



Junta de Andalucía
Consejería de Educación y Deporte

UT 1

Introducción a la programación

—

Módulo de Programación 1º DAW



This work is licensed under a [Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License](https://creativecommons.org/licenses/by-nc-sa/4.0/).

Autor: Fran Gómez



Objetivos

RA1: Reconoce la estructura de un programa informático, identificando y relacionando los elementos propios del lenguaje de programación utilizado.

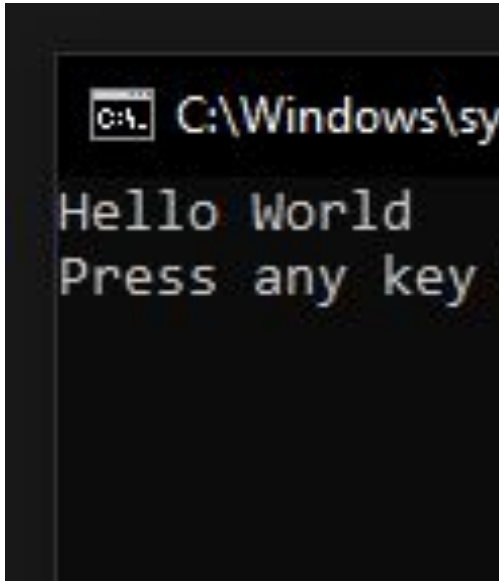
- Conocer los conceptos básicos relacionados con la programación y el diseño de aplicaciones
- Describir los paradigmas de programación más usados
- Aprender a utilizar sistemas de descripción de programas de alto nivel

Índice de contenidos

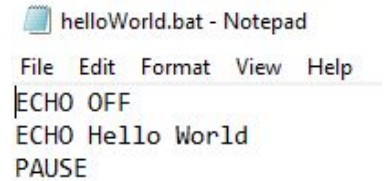
1. Introducción
 - 1.1. Evolución de la programación
 - 1.2. Conceptos básicos
 - 1.3. Clasificar los lenguajes de programación
2. Paradigmas de programación
 - 2.1. Declarativa
 - 2.2. Imperativa
 - 2.3. Estructurada
 - 2.4. Modular (cohesión y acoplamiento)
3. Elementos de un programa
4. Pseudocódigo
5. Diagramas de flujo
6. Tablas de decisión
7. Ciclo de vida del software
8. Recursos y referencias



Introducción



```
C:\Windows\system32>
Hello World
Press any key
```



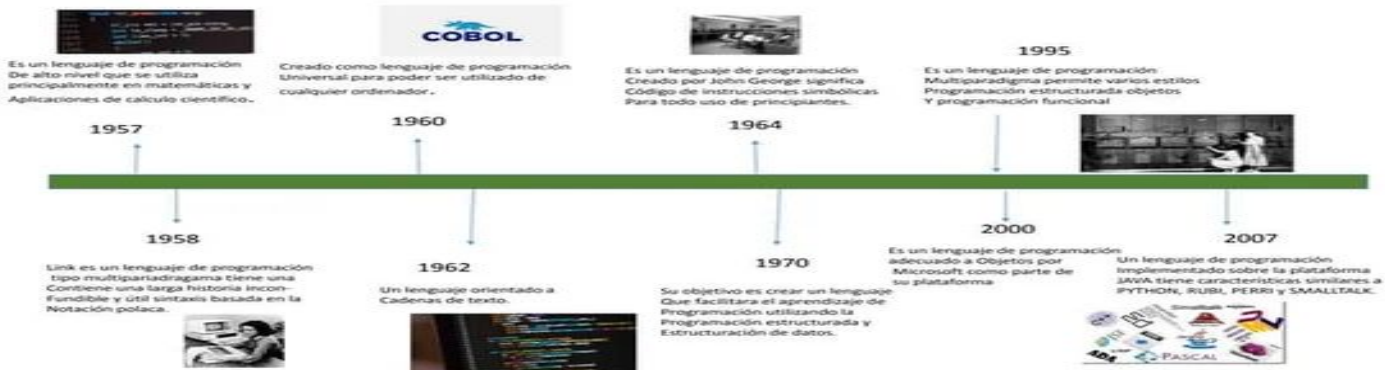
```
helloWorld.bat - Notepad
File Edit Format View Help
ECHO OFF
ECHO Hello World
PAUSE
```

Orígenes de la programación

Ejercicio

Crea una línea del tiempo con los items que más destacarías en la historia de la programación.

1. Año
2. Nombre (logotipo)
3. Creador
4. Una característica

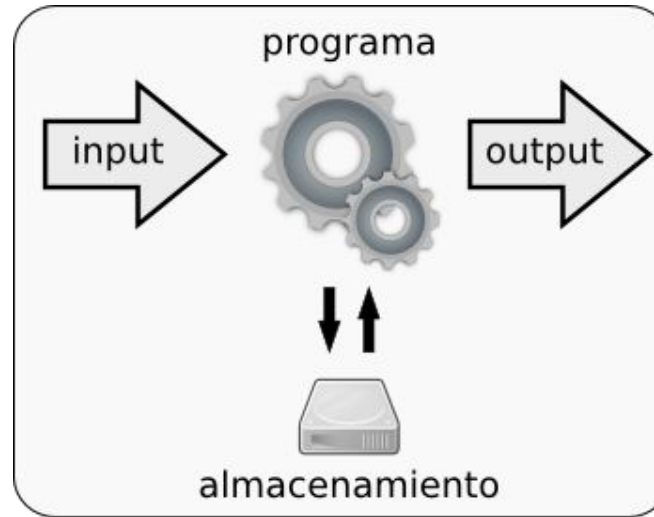


Conceptos de programación

- Programación
- Algoritmo
- Programar
- Código
- Lenguaje binario
- Lenguaje ensamblador
- Lenguajes de alto nivel

- Entrada
- Salida
- Cambio de estado

- Lógica
- Datos



Clasificación de los lenguajes

- Propósito:
 - General vs específico
- Tipo:
 - Scripting (o interpretado)
 - Compilado
 - Marcado
- Uso:
 - Desarrollo Web
 - Juegos
 - Sistemas
 - Móvil
 - IA, BigData, Machine Learning...
- Plataforma:
 - Web
 - Frontend
 - Backend
 - Desktop
 - Móvil
 - Embeded
- Paradigma:

Paradigmas de programación

Ejercicio

Cita todos los paradigmas de programación que encuentres en la Web y un lenguaje de ejemplo

- **Imperativa** \Rightarrow Instrucciones paso a paso
- **Estructurada** o procedural \Rightarrow Instrucciones siguiendo estructuras
- **Declarativa** \Rightarrow Resultado deseado
- **Modular** \Rightarrow Divide el programa en partes
- **Orientada a objetos** \Rightarrow Agrupa Datos y Lógica

Otros:

- Funcional \Rightarrow funciones matemáticas
- Lógico \Rightarrow reglas lógicas

Son subtipos de **Declarativa**

Paradigmas de programación

Característica	Declarativa	Imperativa	Estructural	Modular	Orientada a Objetos
Definición	Se enfoca en el <i>qué</i> se quiere lograr, sin describir el <i>cómo</i> .	Se enfoca en el <i>cómo</i> se deben realizar las tareas, con pasos detallados.	Extensión del paradigma imperativo, pero centrada en bloques de código llamados estructuras.	Divide el programa en módulos o unidades independientes que pueden ser reutilizados.	Organiza el código en objetos, que combinan datos y comportamiento.
Ejemplo de lenguajes	SQL, Prolog, Haskell	C, Assembly, Python (en algunos contextos), JavaScript	C, Pascal, ALGOL, Fortran	Ada, Modula-2, Python (por sus módulos)	Java, C++, Python, Ruby, Smalltalk

Programación Declarativa

Queremos programar la obtención de los clientes de Madrid mayores de 30 años.

```
SELECT nombre, apellido  
FROM clientes  
WHERE ciudad = 'Madrid' AND edad > 30;
```

Declaración, no instrucción: No le dices al ordenador cómo buscar a los clientes, simplemente **declaras** que quieres los nombres y apellidos de aquellos que cumplen las condiciones.

Foco en el resultado: Te centras en el qué quieres obtener, no en el **cómo** lo vas a obtener.

Motor de base de datos: El motor de base de datos se encarga de optimizar la consulta y encontrar los datos que cumplen con los criterios especificados.

Programación Imperativa

Queremos programar la elaboración de un sándwich

```
#include <stdio.h>

int main() {
    printf("Tomando una rebanada de pan\n");
    printf("Untando mantequilla\n");
    printf("Colocando queso\n");
    printf("Tomando otra rebanada\n");
    printf("Cerrando el sándwich\n");
    return 0;
}
```

Instrucciones secuenciales: Cada línea de código representa una instrucción específica que se ejecuta una después de la otra.

Estado mutable: El estado del sándwich va cambiando a medida que se ejecutan las instrucciones.

Foco en el proceso: Nos enfocamos en describir cómo hacer el sándwich, paso a otro.

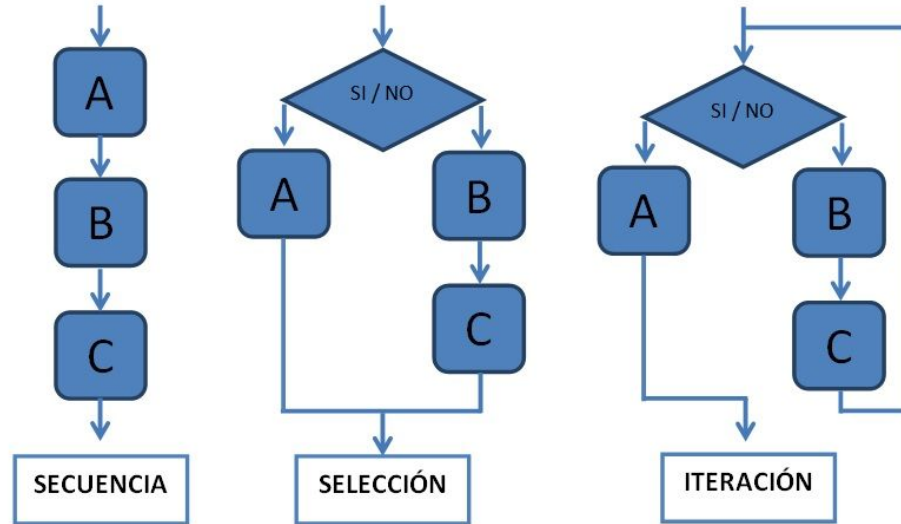
Ejercicio 2

Rellena una tabla como la siguiente, comparando los distintos lenguajes de programación

Lenguaje	Propósito	Tipo	Uso Principal	Plataforma	Paradigma	Ventajas	Desventajas
Python	General	Interpretado	Ciencia de datos, ML, desarrollo web	Web, ciencia de datos, ML	Multiparadigma	Fácil de aprender, gran comunidad	Velocidad de ejecución
JavaScript							
Java							
C#							
C							
C++							
PHP							
GO							
Rust							
Swift							
Ruby							
Bash/Batch/PowerShell							

Programación Estructurada

- Secuencial
- Alternativa
- Iterativa

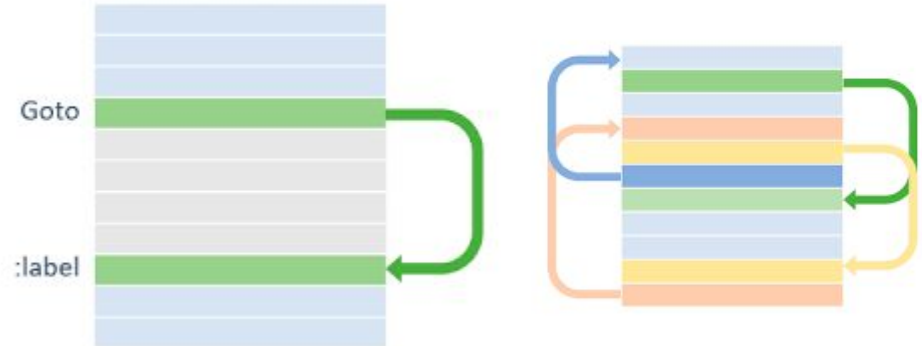


Ideas capacitación
Programación estructurada.

Programación Estructurada

Sentencia GOTO

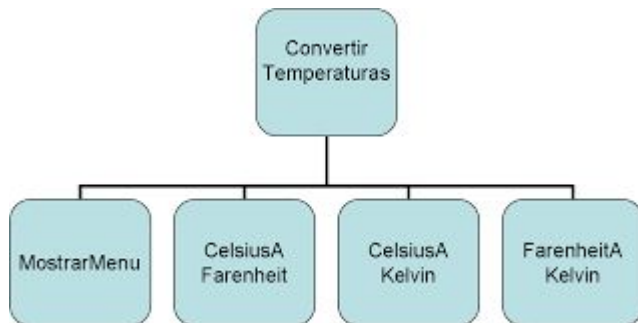
Código espagueti



```
goto etiqueta;  
  
...  
...  
...  
  
etiqueta: sentencia;
```

Programación Modular

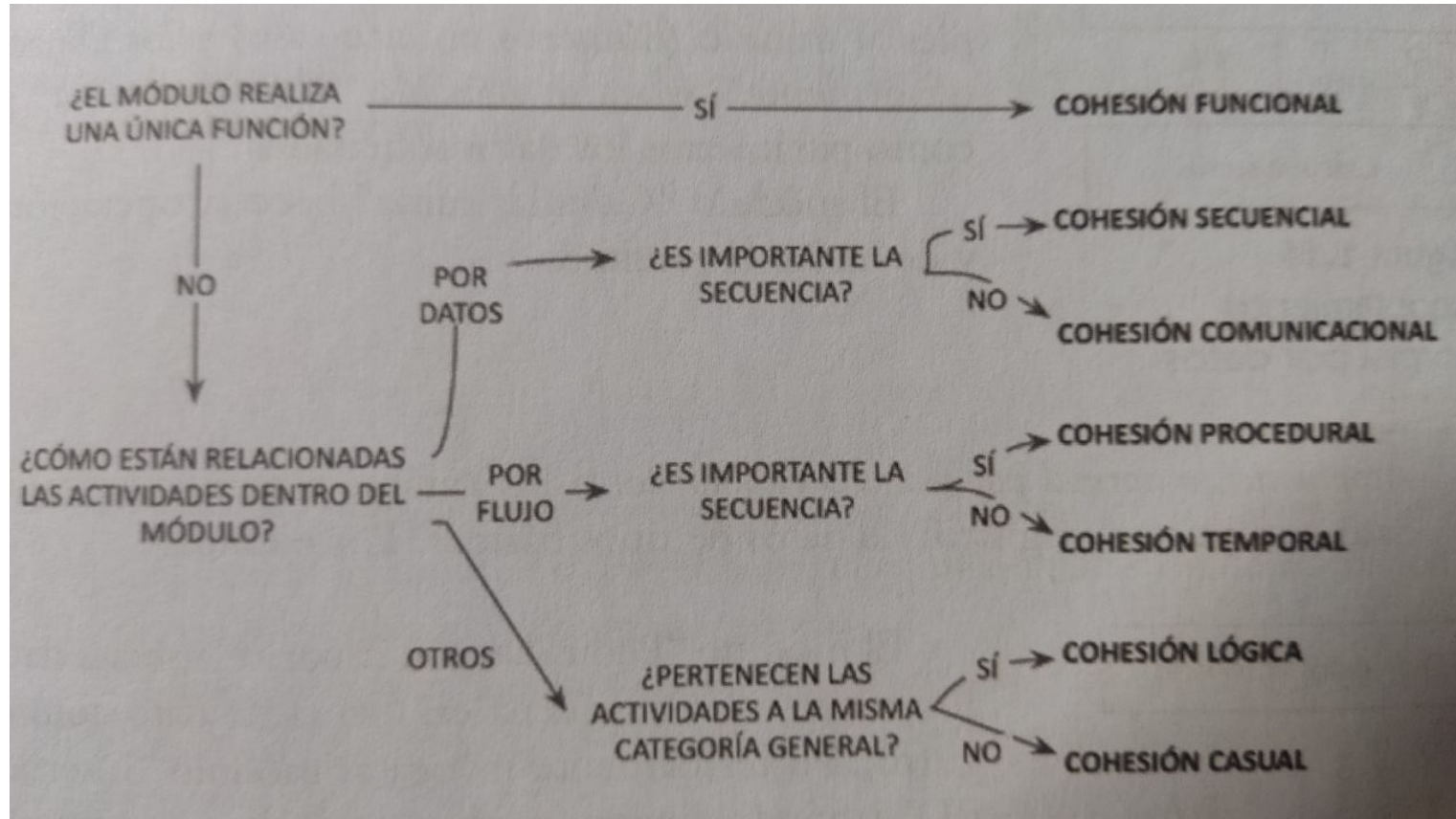
- Los programas crecen y se hacen más complejos \Rightarrow dividirlos en módulos
- Módulo = subprograma o rutina \Rightarrow funciones
- Librerías



¿Cómo dividir en módulos un programa?

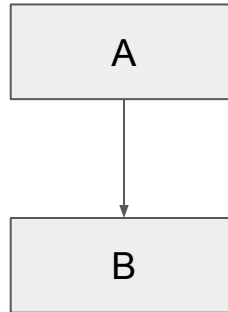
- Único punto de entrada y único punto de salida
- Función bien definida
- Caja negra
- Pequeños
- Máxima cohesión
- Mínimo acoplamiento

Cohesión

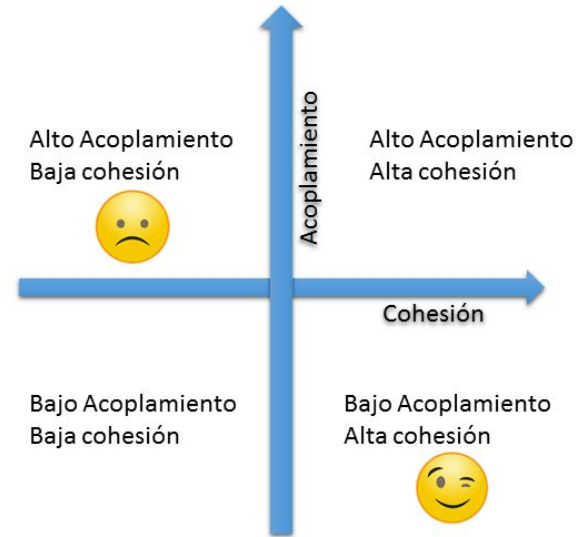
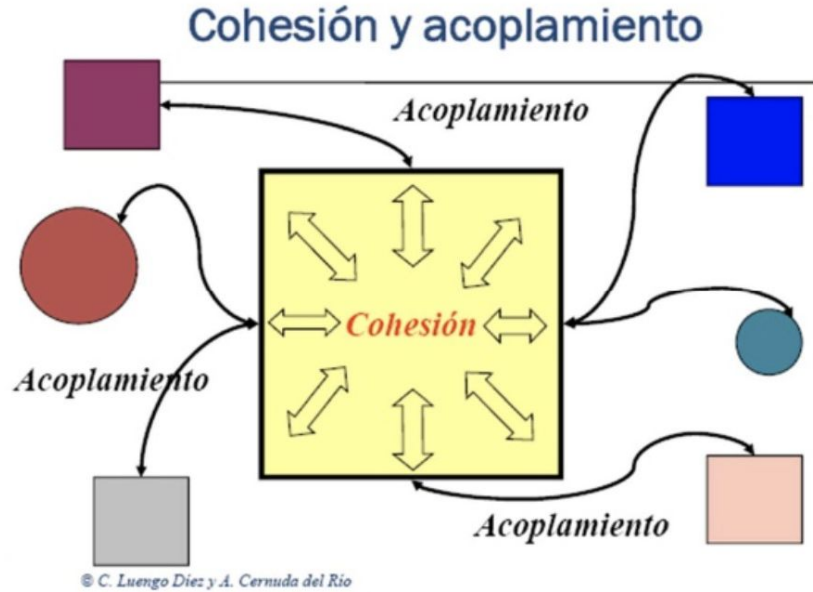


Acoplamiento

Si para hacer cambios en un módulo del programa es necesario hacer cambios en otro módulo distinto, existe acoplamiento entre ambos módulos.



Cohesión y Acoplamiento



Elementos de un programa

1. **Entrada**
 2. Procesamiento
 3. Salida
 4. Palabras reservadas
 5. Comentarios
- **Datos:** La información que el programa recibe del usuario o de otros dispositivos para realizar sus cálculos o tomar decisiones.
 - **Dispositivos de entrada:** Teclado, mouse, escáner, micrófonos, etc.



Elementos de un programa

```
instrucción 1
instrucción 2
.
.
.
instrucción n
```

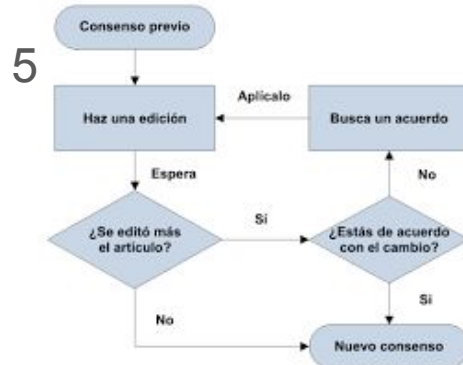
Memoria Variable

10	valor1
1000	pe

- **Instrucciones:** Órdenes. Unidad mínima.
- **Variables:** Espacios en la memoria del ordenador donde se almacenan los datos que el programa está utilizando.
- **Constantes:** como una variable cuyo valor inicial no cambia a lo largo del programa
- **Literales:** datos “*a pelo*” en el código.

1. Entrada
2. **Procesamiento**
3. Salida
4. Palabras reservadas

- **Operadores:** Símbolos que permiten realizar operaciones matemáticas, lógicas o de comparación sobre los datos.
- **Estructuras de control:** Mecanismos que permiten controlar el flujo de ejecución del programa, como las condicionales (si, entonces, sino) y los bucles (para, mientras).



Elementos de un programa

- | | | | |
|----|----------------------|-----------------|---|
| 1. | Entrada | ● | Expresiones: Son combinaciones de literales, constantes, variables y operadores para ejecutar una operación. |
| 2. | Procesamiento | | |
| 3. | Salida | | |
| 4. | Palabras reservadas | 7 + 3 | edad < 12 PI * r^2 |
| 5. | Comentarios | ● | Asignación: Operación que toma el valor de una expresión y lo almacena en una variable |
| | | nombre = "Fran" | suma = 2 + 3 |

Elementos de un programa

1. Entrada
2. Procesamiento
3. Salida
4. Palabras reservadas

- **Resultados:** La información que el programa genera a partir del procesamiento de los datos de entrada.
- **Dispositivos** de salida: Pantalla, impresora, altavoces, etc.



Elementos de un programa

- | | |
|-------------------------------|---|
| 1. Entrada | Comandos que entiende el programa |
| 2. Procesamiento | |
| 3. Salida | |
| 4. Palabras reservadas | <i>Algoritmo</i>
<i>FinAlgoritmo</i>
<i>Escribe</i>
<i>Lee</i>
<i>...</i> |
| 5. Comentarios | |

Elementos de un programa

1. Entrada
 2. Procesamiento
 3. Salida
 4. Palabras reservadas
 5. **Comentarios**
- Textos que añaden claridad al código, explicando qué hace cada parte del programa. Son muy útiles para documentar el código y facilitar su comprensión por parte de otros programadores o por uno mismo en el futuro.

Ejemplo

Programa que calcula el área de un círculo.

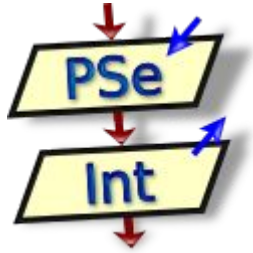
Entrada: El radio del círculo (un número ingresado por el usuario).

Procesamiento: Una variable llamada "radio" para almacenar el valor ingresado. Una constante llamada "pi" con el valor 3.14159. Una sentencia que multiplica el radio por sí mismo y luego por pi para calcular el área.

Salida: El resultado del cálculo (el área del círculo), mostrado en pantalla.

Comentarios: Explicaciones sobre cómo se realiza el cálculo y el significado de cada variable.

Pseudocódigo

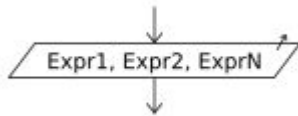
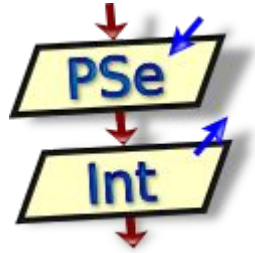


Crear un algoritmo sin usar un lenguaje de programación concreto.

- Lenguaje informal → Cada uno con sus palabras
- Ayuda a ver la lógica antes de pasar a codificar en un lenguaje real
- Facilita comunicación entre programadores
- Sirve para documentar el código
- Útil para aprender a programar

```
1  Algoritmo calcular_area_circulo
2      // Declaración de variables
3      Definir radio, area Como Real
4
5      // Entrada de datos
6      Escribir "Ingrese el radio del círculo: "
7      Leer radio
8
9      // Cálculo del área
10     area ← pi * radio * radio
11
12     // Salida de resultados
13     Escribir "El área del círculo es: ", area
14
15 FinAlgoritmo
```

Pseudocódigo



Estructuras de
control **secuencial**

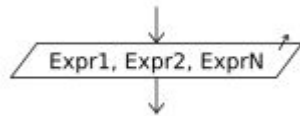
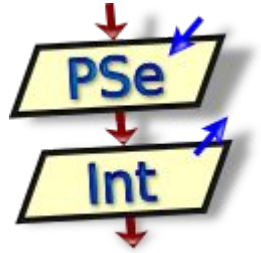
Ejercicios:

Instalamos PSeInt y empezamos a escribir nuestro pseudocódigo.

1. **Hola mundo:** Escribe un programa que muestre por pantalla “Hola mundo”
2. **Hola usuario:** Escribe un programa que muestre por pantalla “Hola “ y el nombre introducido por teclado.

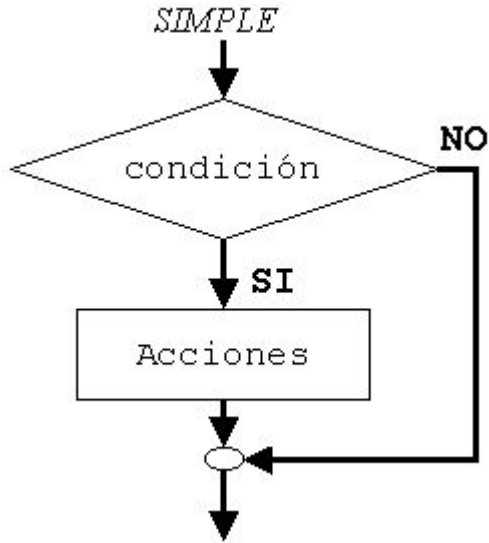
Ejemplo: si introducimos “Pepito” mostraría “*Hola Pepito*”. ¿Cómo harías para mostrar una admiración al final? Ej: “Hola Pepito!”

Pseudocódigo



Estructuras de
control **secuencial**

2. **Hola usuario:** Escribe un programa que muestre por pantalla “Hola “ y el nombre introducido por teclado. Ejemplo: si introducimos “Pepito” mostraría “Hola Pepito”. ¿Cómo harías para mostrar una admiración al final? Ej: “Hola Pepito!”
3. **suma 2 números:** Escribe un programa que sume dos números enteros introducidos por teclado y muestre el resultado por pantalla. Utiliza variables para cada número.



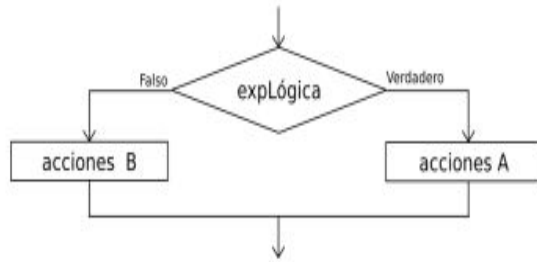
Estructuras de control **alternativas simples**

Pseudocódigo

4. **Login:** Escribe un programa que pida el nombre de usuario y si es correcto muestre “¡Bienvenido <usuario>!”.

Un usuario será correcto si coincide con el valor establecido literalmente por el programador en el código.

Pseudocódigo



Sentencias de control **alternativas múltiples**

5. Ahora añade que si el usuario no es correcto, entonces muestre “Usuario incorrecto”
6. **Mayor o menor:** Solicitar al usuario dos números y mostrar cuál es el mayor.
¿Cómo harías para evaluar además si son iguales?
7. **Par o impar:** Pedir un número al usuario y determinar si es par o impar.
8. **Calculadora básica:** Crear un programa que realice las cuatro operaciones básicas (suma, resta, multiplicación y división) según la opción que seleccione el usuario.