Introducción a la programación

¿Qué es programar? Programación estructurada

Programar es proporcionar las instrucciones necesarias a una máquina o aparato para que realice su función de manera automática.

En todo programa existe una serie de elementos que siempre van a estar presentes:

- **Entradas**: datos o información que hay que suministrar a la máquina para que pueda realizar las operaciones deseadas.
- **Salidas**: información o resultado que la máquina muestra al exterior.
- **Almacenamiento**: los datos, mientras son procesados por la máquina, necesitan almacenarse de alguna forma para trabajar con ellos. Aquí hablaremos de variables, vectores, matrices u otras estructuras complejas de almacenamiento de datos, si bien en este curso nos limitaremos a trabajar con variables.
- **Procesamiento**: operaciones al que se someten los datos para obtener los resultados deseados. Pueden ser aritméticas, lógicas, repeticiones, condicionales, etc...

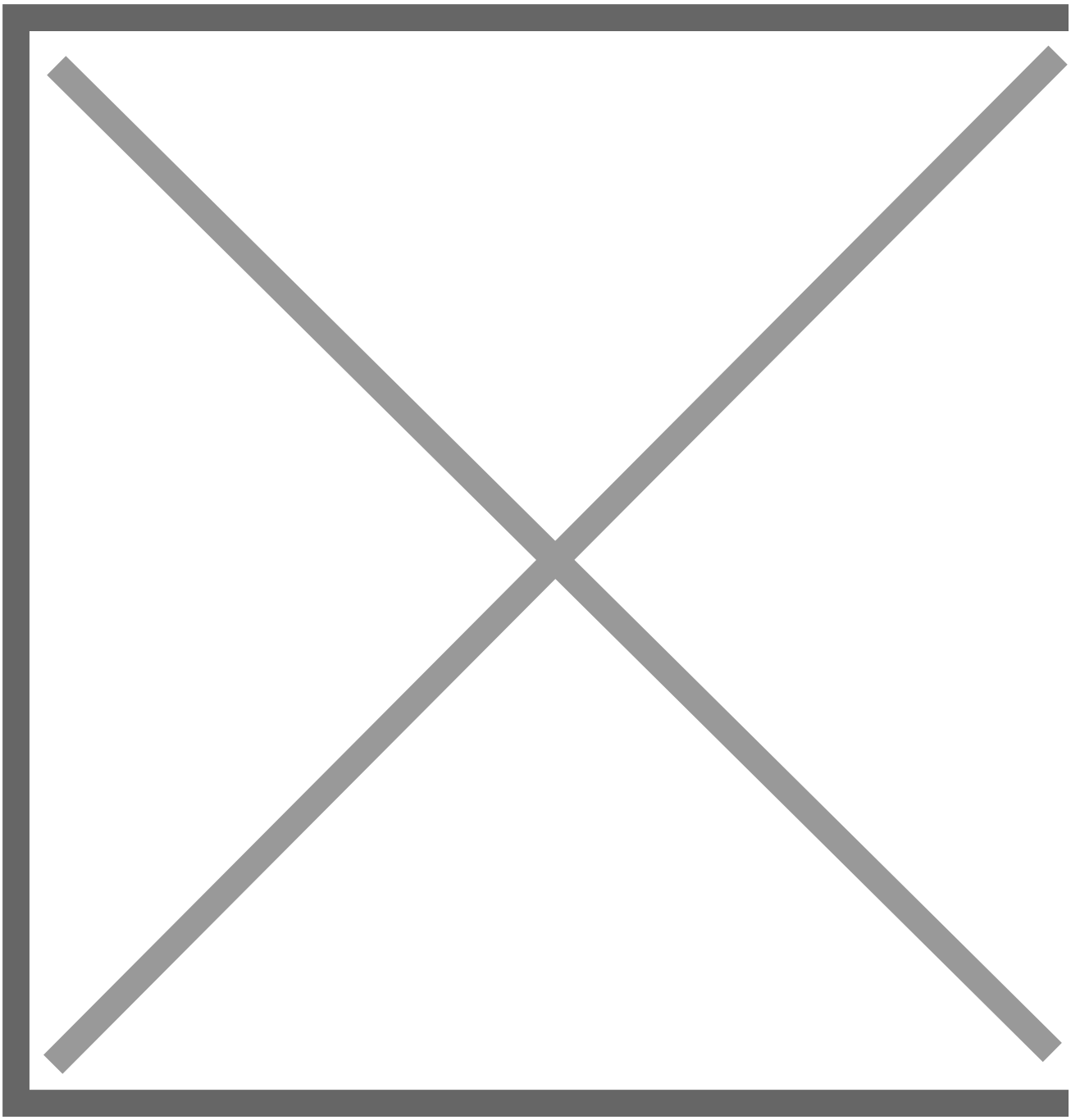
En programas extensos es probable que existan fragmentos de código que deban repetirse con alguna frecuencia. Para la automatización de ese proceso, y manteniendo un principio de programación llamado DRY explicado posteriormente, recurriremos a las **funciones**.

El desarrollo de un programa consta de los siguientes pasos:

1. Definición y análisis del problema.
2. Diseño del algoritmo mediante diagramas de flujo.
3. Codificación del programa: obtención del código fuente utilizando el lenguaje de programación elegido.
4. Compilación: conversión del código fuente al lenguaje máquina.
5. Depuración de errores y verificación del programa.
6. Explotación: documentación y mantenimiento. En este curso por la sencillez de los programas realizados no entraremos en este paso, si bien es fundamental que todo buen programa esté acompañado de una buena documentación y de un buen plan de actualizaciones.

Cuando hablamos de programación estructurada hablamos una forma de programar que se basa únicamente en la combinación de tres órdenes para el procesamiento de los datos:

1. **Secuencia.** La estructura secuencial es la que se da de forma natural en el lenguaje, porque las sentencias se ejecutan en el orden en el que aparecen en el programa, es decir, una detrás de la otra.
2. **Selección o condicional.** La estructura condicional se basa en que una sentencia se ejecuta según el valor que se le atribuye a una variable booleana. Por tanto, esta estructura se puede ejecutar de dos formas distintas, dependiendo del valor que tenga su variable. Para este fin, los lenguajes utilizan la estructura ***if*** o ***Si*** en español.
3. **Iteración (ciclo o bucle).** La estructura de repetición ejecuta una o un conjunto de sentencias siempre que una variable booleana sea verdadera. Para los bucles o iteraciones, los lenguajes de programación usan las estructuras ***while*** y ***for***. (***Mientras*** y ***para*** en español)



La programación estructurada se contrapone habitualmente a la programación orientada a objetos. No obstante existen una serie de principios a la hora de diseñar el software que son comunes:

- **KISS**: acrónimo de Keep It Simple, Stupid! Este principio nos dice que cualquier sistema va a funcionar mejor si se mantiene sencillo que si se vuelve complejo. La sencillez tiene que ser una meta en el desarrollo y la complejidad innecesaria debe ser eliminada.
- **DRY**: acrónimo de Don't Repeat Yourself. Lo que se intenta evitar con este principio es la duplicidad del código, en primer lugar por el principio anterior, y también porque el

mantenimiento posterior se vuelve más difícil ya que no sabemos donde tenemos que modificar las cosas porque se repiten en diversas ocasiones a lo largo del programa y las inconsistencias se multiplican.

Para más información sobre las diferencias entre estos dos tipos de programación consultar aquí.

Para más información sobre los principios de diseño del software consultar aquí.

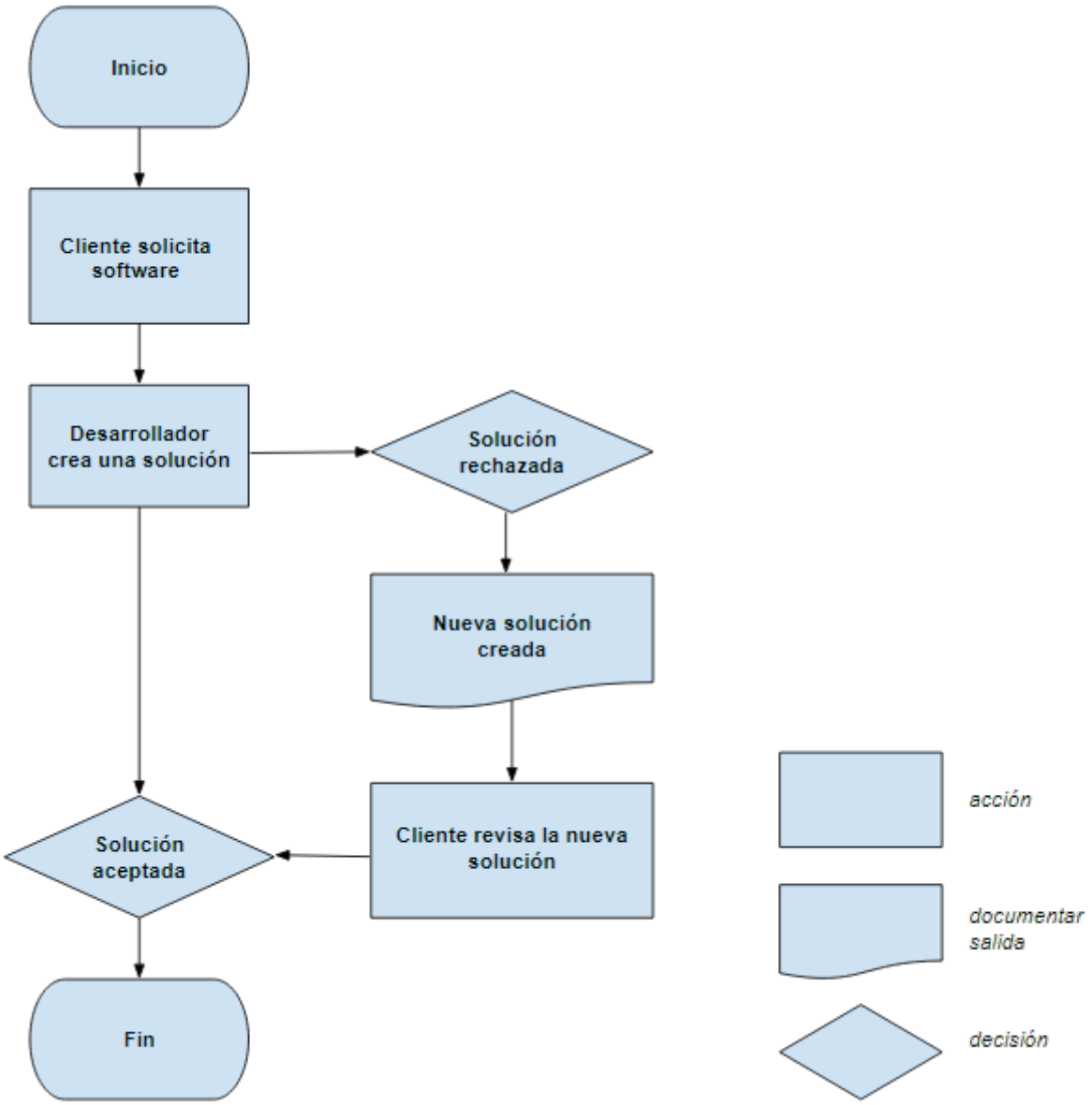
Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU

Diagramas de flujo

Un **algoritmo** es una sucesión de pasos que se deben realizar para resolver un problema.

Un **diagrama de flujo** es una forma de representar un proceso o algoritmo de manera visual, estructurada y organizada. Es una herramienta muy útil para organizar y estructurar una tarea de programación antes de entrar directamente con el código.

Ejemplo de diagrama de flujo:

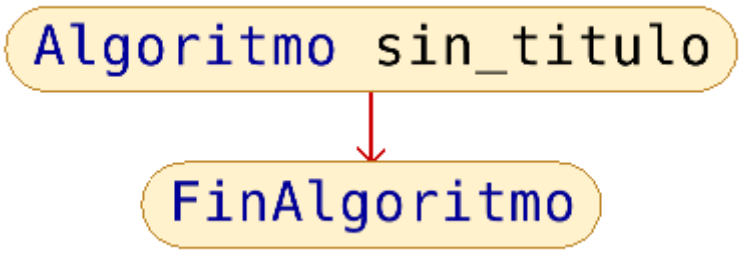


Aunque existen diversos programas informáticos para la realización de diagramas de flujo, en este curso utilizaremos una funcionalidad de PSeInt que lo permite. No obstante, y atendiendo a la cada vez más abundante evidencia científica al respecto, recomendamos que los diagramas de flujo se aborden con el alumnado en primer lugar **con papel y boli, y si es pertinente de forma colaborativa**, dejando para el software simplemente su edición final para incluir como documentación del programa.

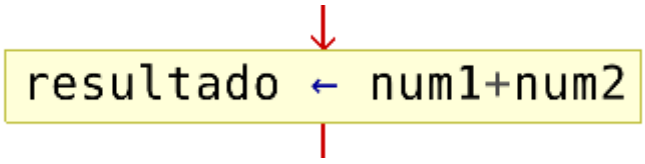
Elementos del diagrama de flujo

Un diagrama de flujo está formado fundamentalmente por los siguientes elementos:

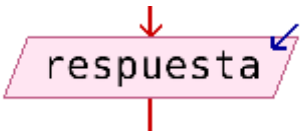
- **Línea o flechas del flujo**: Indica la instrucción que se va a realizar, o la dirección del flujo del proceso. Muestra el carácter secuencial del programa.
- **Terminal o inicio y final**: Es la forma en la cual se indica el “inicio del proceso” y “final del proceso”. Su icono suele ser un *rectángulo con las esquinas redondeadas.*



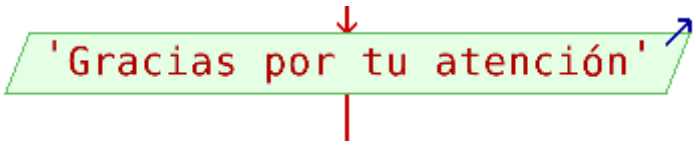
- **Asignación/ definición**: Permite asignar un valor o definir una variable, donde se almacenan los datos que se necesitan para resolver el problema. Suele representarse con un rectángulo.



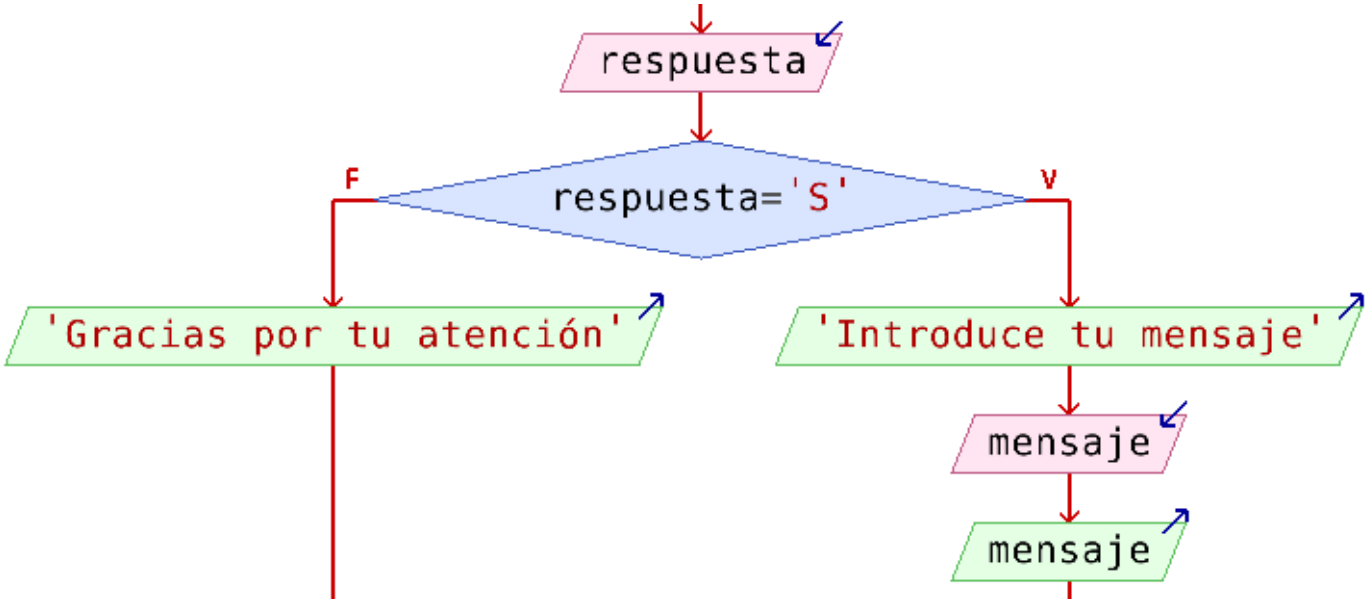
- **Datos de entrada**: Indica la recepción de datos en la entrada. Se representa con un recuadro con las esquinas inclinadas y una flecha hacia dentro.



- **Datos de salida**: Indica la impresión de datos en la salida. Se representa con un recuadro con las esquinas inclinadas y una flecha hacia fuera.



- **Decisión**: Indica que desde el punto en que nos encontramos, puede haber más de un camino para seguir, según la condición dada. En este caso se usa un rombo.



- **Otros**: Como algunos tipos de bucles, los iremos viendo más adelante.

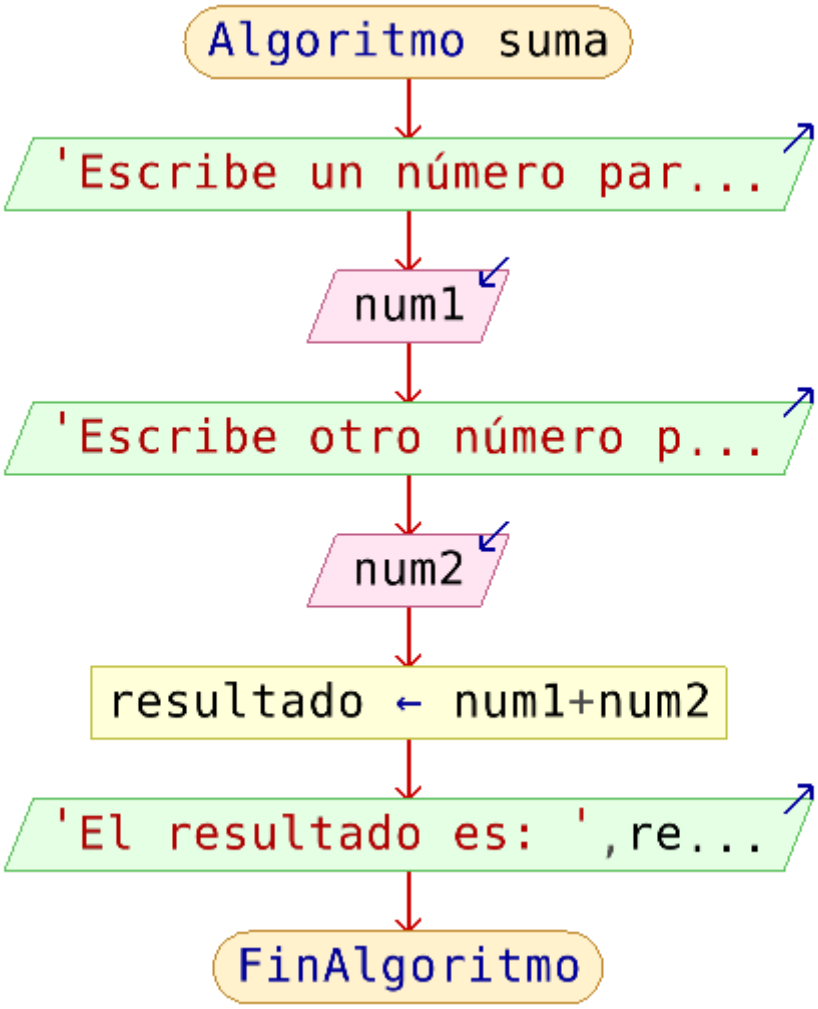
Ejemplos sencillos de diagramas de flujo

Ejemplo 1: Diagrama de flujo de un programa que a partir de dos números los suma y muestra el resultado en pantalla

SOLUCIÓN:

- **Salidas**: tiene que pedir los dos números a sumar, y luego tiene que mostrar el resultado.
- **Entradas**: los dos números a sumar.
- **Almacenamiento**: dos variables que almacenen los números a sumar (num1 y num2) , y una tercera que almacene el resultado (resultado)
- **Operación**: suma.

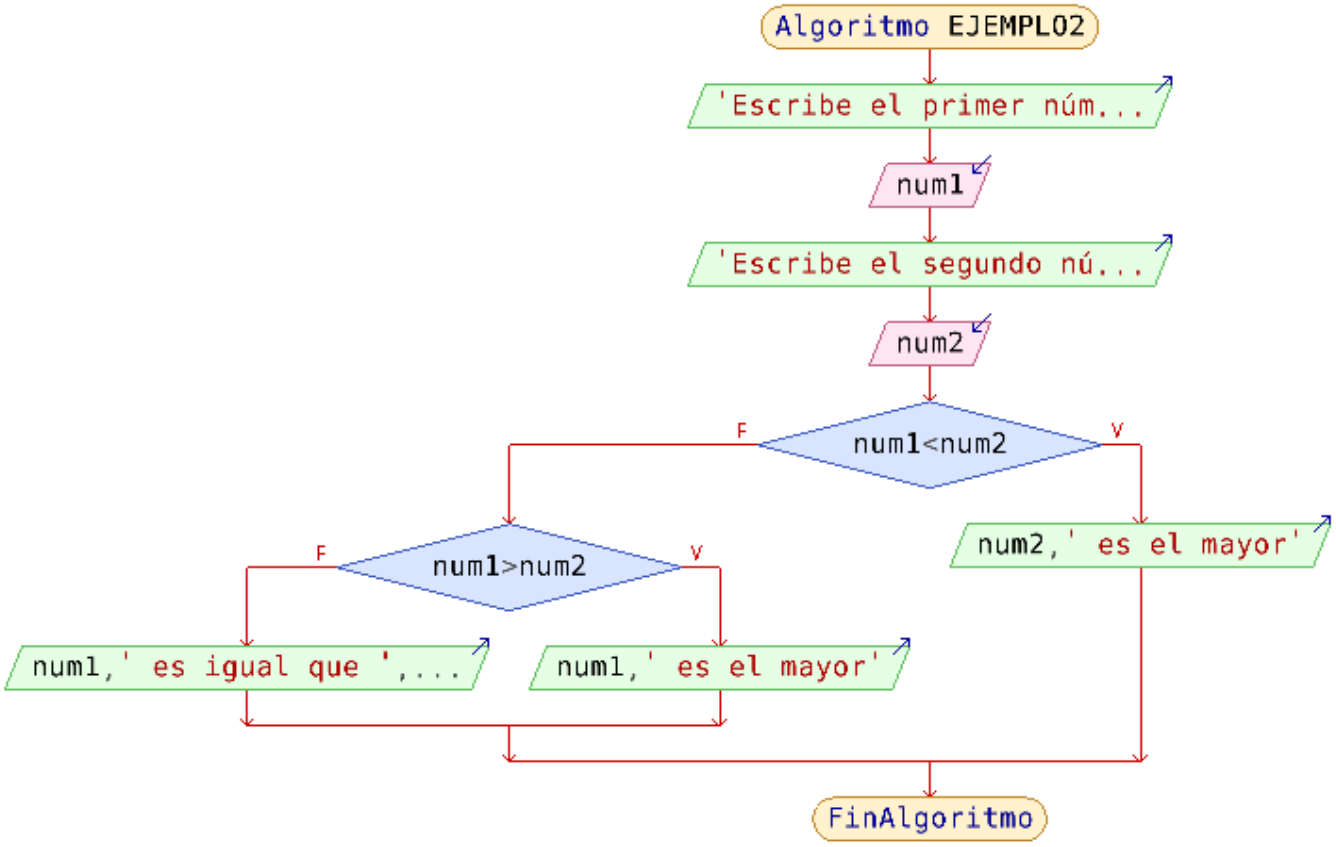
Diagrama de flujo:



Ejemplo 2: Diagrama de flujo de un programa que a partir de dos números compara cuál es mayor y lo muestra en pantalla.

- **Salidas**: tiene que pedir los dos números a comparar, y luego tiene que mostrar el resultado de la comparación.
- **Entradas**: los dos números a comparar.
- **Almacenamiento**: dos variables que almacenen los números a comparar (num1 y num2)
- **Operación**: comparación lógica.

Diagrama de flujo:



Financiado por el Ministerio de Educación y Formación Profesional y por la Unión Europea - NextGenerationEU