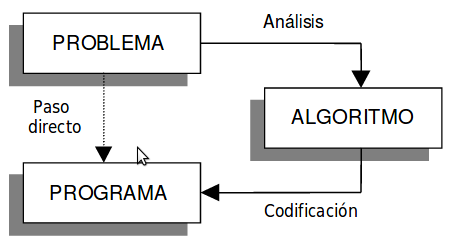
|  |  |
| --- | --- |
|  |  |
| ALGORITMOS |  |
|  | CURSO: SOFTWARE  ALUMNO: ARIEL CALDERÓN CUEVA DOCENTE: MSc. ENRIQUE BAÑO LEÓN |
|  |  |
|  |  |

# ¿Qué es un algoritmo?

Un algoritmo es un procedimiento por etapas. Es **un conjunto de reglas que hay que seguir para realizar una tarea** o resolver un problema.



Es **la capacidad de definir pasos claros** para resolver un problema.

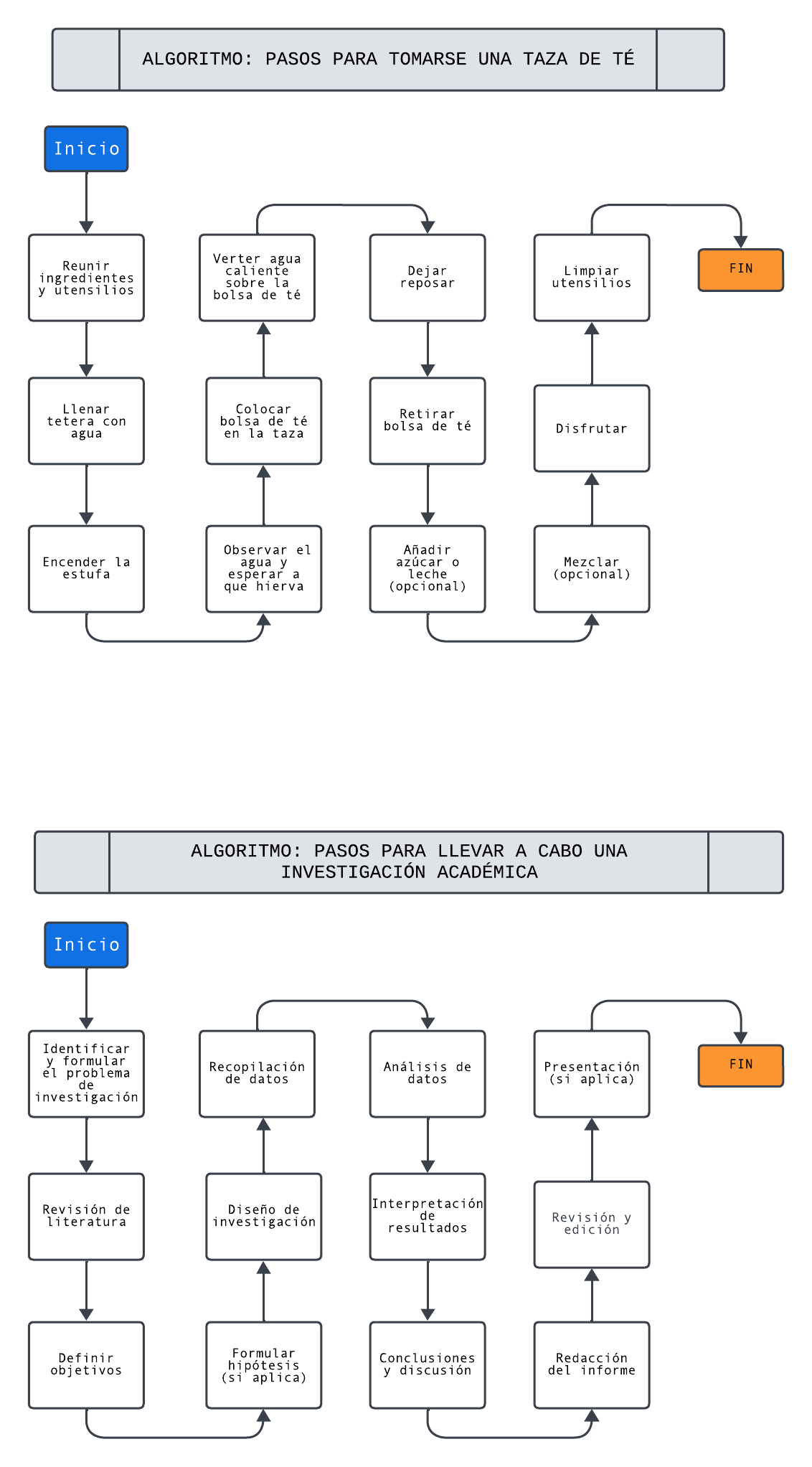
De hecho, utilizamos esta forma de pensar **a diario y a menudo sin darnos cuenta**. En la era de la **Data Science**, el **Machine Learning** y la **Inteligencia Artificial**, los algoritmos son más importantes que nunca y representan el combustible de la nueva revolución industrial.

En **el** **campo de la programación informática**, los algoritmos son conjuntos de reglas que indican al ordenador cómo ejecutar una tarea. En realidad, un programa informático es un algoritmo que indica al ordenador qué pasos debe realizar y en qué orden para llevar a cabo una tarea específica. Se escriben utilizando un lenguaje de programación.

### ¿Cuáles son los diferentes tipos de algoritmos?

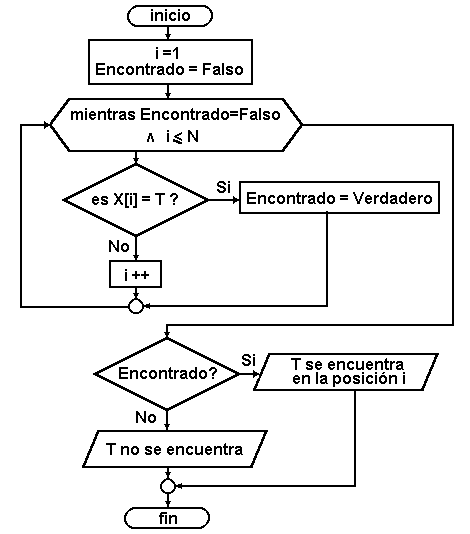
* Algoritmos de Búsqueda
* Algoritmos de Ordenamiento
* Algoritmos Recursivos
* Algoritmos de Grafos
* Algoritmos de Árboles
* Algoritmos de Caminos Mínimos
* Algoritmos de Flujo en Redes
* Algoritmos de Cadena de Texto
* Algoritmos de Machine Learning
* Algoritmos Genéticos

# Algoritmos cualitativos

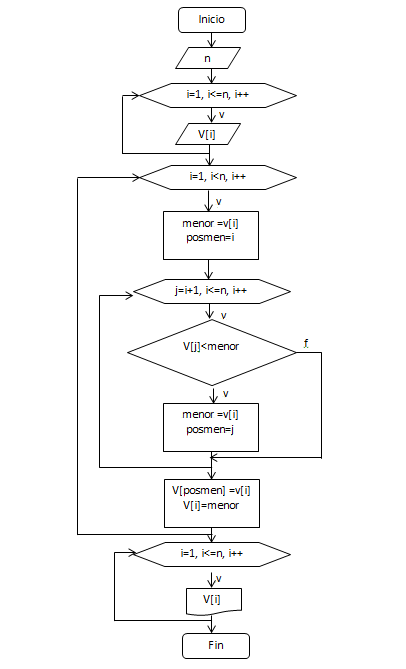


# Algoritmos cuantitativos

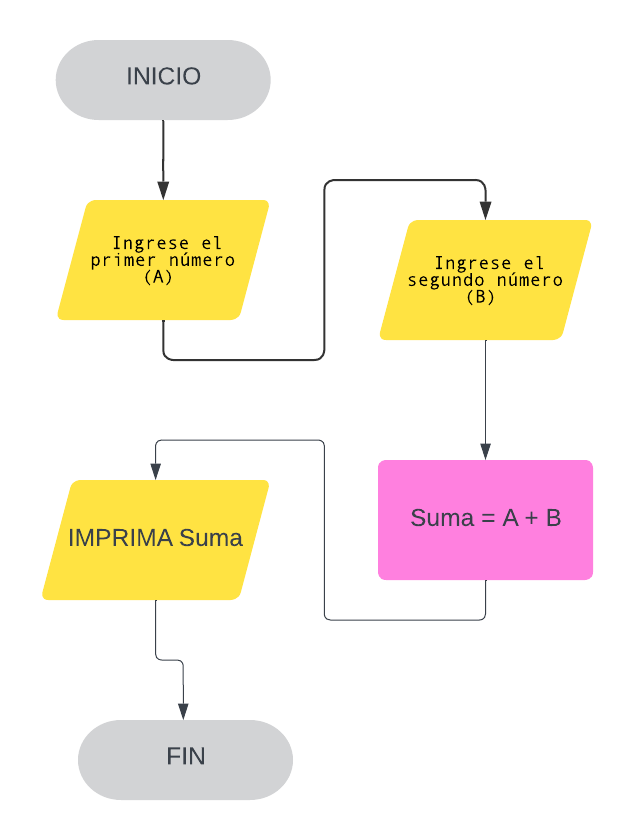
**ALGORITMO: BUSQUEDA SECUENCIAL**

****

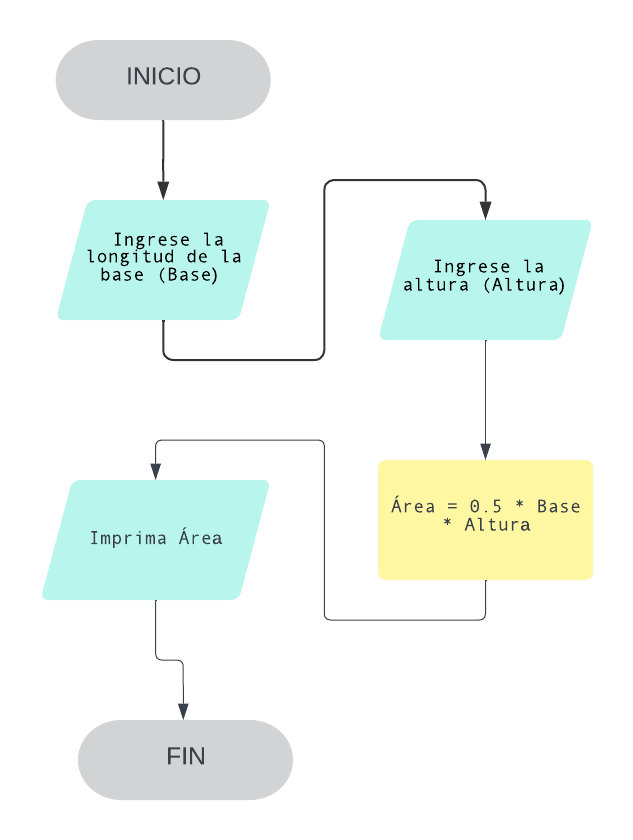
**ALGORITMO: ORDENAMIENTO DE UNA MATRIZ DE VALORES**

****

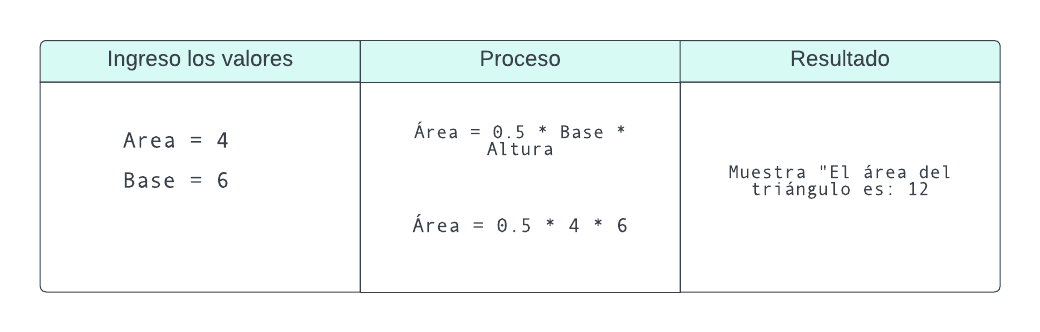
**Algoritmo: Suma de dos números**



**Algoritmo: Área de un triángulo**

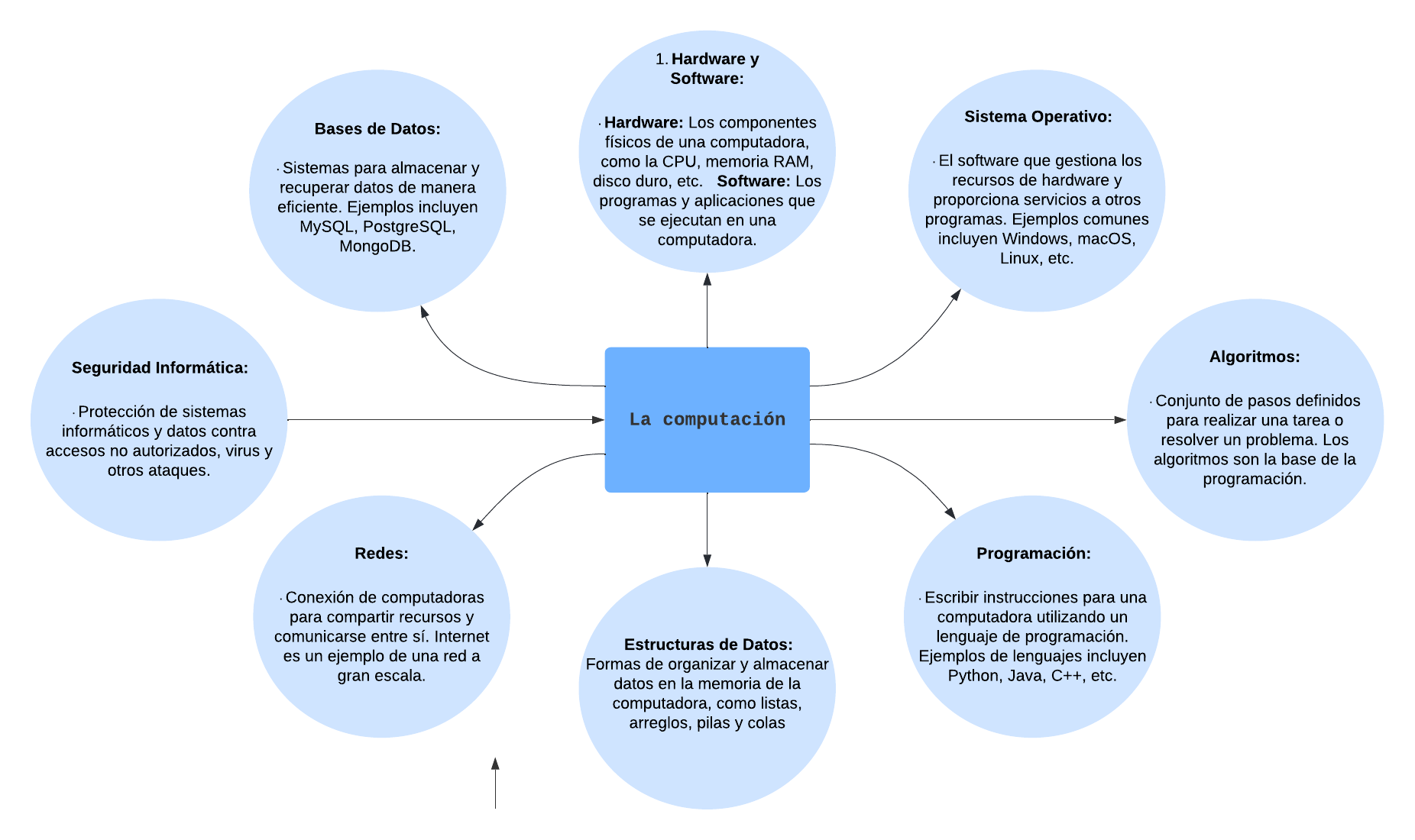


**Prueba de escritorio**



**LA COMPUTACIÓN, PROGRAMACIÓN Y PROGRAMAS**

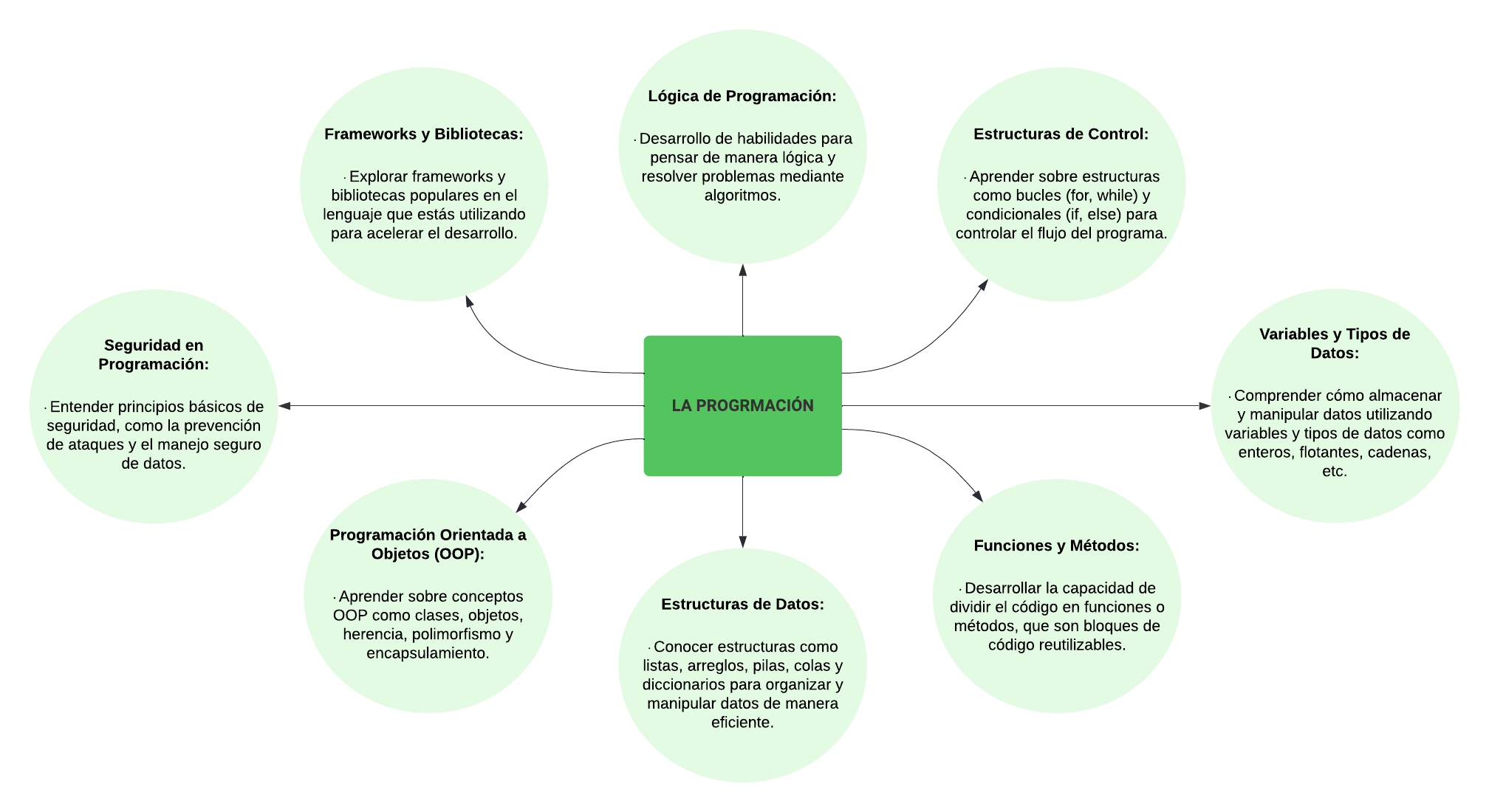
La computación es una disciplina que se ocupa del estudio y desarrollo de sistemas y programas informáticos, así como del procesamiento de información utilizando tecnologías de la computadora. Implica el diseño, la construcción y la utilización de sistemas informáticos y programas para resolver problemas, realizar tareas específicas y procesar datos.



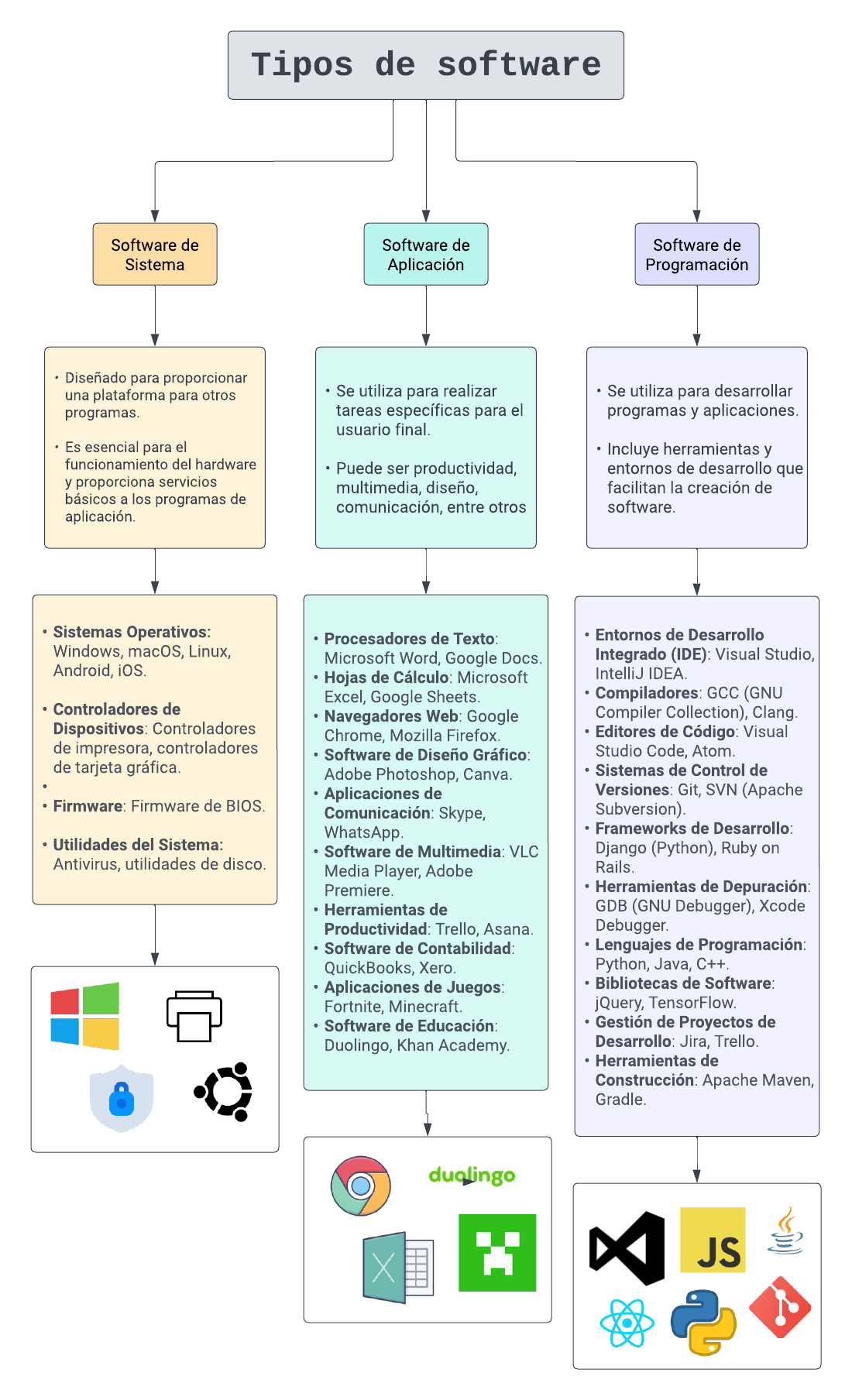
¿Qué es la programación?

La programación, en el contexto de la informática y la tecnología, se refiere al proceso de diseñar, codificar, depurar y mantener el código fuente de un programa de computadora. Un programa de computadora es un conjunto de instrucciones que le indican a una computadora cómo realizar una tarea específica. La programación es la habilidad de escribir estas instrucciones de manera clara y precisa para que la computadora las pueda ejecutar.





+



**Tipos de datos**

En el ámbito de la ciencia y la investigación, existen varios tipos de datos, y su importancia depende del contexto y del tipo de estudio.

**Tipos de Datos Numéricos:**

* **INT**: Números enteros.
* **DECIMAL** o **NUMERIC**: Números decimales o de punto flotante.
* **FLOAT**: Números de punto flotante de precisión simple.
* **DOUBLE**: Números de punto flotante de precisión doble.

**Tipos de Datos de Texto:**

* **CHAR(n)**: Cadenas de caracteres de longitud fija.
* **VARCHAR(n)**: Cadenas de caracteres de longitud variable.
* **TEXT**: Cadenas de texto de longitud variable (puede almacenar una cantidad mayor de caracteres que VARCHAR).

**Tipos de Datos de Fecha y Hora:**

* **DATE**: Fecha (año, mes, día).
* **TIME**: Hora del día.
* **DATETIME** o **TIMESTAMP**: Combinación de fecha y hora.

**Tipos de Datos Lógicos:**

* **BOOLEAN** o **BOOL**: Representa valores de verdad (verdadero o falso).
* **Tipos de Datos Binarios:**
* **BINARY(n)**: Datos binarios de longitud fija.
* **VARBINARY(n)**: Datos binarios de longitud variable.
* **BLOB**: Grandes objetos binarios, como imágenes o archivos.

**Tipos de Datos de Enumeración y Conjuntos:**

* **ENUM**: Lista de valores permitidos.
* **SET**: Conjunto de valores, donde se pueden seleccionar múltiples valores.
* **Tipos de Datos Especiales:**
* **NULL**: Representa la ausencia de un valor.
* **JSON**: Almacena datos en formato JSON.

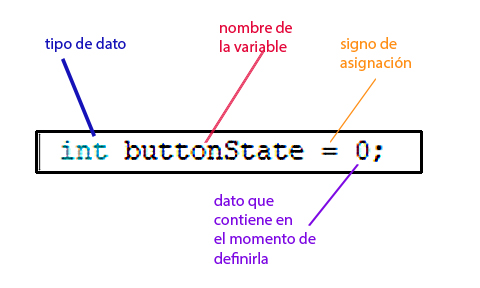
**Tipos de Datos Geoespaciales:**

Algunos sistemas de bases de datos, como PostgreSQL y MySQL con extensiones espaciales, admiten tipos de datos para almacenar y consultar información geoespacial.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nombre** | **Memoria requerida** | **Rango** | **Descripción** |
| Booleano | 1bit | 1 ~ 0 | Verdad - Falso |
| Byte | 1 byte (8 Bits) | 0 ~ 255 | Byte sin signo. |
| ByteSig | 1 byte (8 Bits) | (-128) ~ 127 | Byte con signo. |
| Word | 2 byte (16 Bits) | 0 ~ 65.535 | Word sin signo. |
| WordSig | 2 byte (16 Bits) | (-32768) ~ 32767 | Word con signo. |
| Entero | 4 byte (32 Bits) | 0 ~ 4.294.967.295 | Entero sin signo. |
| EnteroSig | 4 byte (32 Bits) | (-2.147.483.648) ~ 2.147.483.647 | Entero con signo. |
| Real | 8 byte (64 Bits) | (-1,79769313486232^308) ~ (-4,94065645841247^-324) | Número con coma flotante de doble precisión |
| Decimal | 8 byte (64 Bits) | (-922.337.203.685.477,5800) ~ 922.337.203.685.477,5800 | Número con coma fija de 4 decimales. |
| Cadena | 1 byte por carácter | 0 ~ 2000 millones de caracteres | Cadena de caracteres alfanumérica. |

**Constantes & variables**

Las variables se almacenan en la memoria RAM y el espacio de memoria que ocupan (en bytes) depende de su tipo. Una constante tiene las mismas características que una variable excepto el hecho de que su valor asignado no puede ser cambiado durante la ejecución de programa.



**VARIABLES**

Cada variable debe ser declarada antes de ser utilizada en el programa. Como las variables se almacenan en la memoria RAM, es necesario reservar el espacio para ellas (uno, dos o más bytes). Al escribir un programa, usted sabe qué tipo de datos quiere utilizar y qué tipo de datos espera como resultado de una operación, mientras que el compilador no lo sabe. No se olvide de que el programa maneja las variables con los nombres asignados. El compilador las reconoce como números en la memoria RAM sin conocer su tamaño y formato. Para mejorar la legibilidad de código, las variables se declaran con frecuencia al principio de las funciones:

<tipo> variable;

Es posible declarar más de una variable de una vez si tienen el mismo tipo.

<tipo> variable1, variable2, variable3;

Aparte del nombre y del tipo, a las variables se les asignan con frecuencia los valores iniciales justamente enseguida de su declaración. Esto no es un paso obligatorio, sino ‘una cuestión de buenas costumbres’. Se parece a lo siguiente:

unsigned int peso; // Declarar una variable llamada peso

peso = 20; // Asignar el valor 20 a la variable peso

Un método más rápido se le denomina declaración con inicialización (asignación de los valores iniciales):

unsigned int peso = 20; // peso está declarado y su valor es 20

Si hay varias variables con el mismo valor inicial asignado, el proceso se puede simplificar:

unsigned int peso1 = peso2 = peso3 = 20;

int valor\_inicial = un\_mínimo\_de\_petróleo = 0;

**CONSTANTES**

Similar a las variables, las constantes deben ser declaradas antes de ser utilizadas en el programa. En mikroC, no es obligatorio especificar el tipo de constante al declararla. Por otra parte, las constantes deben ser inicializadas a la vez que se declaran. El compilador reconoce las constantes por su prefijo const utilizado en la declaración. Dos siguientes declaraciones son equivalentes:

const int MINIMUM = -100; // Declarar constante MINIMUM

const MINIMUM = -100; // Declarar constante MINIMUM

Las constantes pueden ser de cualquier tipo, incluyendo cadenas:

const T\_MAX = 3.260E1; // constante de punto flotante T\_MAX

const I\_CLASS = 'A'; // constante carácter I\_CLASS

const Mensaje = "Presione el botón IZQUIERDA"; // constante de cadena Mensaje

Las constantes de enumeración son un tipo especial de constantes enteras que hace un programa más comprensible al asignar los números ordinales a las constantes. Por defecto, el valor 0 se asigna automáticamente a la primera constante entre llaves, el valor 1 a la segunda, el valor 2 a la tercera etc.

enum surtidores {AGUA,GASÓLEO,CLORO}; // AGUA = 0; GASÓLEO = 1; CLORO = 2

Es posible introducir directamente el valor de una constante dentro de la lista de enumeraciones. El incremento se detiene al asignar un valor a un elemento de matriz, después se reinicia a partir del valor asignado. Vea el siguiente ejemplo:

enum surtidores {AGUA,GASÓLEO=0,CLORO}; // AGUA = 0; GÁSOLEO = 0; CLORO = 1

Las constantes de enumeración se utilizan de la siguiente manera:

int Velocidad\_de\_ascensor

enum motor\_de\_ascensor {PARADA,INICIO,NORMAL,MÁXIMO};

Velocidad\_de\_ascensor = NORMAL; // Velocidad\_de\_ascensor = 2

## **Variables**

void main() {

// Declaramos una variable para guardar la altura

double altura = 2.5;

// Declaramos una variable para guardar el número de ruedas

int ruedas = 4;

// Calculo el área de un rectángulo de base 3 metros y altura 2.5 metros

println(altura \* 3.0);

// Calculo lo que cuesta cambiar las cuatro ruedas si una vale 150?

println(ruedas \* 150);

}

**CONSTANTES**

void main() {

// Declaramos una constante

final double GRAVEDAD = 9.8;

// Variable con la altura en metros a la que está situado el objeto

double altura = 10;

// Cálculo del tiempo que tarda en caer

double tiempo = sqrt(2.0\*altura/GRAVEDAD);

println(tiempo);

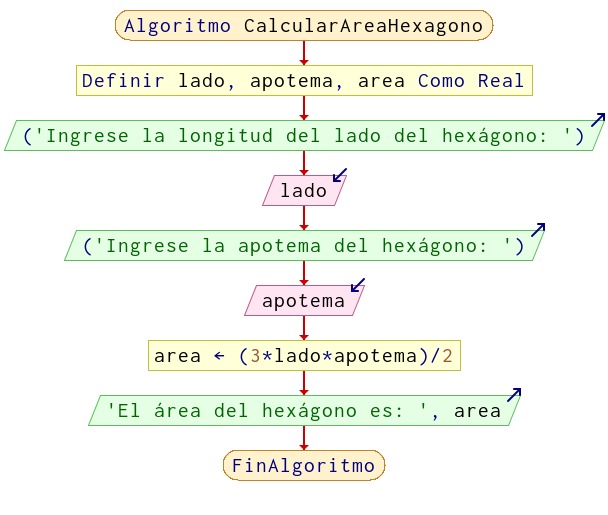
}

¿**Qué son los diagramas de flujo?**

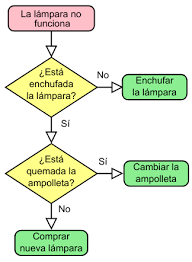
Un diagrama de flujo es un diagrama que describe un proceso, sistema o algoritmo informático. Se usan ampliamente en numerosos campos para documentar, estudiar, planificar, mejorar y comunicar procesos que suelen ser complejos en diagramas claros y fáciles de comprender

Los diagramas de flujo emplean rectángulos, óvalos, diamantes y otras numerosas figuras para definir el tipo de paso, junto con flechas conectoras que establecen el flujo y la secuencia. Pueden variar desde diagramas simples y dibujados a mano hasta diagramas exhaustivos creados por computadora que describen múltiples pasos y rutas

Los diagramas de flujo a veces se denominan con nombres más especializados, como diagrama de flujo de procesos, mapa de procesos o diagrama de flujo funcional.

****

****



**Las estructuras lógicas**  
  
Las estructuras lógicas son fundamentales en la programación y se utilizan para controlar el flujo de ejecución de un programa. Las principales estructuras lógicas son las condicionales y las estructuras de repetición.

Las utilizamos cuando en el desarrollo de la solución de un problema debemos tomar una decisión, para establecer un proceso o señalar un camino alternativo a seguir.

Estas estructuras lógicas son esenciales para controlar el flujo de ejecución en los programas y **permiten que los desarrolladores implementen algoritmos más complejos y eficientes**.

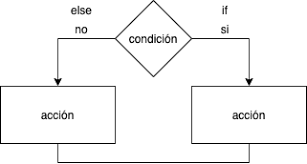


Estructuras Condicionales:

Las estructuras condicionales permiten que un programa tome decisiones basadas en una condición evaluada como verdadera o falsa. Comparan una variable contra otro(s) valor(es), para que con base al resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite.

En la mayoría de los lenguajes de programación, la estructura condicional más común es la declaración "if" (si), que se utiliza para ejecutar un bloque de código si una condición es verdadera.

También pueden incluir "else" (sino), que se ejecuta si la condición es falsa, y "else if" (sino si), que permite evaluar múltiples condiciones de manera secuencial.



Ejemplo en pseudocódigo:

si (condición) entonces

// código a ejecutar si la condición es verdadera

sino

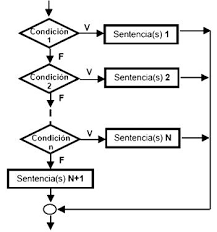
// código a ejecutar si la condición es falsa

fin si

Operadores lógicos (verdadero o falso):

Los operadores lógicos, como AND, OR y NOT, se utilizan para combinar y modificar condiciones en las estructuras condicionales.

Los operadores lógicos también se denominan operadores Booleanos. Devuelve True cuando Expresión1 y Expresión2 son verdaderas. Devuelve True cuando Expresión1 o Expresión2 es verdadera.



Ejemplo en pseudocódigo:

si (condición1 AND condición2) entonces

// código a ejecutar si ambas condiciones son verdaderas

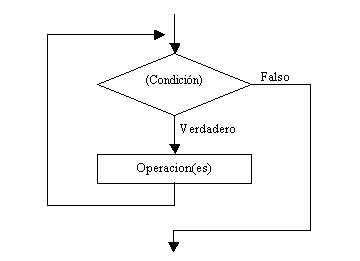
fin si

Estructuras de repetición

Se utilizan para realizar un determinado tipo de instrucciones en un número finito de veces. Se caracterizan por tener un punto inicial de partida, una condición la cual se encarga de ejecutar un número determinado de acciones hasta que esta condición no sea válida

Las estructuras de repetición más comunes son "for" y "while".

Un bucle "for" generalmente se utiliza cuando se conoce de antemano el número de iteraciones, mientras que un bucle "while" se utiliza cuando no se sabe cuántas veces se debe repetir el código.



Ejemplo en pseudocódigo de un bucle "for":

para i desde 1 hasta 10 hacer

// código a ejecutar en cada iteración

fin para

Ejemplo en pseudocódigo de un bucle "while":

mientras (condición) hacer

// código a ejecutar mientras la condición sea verdadera

fin mientras

**Identificadores**:

Los identificadores son nombres que se utilizan para identificar variables, funciones, clases u otros elementos en un programa. Deben seguir ciertas reglas, como comenzar con una letra, no contener espacios y ser significativos para facilitar la comprensión del código.

Los identificadores **son esenciales para referenciar** y manipular datos en un programa.

**Constantes:**

Las constantes **son valores que no cambian durante la ejecución de un programa**. Pueden ser numéricas, como 3.14, o cadenas de texto, como "Hola, mundo". El uso de constantes mejora la legibilidad del código y facilita la actualización de valores en un solo lugar.

**Operadores y Expresiones:**

Los operadores **son símbolos que realizan operaciones** sobre uno o más operandos. Las expresiones son combinaciones de valores y operadores que se evalúan para producir un resultado. Ejemplos de operadores incluyen aritméticos (+, -, \*, /), lógicos (&&, ||) y de comparación (==, !=).

**Abstracción de Datos:**

La abstracción de datos implica ocultar los detalles internos y **mostrar solo las características esenciales de un objeto**. En programación, las estructuras de datos y las clases permiten la abstracción de datos al encapsular información y proporcionar interfaces para interactuar con ella.

La abstracción de datos facilita el manejo de la complejidad al dividir un sistema en componentes más manejables.

**Variables Temporales:**

Las variables temporales son aquellas que se utilizan **para almacenar datos de manera temporal durante la ejecución de un programa**. Son útiles para realizar cálculos intermedios o almacenar valores temporales que se requieren en un momento específico. Las variables temporales ayudan a gestionar la manipulación de datos y facilitan la comprensión del flujo de un programa.

# Ejemplo de identificador para una variable

nombre\_usuario = "JohnDoe"

# Ejemplo de constantes

PI = 3.14159

SALUDO = "Hola, mundo"

# Ejemplo de operadores y expresiones

* a = 5 b = 2
* suma = a + b
* resta = a - b
* multiplicacion = a \* b
* division = a / b
* resultado\_logico = (a > b) and (a != 0)

# Ejemplo de abstracción de datos con una clase

class Persona:

def \_\_init\_\_(self, nombre, edad):

self.nombre = nombre

self.edad = edad

def obtener\_info(self):

return f"Nombre:{self.nombre},

Edad: {self.edad}"

# Uso de la clase

persona1 = Persona("Alice", 25)

print(persona1.obtener\_info())

construcción dE algoritmoS Y programas

En resumen, en el contexto de la programación, los identificadores y constantes son esenciales para dar nombre y significado a los elementos del programa. Los operadores y expresiones permiten realizar operaciones y cálculos. La abstracción de datos ayuda a manejar la complejidad al encapsular información, y las variables temporales son útiles para almacenar datos de manera temporal durante la ejecución del programa. Todos estos conceptos se entrelazan para construir programas efectivos y comprensibles.

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  | |
|  | Bibliografía  [Algoritmos y estructuras de datos - https://concepto.de](https://concepto.de/computacion/#:~:text=La%20computaci%C3%B3n%20es%20la%20ciencia,Estructura%20de%20datos%20y%20Algoritmos).  [Definicion de algoritmos - https://conceptodefinicion.de/computacion/](https://conceptodefinicion.de/computacion/)  [Informatica - https://es.wikipedia.org](https://es.wikipedia.org/wiki/Inform%C3%A1tica)  [Introduccion a la computacion - https://es.wikiversity.org](https://es.wikiversity.org/wiki/Introducci%C3%B3n_a_la_computaci%C3%B3n)  [Definicion de computacion - https://definicion.de/computacion/](https://definicion.de/computacion/)  [Tipos de datos de SQL - IBM](https://www.ibm.com/docs/es/iis/11.5?topic=stage-sql-data-types%7b)  [Constante & variables - https://www.mikroe.com](https://www.mikroe.com/ebooks/microcontroladores-pic-programacion-en-c-con-ejemplos/variables-y-constantes%23:~:text=Las%20variables%20se%20almacenan%20en,bytes)%20depende%20de%20su%20tipo.&text=Una%20constante%20tiene%20las%20mismas,durante%20la%20ejecuci%C3%B3n%20de%20programa).  <Que-es-un-diagrama-de-flujo> - <https://www.lucidchart.com>  [Tipos de datos - https://www.tecnologias-informacion.com](https://www.tecnologias-informacion.com/tipos-sql.html%7b)  [Identificadores en la programacion con C++ - https://www2.eii.uva.es{](https://www2.eii.uva.es/fund_inf/cpp/temas/2_tipos_variables/identificadores.html)  [Constantes y variables - https://www.ibm.com](https://www.ibm.com/docs/es/tcamfma/6.3.0?topic=tesl-constants-variables-4%7b)  [Operadores y expresiones - https://www.luisllamas.es](https://www.luisllamas.es/programacion-operadores-y-expresiones/)  [Abstraccion de datos - https://www.cursosdesarrolloweb.es](https://www.cursosdesarrolloweb.es/blog/abstraccion-programacion-orientada-objetos) | |  |
|  | |