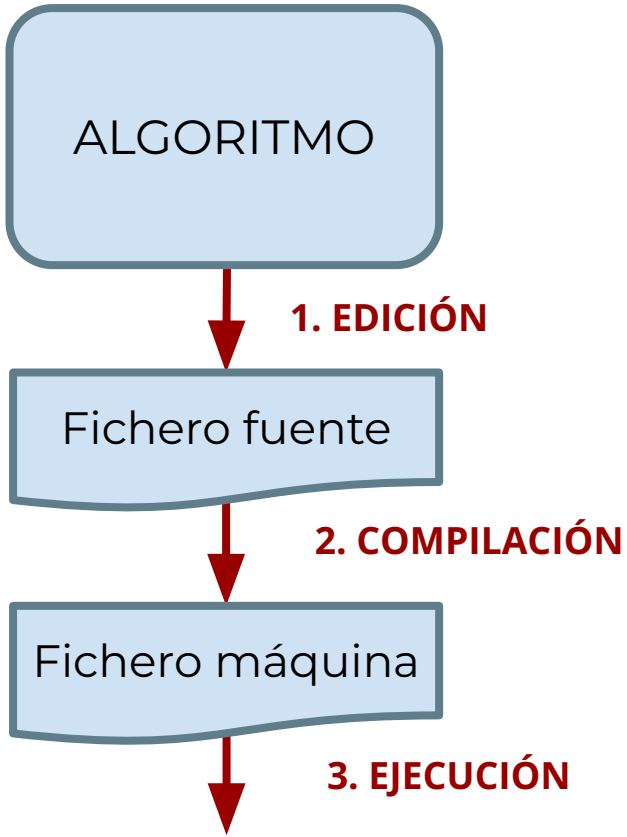
PROGRAMACIÓN ESTRUCTURADA

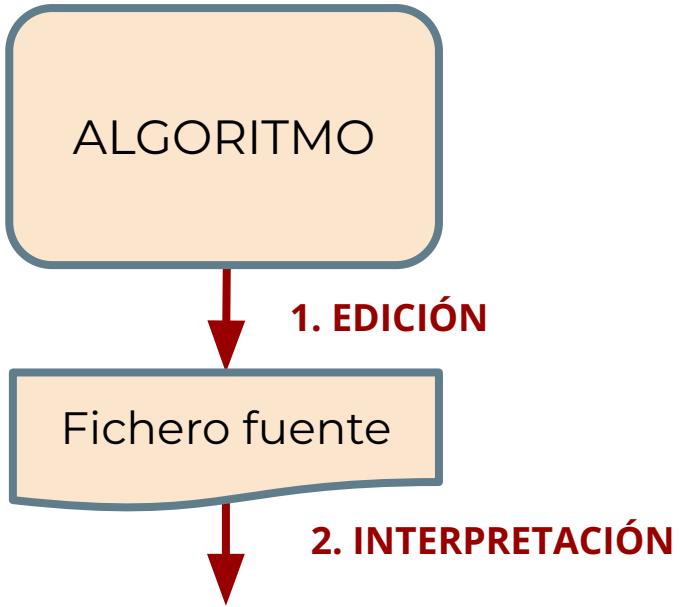
PROGRAMACIÓN OO

Compiladores e intérpretes

- Hay dos formas de traducir un programa escrito en un lenguaje de alto nivel al lenguaje máquina: **interpretar** y **compilar**.
- En la **interpretación** se traduce a lenguaje máquina cada instrucción de manera individual en tiempo de ejecución (de una en una).
- En la **compilación** se traducen de manera global todas las instrucciones por medio de un programa (compilador).
- En los lenguajes compilados se debe hacer siempre una compilación previa para poder ejecutar el programa resultante.

Compiladores e intérpretes





Representación de algoritmos

Representación de algoritmos

- Recordemos la definición de algoritmo: conjunto de reglas o instrucciones que permiten resolver un problema en un número finito de pasos.
- Para resolver un mismo problema pueden haber infinidad de algoritmos distintos que lo resuelvan.
- La calidad de los algoritmos se podrá medir atendiendo a dos factores:
  - → El tiempo necesario para ejecutarlo.
  - → Los recursos, en términos de memoria principal y de almacenamiento que se necesitan para implementarlo.

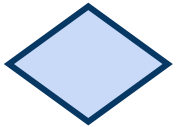
Representación de algoritmos

- Para representar un algoritmo se suele utilizar algún método que permite independizarlo del lenguaje de programación utilizado.
- Esto nos permitirá codificar más tarde a través de cualquier lenguaje de programación.
- Existen diversas técnicas, podemos destacar:
  - **Diagramas de flujo** (ordinogramas) Utilizan símbolos gráficos estándar y escriben los pasos del algoritmo dentro de los símbolos. A través de las líneas de flujo se indican la secuencia en que se deben ejecutar
  - Lenguajes de descripción de algoritmos (**Pseudocódigo**).

Diagramas de flujo. Ordinogramas

Terminal: Representa el inicio y fin de un programa

Proceso: Operaciones generales



Decisión: Indica operaciones lógicas o de comparación, así como expresiones

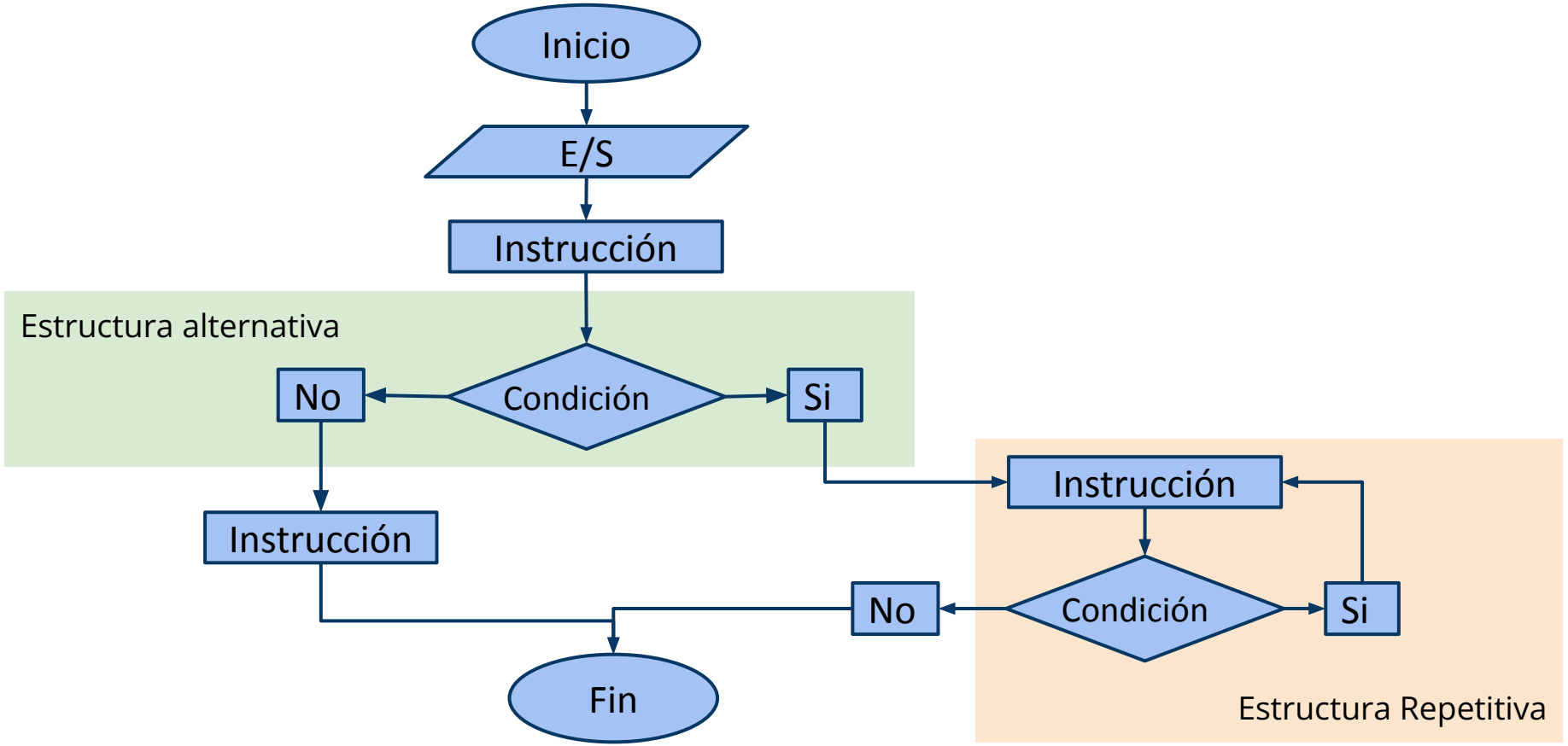
Entrada/Salida: Nos permite introducir y mostrar datos

Conector: Enlaza dos partes de un programa.

Línea de flujo: Indica la dirección de ejecución del algoritmo.

Subprograma: Usado para realizar una llamada a un subprograma o subrutina.

Diagramas de flujo. Ordinogramas



Pseudocódigo

- Es un lenguaje para la descripción de algoritmos que procura mantener las propiedades de los diagramas de flujo: sencillez, claridad, normalización y flexibilidad.
- Su notación se basa en el **lenguaje natural**.
- La **estructura general del pseudocódigo** está dividida en dos partes:
  - → Nombre y entorno del programa: tipos, nombre de las variables y constantes que se utilizan.
  - → Desarrollo ordenado y estructurado del algoritmo: esta segunda parte se corresponde con el diagrama de flujo.

Pseudocódigo

Ejemplo de algoritmo diseñado con Pseudocódigo:

```
PROGRAMA NombreDelPrograma
VARIABLES
    Declaración de variables
ALGORITMO
    Cuerpo del programa
FIN
```

Representación de algoritmos

Diseñar un algoritmo que obtenga el área de un círculo a partir de su radio

PROGRAMA AreaCirculoPrograma

VARIABLES

Area, Radio REAL

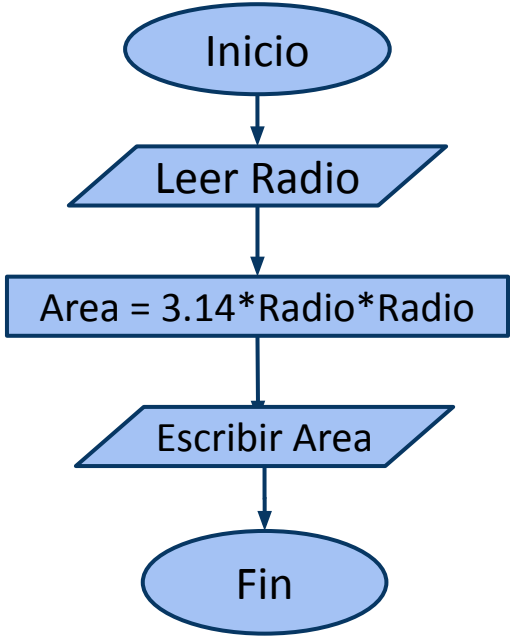
ALGORITMO

LEER Radio

Area = 3.14 * Radio * Radio

ESCRIBIR Area

FIN



Pseudocódigo

- Todo pseudocódigo debe posibilitar la descripción de los diferentes elementos de un algoritmo, es decir, debe permitir:
  - → Representar los **objetos** del programa:
    - Declaración de constantes, variables y tipos de datos.
    - Declaración de subprogramas.
  - → Representar las diferentes **acciones** que se llevan a término durante la ejecución del algoritmo mediante:
    - Instrucciones simples como asignaciones, entrada, salida, etc.
    - Instrucciones de control de flujo.
    - Subprogramas

Elementos de un algoritmo

Elementos de un algoritmo

Un algoritmo está formado por:

1. **Datos**: son los valores que describen los objetos con los que opera un ordenador, como pueden ser números o texto. Los datos se almacenan en variables y constantes.
2. **Expresiones**: es una combinación de constantes, variables, operadores, paréntesis y nombres de funciones.
3. **Instrucciones**: acciones básicas que implementan los algoritmos, como leer o escribir en variables.

Datos

- Un dato es cualquier información dispuesta de manera adecuada para su tratamiento por un ordenador.
- El **tipo de dato** define:
    - Un conjunto de valores.
    - Un conjunto de operaciones permitidas sobre ellos.

Los tipos de datos que vamos a utilizar son: **ENTERO, REAL, TEXTO y BOOLEAN**, los utilizaremos para representar cantidades numéricas, cadenas de texto y los valores verdadero y falso.

Datos y variables

En programación los datos se guardan en **variables**, son espacios de memoria que se reservan para poder almacenar un **tipo de datos**.

Como el propio nombre indica las variables pueden ser modificadas a lo largo de la ejecución de un programa. **Para facilitar la lectura del código el tipo de datos de las variables lo vamos a escribir en mayúsculas**.

Ejemplos:

```
PROGRAMA area
VARIABLES
      base, altura ENTERO
ALGORITMO
      .....
FIN
```

```
PROGRAMA datos
VARIABLES
      nombre, apellidos TEXTO
      mayorEdad BOOLEAN
ALGORITMO
      .....
FIN
```

Expresiones

- Las expresiones se clasifican en:
  - **Aritméticas**: el resultado será numérico.
    - Operandos: constantes y variables numéricas.
    - Operadores: aritméticos.

Operador	Operación	Operador	Operación
+	Suma o signo	+=	Suma y asignación
-	Resta o signo	-=	Resta y asignación
*	Multiplicación	*=	Multiplicación y asignación
/	División	/=	División y asignación
%	Módulo	%=	Módulo y asignación
++	Incremento en 1	--	Decremento en 1

Expresiones

- Ejemplos:

	Expresión	Resultado	Expresión	Resultado
Enteros	3+5	8	2*6	12
	7/2	3	7%2	1
Reales	3.5+5.6	9.1	3.1*2.0	6.2
	15.0/2.0	7.5	7.0%2.0	1.0

Ejemplos de algoritmos con expresiones aritméticas

```
PROGRAMA expr1
VARIABLES
        base, altura, area ENTERO
ALGORITMO
        base = 5
        altura = 10
        area = base * altura
FIN
```

area vale 50

```
PROGRAMA expr2
VARIABLES
        a, b, c ENTERO
ALGORITMO
        a = 3
        b = 2
        c = b % a
        c++
FIN
```

c vale 2

c vale 3

Ejemplos de algoritmos con expresiones aritméticas

```
PROGRAMA expr3
VARIABLES
    a, b, c ENTERO
ALGORITMO
    a = 3
    b = 2
    c = a + b
    c *= a
    c--
FIN
```

c vale 5
c vale 15
c vale 14

```
PROGRAMA expr4
VARIABLES
    a, b, c, d ENTERO
ALGORITMO
    a = 2
    b = 1
    c = a%b
    d = c - b
    d += a + b + c
FIN
```

c vale 0
d vale -1
d vale 2

Ejemplos de algoritmos con expresiones aritméticas

```
PROGRAMA expr5
VARIABLES
        a, b, c REAL
ALGORITMO
        a = 2
        b = 3
        c = 0.5
        a*= b
        a*= c
        a*= c
        a*= b
FIN
```

a vale 6
a vale 3
a vale 1.5
a vale 4.5

```
PROGRAMA expr6
VARIABLES
        a, b ENTERO
        c REAL
ALGORITMO
        a = 2
        b = 3
        c = 0.5
        a*= b
        a*= c
        a*= c
        a*= b
FIN
```

a vale 6
a vale 3
a vale 1
a vale 3

Expresiones

Además también tenemos las **expresiones lógicas**, cuyo resultado será de tipo booleano. Es decir, verdadero o falso. Dichas expresiones están formadas por:

- Operandos: constantes, variables y expresiones lógicas.
- Operadores: **relacionales (==, <, >, <=, >=, !=)**.

Operador	Operación
==	Igual
!=	Distinto
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Ejemplos de algoritmos con expresiones

```
PROGRAMA expr7
VARIABLES
    a, b, c BOOLEAN
ALGORITMO
    a = 1 > 2
    b = 1 < 2
    c = (1*2+1) > (1+2*1)
FIN
```

a vale false
b vale true
c vale false

```
PROGRAMA expr8
VARIABLES
    x, y ENTERO
    b1, b2, b3 BOOLEAN
ALGORITMO
    x = 5
    y = 1
    b1 = x != y
    b2 = x < y
    b3 = (x + y) == 6
FIN
```

b1 vale true
b2 vale false
b3 vale true

Ejemplos de algoritmos con expresiones

```
PROGRAMA expr9
VARIABLES
    a, b, c BOOLEAN
ALGORITMO
    a = 4 > 5
    b = 6 != 3
    c = a == b
FIN
```

a vale false
b vale true
c vale false

```
PROGRAMA expr10
VARIABLES
    x ENTERO
    b1, b2 BOOLEAN
ALGORITMO
    x = 6
    b1 = x <= 7
    b2 = (4 + x) > 10
    b1 = b2
    b1 = b1 == b2
FIN
```

b1 vale true
b2 vale false
b1 vale false
b1 vale true

Expresiones

- Las expresiones pueden agruparse a través de operadores lógicos. Este tipo de operadores se utilizan para hacer varias comprobaciones de forma conjunta, dichas expresiones pueden utilizar :
  - Operandos: constantes, variables y expresiones lógicas.
  - Operadores relacionales
  - Operadores **lógicos (not, and, or, xor)**

Operador	Operación	Significado
!	NOT	Negación lógica
&&	AND	Conjunción lógica (Y lógico)
||	OR	Disyunción (O lógico)
^	XOR	Disyunción Exclusiva (XOR)

Expresiones

x	y	x && y	x || y	x ^ y	!x
false	false	false	false	false	true
false	true	false	true	true	true
true	false	false	true	true	false
true	true	true	true	false	false

```
(valor >= 15) && (valor <= 20) //true si valor entre 15 y 20
(valor <= 15) && (valor >= 20) //false, condición imposible
(valor > 15) || (valor == 15) //true si valor es mayor o igual que 15
(valor > 15) || (valor < 15) //true para cualquier valor menos el 15
(valor >= 15) && (valor <= 20) || (valor > 30 ) //true si valor entre 15 y 20
o mayor que 30
!(valor > 15) //true si valor es menor o igual que 15
```

Ejemplos de algoritmos con expresiones

```
PROGRAMA expr11
VARIABLES
        a, b, c BOOLEAN
ALGORITMO
        a = TRUE
        b = FALSE
        c =  a && b
        c = a || b
        c = a && (6 != 3)
        c = a && (6 == 3)
FIN
```

c vale false
c vale true
c vale true
c vale false

```
PROGRAMA expr12
VARIABLES
        x ENTERO
        b1, b2 BOOLEAN
ALGORITMO
        x = 10
        b1 = (x == 10)
        x+=x
        b2 = (x == 10)
        b1 = b1 && b2
        b2 = !b1 || b2
FIN
```

b1 vale true
b2 vale false
b1 vale false
b2 vale true

Ejemplos de algoritmos con expresiones

```
PROGRAMA expr13
VARIABLES
        x, y ENTERO
        b1, b2 BOOLEAN
ALGORITMO
        x = 10
        y = 10
        b1 = (x+y > 20) && x > 10 || y-x < 10
        x+=y+x
        b2 = !(x == 20) && !b1
        b1 = b1 && b2 || b1 || b2
        b2 = b1 || !b2 && b2 && b1
FIN
```

b1 vale true
b2 vale false
b1 vale true
b2 vale true

El operador AND lógico (&&) tiene una mayor prioridad que el operador OR lógico (||)

En este último ejemplo, primero se evalúa toda la expresión con ANDs

b2 = b1 || **!b2 && b2 && b1**

Instrucciones en pseudocódigo

Instrucciones

- Las instrucciones se clasifican en:
  - **Declarativas**: anuncian el uso de variables y constantes, se debe indicar el nombre de la variable y el tipo de dato que contendrá.
  - **Primitivas**: las ejecuta el procesador de manera inmediata; no dependen de ningún otro objeto o condición.
  - **Control**: evalúan expresiones lógicas para controlar la ejecución de otras instrucciones o alterar el orden de ejecución normal (ejecución secuencial).

Declaración

- **Declarativas**: anuncian el uso de variables y constantes, se debe indicar el nombre de la variable y el tipo de dato que contendrá, para diferencia el tipo lo pondremos en mayúsculas, ENTERO o REAL.

```
PROGRAMA area
VARIABLES
    base, altura, area ENTERO
ALGORITMO
    Cuerpo del programa
FIN
```

Primitivas

- **Primitivas**: las ejecuta el procesador de manera inmediata; no dependen de ningún otro objeto o condición.
  - → **Entrada**: lee valores y los asigna a un identificador en memoria
  - → **Salida**: envía datos a un dispositivo desde la memoria.
  - → **Asignación**: obtiene el resultado de una expresión y la guarda en un identificador en memoria.
- Las primitivas que utilizaremos serán **LEER** y **ESCRIBIR**, y para diferenciarlas del resto del código, también las escribiremos en mayúsculas, con el contenido entre paréntesis.
- Al utilizar la primitiva ESCRIBIR, podemos concatenar texto en lenguaje natural con los valores de las variables, para ello debemos utilizar el operador suma (+).
  - → Ejemplo: **ESCRIBIR(“El resultado de la suma es “+resultado)**
  - → Si la variable ‘resultado ‘tiene guardado en su interior un 20, la salida del programa sería → El resultado de la suma es 20.

Primitivas

Algoritmo que dada la base y la altura de un rectángulo, obtiene su área:

```
PROGRAMA area
VARIABLES
    base, altura, area ENTERO
ALGORITMO
    ESCRIBIR (“Introduce la base y la altura”)
    LEER (base, altura)
    area = base*altura
    ESCRIBIR (“El área del rectángulo es “+area)
FIN
```

ENTRADA	SALIDA
base = 3, altura = 4	El área del rectángulo es 12

Ejercicio

Escribe un algoritmo en pseudocódigo que dada una distancia en millas marinas, las escriba en metros, se debe tener en cuenta que 1 milla marina es igual a 1852 metros.

Ejercicio

```
PROGRAMA calcularMillas
VARIABLES
    millas, metros ENTERO
ALGORITMO
    ESCRIBIR (“Introduce el n° de millas”)
    LEER (millas)
    metros = millas*1852
    ESCRIBIR (“Metros: “+ metros)
FIN
```

ENTRADA	SALIDA
millas = 3	Metros: 5556

Ejercicio

Diseñar un algoritmo expresado en pseudocódigo que, introduciendo por teclado el precio y el porcentaje descontado, escriba el precio pagado en una compra.

Ejercicio

```
PROGRAMA calcularPrecio
VARIABLES
    tarifa, porcentaje, precioPagado ENTERO
ALGORITMO
    ESCRIBIR (“Introduce tarifa y porcentaje”)
    LEER (tarifa, porcentaje)
    precioPagado = tarifa - (tarifa*porcentaje/100)
    ESCRIBIR (“El precio a pagar es: “+precioPagado)
FIN
```

ENTRADA	SALIDA
tarifa = 1000, porcentaje = 15	El precio a pagar es: 850

Estructuras de control

Estructuras de control: evalúan expresiones lógicas para controlar la ejecución de otras instrucciones o alterar el orden de ejecución normal:

- **Alternativas**: simples, dobles, múltiples
- **Repetitivas**: mientras, repetir...mientras, para...hacer

Para representar el código dentro de una estructura de control, lo escribiremos **tabulando el código**, es decir, dejando un margen por la izquierda. En caso de querer representar estructuras anidadas, podremos ir añadiendo tabulaciones a medida que las necesitemos.

Estructuras de control alternativas

Alternativas: se ejecuta una instrucción o conjunto de instrucciones después de evaluar una condición.

SIMPLE:

```
SI (condición )
    instrucción1
    instrucción2
instrucción3
```

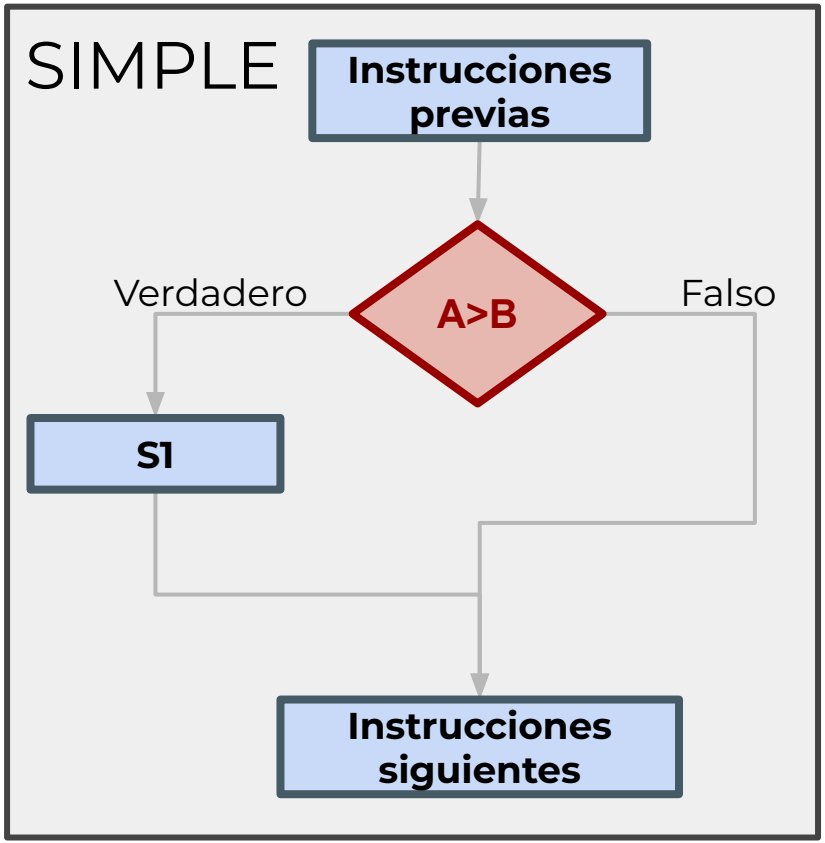
DOBLE:

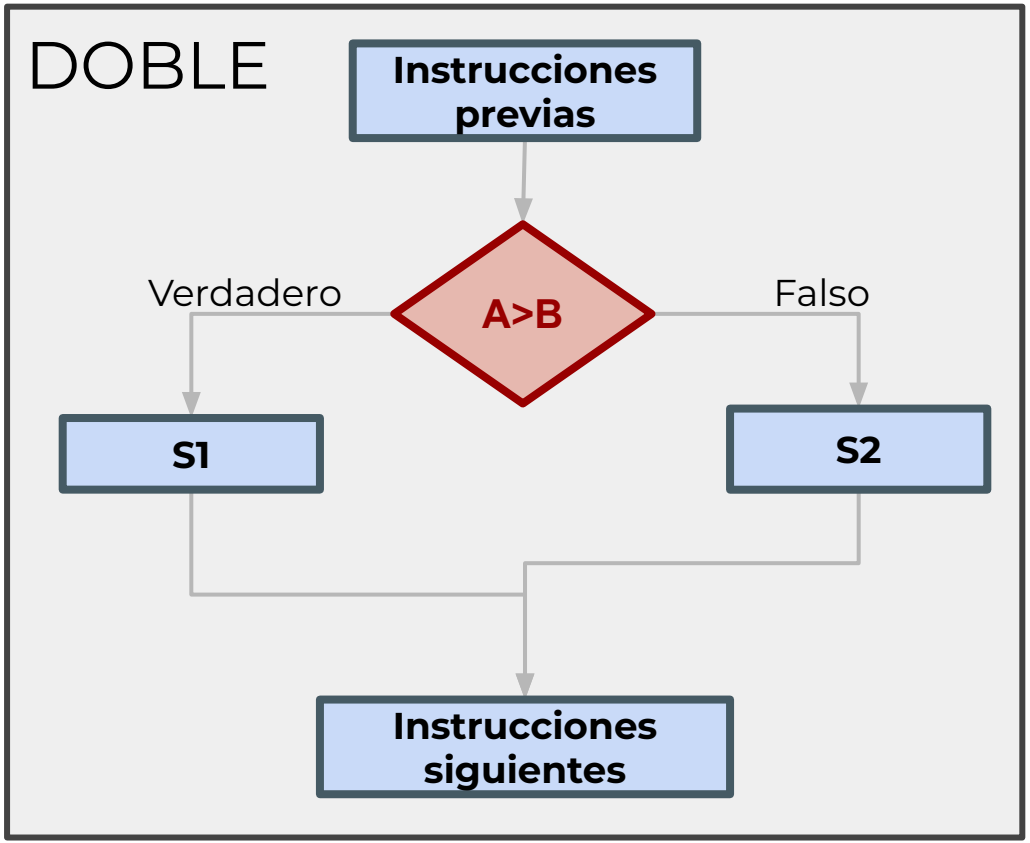
```
SI (condición )
    instrucción1
    instrucción2
SINO
    instrucción3
instrucción4
```

ANIDADA:

```
SI (condición1 )
    instrucción1
    instrucción2
SINO
    SI (condición2 )
        instrucción3
        instrucción4
    SINO
        instrucción5
instrucción6
```

Estructuras de control alternativas





Estructuras de control alternativas

Diseñar un algoritmo que determine si un número es par.

Estructuras de control alternativas

```
PROGRAMA esPar
VARIABLES
        num ENTERO
ALGORITMO
        ESCRIBIR(“Introduce un número”)
        LEER(num)
        SI (num % 2 == 0)
                ESCRIBIR (“El número” + num + “ es PAR”)
        SINO
                ESCRIBIR (“El número” + num + “ es IMPAR”)
FIN
```

Podemos utilizar el operador suma para concatenar más de una vez

ENTRADA	SALIDA
num = 6	El número 6 es PAR

Estructuras de control alternativas

Diseñar un algoritmo que determine si un número N es divisible por M.

Estructuras de control alternativas

```
PROGRAMA esMultiplo
VARIABLES
        numN, numM ENTERO
ALGORITMO
        ESCRIBIR("Introduce N y M")
        LEER(numN, numM)
        SI (numN % numM == 0)
                ESCRIBIR ("El número" + numN + "es divisible entre"+numM)
        SINO
                ESCRIBIR ("El número" + numN + "NO es divisible entre"+numM)
FIN
```

ENTRADA	SALIDA
numN= 15, numM= 5	El número 15 es divisible entre 5

Estructuras de control alternativas

Diseñar un algoritmo que lea dos valores numéricos X e Y, determine si son iguales y, en caso de no serlo, determine cuál de ellos es el mayor.

Estructuras de control alternativas

```
PROGRAMA sonIguales
VARIABLES
        num1, num2 ENTERO
ALGORITMO
        ESCRIBIR (“Introduce dos números”)
        LEER (num1, num2)
        SI (num1 == num2)
                ESCRIBIR (num1+ “=“+num2)
        SINO
                SI (num1 > num2)
                        ESCRIBIR (num1 +  “>“ + num2)
                SINO
                        ESCRIBIR (num1 +  “<“ + num2)
FIN
```

ENTRADA	SALIDA
num1 = 5, num2 = 6	5 < 6

Estructuras de control repetitivas

Repetitivas: bloque de código que puede ejecutarse más de una vez.

MIENTRAS (condición)

instrucción1
instrucción2
.....
instrucciónN

instrucción4

- ✓ *Se ejecutará mientras la condición sea cierta.*
- ✓ *Es posible que nunca se ejecuten las instrucciones.*

REPETIR

instrucción1
instrucción2
.....
instrucciónN

MIENTRAS (condición)

instrucción4

- ✓ *Se ejecutará mientras se cumpla la condición.*
- ✓ *Las instrucciones se ejecutan como mínimo una vez.*

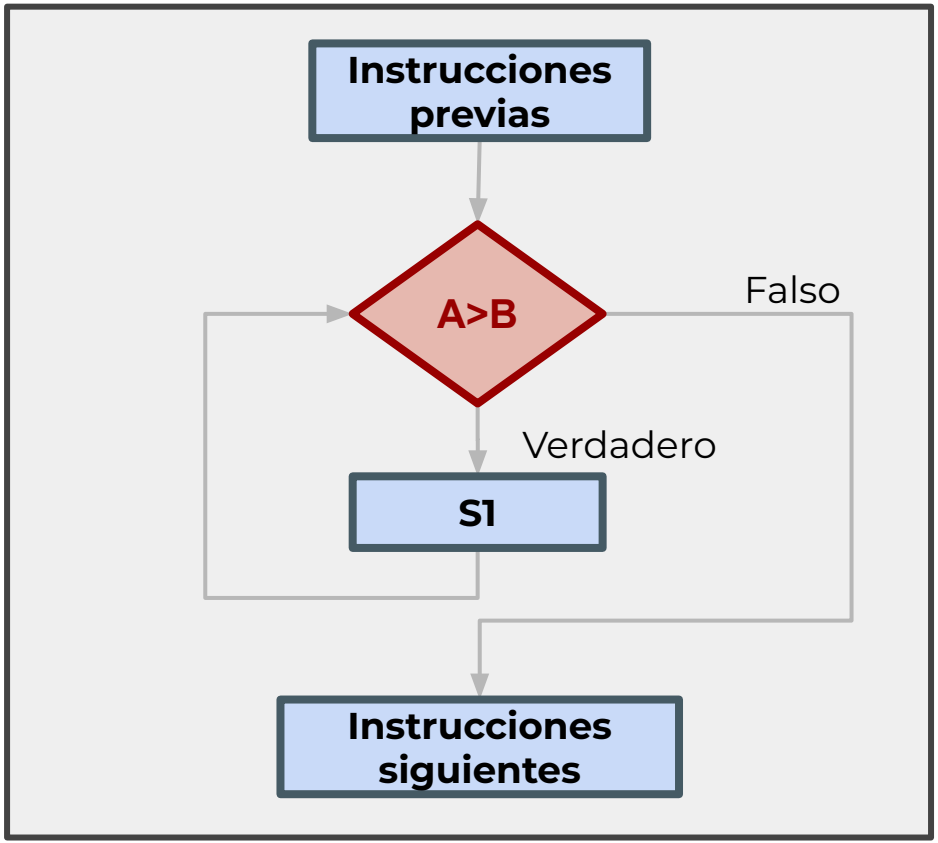
PARA (Vi; Vf; Vc)

instrucción1
instrucción2
.....
instrucciónN

instrucción4

- ✓ *Se ejecutará mientras se cumpla la condición.*
- ✓ *Las instrucciones se ejecutan un número determinado de veces.*
- ✓ *Se indica el valor inicial(Vi), valor final(Vf) y el incremento(Vc).*

Estructuras de control repetitivas



MIENTRAS

MIENTRAS:(condición)

HACER
    instrucción1
    instrucción2
    .....
    instrucciónN

instrucción4

- ✓ *Se ejecutará mientras la condición sea cierta.*
- ✓ *Es posible que nunca se ejecuten las instrucciones.*

Estructuras de control repetitivas

Diseñar un algoritmo que calcule la potencia de un número, dada la base y el exponente, ambos deben ser positivos.

Estructuras de control repetitivas

```
PROGRAMA calculaPotencia
VARIABLES
        base, exponente, contador, potencia ENTERO
ALGORITMO
        LEER (base, exponente)
        potencia = 1
        contador = 1
        MIENTRAS (contador < exponente)
                potencia = potencia*base
                contador = contador + 1
        ESCRIBIR(base+“elevado a”+ exponente + ”=“ + potencia)
FIN
```

ENTRADA	SALIDA
base = 2, exponente = 4	2 elevado a 4 = 8

base	exp	cont	poten
2	4	1	1
2	4	2	2
2	4	3	4
2	4	4	**8**

SOLUCIÓN NO VÁLIDA

Estructuras de control repetitivas

```
PROGRAMA calculaPotencia
VARIABLES
        base, exponente, contador, potencia ENTERO
ALGORITMO
        LEER (base, exponente)
        potencia = 1
        contador = 1
        MIENTRAS (contador <= exponente)
                potencia = potencia*base
                contador = contador + 1
        ESCRIBIR(base+ “elevado a”+ exponente+”=“+potencia)
FIN
```

ENTRADA	SALIDA
base = 2, exponente = 4	2 elevado a 4 = 16

base	exp	cont	poten
2	4	1	1
2	4	2	2
2	4	3	4
2	4	4	8
2	4	5	**16**

SOLUCIÓN 1, modificamos la condición del bucle

Estructuras de control repetitivas

```
PROGRAMA calculaPotencia
VARIABLES
        base, exponente, contador, potencia ENTERO
ALGORITMO
        LEER (base, exponente)
        potencia = 1
        contador = 0
        MIENTRAS (contador < exponente)
                potencia = potencia*base
                contador = contador + 1
        ESCRIBIR(base+ “elevado a”+ exponente+”=“+potencia)
FIN
```

ENTRADA	SALIDA
base = 2, exponente = 4	2 elevado a 4 = 8

base	exp	cont	poten
2	4	0	1
2	4	1	2
2	4	2	4
2	4	3	8
2	4	4	**16**

SOLUCIÓN 2, ponemos el contador a cero

Estructuras de control repetitivas

Leer dos números enteros, A y B. Calcular C = A*B mediante sumas sucesivas e imprimir el resultado.

Estructuras de control repetitivas

```
PROGRAMA calculaProducto
VARIABLES
        num1, num2, producto,contador ENTERO
ALGORITMO
        ESCRIBIR (“Introduce numero1 y numero2”)
        LEER (num1, num2)
        producto = 0
        contador = 0
        MIENTRAS (contador < num2)
                producto = producto + num1
                contador = contador + 1
        ESCRIBIR(num1+ “*”+ num2+”=“+producto)
FIN
```

num1	num2	cont	prod
5	4	0	0
5	4	1	5
5	4	2	10
5	4	3	15
5	4	4	**20**

ENTRADA	SALIDA
num1 = 5, num2 = 4	5*4=20

Estructuras de control repetitivas

```
PROGRAMA calculaProducto
VARIABLES
        num1, num2, producto,contador ENTERO
ALGORITMO
        ESCRIBIR (“Introduce numero1 y numero2”)
        LEER (num1, num2)
        producto = 0
        contador = 0
        MIENTRAS (contador < num2)
                producto = producto + num1
                contador = contador + 1
        ESCRIBIR(num1+ “*”+ num2+”=“+producto)
FIN
```

num1	num2	cont	prod
0	4	0	0
0	4	1	0
0	4	2	0
0	4	3	0
0	4	4	**0**

ENTRADA	SALIDA
num1 = 0, num2 = 4	0*4=0

Estructuras de control repetitivas

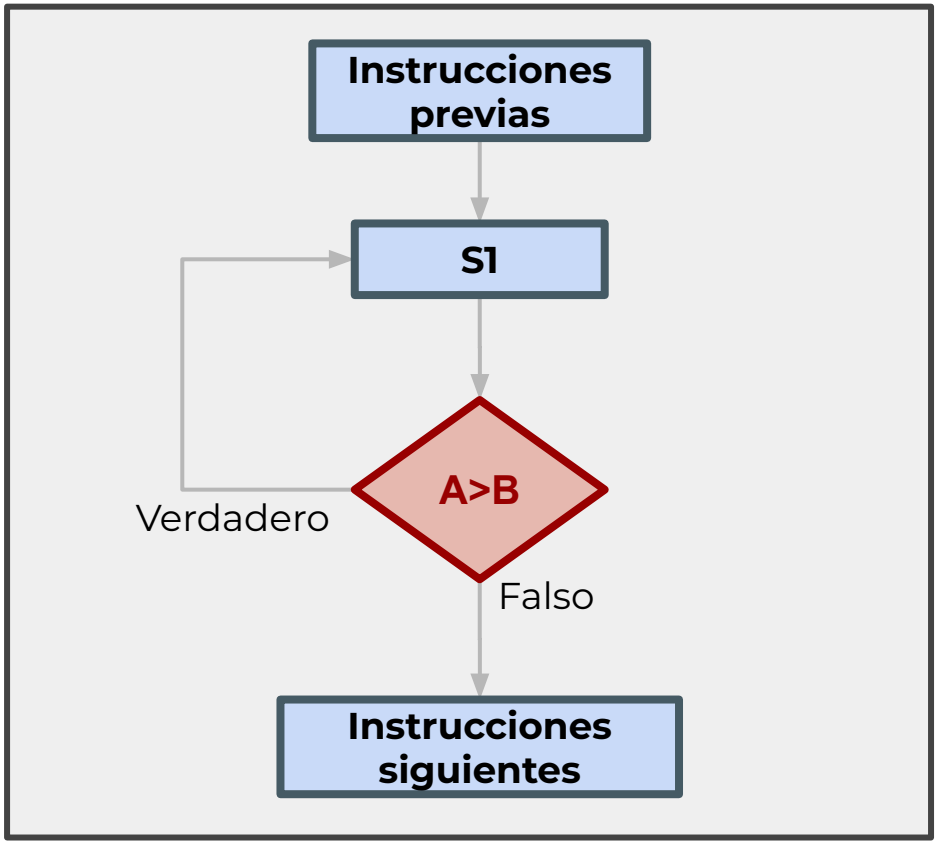
```
PROGRAMA calculaProducto
VARIABLES
        num1, num2, producto,contador ENTERO
ALGORITMO
        ESCRIBIR (“Introduce numero1 y numero2”)
        LEER (num1, num2)
        producto = 0
        contador = 0
        SI (num1 != 0)
                MIENTRAS (contador < num2)
                        producto = producto + num1
                        contador = contador + 1
        ESCRIBIR(num1+ “*”+ num2+”=“+producto)
FIN
```

Mejoramos el algoritmo en los casos donde el primer valor es 0

num1	num2	cont	prod
0	4	0	**0**

ENTRADA	SALIDA
num1 = 0, num2 = 4	0*4=0

Estructuras de control repetitivas



REPETIR...MIENTRAS

REPETIR

instrucción1
instrucción2
.....
instrucciónN

MIENTRAS(condición)

instrucción4

- ✓ *Se ejecutará mientras se cumpla la condición.*
- ✓ *Las instrucciones se ejecutan como mínimo una vez.*

Estructuras de control repetitivas

Calcular mediante un algoritmo repetitivo la suma de los N primeros números naturales, siendo N un número > 1.

Estructuras de control repetitivas

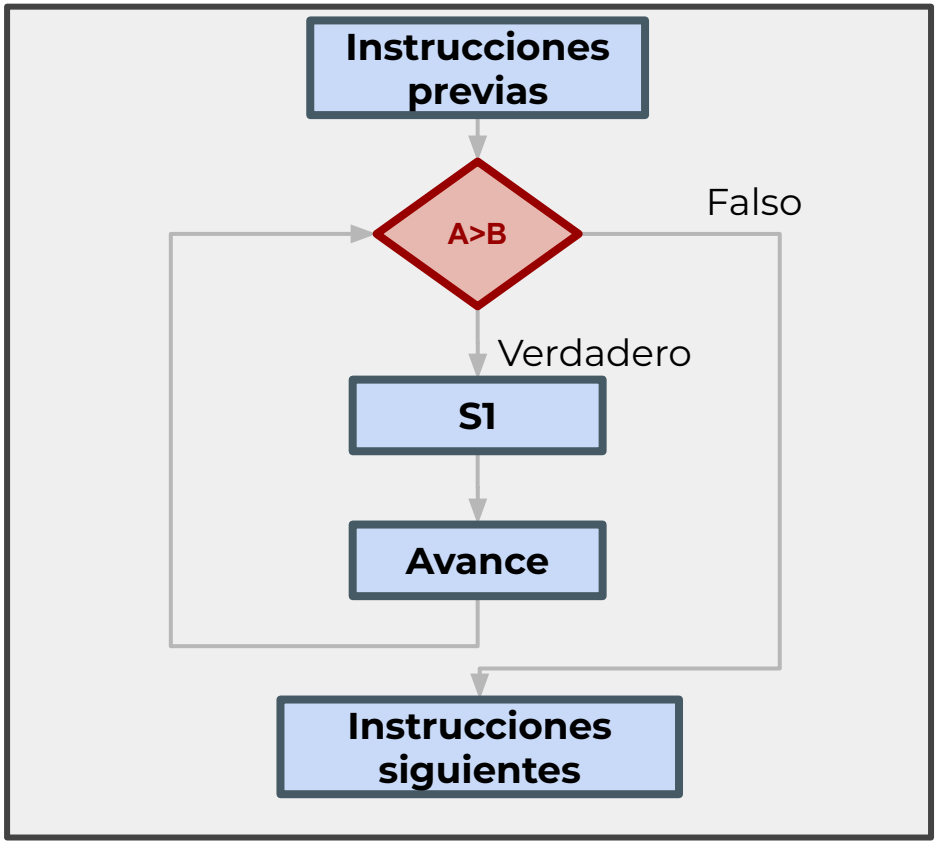
```
PROGRAMA calculaSuma
VARIABLES
        N, contador, suma ENTERO
ALGORITMO
        ESCRIBIR("Introduce un número")
        LEER (N)
        contador = 1
        suma = 0
        REPETIR
                suma += contador
                contador++
        MIENTRAS (contador <= N)
        ESCRIBIR ("La suma de los "+N + "números es " + suma)
FIN
```

Para N = 5

suma	cont
0	1
1	2
3	3
6	4
10	5
15	6

ENTRADA	SALIDA
N= 5	La suma de los 5 números es 15

Estructuras de control repetitivas



PARA...

PARA (Vi; Vf; Vc)

    instrucción1
    instrucción2
    .....
    instrucciónN

instrucción4

- ✓ *Se ejecutará mientras se cumpla la condición.*
- ✓ *Las instrucciones se ejecutan un número determinado de veces.*
- ✓ *Se indica el valor inicial(Vi), valor final(Vf) y el incremento(Vc).*

Estructuras de control repetitivas

Diseñar un algoritmo que lea 5 valores numéricos y calcule su producto, se debe utilizar un bucle PARA.

Estructuras de control repetitivas

```
PROGRAMA calculaProducto1
VARIABLES
        num, producto, contador ENTERO
ALGORITMO
        producto = 1
        PARA (contador = 1;  contador <= 5;  contador++)
                LEER (num)
                producto = producto*num
        ESCRIBIR("El producto de los 5 números es"+ producto)
FIN
```

Se pasa 5 veces por el bucle

num	prod	cont
-	1	1
3	3	2
5	15	3
2	30	4
7	210	5
2	420	6

ENTRADA	SALIDA
num = 3, 5, 2, 7, 2	El producto de los 5 números es 420

Estructuras de control repetitivas

```
PROGRAMA calculaProducto2
VARIABLES
        num, producto, contador ENTERO
ALGORITMO
        producto = 1
        PARA (contador = 0;  contador < 5;  contador++)
                LEER (num)
                producto = producto*num
        ESCRIBIR("El producto de los 5 números es"+ producto)
FIN
```

Se pasa 5 veces por el bucle

num	prod	cont
-	1	0
3	3	1
5	15	2
2	30	3
7	210	4
2	420	5

ENTRADA	SALIDA
num = 3, 5, 2, 7, 2	El producto de los 5 números es 420

Estructuras de control repetitivas

Diseña un algoritmo que sume los números pares entre el 1 y el 10.

Estructuras de control repetitivas

SOLUCIÓN CON BUCLE MIENTRAS

```
PROGRAMA calculaSumaPares1
VARIABLES
        contador, suma ENTERO
ALGORITMO
        contador = 2
        suma = 0
        MIENTRAS (contador <= 10)
                SI(contador%2 == 0)
                        suma += contador
                contador++
        ESCRIBIR (“La suma de los pares es” + suma)
FIN
```

SALIDA
La suma de los pares es 30

suma	contador
0	2
2	3
2	4
6	5
6	6
12	7
12	8
20	9
20	10
30	11

Estructuras de control repetitivas

```
PROGRAMA calculaSumaPares2
VARIABLES
        contador, suma ENTERO
ALGORITMO
        contador = 2
        suma = 0
        MIENTRAS (contador <= 10)
                suma+=contador
                contador+=2
        ESCRIBIR (“La suma de los pares es” + suma)
FIN
```

SOLUCIÓN MÁS EFICIENTE

SALIDA
La suma de los pares es 30

suma	contador
0	2
2	4
6	6
12	8
20	10
30	12

Estructuras de control repetitivas

SOLUCIÓN CON BUCLE PARA

```
PROGRAMA calculaSumaPares3
VARIABLES
        contador, suma ENTERO
ALGORITMO
        suma=0
        PARA(contador = 2; contador <= 10; contador += 2)
                suma = suma+contador
        ESCRIBIR (“La suma de los pares es” + suma)
FIN
```

SALIDA
La suma de los pares es 30

suma	contador
0	2
2	4
6	6
12	8
20	10
30	12

Estructuras de control repetitivas

Realizar un programa que lee diferentes puntuaciones hasta que se introduce un 0, y calcula la puntuación total de la siguiente manera:

- Si la puntuación está comprendida entre 2000 y 3000 puntos (ambas cantidades incluidas), calcular la puntuación total multiplicando los puntos por la base 0,3.
- Si la puntuación no está comprendida en este intervalo, multiplicar los puntos por la base 0,2.

Estructuras de control repetitivas

```
PROGRAMA calculaPuntuacion
VARIABLES
        puntuacion, total ENTERO
ALGORITMO
        total=0
        LEER (puntuacion)
        MIENTRAS (puntuacion != 0)
                SI (puntuación >= 2000 && puntuacion <= 3000)
                        total += puntuacion*0,3
                SINO
                        total += puntuacion*0,2
                LEER (puntuacion)
        ESCRIBIR (“La puntuación total es:”+ puntuacion)
FIN
```

puntuacion	total
-	0
1000	200
2000	800
3000	1700
0	1700

ENTRADA	SALIDA
puntuacion = 1000, 2000, 3000, 0	La puntuación total es: 1700

Estructuras de control repetitivas

Desarrollar un algoritmo para calcular e imprimir el factorial de un número.

Ejemplo:

5! = 5 x 4 x 3 x 2 x 1 = 120

Estructuras de control repetitivas

```
PROGRAMA factorial
VARIABLES
        num, contador, factorial ENTERO
ALGORITMO
        ESCRIBIR (“Introduce numero”)
        LEER (num)
        factorial = 1
        contador = num
        MIENTRAS (contador >= 1)
                factorial = factorial*contador
                contador--
        ESCRIBIR(“El factorial de”+ num + ”es“ + factorial)
FIN
```

fact	cont
1	5
5	4
20	3
60	2
120	1
120	0

ENTRADA	SALIDA
num = 5	El factorial de 5 es 120

Estructuras de control repetitivas

```
PROGRAMA factorial
VARIABLES
        num, contador, factorial ENTERO
ALGORITMO
        ESCRIBIR (“Introduce numero”)
        LEER (num)
        factorial = 1
        contador = num
        MIENTRAS (contador > 1)
                factorial = factorial*contador
                contador--
        ESCRIBIR(“El factorial de”+ num + ”es“ + factorial)
FIN
```

1 pasada menos por el bucle no afecta al resultado final

fact	cont
1	5
5	4
20	3
60	2
120	1

ENTRADA	SALIDA
num = 5	El factorial de 5 es 120

Estructuras de control repetitivas

SOLUCIÓN CON BUCLE PARA

```
PROGRAMA factorial
VARIABLES
        num, contador, factorial ENTERO
ALGORITMO
        ESCRIBIR (“Introduce numero”)
        LEER (num)
        factorial = 1
        PARA (contador = num; contador > 1; contador-- )
                factorial *= contador
        ESCRIBIR(“El factorial de”+ num + ”es“ + factorial)
FIN
```

fact	cont
1	5
5	4
20	3
60	2
120	1

ENTRADA	SALIDA
num = 5	El factorial de 5 es 120

Seguimiento de algoritmos
Algoritmo 1

Seguimiento de algoritmos

Indica los valores de A y B tras finalizar el siguiente algoritmo:

```
PROGRAMA algoritmo1
VARIABLES
    A,B Entero
ALGORITMO
    A=2
    B=2
    MIENTRAS(A>0)
        A--
        B++
    MIENTRAS(B>0)
        A++
        B--
    B+=A
    A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2

```
PROGRAMA algoritmo1
VARIABLES
      A,B Entero
ALGORITMO
      A=2
      B=2
      MIENTRAS(A>0)
            A--
            B++
      MIENTRAS(B>0)
            A++
            B--
      B+=A
      A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3

```
PROGRAMA algoritmo1
VARIABLES
        A,B Entero
ALGORITMO
        A=2
        B=2
        MIENTRAS(A>0)
                A--
                B++
        MIENTRAS(B>0)
                A++
                B--
        B+=A
        A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3
0	4

```
PROGRAMA algoritmo1
VARIABLES
    A,B Entero
ALGORITMO
    A=2
    B=2
    MIENTRAS(A>0)
        A--
        B++
    MIENTRAS(B>0)
        A++
        B--
    B+=A
    A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3
0	4
1	3

```
PROGRAMA algoritmo1
VARIABLES
    A,B Entero
ALGORITMO
    A=2
    B=2
    MIENTRAS(A>0)
        A--
        B++
    MIENTRAS(B>0)
        A++
        B--
    B+=A
    A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3
0	4
1	3
2	2

```
PROGRAMA algoritmo1
VARIABLES
        A,B Entero
ALGORITMO
        A=2
        B=2
        MIENTRAS(A>0)
                A--
                B++
        MIENTRAS(B>0)
                A++
                B--
        B+=A
        A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3
0	4
1	3
2	2
3	1

```
PROGRAMA algoritmo1
VARIABLES
        A,B Entero
ALGORITMO
        A=2
        B=2
        MIENTRAS(A>0)
                A--
                B++
        MIENTRAS(B>0)
                A++
                B--
        B+=A
        A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3
0	4
1	3
2	2
3	1
4	0

```
PROGRAMA algoritmo1
VARIABLES
    A,B Entero
ALGORITMO
    A=2
    B=2
    MIENTRAS(A>0)
        A--
        B++
    MIENTRAS(B>0)
        A++
        B--
    B+=A
    A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3
0	4
1	3
2	2
3	1
4	0
4	4

```
PROGRAMA algoritmo1
VARIABLES
    A,B Entero
ALGORITMO
    A=2
    B=2
    MIENTRAS(A>0)
        A--
        B++
    MIENTRAS(B>0)
        A++
        B--
    B+=A
    A*=B
FIN
```

Seguimiento de algoritmos

A	B
2	2
1	3
0	4
1	3
2	2
3	1
4	0
4	4
16	**4**

```
PROGRAMA algoritmo1
VARIABLES
        A,B Entero
ALGORITMO
        A=2
        B=2
        MIENTRAS(A>0)
                A--
                B++
        MIENTRAS(B>0)
                A++
                B--
        B+=A
        A*=B
FIN
```

Seguimiento de algoritmos
Algoritmo 2

Seguimiento de algoritmos

Indica los valores de A, B y C tras finalizar el siguiente algoritmo:

```
PROGRAMA algoritmo2
VARIABLES
        A,B,C       Entero
ALGORITMO
        A=1
        B=0
        C=0
        SI(B!=C || A!=B && A==B)
                A++
                B+=A
        SI(C%A==0)
                C+=A
        A*=2
        B-=2
        C/=2
FIN
```

Seguimiento de algoritmos

A	B	C
1	0	0

```
PROGRAMA algoritmo2
VARIABLES
        A,B,C      Entero
ALGORITMO
        A=1
        B=0
        C=0
        SI(B!=C || A!=B && A==B)
                A++
                B+=A
        SI(C%A==0)
                C+=A
        A*=2
        B-=2
        C/=2
FIN
```

Seguimiento de algoritmos

A	B	C
1	0	0
1	0	0

```
PROGRAMA algoritmo2
VARIABLES
        A,B,C      Entero
ALGORITMO
        A=1
        B=0
        C=0
        SI(B!=C || A!=B && A==B)
                A++
                B+=A
        SI(C%A==0)
                C+=A
        A*=2
        B-=2
        C/=2
FIN
```

Seguimiento de algoritmos

A	B	C
1	0	0
1	0	0
1	0	1

```
PROGRAMA algoritmo2
VARIABLES
        A,B,C      Entero
ALGORITMO
        A=1
        B=0
        C=0
        SI(B!=C || A!=B && A==B)
                A++
                B+=A
        SI(C%A==0)
                C+=A
        A*=2
        B-=2
        C/=2
FIN
```

Seguimiento de algoritmos

A	B	C
1	0	0
1	0	0
1	0	1
2	0	1

```
PROGRAMA algoritmo2
VARIABLES
        A,B,C      Entero
ALGORITMO
        A=1
        B=0
        C=0
        SI(B!=C || A!=B && A==B)
                A++
                B+=A
        SI(C%A==0)
                C+=A
        A*=2
        B-=2
        C/=2
FIN
```

Seguimiento de algoritmos

A	B	C
1	0	0
1	0	0
1	0	1
2	0	1
2	-2	1

```
PROGRAMA algoritmo2
VARIABLES
        A,B,C      Entero
ALGORITMO
        A=1
        B=0
        C=0
        SI(B!=C || A!=B && A==B)
                A++
                B+=A
        SI(C%A==0)
                C+=A
        A*=2
        B-=2
        C/=2
FIN
```

Seguimiento de algoritmos

A	B	C
1	0	0
1	0	0
1	0	1
2	0	1
2	-2	1
2	**-2**	**0**

```
PROGRAMA algoritmo2
VARIABLES
        A,B,C      Entero
ALGORITMO
        A=1
        B=0
        C=0
        SI(B!=C || A!=B && A==B)
                A++
                B+=A
        SI(C%A==0)
                C+=A
        A*=2
        B-=2
        C/=2
FIN
```

Seguimiento de algoritmos
Algoritmo 3

Seguimiento de algoritmos

Indica los valores de A, B y C tras finalizar el siguiente algoritmo:

```
PROGRAMA algoritmo3
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            B*=C
            C++
            A--
      A+=B
      C-=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2

```
PROGRAMA algoritmo3
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            B*=C
            C++
            A--
      A+=B
      C-=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
4	6	3

```
PROGRAMA algoritmo3
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            B*=C
            C++
            A--
      A+=B
      C-=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
4	6	3
3	18	4

```
PROGRAMA algoritmo3
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            B*=C
            C++
            A--
      A+=B
      C-=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
4	6	3
3	18	4
21	18	4

```
PROGRAMA algoritmo3
VARIABLES
    A,B,C   Entero
ALGORITMO
    A=5, B=3, C=2
    MIENTRAS(A>C)
        B*=C
        C++
        A--
    A+=B
    C-=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
4	6	3
3	18	4
21	18	4
21	**18**	**-17**

```
PROGRAMA algoritmo3
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            B*=C
            C++
            A--
      A+=B
      C-=A
FIN
```

Seguimiento de algoritmos
Algoritmo 4

Seguimiento de algoritmos

Indica los valores de A, B y C tras finalizar el siguiente algoritmo:

```
PROGRAMA algoritmo4
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            SI(A==B || A>=C+B)
                  B*=C
            C++
      A%=B
      C/=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2

```
PROGRAMA algoritmo4
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            SI(A==B || A>=C+B)
                  B*=C
            C++
      A%=B
      C/=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
5	6	3

```
PROGRAMA algoritmo4
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            SI(A==B || A>=C+B)
                  B*=C
            C++
      A%=B
      C/=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
5	6	3
5	6	4

```
PROGRAMA algoritmo4
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=5, B=3, C=2
      MIENTRAS(A>C)
            SI(A==B || A>=C+B)
                  B*=C
            C++
      A%=B
      C/=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
5	6	3
5	6	4
5	6	5

```
PROGRAMA algoritmo4
VARIABLES
    A,B,C   Entero
ALGORITMO
    A=5, B=3, C=2
    MIENTRAS(A>C)
        SI(A==B || A>=C+B)
            B*=C
        C++
    A%=B
    C/=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
5	6	3
5	6	4
5	6	5
5	6	5

```
PROGRAMA algoritmo4
VARIABLES
    A,B,C   Entero
ALGORITMO
    A=5, B=3, C=2
    MIENTRAS(A>C)
        SI(A==B || A>=C+B)
            B*=C
        C++
    A%=B
    C/=A
FIN
```

Seguimiento de algoritmos

A	B	C
5	3	2
5	6	3
5	6	4
5	6	5
5	6	5
5	**6**	**1**

```
PROGRAMA algoritmo4
VARIABLES
    A,B,C   Entero
ALGORITMO
    A=5, B=3, C=2
    MIENTRAS(A>C)
        SI(A==B || A>=C+B)
            B*=C
        C++
    A%=B
    C/=A
FIN
```

Seguimiento de algoritmos
Algoritmo 5

Seguimiento de algoritmos

Indica los valores de A, B y C tras finalizar el siguiente algoritmo:

```
PROGRAMA algoritmo5
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=0, B=0, C=0
      MIENTRAS(A<5)
            SI(A%2==0)
                  A++
            B++
            A++
      A+=B
      C+=A+B
FIN
```

Seguimiento de algoritmos

A	B	C
0	0	0

```
PROGRAMA algoritmo5
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=0, B=0, C=0
      MIENTRAS(A<5)
            SI(A%2==0)
                  A++
            B++
            A++
      A+=B
      C+=A+B
FIN
```

Seguimiento de algoritmos

A	B	C
0	0	0
2	1	0

```
PROGRAMA algoritmo5
VARIABLES
      A,B,C    Entero
ALGORITMO
      A=0, B=0, C=0
      MIENTRAS(A<5)
            SI(A%2==0)
                  A++
            B++
            A++
      A+=B
      C+=A+B
FIN
```

Seguimiento de algoritmos

A	B	C
0	0	0
2	1	0
4	2	0

```
PROGRAMA algoritmo5
VARIABLES
    A,B,C    Entero
ALGORITMO
    A=0, B=0, C=0
    MIENTRAS(A<5)
        SI(A%2==0)
            A++
        B++
        A++
    A+=B
    C+=A+B
FIN
```

Seguimiento de algoritmos

A	B	C
0	0	0
2	1	0
4	2	0
6	3	0

```
PROGRAMA algoritmo5
VARIABLES
    A,B,C    Entero
ALGORITMO
    A=0, B=0, C=0
    MIENTRAS(A<5)
        SI(A%2==0)
            A++
        B++
        A++
    A+=B
    C+=A+B
FIN
```

Seguimiento de algoritmos

A	B	C
0	0	0
2	1	0
4	2	0
6	3	0
9	3	0

```
PROGRAMA algoritmo5
VARIABLES
    A,B,C    Entero
ALGORITMO
    A=0, B=0, C=0
    MIENTRAS(A<5)
        SI(A%2==0)
            A++
        B++
        A++
    A+=B
    C+=A+B
FIN
```

Seguimiento de algoritmos

A	B	C
0	0	0
2	1	0
4	2	0
6	3	0
9	3	0
9	**3**	**12**

```
PROGRAMA algoritmo5
VARIABLES
    A,B,C    Entero
ALGORITMO
    A=0, B=0, C=0
    MIENTRAS(A<5)
        SI(A%2==0)
            A++
        B++
        A++
    A+=B
    C+=A+B
FIN
```

Seguimiento de algoritmo con bucles anidados

Seguimiento de algoritmos con bucles anidados

Escribe la salida del siguiente algoritmo:

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

i	j
4	-
4	4
4	3
4	2
4	1
4	0
4	-1

SALIDA

43210

Fin del primer bucle interno

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

i	j
3	-1
3	3
3	2
3	1
3	0
3	-1

SALIDA

43210

3210

Fin del segundo bucle interno

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

i	j
2	-1
2	2
2	1
2	0
2	-1

SALIDA

43210

3210

210

Fin del tercer bucle interno

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

i	j
1	-1
1	1
1	0
1	-1

SALIDA

43210

3210

210

10

Fin del cuarto bucle interno

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

i	j
0	-1
0	0
0	-1

SALIDA

43210

3210

210

10

0

Fin del quinto bucle interno

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

i	j
0	-1
0	0
0	-1
-1	-1

SALIDA

43210

3210

210

10

0

Fin del bucle externo

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

Escribe la salida del siguiente algoritmo:

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i > 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

Escribe la salida del siguiente algoritmo:

43210
3210
210
10

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i > 0)
                j = i
                MIENTRAS (j >= 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

Escribe la salida del siguiente algoritmo:

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j > 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

Escribe la salida del siguiente algoritmo:

4321

321

21

1

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i >= 0)
                j = i
                MIENTRAS (j > 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

Escribe la salida del siguiente algoritmo:

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i > 0)
                j = i
                MIENTRAS (j > 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Seguimiento de algoritmos con bucles anidados

Escribe la salida del siguiente algoritmo:

4321

321

21

1

```
PROGRAMA algoritmo6
VARIABLES
        i, j,    Entero
ALGORITMO
        i = 4
        MIENTRAS (i > 0)
                j = i
                MIENTRAS (j > 0)
                        ESCRIBIR j
                        j--
                ESCRIBIR Cambio de línea
                i--
FIN
```

Ejercicios adicionales

Ejercicios adicionales

1. Leer dos números A y B e intercambiar el valor de sus variables.
2. Dado un tiempo en minutos, calcula los días, horas y minutos que le corresponden.
3. Leer 5 números enteros y decir si alguno ha sido negativo e indicar el valor máximo introducido.
4. Dado un número, determinar cuántos dígitos tiene.
5. Leer una cantidad ‘N’ mayor que cero, y luego introducir ‘N’ números enteros positivos. Se pide imprimir el mayor y el menor y las veces que aparece cada uno.

Ejercicios adicionales

1. Leer dos números A y B e intercambiar el valor de sus variables.

```
PROGRAMA intercambio
VARIABLES
        a, b, c  ENTERO
ALGORITMO:
        ESCRIBIR (“Introduce dos números”)
        LEER(a, b)
        c = a
        a = b
        b = c
        ESCRIBIR (“A: “ + a + ” B: “ + b)
FIN
```

ENTRADA	SALIDA
a = 5, b = 7	A:7 B:5

Ejercicios adicionales

2. Dado un tiempo en minutos, calcula los días, horas y minutos que le corresponden.

```
PROGRAMA calcularDiasHorasMinutos
VARIABLES
        tiempo, dias, horas, minutos, resto, minutosDia ENTERO
ALGORITMO:
        ESCRIBIR (“Introduce el tiempo en minutos”)
        LEER (tiempo)
        minutosDia = 60*24
        dias = tiempo / minutosDia
        resto = tiempo % minutosDia
        horas = resto/60
        minutos = resto % 60
        ESCRIBIR (“Días: “+ dias + “ - Horas: ” + horas + “ - Minutos: ”+ minutos)
FIN
```

SOLUCIÓN VÁLIDA CON TRUNCADO, mismo comportamiento por defecto que JAVA

ENTRADA	SALIDA
tiempo = 1500	Días: 1 - Horas: 1 - Minutos: 0

Ejercicios adicionales

3. Leer 5 números enteros y decir si alguno ha sido negativo e indicar el valor máximo introducido.

```
PROGRAMA ejercicio3
VARIABLES
        num, max, contador ENTERO
        negativo BOOLEAN
ALGORITMO
        contador = 0
        max = 0
        negativo = FALSE
        MIENTRAS (contador < 5)
                LEER(num)
                SI (num < 0)
                        negativo = TRUE
                SI (num > max)
                        max = num
                contador++
        SI(negativo == TRUE)
                ESCRIBIR (“Algún número ha sido negativo.”)
        SINO
                ESCRIBIR (“No hay ningún número negativo.”)
        ESCRIBIR(“El número máximo ha sido “+ max)
FIN
```

Ejercicios adicionales

3. Leer 5 números enteros y decir si alguno ha sido negativo e indicar el valor máximo introducido.

Si todos son negativos, max será 0, se puede solucionar iniciando max al primer elemento que leemos

ENTRADA	SALIDA
num = 5, num = 6, num = 3, num = -2, num = 1	Algún número ha sido negativo. El número máximo ha sido 6

```
PROGRAMA ejercicio3
VARIABLES
        num, max, contador ENTERO
        negativo BOOLEAN
ALGORITMO
        contador = 0
        negativo = FALSE
        MIENTRAS (contador < 5)
                LEER(num)
                SI (contador == 0)
                        max = num
                SI (num < 0)
                        negativo = TRUE
                SI (num > max)
                        max = num
                contador++
        SI(negativo == TRUE)
                ESCRIBIR (“Algún número ha sido negativo.”)
        SINO
                ESCRIBIR (“No hay ningún número negativo.”)
        ESCRIBIR(“El número máximo ha sido “+ max)
FIN
```

Ejercicios adicionales

4. Dado un número, determinar la cantidad de dígitos que tiene

num	digitos
1014	0
101	1
10	2
1	3
0	4

```
PROGRAMA ejercicio4
VARIABLES
      num, digitos ENTERO
ALGORITMO
      ESCRIBIR    (“Introduce un número”)
      LEER (num)
      digitos = 0
      MIENTRAS (num != 0)
            num /=10
            digitos++
      ESCRIBIR    (“Cantidad de dígitos: “+ digitos)
FIN
```

SOLUCIÓN VÁLIDA CON TRUNCADO

ENTRADA	SALIDA
a = 1014	Cantidad de dígitos: 4

Ejercicios adicionales

5. Leer una cantidad ‘N’ mayor que cero, y luego introducir ‘N’ números enteros positivos. Se pide imprimir el mayor y el menor y las veces que aparece cada uno.

ENTRADA	SALIDA
N = 5 num = 1 num = 5 num = 5 num = 1 num = 6	El mínimo es 1 y aparece 2 veces. El máximo es 6 y aparece 1 veces.

```
PROGRAMA numeros
VARIABLES
        n, num, cont, max, min, contMax, contMin ENTERO
ALGORITMO
        ESCRIBIR (“introduce N”)
        LEER(n)
        cont = 0
        MIENTRAS (cont < n)
                ESCRIBIR (“Introduce un número”)
                LEER (num)
                SI (cont == 0 )
                    max = num, min = num, contMax = 1, contMin = 1
                SINO
                    SI (num > max)
                        max = num
                        contMax= 1
                    SINO
                        SI (num == max)
                            contMax++
                    SI (num < min)
                        min = num
                        contMin = 1
                     SINO
                        SI (num == min)
                            contMin ++
                  cont++
        ESCRIBIR (“El mínimo es ” + min + “ y aparece “ + numMin + “ veces.”)
        ESCRIBIR (“El máximo es ” + max + “ y aparece “ + numMax + “ veces.”)
FIN
```

Ejercicios de ampliación

Ejercicios de ampliación

1. Dado un número calcular el producto de los dígitos distintos de cero.
2. Realizar un algoritmo que lea una serie de números enteros y verifique si están ordenados ascendentemente o no, informando con un mensaje. La serie finalizará cuando se introduzca un cero
3. Dado un número indicar si está ordenado. Un número está ordenado cuando el dígito que tiene a la derecha es mayor que él.

Bibliografía:

Allen Weiss, M. (2007). *Estructuras de datos en JAVA*. Madrid: Pearson

Froufe Quintas, A. (2002). *JAVA 2: Manual de usuario y tutorial*. Madrid: RA-MA

J. Barnes, D. *Programación orientada a objetos en JAVA*. Madrid: Pearson

Desing Patterns. Elements of Reusable. OO Software

JAVA Limpio. Pello Altadi. Eugenia Pérez

Apuntes de Programación de Anna Sanchis Perales

Apuntes de Programación de Lionel Tarazón Alcocer

Ilustraciones:

https://pixabay.com/

https://freepik.es/

Preguntas