

BIENVENIDO



PROGRAMADOR

INDICE





**le
BIMESTRE**

SESIÓN 01

¿QUÉ ES LA PROGRAMACIÓN?

La programación informática es el arte del proceso por el cual se limpia, codifica, traza y protege el código fuente de programas computacionales, en otras palabras, es indicarle a la computadora lo que tiene que hacer.



¿QUÉ GENERA LA PROGRAMACIÓN?

CREATIVIDAD

La programación despierta la creatividad

LÓGICA

La programación es lógica



DISCIPLINA

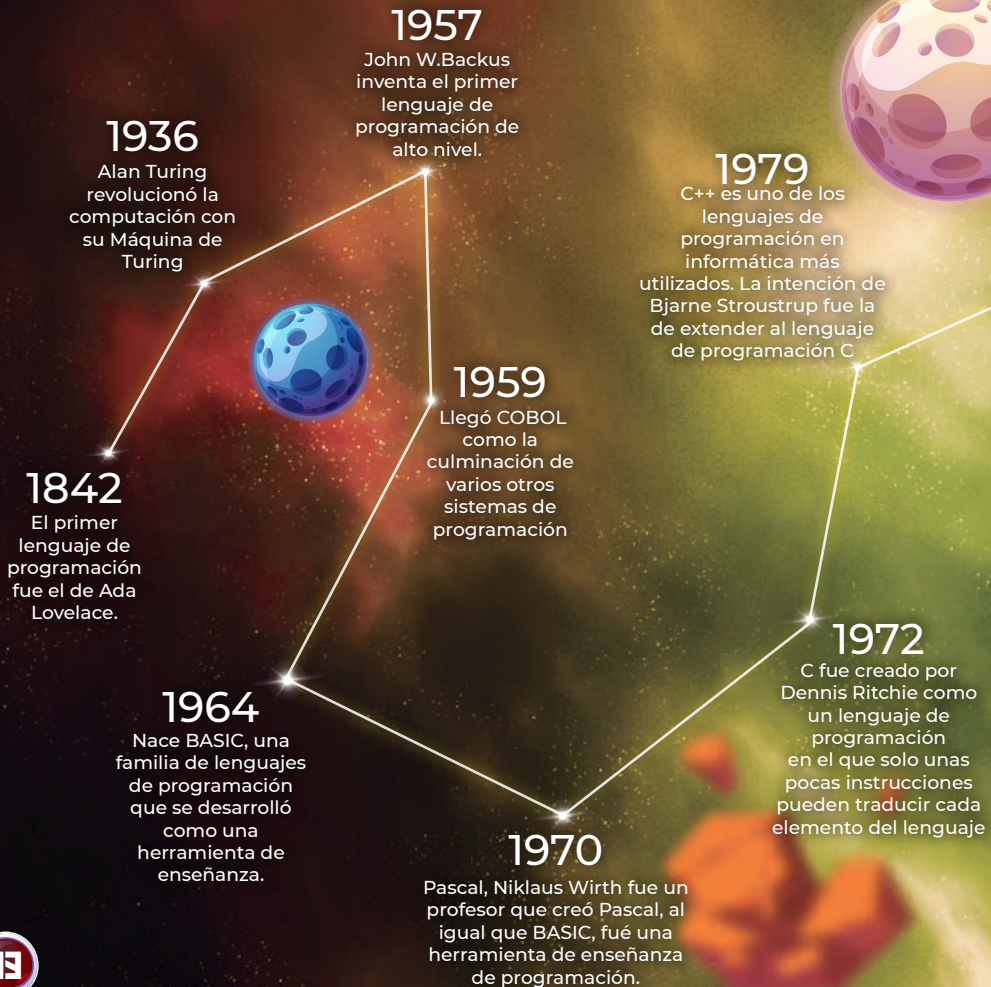
Para ser bueno programando se necesita constante estudio

INNOVAR

la programación como herramienta de la tecnología



LA HISTORIA DE LA PROGRAMACIÓN



1991

HTML, Python y Visual Basic legó a la década de Internet

1995

Aparecen Java, JavaScript y PHP, definiendo la manera en la que entendemos el mundo.

2001

Nació C# Con la llegada de la década de los 2000, llegaron nuevos lenguajes de programación, como Action Script.

2006

Aparece Scratch, la "revolución" de los lenguaje de programación

2013

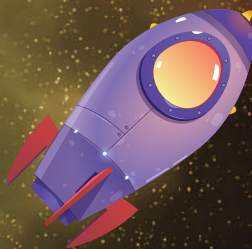
Se crearon diferentes lenguajes específicos para agilizar la programación.

2009

Go también conocido como Goland, es un lenguaje de programación diseñado por Google

2012

Kotlin es uno de los lenguajes de programación en informática de moda



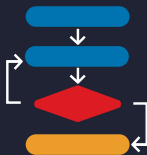
¿QUÉ SON LOS PARADIGMAS DE PROGRAMACIÓN?

Los paradigmas son los diferentes estilos de usar la programación para resolver un problema.



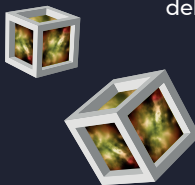
PROGRAMACIÓN ESTRUCTURADA

Programación secuencial con lo que todos aprendemos a programar. Usa ciclos y condiciones.



PROGRAMACIÓN ORIENTADA A OBJETOS

Divide los componentes del programa en objetos que tienen datos y comportamiento y se comunican entre sí.



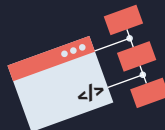
PROGRAMACIÓN REACTIVA

Divide el programa en tareas pequeñas que son ejecutadas por funciones.



PROGRAMACIÓN FUNCIONAL

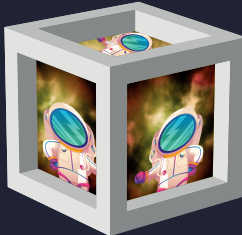
Divide el programa en tareas pequeñas que son ejecutadas por funciones.



¿QUÉ ES LA PROGRAMACIÓN ORIENTADA A OBJETOS?

- Los objetos se crean a partir de una plantilla llamada **clase**. Cada objeto es una instancia de su clase.

CLASE

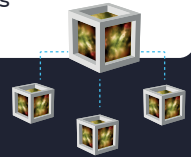


INSTANCIACIÓN

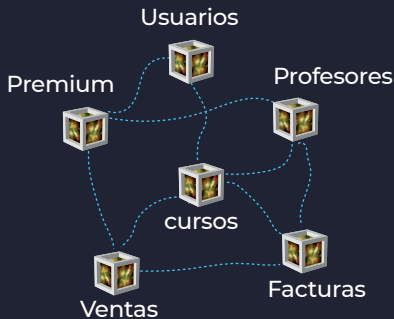
OBJETO



- Los objetos tienen **datos (atributos)** y **funcionalidades (métodos)**.



- En una aplicación los objetos están separados **pero se comunican entre ellos**.



ATRIBUTOS

- Nombres
- Apellidos
- Correo
- Contraseña
- Premium



MÉTODOS

- Editar perfil
- Iniciar Sesión
- Cerrar sesión
- Cambiar contraseña
- Pasar a premium



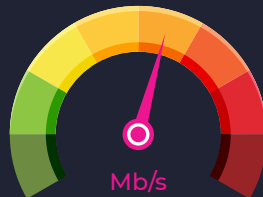
Puedes programar con este paradigma **en la mayoría de lenguajes**.



¿QUÉ ES LA PROGRAMACIÓN REACTIVA?



La Programación Reactiva(Rx) consiste en analizar flujos de datos asíncronos(streams) y reaccionar a sus cambios.



Observable

Objeto que entregará un conjunto de valores en el futuro.

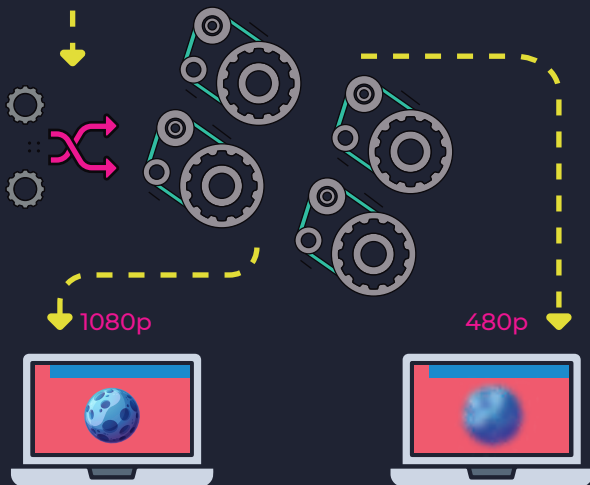
● Pipe

Envía los resultados de un operador a otro.



● Operator

Funciones puras que realizan operaciones.



Observer

Se suscribe a un observable y es capaz de reaccionar cuando los datos cambian.

Con esta técnica, Netflix puede observar tu velocidad de conexión y cambiar la calidad del video.



LENGUAJE DE PROGRAMACIÓN SEGÚN EL PARADIGMA

En la programación hay **muchas maneras de resolver un problema** y cada una de ellas **constituye un paradigma**.



FUNCIONALES



No se especifica cómo se deben hacer las cosas sino **qué debe hacer el programa**.

MULTIPARADIGMA



Pueden programar de varias maneras. **El programador escoge el paradigma adecuado para cada trabajo.**



REACTIVOS



Permiten que los programas **reaccionen a eventos o cambios en los datos**. Existen librerías como RX que **pueden aplicarse a casi todos los lenguajes**.

ORIENTADOS A OBJETOS



Todo se trata como un objeto, tienen **características y comportamientos diferentes**.



SESIÓN 02

GENERACIONES DE LENGUAJE DE PROGRAMACIÓN

- Los lenguajes de primera generación, o 1GL, lenguajes de bajo nivel que son lenguaje de máquina.
- Los lenguajes de segunda generación, o 2GL, lenguajes de bajo nivel que generalmente consisten en lenguajes ensamblados.
- Los lenguajes de tercera generación, o 3GL, lenguajes de alto nivel como C.
- Los lenguajes de cuarta generación, o 4GL, son idiomas que consisten en declaraciones similares a las declaraciones en un lenguaje humano. Los lenguajes de cuarta generación se usan comúnmente en la programación de bases de datos y scripts.
- Los idiomas de quinta generación, o 5GL, son lenguajes de programación que contienen herramientas visuales para ayudar a desarrollar un programa. Un buen ejemplo de un lenguaje de quinta generación es Visual Basic.



NIVELES DE LOS LENGUAJES DE PROGRAMACIÓN

BAJO NIVEL:

1). LENGUAJE DE MÁQUINA

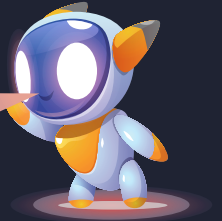
¿QUÉ ENTIENDE UNA COMPUTADORA?

Las computadoras NO entienden directamente letras, números, archivos o fotos. En realidad solo entienden unos y ceros.



mmm... Khe?

Inserta texto en word y abre una imagen en photoshop.



01000100 01100001
01101100 01100101
00100000 01101100.

¿Para mí?



A esto se le llama **SISTEMA BINARIO** por dentro usan transistores que se encargan de guardar los ceros y unos para procesar la información.



funciona como un bombillo



Hay sistemas como el **ASCII** que se encargan de traducir esto a objetos que los humanos entiendan



1). LENGUAJE DE ENSAMBLADOR

- El problema es que la computadora no comprende el código ensamblador, por lo que necesitamos una forma de convertirlo a código de máquina, que la computadora sí entiende.
- Los programas de lenguaje ensamblador se traducen al lenguaje de máquina mediante un programa llamado ensamblador.



ALTO NIVEL:

- Los idiomas de alto nivel nos permiten escribir códigos de computadora usando instrucciones que se asemejan al lenguaje hablado cotidiano (por ejemplo: imprimir, si, mientras) que luego se traducen al lenguaje de máquina para ser ejecutados.
- Los programas escritos en un lenguaje de alto nivel deben ser traducidos al lenguaje de máquina antes de que puedan ser ejecutados.



DIFERENCIAS

ENTRE LOS LENGUAJES DE PROGRAMACIÓN



ALTO NIVEL

Es un lenguaje que entienden los **HUMANOS.**

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main() {
6
7      cout << "Hola Pedro" << endl;
8
9      return 0;
10 }
```

- Está orientado al **software**.
- Utilizan **menos instrucciones** para realizar una acción.
- Te permite programar **aplicaciones y videojuegos**



BAJO NIVEL

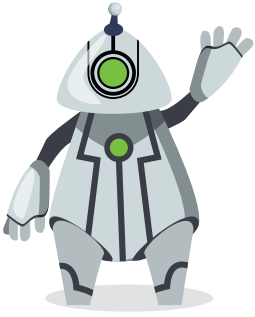
Son instrucciones para el **PROCESADOR.**

```
1  COPY      START      2010H
2           LDX          ZERO
3  MOVECH    LDCH        STR1, X
4           STCH         STR2,X
5           TIX          FOUR
6           JLT          MOVECH
7  STR1      BYTE        C'HOLA'
8  STR2      RESB        4
9  ZERO      WORD        0
10 FOUR      WORD        4
10  END
```

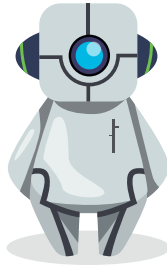
- Está orientado al **hardware**.
- Nos ayuda a entender **cómo funcionan las instrucciones** en la computadora.
- Puedes construir **sistemas operativos y núcleos**.



ESTRUCTURA EN CAPAS DE LOS LENGUAJES DE PROGRAMACIÓN

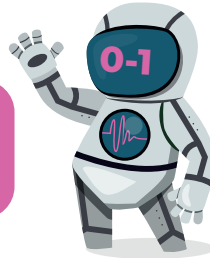


LENGUAJE DE ALTO NIVEL
(JAVA, C#, PHP, PYTHON, ETC.)

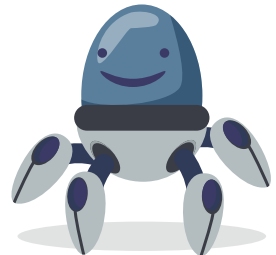


LENGUAJE DE BAJO NIVEL
ENSAMBLADOR

CÓDIGO MÁQUINA
(BINARIO [0-1])



HARDWARE
(PARTE FÍSICA DE LA COMPUTADORA)



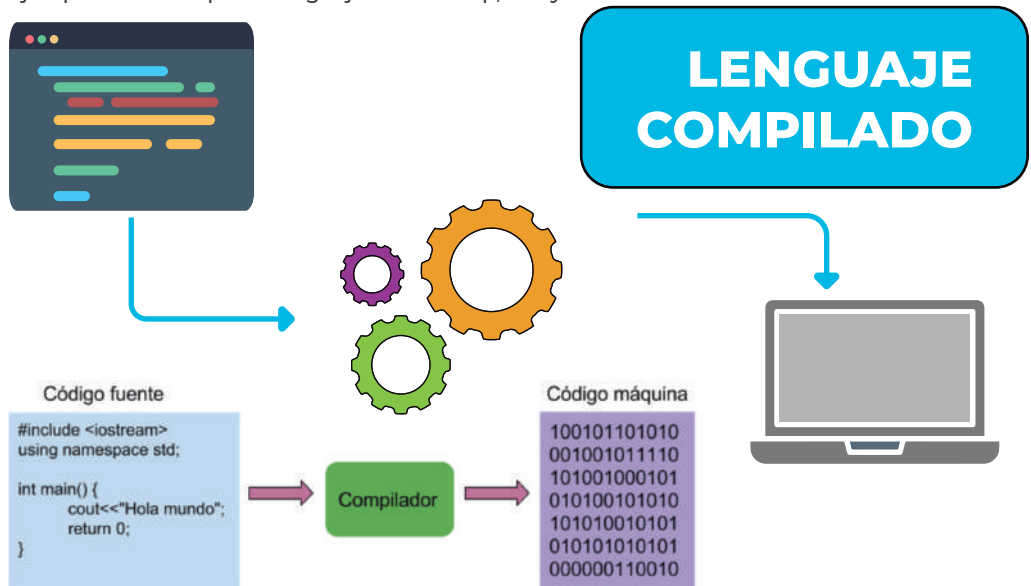
SESIÓN 03

TIPOS DE LENGUAJES DE PROGRAMACIÓN

LENGUAJE COMPILADO

En este tipo de lenguaje el código se compila, ¿para qué? para crear un paquete de código máquina (código binario), así el computador puede ejecutar las instrucciones, ¿sabes por qué?, ¡porque ahora el código se encuentra en su idioma!

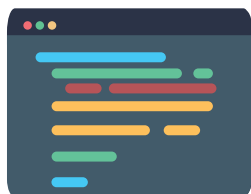
Ejemplos de este tipo de lenguaje son C Sharp, C++ y Go.




LENGUAJE INTERPRETADO

Este lenguaje ¡ya no cuenta con un compilador! El código va directo a la máquina quien ahora tiene un intérprete, que traduce el código y lo convierte a su lenguaje, entonces ¿Un compilador es lo mismo que un intérprete?, bueno, digamos que tienen la misma funcionalidad (traducir), pero su diferencia radica en que el intérprete lo realiza al momento de ejecución (cuando lo solicitas) y al ser en tiempo real puede alentar el proceso.

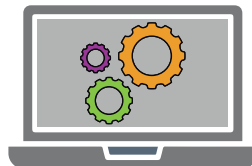
Lenguajes de este tipo son JavaScript, PHP, Python y Ruby.



LENGUAJE INTERPRETADO



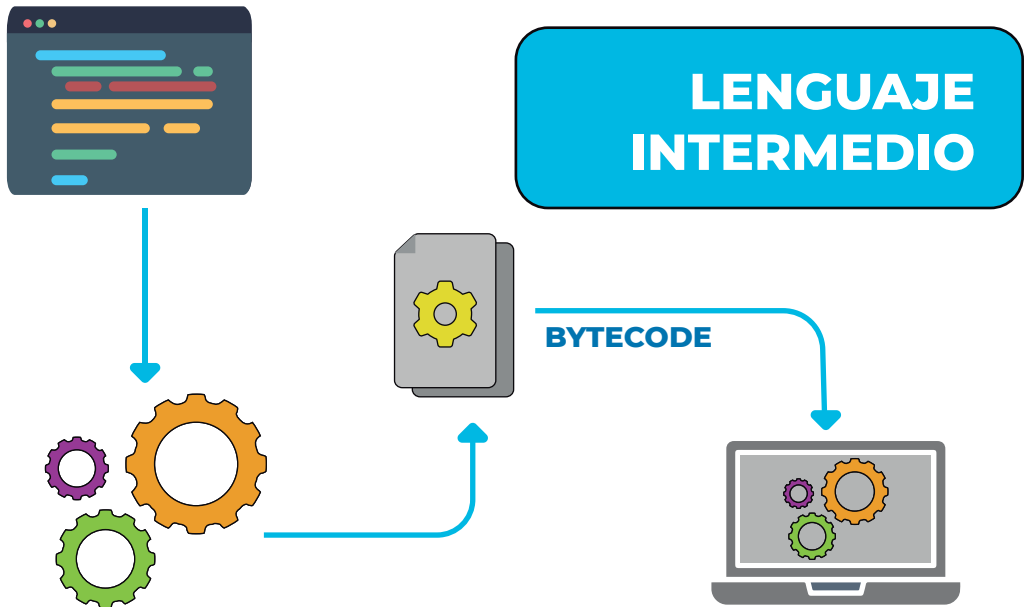
```
public class Metereologia {  
  
    /**  
     * Método principal de la clase  
     * @param args el argumento del método  
     */  
    public static void main(String[] args) {  
  
        //declarar las variables  
        int cantidad,tempNegativas;  
        double valorTemperatura,sumaTemperatura;  
        double mayorTemperatura,menorTemperatura;  
    }  
}
```



LENGUAJE INTERMEDIO

A diferencia de los otros lenguajes este cuenta con un paso intermedio, pues después de escribir el código y compilarlo, obtiene un “Bytecode” (otro lenguaje intermedio que también debe ser interpretado), pero, ¿para qué te sirve esto?, bueno, su funcionalidad es sorprendente, ¡porque te será posible ejecutar el código en cualquier sistema operativo!, necesitando solamente el intérprete de Bytecode (Java Virtual Machine).

Dentro de estos lenguajes se encuentran Java, Kotlin y Scala.





¡REPASEMOS!

TIPOS DE LENGUAJE

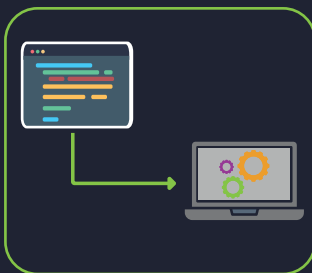
COMPILADO

Convierte el código a binarios que lee el sistema operativo.



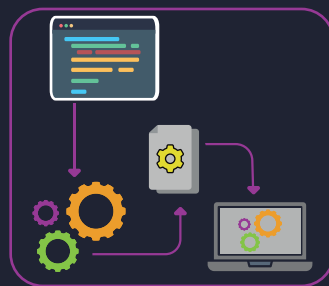
INTERPRETADO

Requiere de un programa que lea la instrucción del código en tiempo real, y la ejecute.



INTERMEDIO

Se compila el código fuente a un lenguaje intermedio y este último se ejecuta en una máquina virtual.



LENGUAJE DE PROGRAMACIÓN SEGÚN SU COMPILACIÓN

La compilación es el proceso de traducir el código de un lenguaje de alto nivel al **lenguaje máquina** que leen las computadoras.

1100101
000010
10110101



COMPILADOS

Se convierte en código binario a través de un compilador.

1

Escribes el código.

2

Pasa por un **proceso de compilación** que genera un **archivo ejecutable**.

3

Ese ejecutable es leído por la computadora.



INTERPRETADOS

Requieren de un programa que lee la instrucción en **tiempo real** y la ejecuta

1

Escribes el código.

2

En la computadora, el programa **interpreta** el código y lo **traduce a lenguaje máquina** en tiempo real



TIPOS DE LENGUAJE SEGÚN SU PROPÓSITO

GENERAL PURPOSE PROGRAMMING LANGUAGE (GLP)

Son lenguajes que pueden usarse para solucionar **cualquier tipo de problema**. Son a los que normalmente llamamos **lenguajes de programación**.



DOMAIN SPECIFIC LANGUAGE (DSL)

Son lenguajes con un **alto nivel de abstracción** creados para solucionar un **tipo muy específico** de problema.



Hay un gran debate sobre si HTML, CSS o SQL son lenguajes de programación o no.
¿Tú que opinas?



SESIÓN 04

¿QUÉ ES UN ALGORITMO?

Como algoritmos denominamos un conjunto ordenado y finito de operaciones simples a través del cual podemos hallar la solución a un problema.

Los algoritmos nos permiten ejecutar una acción o resolver un problema mediante una serie de secuencias, lógicas ordenadas y finitas. Así, dado un estado inicial y una entrada, y siguiendo los sucesivos pasos indicados, se llega al estado final y se obtiene una solución.

¿SABÍAS QUÉ?

Los primeros algoritmos registrados datan de Babilonia, originados en las matemáticas como un método para resolver un problema usando una secuencia de cálculos más simples.

El primer algoritmo famoso es el cálculo del MCD de dos números (Grecia, aproximadamente del s. IV a. C.).



CONOCIENDO A ALAN TURING

Alan Mathison Turing, fue un matemático, lógico, informático teórico, criptógrafo, filósofo, biólogo teórico, maratoniano y corredor de ultradistancia británico.

Turing fue el padre teórico del ordenador y el precursor de la inteligencia artificial. ... En 1945, al finalizar la Segunda Guerra Mundial, Alan Turing recibió la Orden del Imperio Británico. Había liderado con éxito una misión para descifrar mensajes nazis codificados.



CUANDO SE ELABORA UN ALGORITMO SE DEBE TENER EN CUENTA LO SIGUIENTE

- Tienen inicio y fin: todo algoritmo comienza en un estado inicial con una serie de datos específicos, y culmina con una solución o salida.
- Funcionan en secuencia: un algoritmo está compuesto por una serie de pasos ordenados.
- Las secuencias son concretas: cada paso es claro y no deja lugar a la ambigüedad.
- Los algoritmos son abstractos: son modelos o guías para ordenar procesos.
- Realizar pruebas al algoritmo para verificar los resultados



TIPOS DE ALGORITMOS

- Algoritmos computacionales. Un algoritmo cuya resolución depende del cálculo, y que puede ser desarrollado por una calculadora o computadora sin dificultades.
- Algoritmos no computacionales. Aquellos que no requieren de los procesos de un computador para resolverse, o cuyos pasos son exclusivos para la resolución por parte de un ser humano.
- Algoritmos cualitativos. Se trata de un algoritmo en cuya resolución no intervienen cálculos numéricos, sino secuencias lógicas y/o formales.
- Algoritmos cuantitativos. Todo lo contrario, es un algoritmo que depende de cálculos matemáticos para dar con su resolución.



EJEMPLOS DE AGORITMOS EN LA VIDA

RECETAS DE COMIDAS

Explican el paso a paso para crear una comida con una cantidad finita de ingredientes. El estado inicial serían los ingredientes sin procesar y el estado final la comida preparada.



MANUALES

Sirven de guía para ejecutar procesos, desde cómo armar una biblioteca hasta cómo activar un teléfono móvil. En estos casos, el estado final es el producto armado, instalado, encendido, en funcionamiento, etc.

OPERACIONES MATEMÁTICAS

En matemáticas, algunos ejemplos de algoritmos son la multiplicación, en donde seguimos una secuencia de operaciones para obtener un producto; o la división, que nos permite determinar el cociente de dos números. El algoritmo de Euclides, con el cual sacamos el máximo común divisor de dos enteros positivos es otro ejemplo de algoritmo.



¡REPASEMOS!

¿QUÉ ES UN ALGORITMO?

Es la **secuencia de pasos** que resuelve un problema y es la base de la programación.

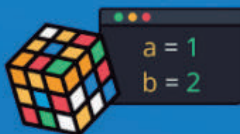
Prof. Alexys Lozada



PARTES DE UN ALGORITMO

ENTRADA

Son los datos que se le dan al algoritmo.



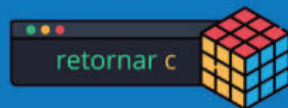
PROCESO

Operaciones que se hacen con los datos.



SALIDA

Resultado final que se obtiene de las operaciones, en este caso será 3.



CARACTERÍSTICAS

PRECISO

Tiene que resolver el problema sin errores.



DEFINIDO

Si ejecutas el algoritmo varias veces, los datos de salida serán iguales en cada repetición.



FINITO

Debe tener un inicio y un final.



LEGIBLE

Cualquier persona que vea el algoritmo debe ser capaz de comprenderlo.



EJEMPLO 1

Elabore un algoritmo que permita ir de la casa al colegio.



OBJETIVO

IR DE LA CASA AL COLEGIO

INICIO

Salir de casa

- 1) Si está lejos del colegio entonces tomar un medio de transporte que lo deje cerca del mismo.
- 2) Si no está lejos del colegio entonces dirigirse caminando hacia él mismo.
- 3) Llegar a la puerta del colegio.

FIN

EJEMPLO 2

Plantee un algoritmo que permita adquirir un boleto para ir al cine.



OBJETIVO

ADQUIRIR UN BOLETO PARA IR AL CINE

INICIO

Dirigirse hacia el teatro donde quiere ver la película

- 1) Si hay gente esperando el boleto entonces hacer la fila y avanzar con la misma hasta llegar a la taquilla
- 2) Si no hay gente esperando comprar el boleto entonces dirigirse a la taquilla
- 3) Comprar el boleto para ver la película.

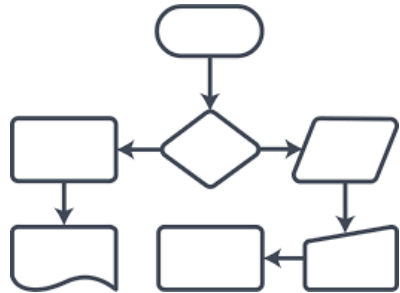
FIN

SESIÓN 05

¿QUÉ ES DIAGRAMA DE FLUJO?

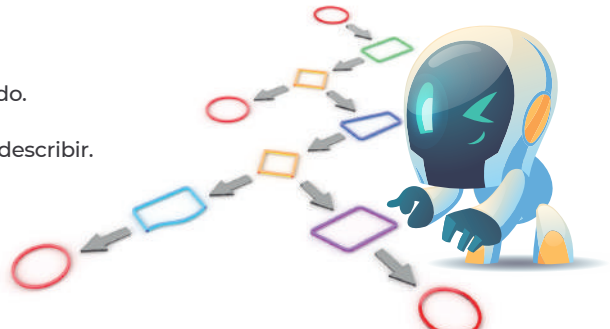
El diagrama de flujo o diagrama de actividades es la representación gráfica del algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.

En Lenguaje Unificado de Modelado (UML), un diagrama de actividades representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un diagrama de actividades muestra el flujo de control general.



CARACTERÍSTICAS









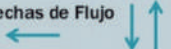
- Identificar las ideas principales a ser incluidas en el diagrama de flujo.
- Identificar quién lo emplea y cómo.
- Establecer el nivel de detalle requerido.
- Determinar los límites del proceso a describir.



PASOS PARA CONSTRUIR UN DIAGRAMA DE FLUJO

- Establecer el alcance del proceso a describir. De esta manera quedará fijado el comienzo y el final del diagrama.
- Identificar y listar las principales actividades/subprocesos que están incluidos en el proceso a describir y su orden cronológico.
- Si el nivel de detalle definido incluye actividades menores, listarlas también. Identificar y listar los puntos de decisión.
- Construir el diagrama respetando la secuencia cronológica y asignando los correspondientes símbolos.
- Asignar un título al diagrama y verificar que esté completo y describa con exactitud el proceso elegido.

DIAGRAMAS

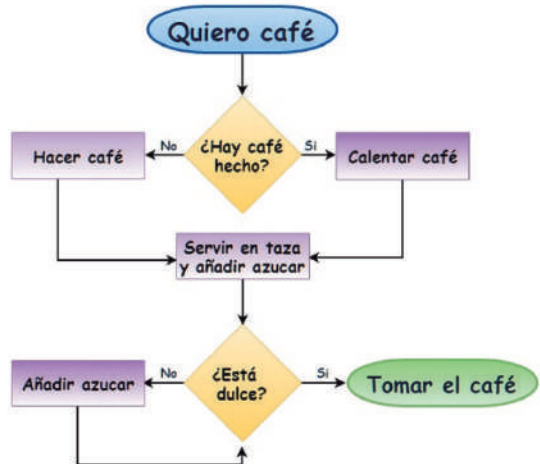
| símbolo | Función | Símbolo | Función |
|--|---|---|---|
| Terminal  | Indicar el inicio y fin del diagrama | Teclado  | Introducir datos manualmente por el teclado |
| Entrada/salida  | Entrada o salida simple de información | Decisión  | Indica operaciones lógicas o de comparación y tienen dos salidas dependiendo del resultado. |
| Proceso  | Realizar cualquier operación o cálculo con la información | | |
| Salida a Impresora  | Salida de información a la impresora | Conectores  | Une dos partes del diagrama a la misma o diferente página |
| Salida a Pantalla  | Mostrar información de salida a la pantalla | Flechas de Flujo  | Indica la dirección del flujo de la información |

EJEMPLOS DE DIAGRAMA DE FLUJO

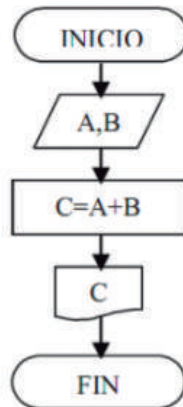


**¿COMO APRENDER
DIAGRAMAS DE FLUJO?**

QUIERO CAFÉ



24.-Desarrolle un algoritmo que le permita leer dos valores y escribir la suma de los dos.



25.-Desarrolle un algoritmo que le permita leer un valor entero, calcular su cuadrado y escribir dicho resultado.

