



Clase 1

Agenda

- Algoritmos
- Diagramas de Flujo
- Pseudocódigo
- Introducción a la Programación
- Dudas



Introducción a Algoritmos



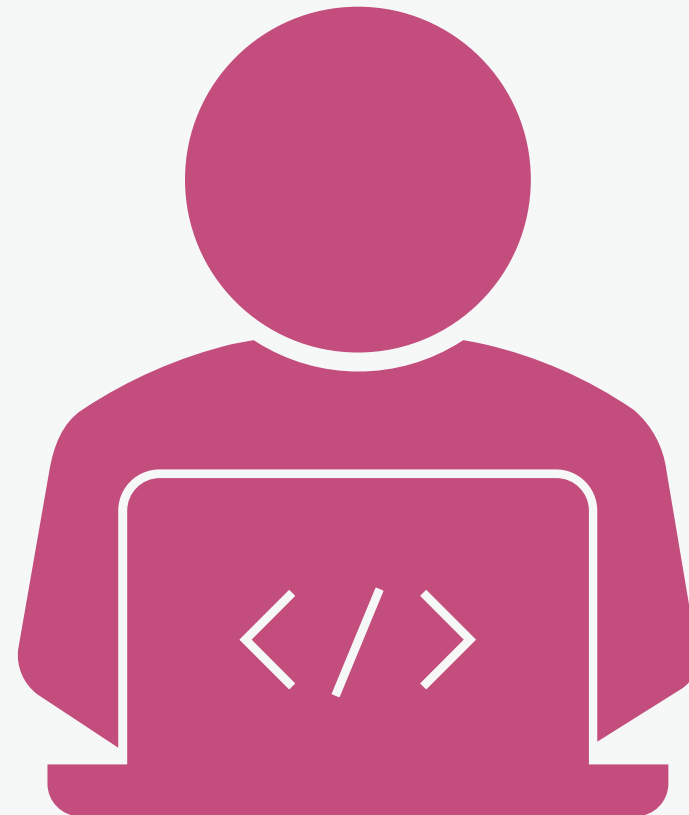
¿Qué son?

Tipos de Algoritmos

Clasificación

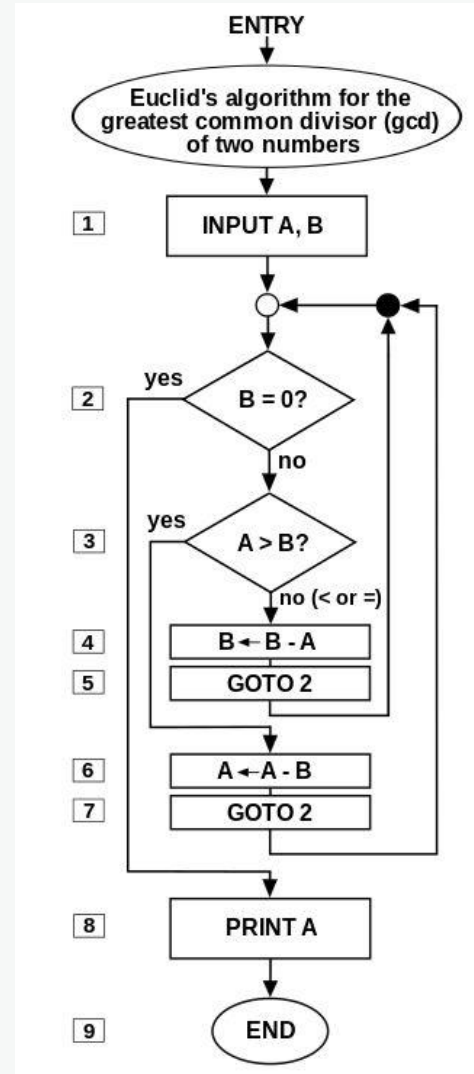
Introducción a Algoritmos

Un algoritmo es una secuencia finita de pasos precisos, bien definidos y ordenados, que permiten resolver un problema específico o realizar una tarea determinada. Estos pasos deben ser lo suficientemente claros y comprensibles para ser ejecutados por una persona o una máquina.

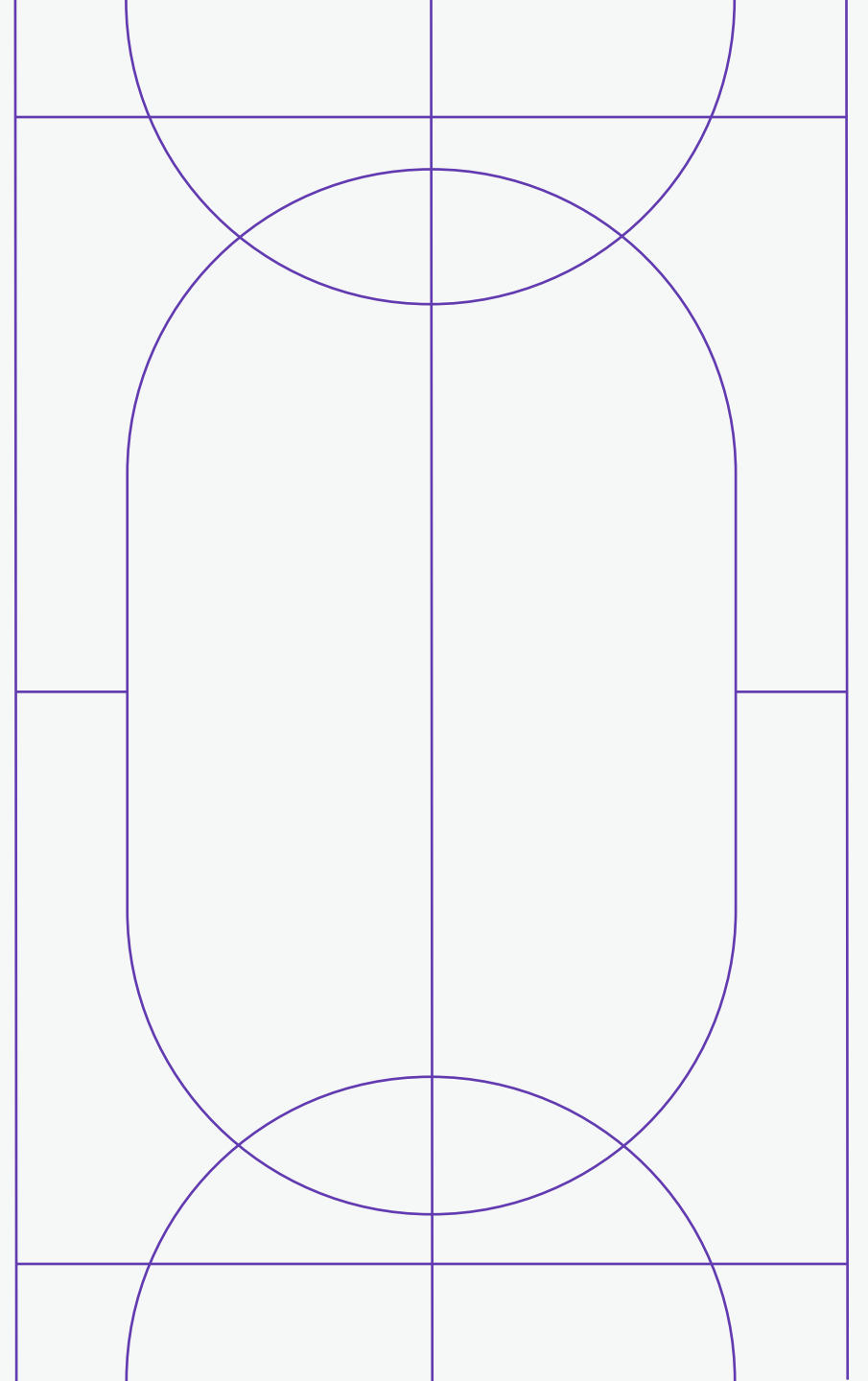


Clasificación de Algoritmos

- Por Implementación
- Por Paradigma de Diseño
- Problemas de Optimización
- Por Campo de Estudio
- Por Complejidad



Clasificación de Algoritmos



Por
Implementación

```
graph TD; A[Por Implementación] --> B[Recursivo]; A --> C[Lógicos]; A --> D[Seriales, Paralelos, Distribuidos]; A --> E[Determinísticos y no Determinísticos]; A --> F[Exactos o Aproximados]; A --> G[Algoritmos Cuánticos]; B --> H[Se emplea a sí mismo]; C --> I[Salida "V", "F"];
```

This diagram classifies algorithms based on their implementation. The root node is 'Por Implementación', which branches into six categories: 'Recursivo', 'Lógicos', 'Seriales, Paralelos, Distribuidos', 'Determinísticos y no Determinísticos', 'Exactos o Aproximados', and 'Algoritmos Cuánticos'. The 'Recursivo' category further branches into 'Se emplea a sí mismo', and the 'Lógicos' category branches into 'Salida "V", "F"'. Each node is represented by a rounded rectangle with a purple border and a light purple fill, with a darker purple shadow box behind it.

Recursivo

Lógicos

Seriales,
Paralelos,
Distribuidos

Determinísticos y
no
Determinísticos

Exactos o
Aproximados

Algoritmos
Cuánticos

Se emplea a sí
mismo

Salida
"V", "F"

Por Paradigma de Diseño



```
graph TD; A[Por Paradigma de Diseño] --> B[Fuerza Bruta o Exhaustiva]; A --> C[Algoritmos "Divide y Vencerás"]; A --> D[Búsqueda y Enumeración]; A --> E[Algoritmos de Randomización]; A --> F[Reducción de Complejidad]; B --> B1[Trabajan hasta encontrar el error]; C --> C1[Dividen el problema para encontrar 1 solución]; E --> E1[Dan salidas aleatorias en cada iteración]; F --> F1[Optimización];
```

The diagram is a hierarchical flowchart. At the top is a box labeled 'Por Paradigma de Diseño'. A horizontal line below it branches into five boxes: 'Fuerza Bruta o Exhaustiva', 'Algoritmos "Divide y Vencerás"', 'Búsqueda y Enumeración', 'Algoritmos de Randomización', and 'Reducción de Complejidad'. From 'Fuerza Bruta o Exhaustiva', a vertical line leads down to 'Trabajan hasta encontrar el error'. From 'Algoritmos "Divide y Vencerás"', a vertical line leads down to 'Dividen el problema para encontrar 1 solución'. From 'Algoritmos de Randomización', a vertical line leads down to 'Dan salidas aleatorias en cada iteración'. From 'Reducción de Complejidad', a vertical line leads down to 'Optimización'. The box 'Búsqueda y Enumeración' has no further connections shown.

Fuerza Bruta o Exhaustiva

Trabajan hasta encontrar el error

Algoritmos "Divide y Vencerás"

Dividen el problema para encontrar 1 solución

Búsqueda y Enumeración

Algoritmos de Randomización

Dan salidas aleatorias en cada iteración

Reducción de Complejidad

Optimización

Por
Problemas de
Optimización

```
graph TD; A[Por Problemas de Optimización] --- B[Programación Lineal]; A --- C[Programación Dinámica]; A --- D[Vuelta Atrás]; A --- E[Algoritmo Voraz]; A --- F[Métodos Heurísticos];
```

The diagram is a hierarchical tree structure. At the top is a box labeled 'Por Problemas de Optimización'. A horizontal line below it connects to five boxes: 'Programación Lineal', 'Programación Dinámica', 'Vuelta Atrás', 'Algoritmo Voraz', and 'Métodos Heurísticos'. Each box has a dark purple shadow and a light purple fill. The entire diagram is set against a light gray background with a purple grid and decorative arcs at the top and bottom.

Programación
Lineal

Programación
Dinámica

Vuelta Atrás

Algoritmo
Voraz

Métodos
Heurísticos

Por Campos de Estudio

Búsqueda

Ordenamiento

Merge

Numéricos

Gráficos

De Cadenas

Geometría Computacional

Combinatoria

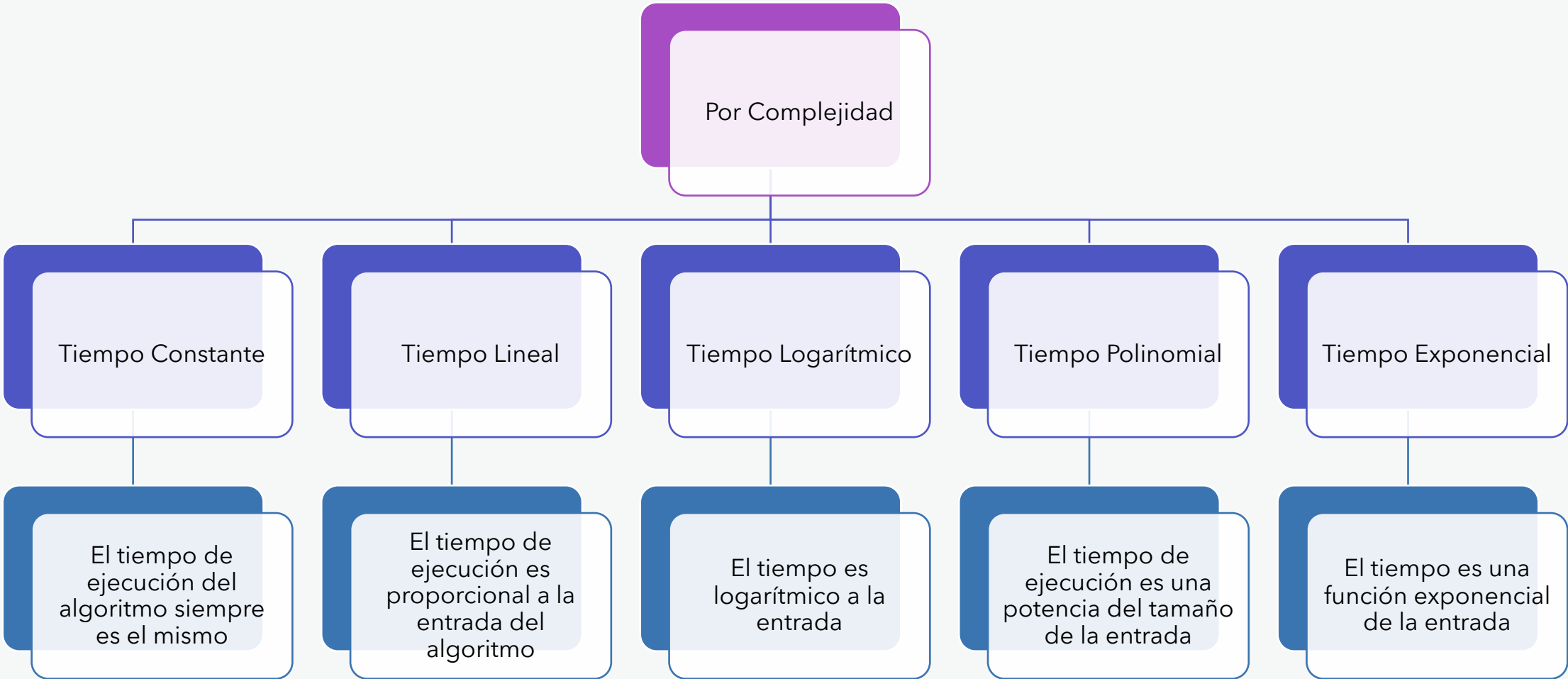
Criptografía

Comprensión/Análisis de Datos

Machine Learning

Parsing

Por Complejidad



```
graph TD; A[Por Complejidad] --> B[Tiempo Constante]; A --> C[Tiempo Lineal]; A --> D[Tiempo Logarítmico]; A --> E[Tiempo Polinomial]; A --> F[Tiempo Exponencial]; B --> B1[El tiempo de ejecución del algoritmo siempre es el mismo]; C --> C1[El tiempo de ejecución es proporcional a la entrada del algoritmo]; D --> D1[El tiempo es logarítmico a la entrada]; E --> E1[El tiempo de ejecución es una potencia del tamaño de la entrada]; F --> F1[El tiempo es una función exponencial de la entrada];
```

The diagram is a hierarchical flowchart. At the top is a light purple box with a dark purple shadow containing the text 'Por Complejidad'. A horizontal line below it branches into five vertical lines, each leading to a light blue box with a dark blue shadow. These boxes are labeled 'Tiempo Constante', 'Tiempo Lineal', 'Tiempo Logarítmico', 'Tiempo Polinomial', and 'Tiempo Exponencial' from left to right. Below each of these boxes is another light blue box with a dark blue shadow, connected by a vertical line. These bottom boxes contain descriptive text: 'El tiempo de ejecución del algoritmo siempre es el mismo', 'El tiempo de ejecución es proporcional a la entrada del algoritmo', 'El tiempo es logarítmico a la entrada', 'El tiempo de ejecución es una potencia del tamaño de la entrada', and 'El tiempo es una función exponencial de la entrada'.

Tiempo Constante

El tiempo de ejecución del algoritmo siempre es el mismo

Tiempo Lineal

El tiempo de ejecución es proporcional a la entrada del algoritmo

Tiempo Logarítmico

El tiempo es logarítmico a la entrada

Tiempo Polinomial

El tiempo de ejecución es una potencia del tamaño de la entrada

Tiempo Exponencial

El tiempo es una función exponencial de la entrada

Diseño de Algoritmos

Definir el Problema

Desarrollo de un Modelo

Especificación del Algoritmo

Designación del Algoritmo

Pruebas y Correcciones del Algoritmo

Análisis del Algoritmo

Implementación del Algoritmo

Pruebas en Programa

Documentación

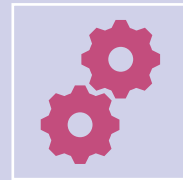
Diagramas de Flujo



¿Qué son?



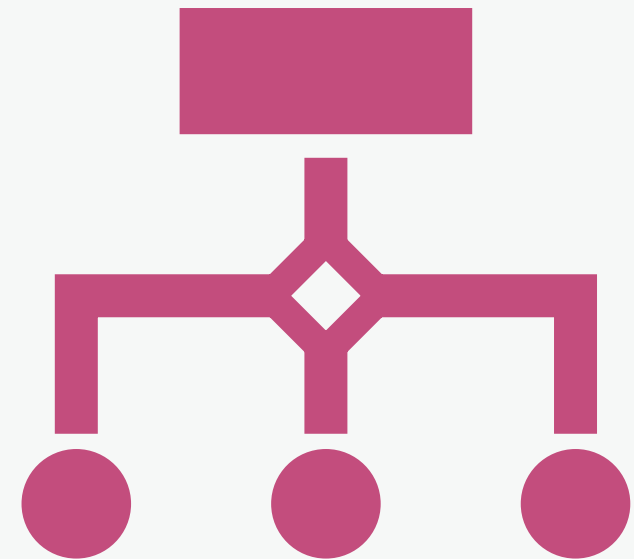
Nomenclatura



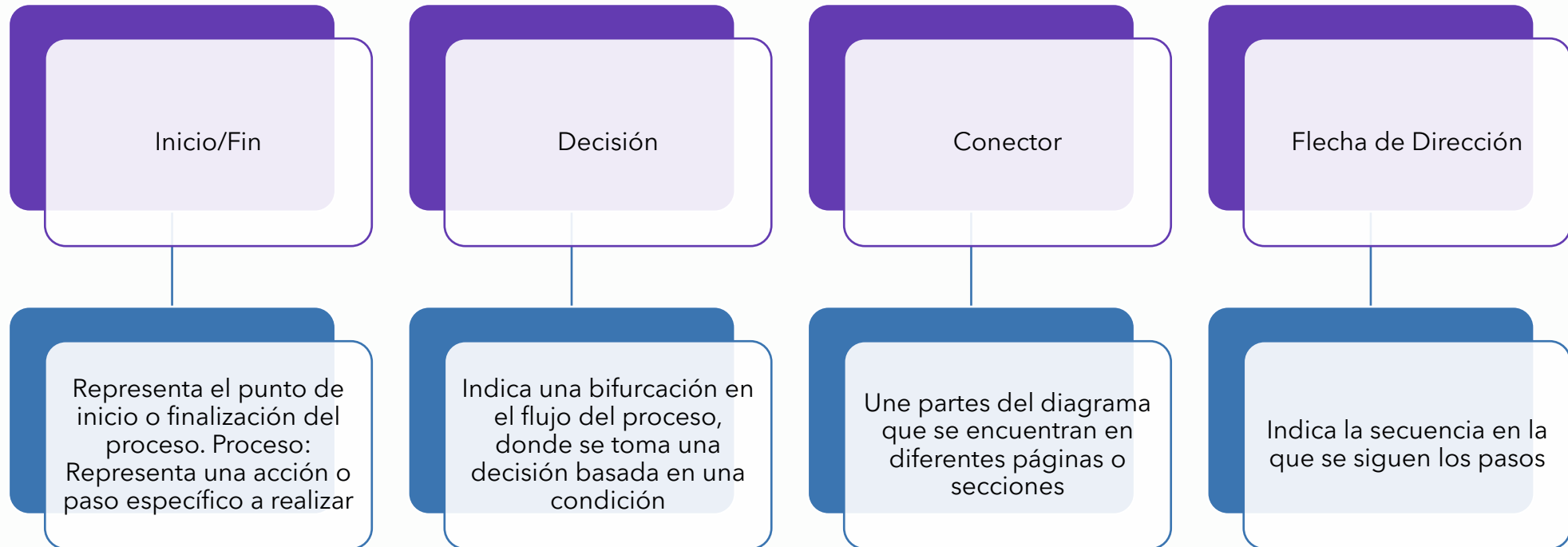
Proceso y Diseño de Algoritmos






Diagramas de Flujo

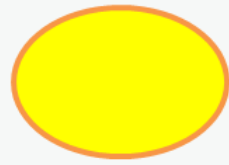
Un diagrama de flujo es una representación gráfica de un algoritmo, proceso o sistema que utiliza símbolos estandarizados para ilustrar la secuencia de pasos y decisiones que se deben seguir para completar una tarea específica



Símbolos Más Utilizados



Símbolo		Función
Líneas de flujo		Conectan los pasos, etapas, decisiones y otros elementos que intervienen en los diagramas
Decisión		Se usan para indicar las elecciones y decisiones realizadas.
Datos		Ofrecen información nueva, de interés o de gran valor para el desarrollo del proceso respresentado.
Actividad		Indican las acciones que se transforman en datos que dan continuidad al proceso.
Inicio / final		Se utiliza cada vez que se indica el problema/ solución en el diagrama de flujo macando el inicio y cierre de mismo.



Inicio y Fin



Entrada-Salida tipo de frente



Operación psíquica y repetitiva



Expresión de asignación algebraica



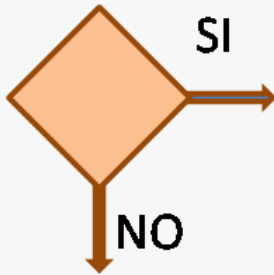
Entrada: Lectura datos por tarjetas perforadas



Conector dentro de pagina



Proceso de llamada de una sub alterna



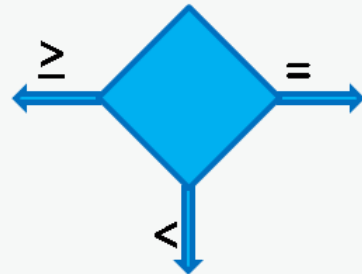
Condiciones de asignatura alternada



Conector dentro de pagina



Representación de datos grabados en la cuarta cinta magnética



Condición de acciones alternativas decisión numérica



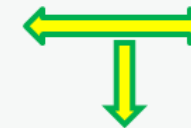
Representación de resultados
Reporte Impreso



Conector fuera y dentro de pagina



Almacenamiento en línea de disco magnético

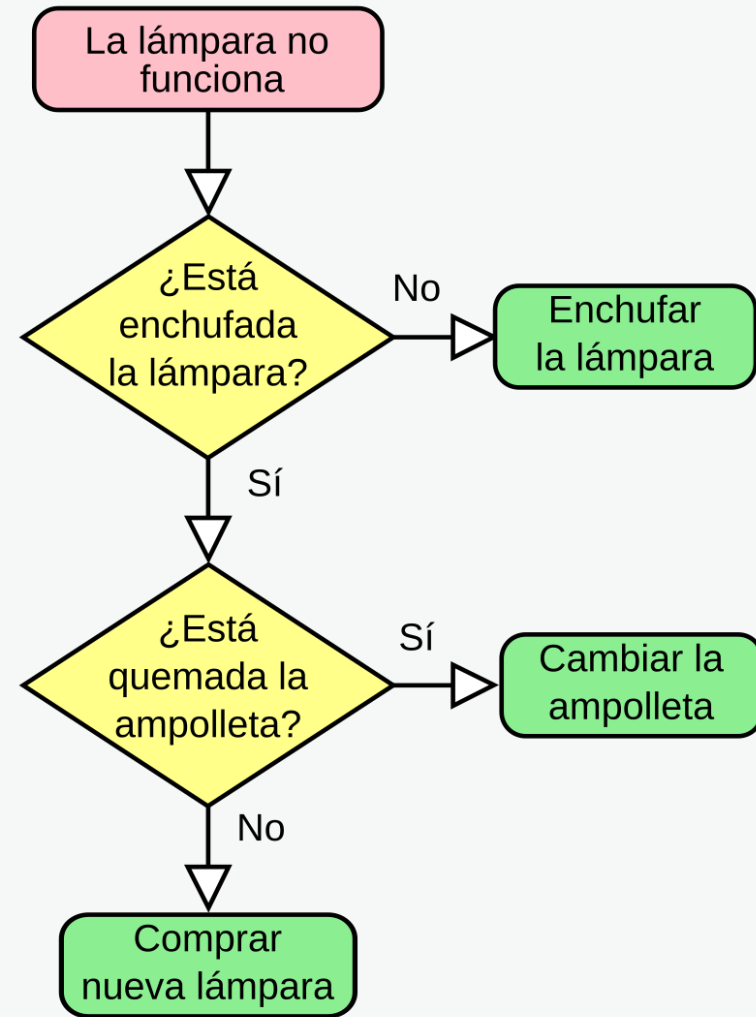


Dirección de un diagrama de flujo

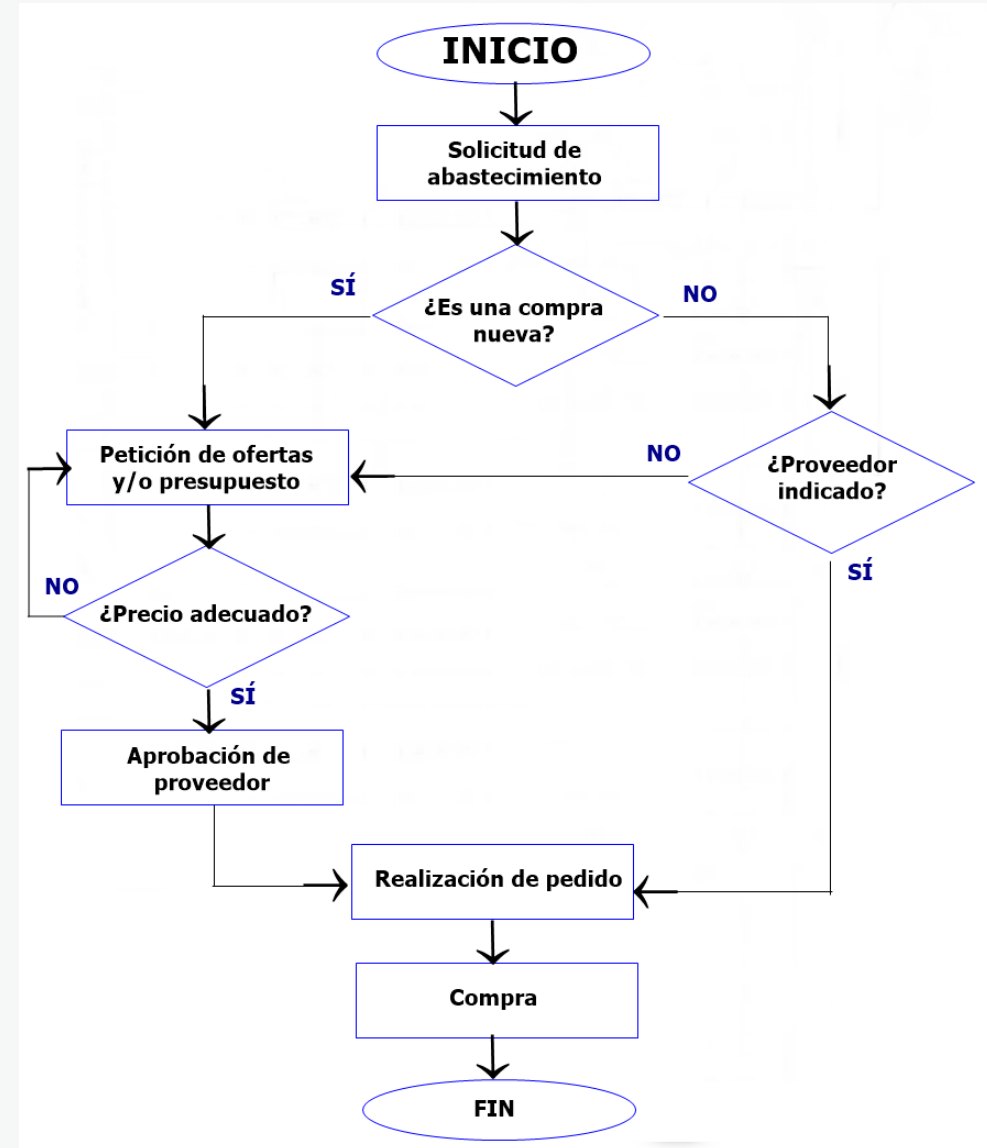
Ejemplos



Funcionamiento de la Lámpara



Proceso de Compra a Proveedores



Pseudocódigo

¿Qué es?

Importancia

Estructuras básicas



¿Qué es el Pseudocódigo?

El pseudocódigo es una descripción informal de un algoritmo o programa, el cuál se puede escribir por medio del lenguaje natural estructurado que puede imitar la lógica del código pero sin seguir las reglas de la sintaxis del lenguaje de programación; teniendo como propósito el mostrar el flujo lógico de un programa o sistema de manera que cualquier persona que tenga o no experiencia en programación pueda comprender y entender los pasos que realiza el algoritmo.

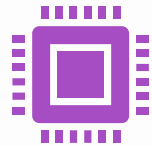
En contraste con el código, el pseudocódigo no se ejecuta en el ordenador, sino que actúa como puente entre la idea y el código final; siendo una forma de planificar el proceso antes de lidiar con detalles técnicos y sintácticos de un lenguaje de programación en concreto.

Importancia del uso del Pseudocódigo



Claridad y simplicidad

Es clave para la descripción lógica de un programa de manera clara, permitiendo expresar ideas sin la necesidad de preocuparse de la sintaxis de un lenguaje específico.



Planificación pre codificación

Facilita la planificación del código, donde los desarrolladores definen los pasos del algoritmo, antes de escribir código real, ayudando a evitar errores de diseño, asegurando que la lógica este lo más clara posible.



Comunicación con el equipo

Permite comunicar la lógica de un programa de manera sencilla a otros miembros del equipo, independientemente del lenguaje que utilicen, resultando útil en proyectos colaborativos.



Optimización

Permite refinar la lógica de un programa antes de su implementación, ayudando en la detección de ineficiencias y mejorar la estructura del algoritmo, facilitando que el programa final sea más eficiente y mantenible.

Estructuras básicas

Lenguaje Simple

Al caracterizarse por utilizar un lenguaje simple y fácil de comprender, se pueden emplear palabras comunes como "Si", "Sino", "Mientras", "Repetir", para describir acciones y decisiones. Permitiendo centrarse en los pasos lógicos en lugar de los detalles de la implementación técnica.

Describir pasos y decisiones

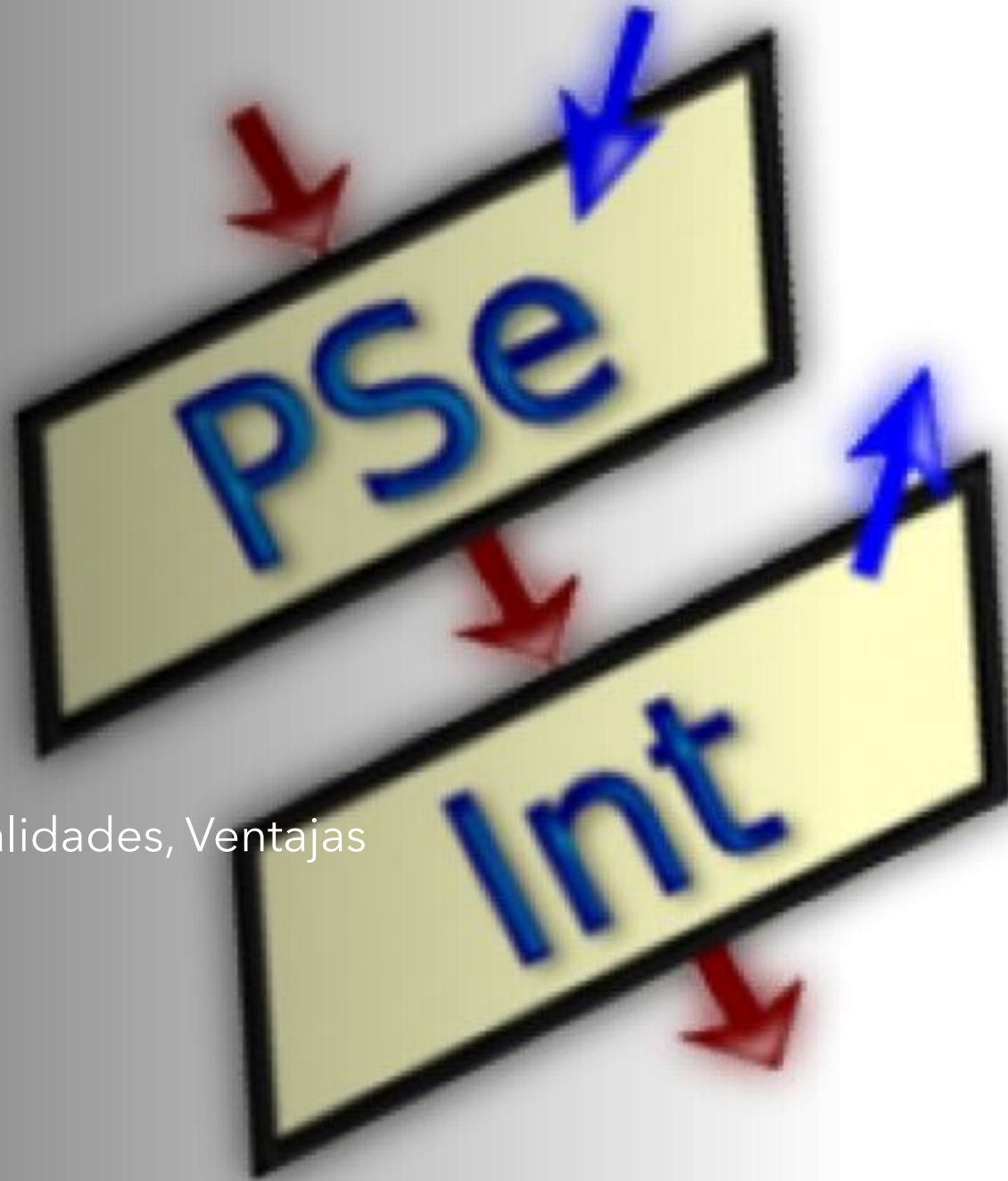
Se utiliza para describir los pasos que debe seguir un programa o algoritmo, como las decisiones lógicas que se deben tomar en función de ciertas condiciones, representando una acción que el programa debe realizar, permitiendo definir las estructuras condicionales, o bucles la definición de la lógica del flujo del programa de manera clara y ordenada.

Uso de estructuras de control básicas

Se pueden utilizar estructuras de control básicas para representar la lógica de un programa, como las condicionales (si, sino) y bucles (mientras, para), resultando fundamentales para el modelado de cualquier algoritmo.

PseInt

¿Qué es?, Funcionalidades, Ventajas



¿Qué es?

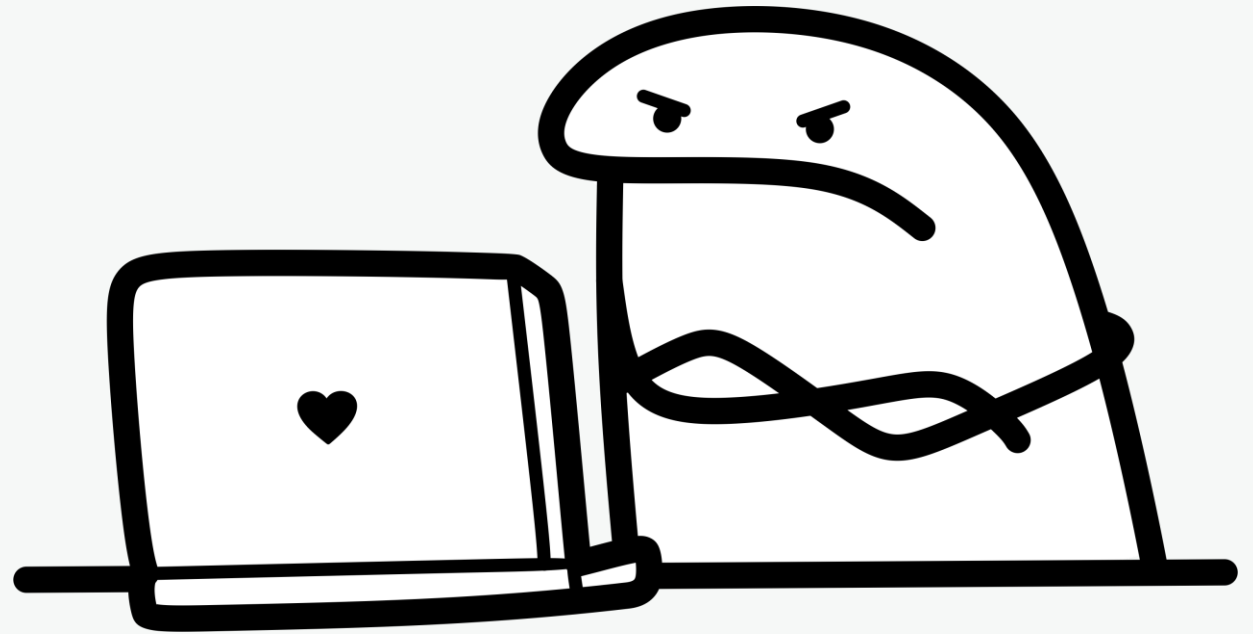
PseInt es un entorno de desarrollo gratuito que permite escribir, ejecutar y depurar pseudocódigo de manera sencilla.

Está pensado para asistir a las personas que inician en la construcción de programas o algoritmos computacionales.



Funcionalidades

- Presentación de herramientas de edición para escritura de algoritmos en pseudocódigo.
- Generación y edición de diagramas de flujo del algoritmo.
- Edición simultánea de múltiples algoritmos.
- Ejecución de los algoritmos.
- Multiplataforma.
- Software libre y gratuito.

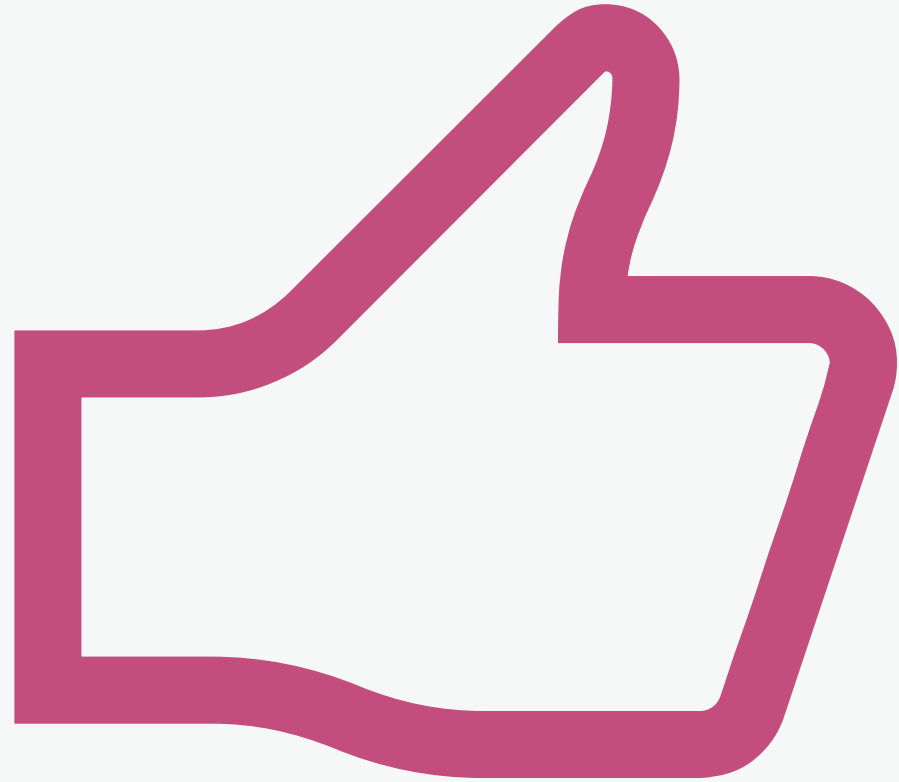


Ventajas

- Facilita el aprendizaje y la práctica de la programación sin necesidad de dominar un lenguaje de programación específico.
- Conversión de algoritmos a código C++.
- Lenguaje de pseudocódigo configurable.
- Edición y generación de diagramas de flujo.



Ejemplos



Condicional

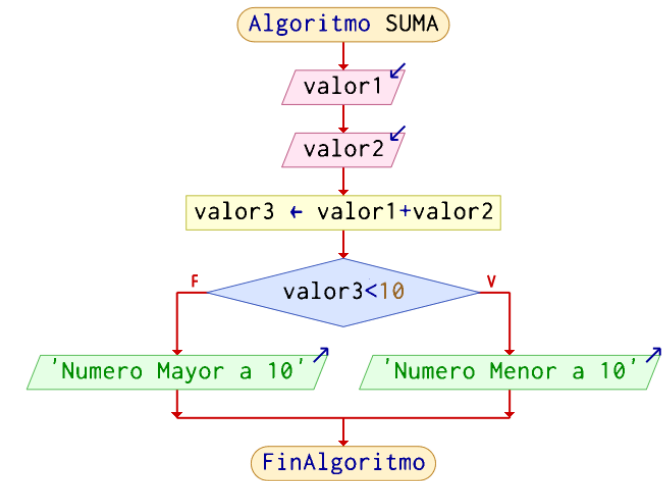
SI (condición)

Instrucción

SINO

Instrucción

FIN_SI



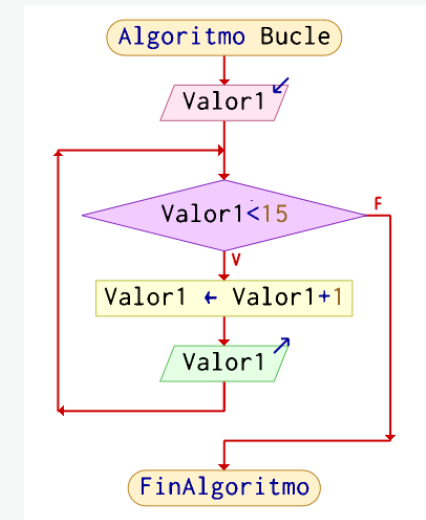
```
Algoritmo SUMA
  Leer valor1
  Leer valor2
  valor3 = valor1 + valor2
  Si valor3 < 10 Entonces
  ..... Escribir "Numero Menor a 10"
  SiNo
  ..... Escribir "Numero Mayor a 10"
  Fin Si
FinAlgoritmo
```

Bucle

MIENTRAS (condición)

Instrucción

FIN_MIENTRAS



Algoritmo Bucle

Leer Valor1

Mientras Valor1 < 15 **Hacer**

..... Valor1 = Valor1 + 1

..... **Escribir** Valor1

FinMientras

FinAlgoritmo

Introducción a la Programación



¿Qué es Programación?

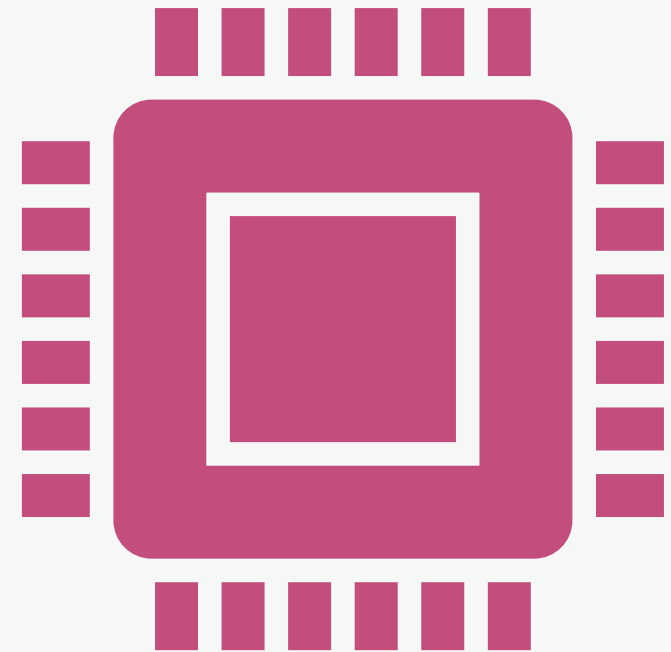
Computadoras

Software

Lenguajes de Programación

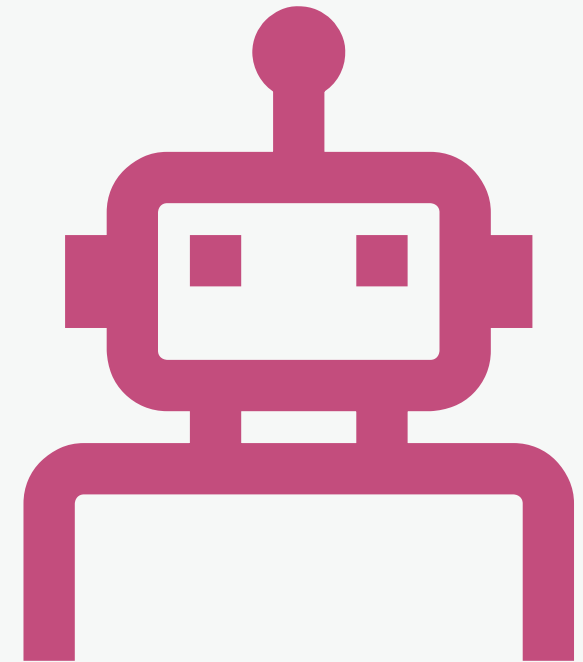
Computadoras

La computadora es una máquina digital que ejecuta comandos para convertirlos en datos convenientes y útiles que posteriormente se envían a las unidades de salida. Un computador está formado físicamente por numerosos circuitos integrados y muchos componentes de apoyo, extensión y accesorios, que en conjunto pueden ejecutar tareas diversas con suma rapidez y bajo el control de un programa (software).



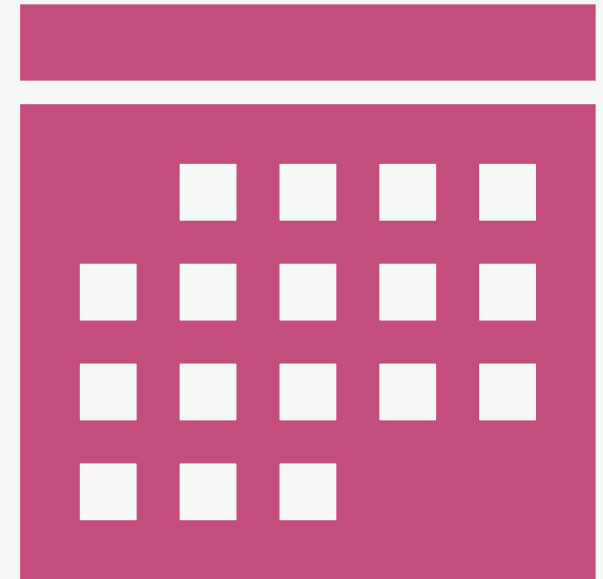
Software

Se conoce como software al soporte lógico de un sistema informático, que comprende el conjunto de los componentes lógicos necesarios que hacen posible la realización de tareas específicas, en contraposición a los componentes físicos que son llamados hardware. La interacción entre el software y el hardware hace operativo un ordenador (u otro dispositivo), es decir, el Software envía instrucciones que el Hardware ejecuta, haciendo posible su funcionamiento.



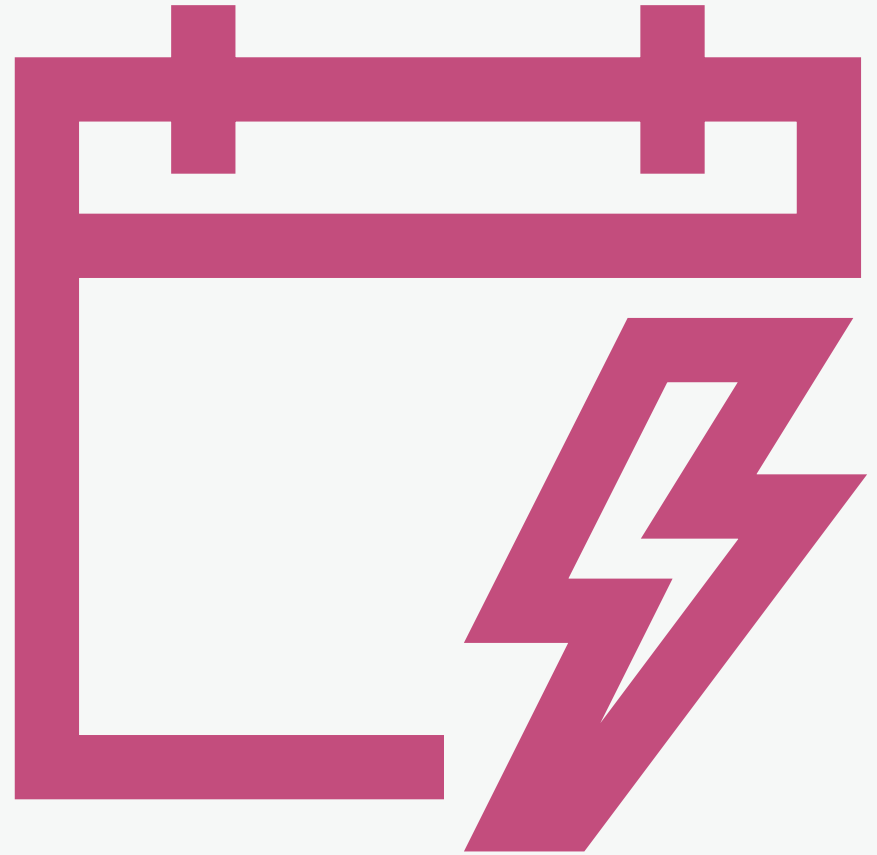
Proceso de Programación

La programación es el proceso utilizado para idear y ordenar las acciones necesarias para realizar un proyecto, preparar ciertas máquinas o aparatos para que empiecen a funcionar en el momento y en la forma deseados o elaborar programas para su empleo en computadoras.



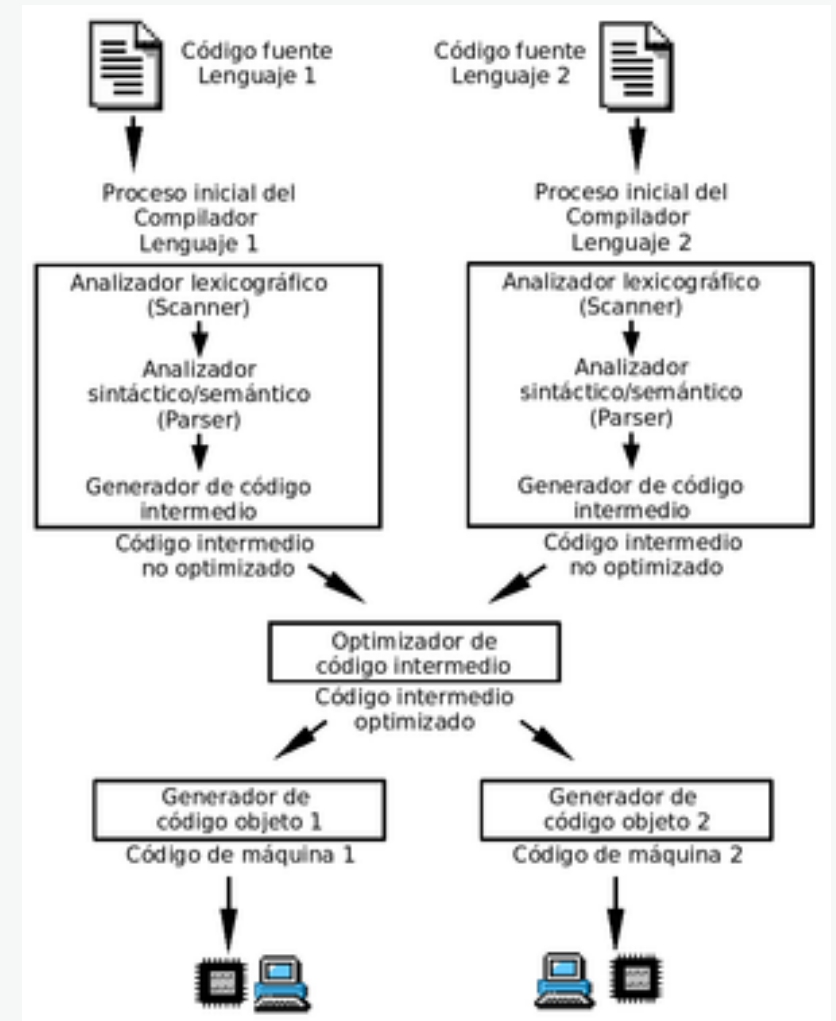
Lenguajes de Programación

Un lenguaje de programación de alto nivel se caracteriza por expresar los algoritmos de una manera adecuada a la capacidad cognitiva humana, en lugar de la capacidad con que los ejecutan las máquinas. Estos lenguajes permiten una máxima flexibilidad al programador a la hora de abstraerse o de ser literal.



Lenguajes de Programación

Permiten un camino bidireccional entre el lenguaje máquina y una expresión casi oral entre la escritura del programa y su posterior compilación. Por lo general suelen estar orientados a objetos, a eventos o a funciones, pudiendo estos combinarse. Asimismo, pueden ser compilados o interpretados. Algunos ejemplos son: Java, PHP, Python, Javascript, C++.



Elementos del Lenguaje

Tipos de Datos

Primitivos

Los datos primitivos son los tipos de datos básicos que están predefinidos en un lenguaje de programación y no pueden ser descompuestos en partes más pequeñas. Estos tipos de datos primitivos son fundamentales para realizar operaciones y cálculos en un programa.

No Primitivos

Los datos no primitivos, también conocidos como tipos de datos compuestos o estructuras de datos, son aquellos que pueden contener múltiples valores y combinan datos primitivos o incluso otros datos no primitivos. Estos tipos de datos no primitivos son construidos por el programador y ofrecen más flexibilidad para modelar estructuras de información complejas.

Dudas o Comentarios

