ALGORITMOS FUNDAMENTALES

CARACTERÍSTICAS DE LOS ALGORITMOS Y TIPOS

Algoritmia

Algoritmo

- Informalmente, un algoritmo es cualquier procedimiento computacional bien definido que toma algún valor, o un conjunto de valores como entrada y produce algún valor, o conjunto de valores como salida.

- Un algoritmo es por lo tanto una secuencia de pasos computacionales que transforma la entrada en la salida.

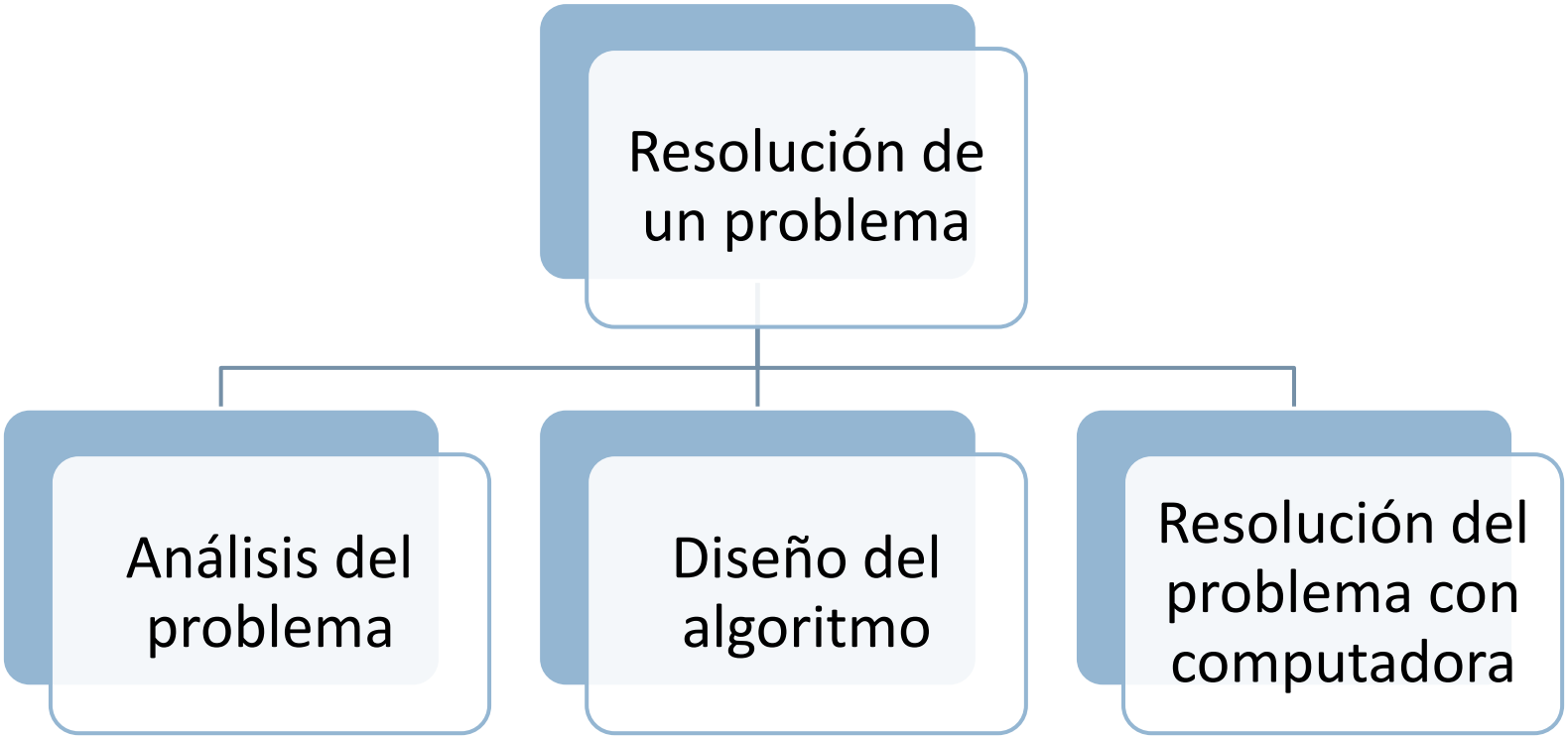
- También podemos ver un algoritmo como una herramienta para resolver un problema computacional bien especificado.

Análisis del enunciado de un problema

- Resulta evidente que, si vamos a diseñar un algoritmo para resolver un determinado problema, tenemos que tener totalmente estudiado y analizado el contexto de dicho problema. Esto implica:
  - Comprender el alcance.
  - Identificar los datos de entrada.
  - Identificar los datos de salida o resultados.

- El análisis anterior es fundamental para poder diseñar una estrategia de solución que será la piedra fundamental para el desarrollo de algoritmo.

Análisis del problema



Teorema de la programación estructurada

- Este teorema establece que todo algoritmo puede resolverse mediante el uso de tres estructuras básicas llamadas “estructuras de control”:
  - La “estructura secuencial” o “acción simple”.
  - La “estructura de decisión” o “acción condicional”.
  - La “estructura iterativa” o “acción de repetición”.

Conceptos de programación

- Estudiamos algoritmos para aplicarlos a la resolución de problemas mediante el uso de la computadora.

- Los lenguajes de programación son lenguajes formales que se componen de un conjunto de palabras, generalmente en inglés, y reglas sintácticas y semánticas.

- Podemos utilizar un lenguajes de programación para escribir o codificar nuestro algoritmo y luego, con un programa especial llamado “compilador”, podremos generar los “unos y ceros” que representan sus acciones.

Codificación de un algoritmo

- Cuando escribimos las acciones de una algoritmo en algún lenguaje de programación decimos que lo estamos codificando. Generalmente cada acción se codifica en una línea de código.

- Al conjunto de líneas de código lo llamamos “código fuente”.

- El código fuente debe de estar contenido en un archivo de texto cuyo nombre debe tener una extensión determinada que dependerá del lenguaje de programación que hayamos utilizado.

Programas de computación

- El proceso para crear un nuevo programa es el siguiente:

  - Diseñar y desarrollar su algoritmo.
  - Codificar el algoritmo utilizando un lenguaje de programación.
  - Compilarlo para obtener el código de máquina o archivo ejecutable.

Entrada y salida de datos

- Llamamos “entrada” al conjunto de datos externos que ingresan al algoritmo. Por ejemplo, el teclado, un lector de código de barras, un scanner de huellas digitales, etc.

- Llamamos “salida” a la información que el algoritmo emite sobre algún dispositivo como puede ser la consola, una impresora, un archivo, etc.

REPRESENTACIÓN DE ALGORITMOS

Pseudocódigo y Diagramas de flujo

Lenguajes algorítmicos

- Llamamos “lenguaje algorítmico” a todo recurso que permita describir con mayor o menor nivel de detalle los pasos que componen un algoritmo.

- El pseudocódigo surge de mezclar un lenguaje natural (por ejemplo, el español) con ciertas convenciones sintácticas y semánticas propias de un lenguaje de programación.

Pseudocódigo

- Calcular el valor de la suma 1 + 2 + 3 + … + 100.

Algoritmo

Se utiliza una variable `Contador` como un contador que genere los sucesivos números enteros, y `Suma` para almacenar las sumas parciales 1, 1 + 2, 1 + 2 + 3 ...

```
1. Establecer Contador a 1
2. Establecer Suma a 0
3. mientras Contador < = 100 hacer
      Sumar Contador a Suma
      Incrementar Contador en 1
   fin_mientras
4. Visualizar Suma
```

- La descripción de un algoritmo usualmente se hace en tres niveles:
  - Descripción de alto nivel. Se establece el problema, se selecciona un modelo matemático y se explica el algoritmo de manera verbal.
  - Descripción formal. Se usa pseudocódigo para describir la secuencia de pasos que encuentran la solución.
  - Implementación. Se muestra el algoritmo expresado en un lenguaje de programación específico o algún objeto capaz de llevar a cabo instrucciones.

Tipos de algoritmos según su función

- Algoritmos de ordenamiento
  - Ordenamiento por inserción
  - Ordenamiento por selección
  - Ordenamiento de burbuja
  - Ordenamiento por mezcla

- Algoritmos de búsqueda
  - Búsqueda secuencial
  - Búsqueda binaria
  - Búsqueda indexada

Técnicas de diseño de algoritmos

- **Algoritmos voraces (greedy):** Seleccionan los elementos más prometedores del conjunto de candidatos hasta encontrar una solución. En la mayoría de los casos la solución no es óptima.

- **Algoritmos paralelos:** Permiten la división de un problema en subproblemas de forma que se puedan ejecutar de forma simultánea en varios procesadores.

- **Algoritmos probabilísticos:** Es un algoritmo donde el resultado o la manera en que se obtiene el resultado depende de la probabilidad. A veces también son llamados algoritmos aleatorios.

- **Algoritmos determinísticos:** El comportamiento del algoritmo es lineal: cada paso del algoritmo tiene únicamente un paso sucesor y otro antecesor.

Técnicas de diseño de algoritmos

- **Algoritmos no determinísticos:** el comportamiento del algoritmo tiene forma de árbol y a cada paso del algoritmo puede bifurcarse a cualquier número de pasos inmediatamente posteriores, además todas las ramas se ejecutan simultáneamente.

- **Divide y vencerás:** Dividen el problema en subconjuntos disjuntos obteniendo una solución de cada uno de ellos para después unirlas, logrando así la solución al problema completo.

- **Metaheurísticas:** Encuentran soluciones aproximadas (no óptimas) a problemas basándose en un conocimiento anterior (a veces llamado experiencia) de los mismos.

Técnicas de diseño de algoritmos

- **Programación dinámica:** Intenta resolver problemas disminuyendo su coste computacional aumentando el coste espacial.

- **Ramificación y acotación:** Se basa en la construcción de las soluciones al problema mediante un árbol implícito que se recorre de forma controlada encontrando las mejores soluciones.

- **Vuelta atrás (backtracking):** Se construye el espacio de soluciones del problema en un árbol que se examina completamente, almacenando las soluciones menos costosas.

Representación gráfica de algoritmos

- Los algoritmos pueden representarse mediante el uso de diagramas.

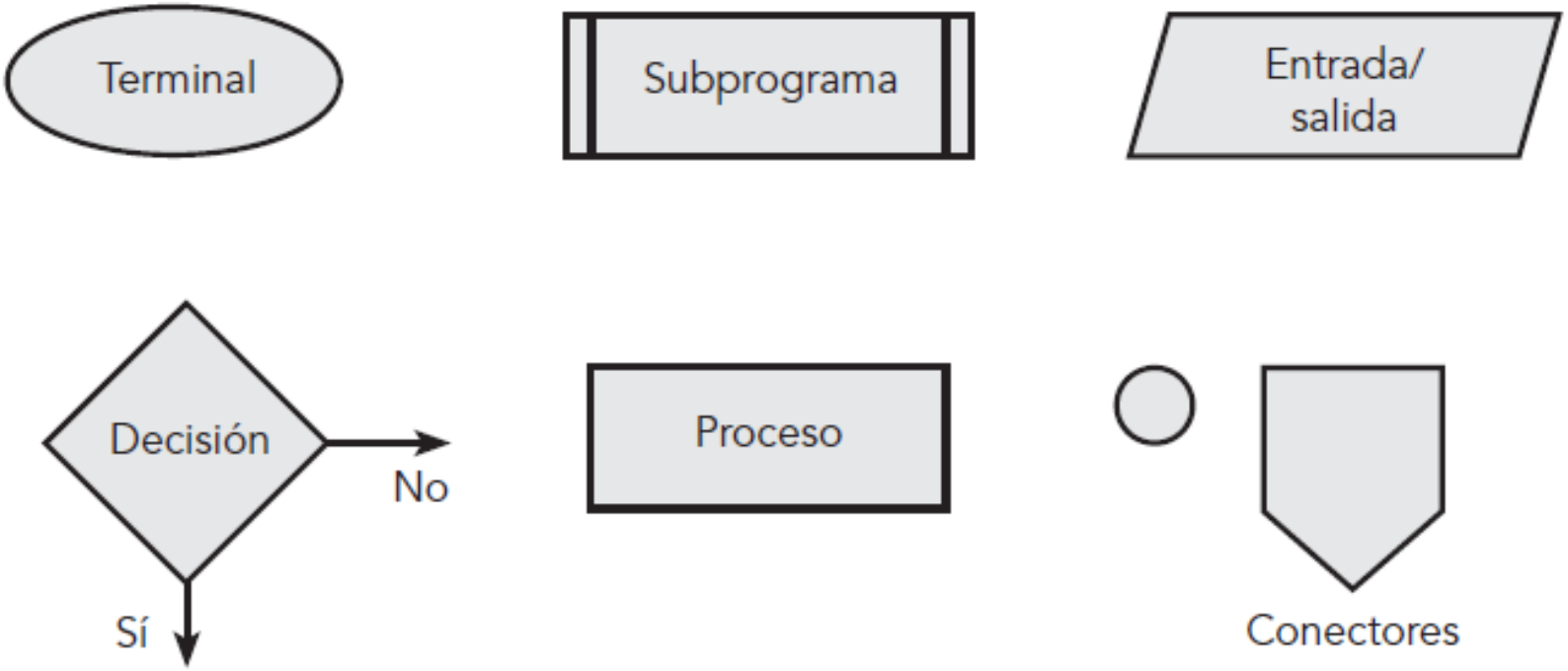
- Los diagramas proveen una visión simplificada de la lógica del algoritmo y son una herramienta importantísima para analizar y documentar los algoritmos o programas.

Diagrama de flujo

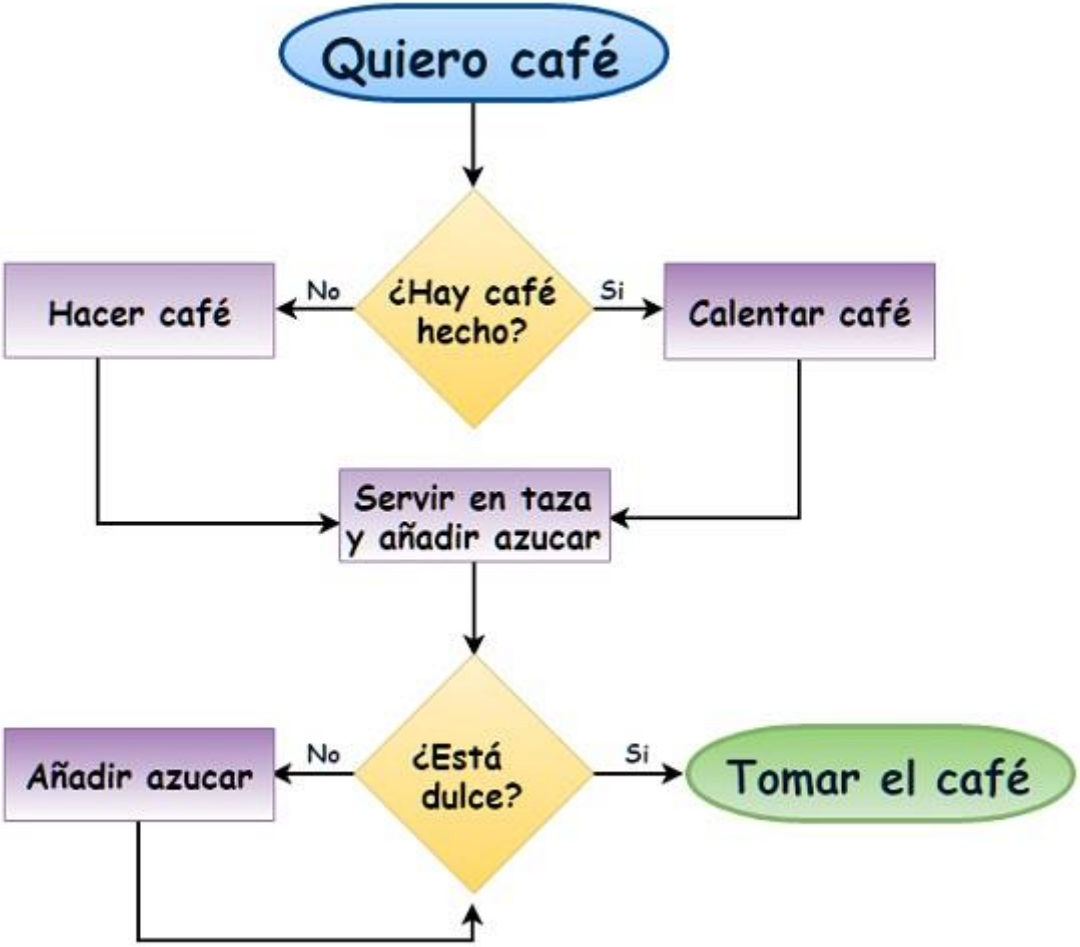
- Los diagramas de flujo son descripciones gráficas de algoritmos; usan símbolos conectados con flechas para indicar la secuencia de instrucciones.

- Los diagramas de flujo son usados para representar algoritmos pequeños, ya que abarcan mucho espacio y su construcción es laboriosa. Por su facilidad de lectura son usados como introducción a los algoritmos, descripción de un lenguaje y descripción de procesos a personas ajenas a la computación.

Símbolos más utilizados en los diagramas de flujo



Ejemplo de diagramas de flujo



Ejemplo de problema

Se desea obtener una tabla con las depreciaciones acumuladas y los valores reales de cada año, de un automóvil comprado por 20 000 dólares en el año 2005, durante los seis años siguientes suponiendo un valor de recuperación o rescate de 2000. Realizar el análisis del problema, conociendo la fórmula de la depreciación anual constante D para cada año de vida útil.

$$D = \frac{\text{costo} - \text{valor de recuperación}}{\text{vida útil}}$$

Entrada
- costo original
- vida útil
- valor de recuperación

$$D = \frac{20\,000 - 2\,000}{6} = \frac{18\,000}{6} = 3\,000$$

Salida
- depreciación acumulada en cada año
- depreciación anual por año
- valor del automóvil en cada año

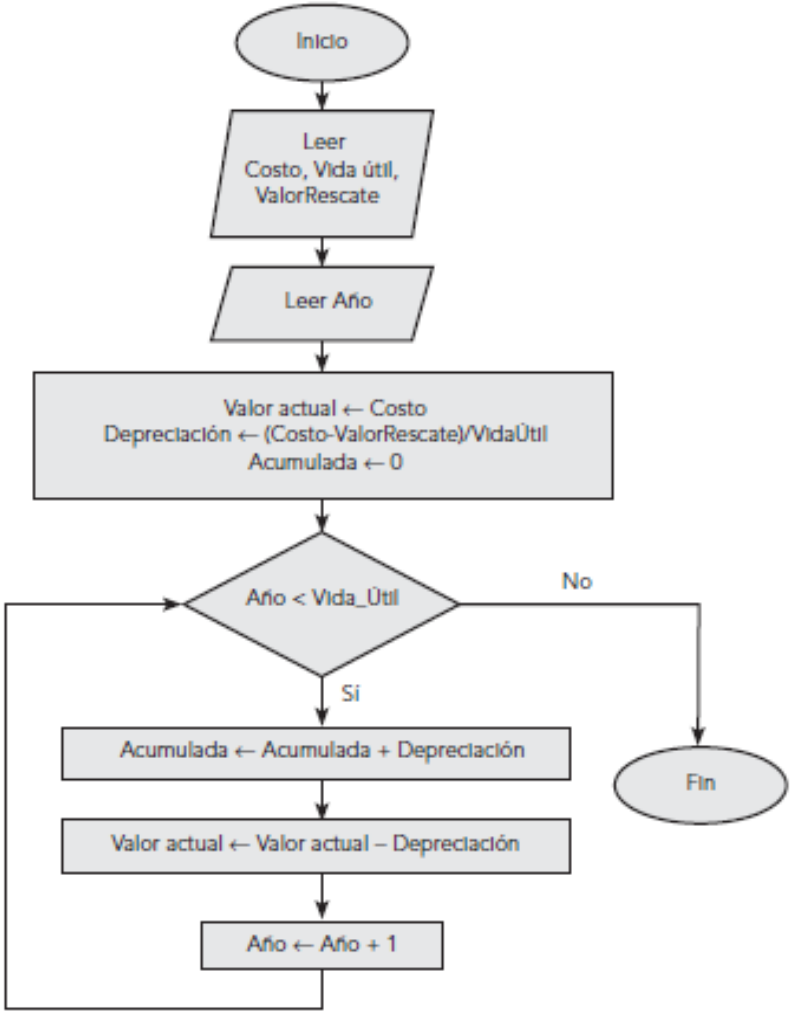
Tabla 2.1 Análisis del problema.

Año	Depreciación	Depreciación acumulada	Valor anual
1 (2006)	3 000	3 000	17 000
2 (2007)	3 000	6 000	14 000
3 (2008)	3 000	9 000	11 000
4 (2009)	3 000	12 000	8 000
5 (2010)	3 000	15 000	5 000
6 (2011)	3 000	18 000	2 000

Diagramas de flujo del problema

Pseudocódigo

```
Previsiones de depreciación
Introducir costo
    vida útil
    valor final de rescate (recuperación)
imprimir cabeceras
Establecer el valor inicial del año
Calcular depreciación
mientras año =< vida útil hacer
    imprimir una línea en la tabla
    incrementar el valor del año
fin de mientras
```



Diagramas de Nassi - Shneiderman

- Representación gráfica de la estructura secuencial o acción simple.

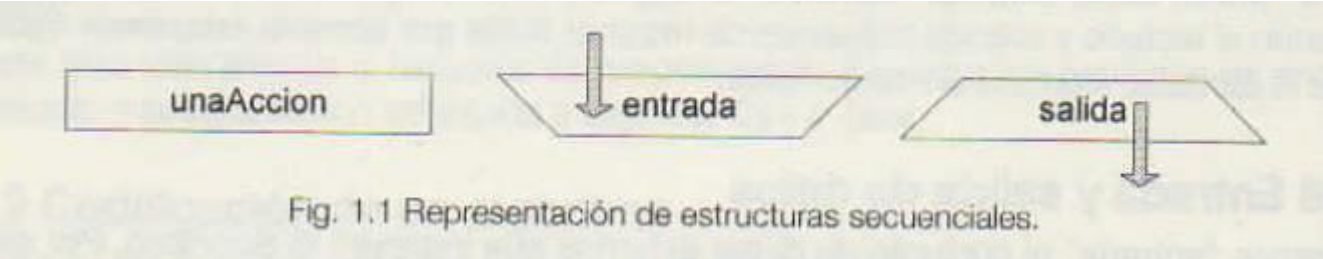


Fig. 1.1 Representación de estructuras secuenciales.

- Representación gráfica de la estructura de decisión.

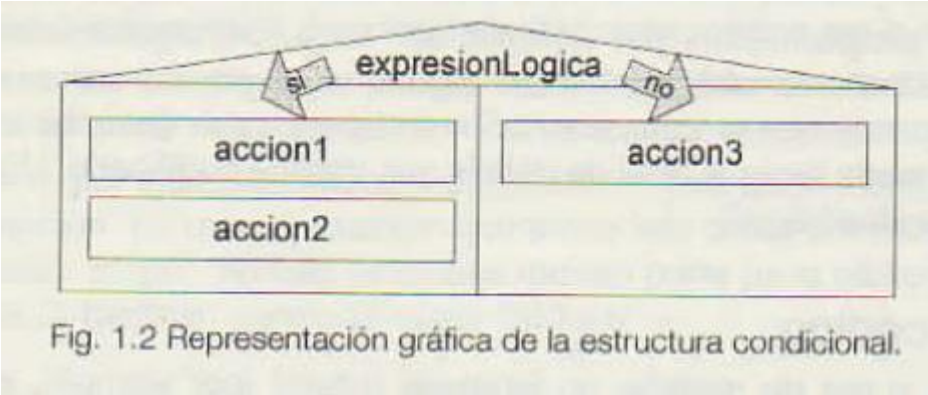


Fig. 1.2 Representación gráfica de la estructura condicional.

Diagramas de Nassi - Shneiderman

- Representación gráfica de la estructura de repetición.

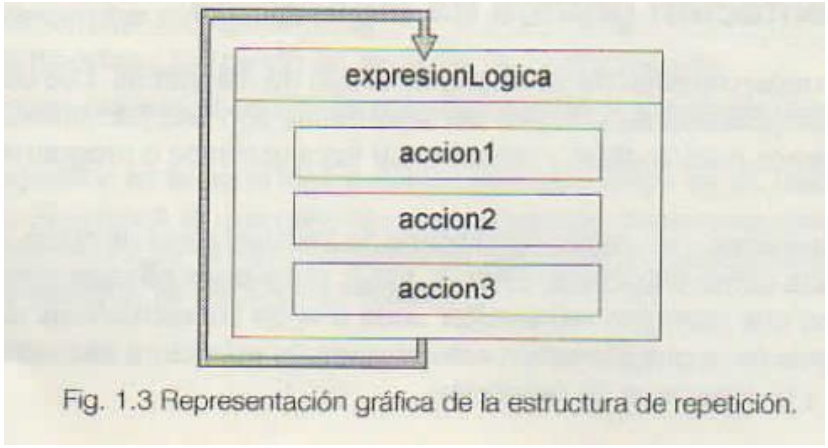


Fig. 1.3 Representación gráfica de la estructura de repetición.

- Representación gráfica de módulos o funciones.

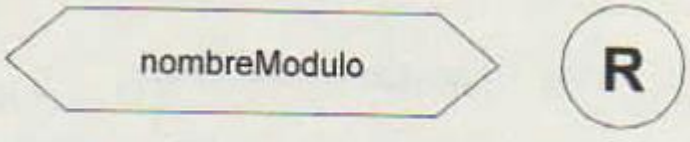


Fig. 1.5 Encabezado y finalización de un módulo.

- Módulo esperar semáforo

Fig. 1.6 Módulo `esperarSemaforo`.

- Diagrama del algoritmo principal

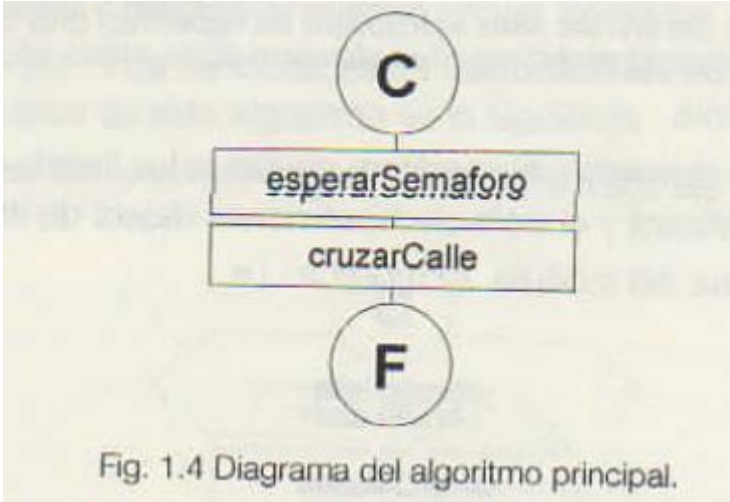


Fig. 1.4 Diagrama del algoritmo principal.

Algoritmo de comparación de dos valores numéricos

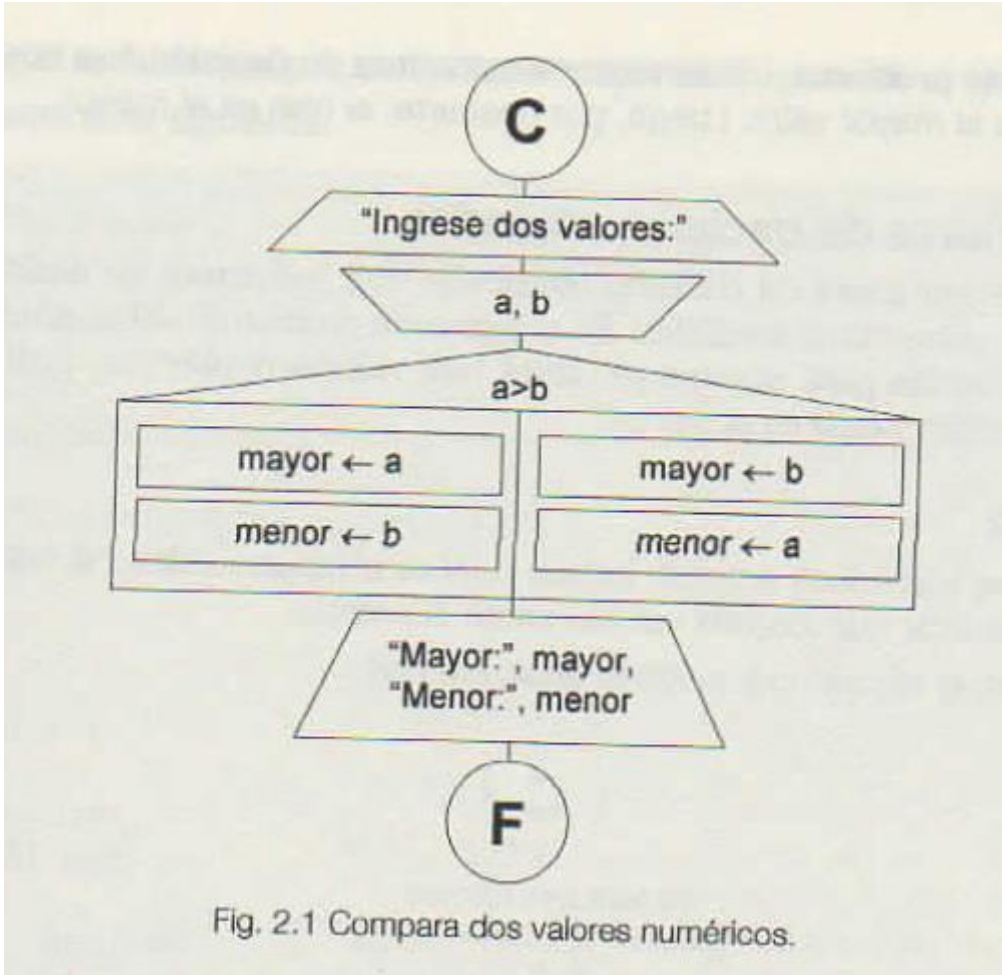


Fig. 2.1 Compara dos valores numéricos.

Código de comparación de dos valores numéricos

```
int main()
{
        int a, b;
        int mayor, menor;
        printf("Ingrese dos valores: ");
        scanf("%d %d", &a, &b);
        if( a > b )  {
                mayor = a;
                menor = b;
        }
        else  {
                mayor = b;
                menor = a;
        }

        printf("Mayor: %d\n", mayor);
        printf("Menor: %d\n", menor);
        return 0;
}
```