Índice de Contenidos

1.- Herramientas de programación.

Durante el diseño de un programa y en sus fases de análisis y programación surge la necesidad de utilizar una herramienta de diseño gráfico para la representación de los flujos de datos manipulados por el mismo, así como la secuencia lógica de las operaciones que constituyen el algoritmo de resolución del problema para el que ha sido creado.

Toda representación gráfica, debe cumplir las siguientes cualidades:

- Sencillez: han de ser fáciles y sencillos.
- Claridad: deben reconocerse todos los elementos empleados por cualquier persona distinta a la que lo diseño.
- Normalización: tanto los diseñadores como los usuarios de los programas deben utilizar las mismas normas de construcción.
- Flexibilidad: todo método de representación debe permitir sin grandes dificultades, posteriores modificaciones de algunas de las partes del algoritmo y la inserción de alguna nueva.

Para la representación de un algoritmo se debe utilizar algún método que permita independizar dicho algoritmo del lenguaje de programación elegido. Esto permitirá que un algoritmo pueda ser codificado indistintamente en cualquier lenguaje.

Los métodos más usuales para la representación de algoritmos son:

- Diagrama de flujo.
- Diagrama de N-S (Nassi-Schneiderman).
- Pseudocódigo.
- Tablas de decisión.

2.- Diagrama de flujo.

Los diagramas de flujo engloban tanto la representación gráfica de la circulación de los datos e información dentro de un programa (organigrama o diagrama de flujo de sistema), como a la representación gráfica de la secuencia de operaciones que se han de realizar dentro del mismo (ordinograma o diagrama de flujo de detalle).

Estas representaciones se corresponden con las distintas fases de un programa:

Organigrama ⟶ análisis
Ordinograma ⟶ programación

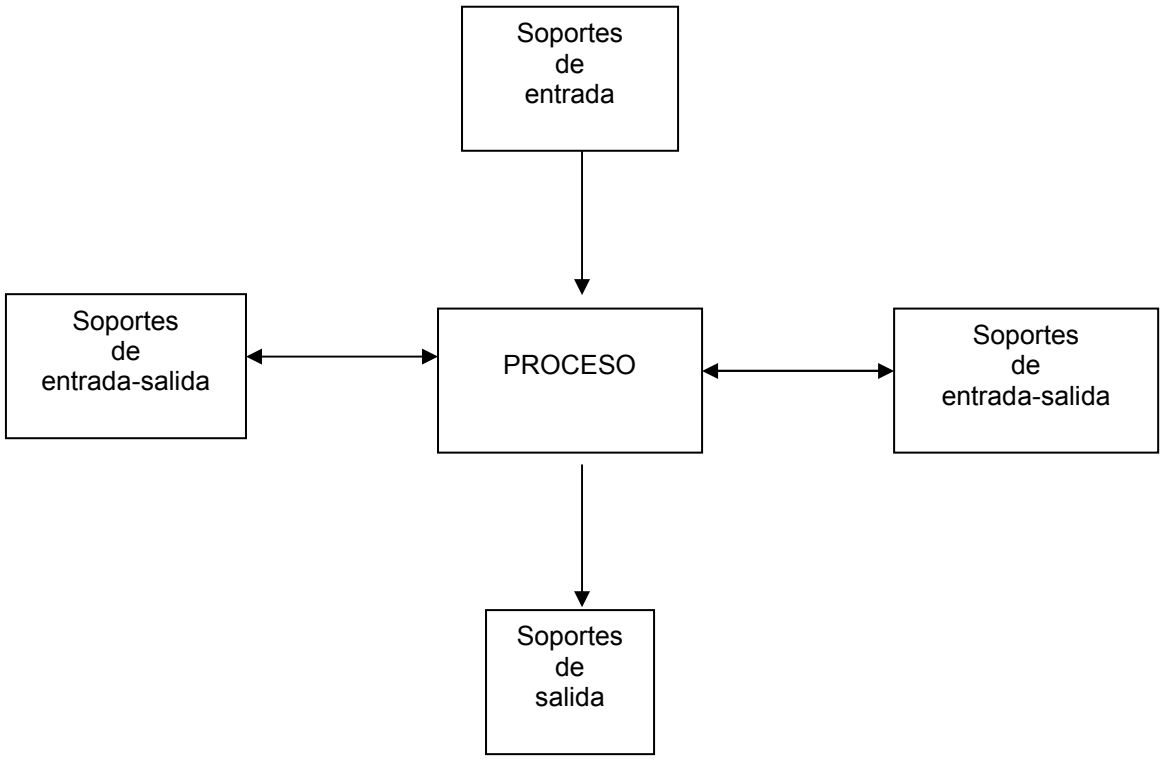
2.1.- Organigramas o diagramas de flujo del sistema.

Son representaciones gráficas del flujo de datos e informaciones que maneja un programa.
En general una aplicación se compone de más de un programa y es necesario realizar un organigrama por cada uno de ellos, siendo recomendable en estos casos la existencia de uno general que represente todo el movimiento de datos de la aplicación.
El conjunto del organigrama permitirá con facilidad la identificación de los siguientes elementos:

- Soportes en que se encuentran los datos.
- El programa y su identificación.
- Soportes donde se encuentran los resultados.
- El flujo de datos.

Para la representación de un organigrama se deben seguir las siguientes reglas:

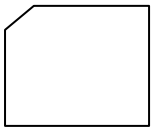
- En el centro figurará el símbolo del proceso. Que representa el programa.
- En la parte superior aparecerán los soportes que suministrarán los datos de entrada.
- En la parte inferior aparecerán los soportes que suministrarán los datos de salida.
- En las zonas de la derecha y de la izquierda aparecerán los soportes de los datos de entrada y salida.



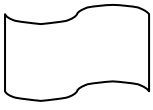
Los símbolos que se utilizan en la confección de organigramas se agrupan en tres bloques:

- Símbolos de soporte: Representan los soportes físicos donde se encuentran los datos de entrada y donde van a ser registrados los resultados.

Tarjeta perforada (E/S)

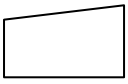


Cinta de papel (E/S)

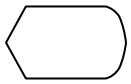


Impresora (S)

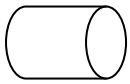
Teclado (E)



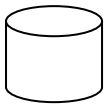
Pantalla (S)



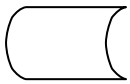
Tambor magnético (E/S)



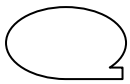
Disco magnético (E/S)



Soporte magnético (E/S)

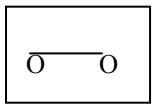


Cinta magnética (E/S)

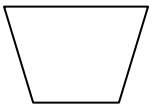


Disco Flexible (E/S)

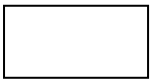
Cinta encapsulada (E/S)



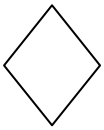
Soporte genérico (E)

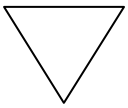


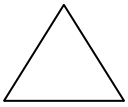
- Símbolos de proceso: Representan el programa o conjunto de operaciones que realizan un determinado trabajo completo.

Proceso 

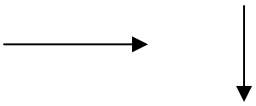
Operación auxiliar 

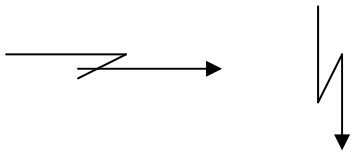
Clasificación de ficheros 

Fusión de ficheros 

Partición de ficheros 

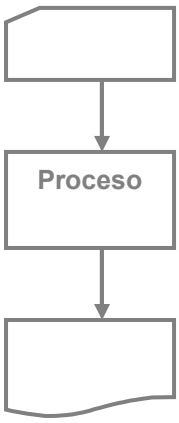
- Líneas de flujo: Indican el sentido del movimiento de los datos e informaciones y si se realiza dicho movimiento a corta o larga distancia.

Dirección del flujo 

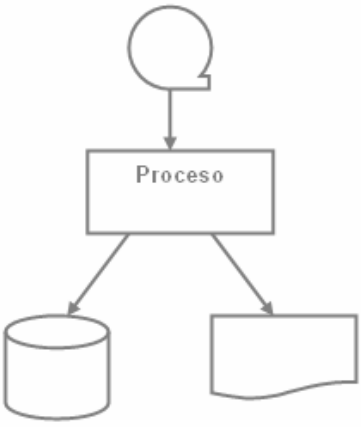
Líneas de teleproceso 

Veamos varios ejemplos de representación de algunos procesos generales mediante organigramas:

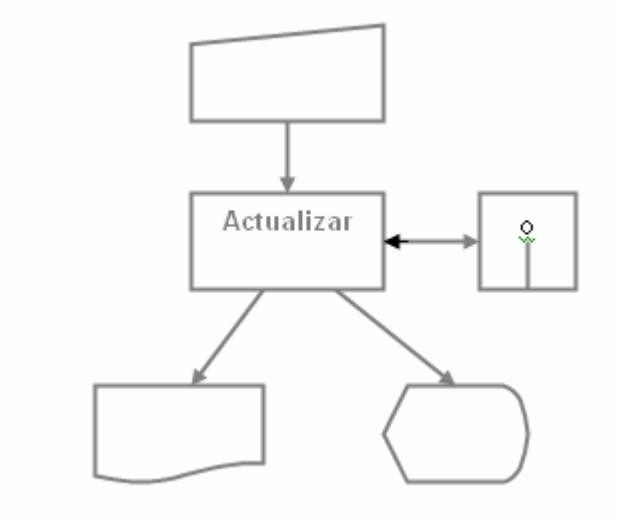
☑ El organigrama de un programa que toma datos de una ficha perforada, realiza unas operaciones y muestra los resultados por impresora sería el siguiente:



☑ **P**ara representar el diagrama de flujo de un programa que lee datos de un fichero en cinta, graba cierta información en el disco duro y produce una salida en la impresora, el organigrama sería:



☑ Programa que hace la actualización de un fichero con un disquete y la entrada de datos por teclado. También hace una consulta de datos por pantalla y se produce una salida por impresora con los datos modificados.



Práctica propuesta

Práctica 3-1.

Realizar el organigrama de un programa que lee datos de un fichero en cinta magnética, graba cierta información en un disco y produce un resultado impreso. El fichero en disco creado en el proceso anterior es leído en otro programa actualizando si procede otro fichero en disco y produciendo un listado de errores.

2.2.- Ordinogramas o diagramas de flujo de detalle.

Son representaciones gráficas de la secuencia lógica de las operaciones que se han de realizar para la resolución de un problema por medio del ordenador.
En la fase de programación, el programador crea para cada programa un ordinograma, a partir del cual realiza la codificación en el correspondiente lenguaje de programación.
Un ordinograma que representa un programa debe reflejar con claridad algunos de los elementos esenciales del mismo:

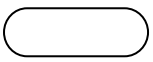
- Comienzo del programa.
- Operaciones.
- Secuencia en que se realizan.
- Final del programa.

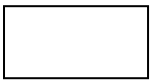
En esta técnica de representación es aconsejable seguir las siguientes recomendaciones:

- El comienzo del programa figurará en la parte superior.
- El flujo de las operaciones irá siempre que sea posible, de arriba abajo y de izquierda a derecha (en cuyo caso se pueden omitir las puntas de flecha).
- El final del programa figurará en la parte inferior.
- Se debe guardar simetría y equilibrio en la composición del conjunto del ordinograma.
- Aunque se permiten, se evitarán siempre en la medida de lo posible los cruces de las líneas de flujo utilizando conectores.
- A un reagrupamiento de líneas de flujo pueden llegar varias de ellas pero solo puede salir una.

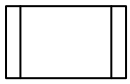
Los símbolos utilizados en la confección de Ordinogramas son los siguientes:

- **Símbolos de operación:**

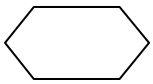
Inicio o Fin 

Operación general 

Operación de E/S general 

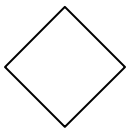
Subprograma 

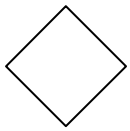
Modificación de instrucción o inicializaciones



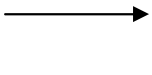
Operación manual 

- **Símbolos de decisión:** Se utilizan para el establecimiento de bifurcaciones o la construcción de estructuras en las que se evalúa una expresión lógica o múltiple, derivándose la secuencia lógica de ejecución de las operaciones entre varios caminos posibles.

Decisión 

Decisión múltiple 

- **Líneas de flujo:** Indican la secuencia lógica de ejecución de las operaciones desde el Inicio al Fin.

Dirección del flujo  

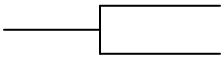
- **Símbolos de conexión**: Se utilizan para la unión de líneas de flujo en los casos de reagrupamiento y de conexión o continuación entre otra parte por cualquier motivo, (cambio de hoja, cruce de líneas).

Reagrupamiento 

Conector 

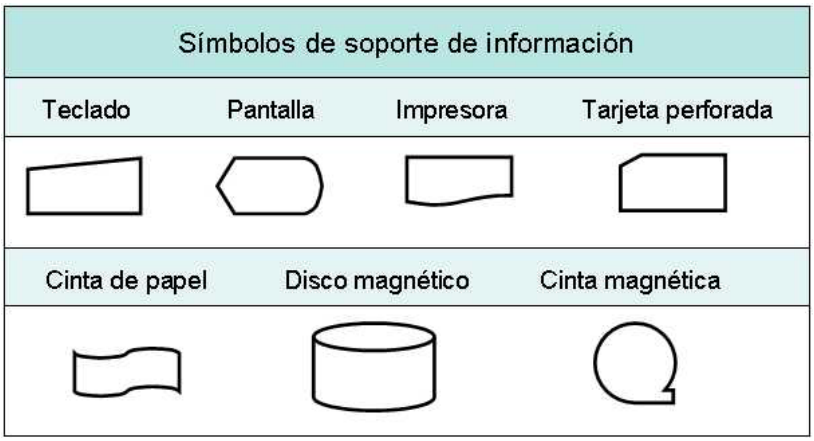
Conector a distinta página 

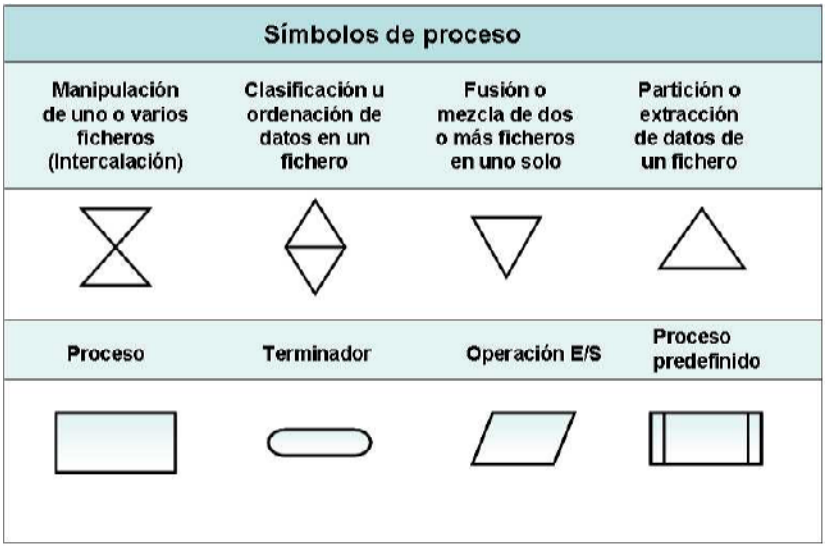
- **Símbolo de comentario:** Se utiliza para aclarar o documentar el diseño del algoritmo con algún comentario que se considere necesario.

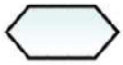
  Comentario 

Resulta necesario indicar dentro de los símbolos la operación específica concebida por el programa.

A modo de resumen, se presenta la siguiente "guía rápida" de símbolos de soporte, proceso, decisión y conexión:





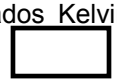
Símbolos de decisión		Líneas de flujo	
Decisión	Bucle	Flechas	Línea conectora
			

Símbolos de conexión			Símbolos infor.
Conector	Conector misma página	Conector distintas páginas	Comentarios
			

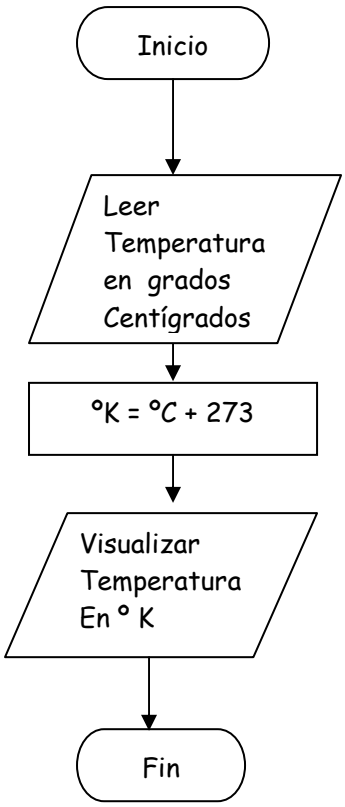
Práctica resuelta

Representar mediante un ordinograma la secuencia de pasos necesaria para que un programa lea una temperatura en grados Centígrados y calcule y escriba en el dispositivo estándar de salida su valor en grados Kelvin.

Análisis del problema

Utilizamos el símbolo de Operación de E/S en general  para **Leer** la temperatura en grados Centígrados. El siguiente paso a realizar es el del cálculo de la transformación de esa temperatura a Grados Kelvin. Esto lo representaremos mediante el símbolo de proceso u operación general  . A continuación se **visualiza** el resultado del cálculo realizado en el paso anterior y, para ello utilizamos, de nuevo, el símbolo de Operación de E/S en general  .

La solución propuesta será la siguiente



Práctica resuelta

Representar mediante un ordinograma el algoritmo que lea dos números por teclado y nos diga por pantalla si son iguales

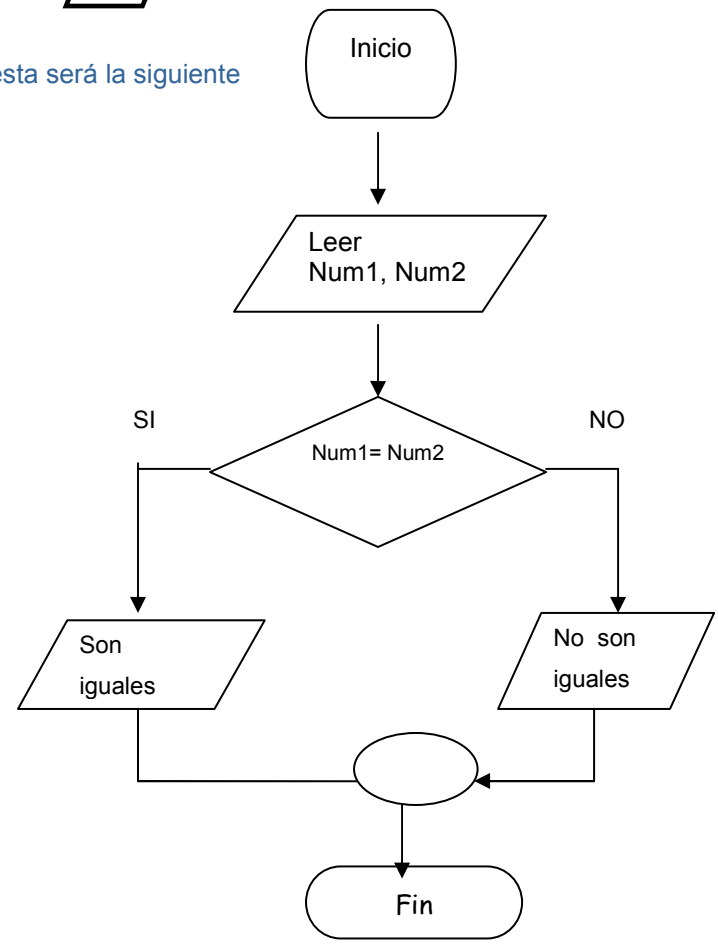
Análisis del problema

Comienza el programa con el símbolo de Inicio . Lo primero que necesitamos es suministrarle al ordenador los dos números con los que queremos trabajar. Éstos serán los datos de entrada y representaremos esta operación mediante el símbolo . Identificaremos a las variables con los nombres Num1 y Num2.

El siguiente paso consiste en comparar el contenidos de las variables, y esto se realiza mediante el símbolo de decisión que nos permite preguntar si Num1 *es igual a* Num2.

Si la repuesta es SI se debe visualizar que los números son iguales. En el caso de que la espuesta sea NO se mostrará el mensaje contrario. El símbolo que utilizaremos para dar a Salida al mensaje será .

La solución propuesta será la siguiente



Prácticas propuestas

Práctica 3-2.

Representar mediante un ordinograma el algoritmo de un programa que lea dos números por teclado y, nos dé como resultado la suma y la resta en pantalla.

Práctica 3-3.

Representar mediante un ordinograma el algoritmo que lea dos números por teclado y nos diga por pantalla si son iguales y si no lo son que nos diga cual es el mayor.

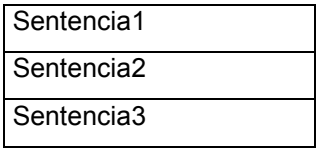
Práctica 3-4.

Hacer el ordinograma de un programa que lee un número del por teclado y comprueba e imprime en los dispositivos estándar de salida si dicho número es nulo.

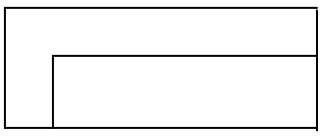
3.- Diagrama de Nassi-Schneiderman.

Los diagramas Nassi-Schneiderman, denominados así por sus inventores, o también N-S, o de Chapin, son una herramienta de programación que favorece la programación estructurada y reúne características gráficas propias de diagramas de flujo y lingüísticas propias de los pseudocódigos. Constan de una serie de cajas contiguas que se leerán siempre de arriba-abajo y se documentarán de la forma adecuada.

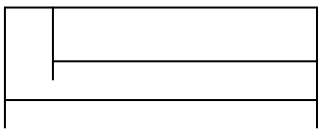
En los diagramas N-S las tres estructuras básicas de la programación estructurada, secuenciales, selectivas y repetitivas, encuentran su representación propia.



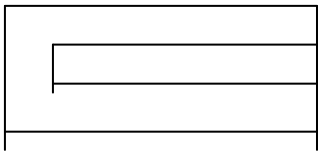
Secuencial



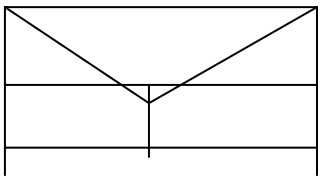
Repetitiva de 0 a n veces



Repetitiva de 1 a n veces



Repetitiva n veces



Selectiva

Pr ejemplo, el algoritmo que lee una temperatura en grados Centígrados y calcula y escribe su valor en grados Kelvin se puede representar mediante el siguiente diagrama N-S.

inicio
leer (Centígrados)
Kelvin ← Centígrados + 273.15
escribir (kelvin)
fin

4.-Tablas de decisión.

Se denomina **tabla de decisión** a una representación tabular de la lógica de un problema, en el que se presentan varias situaciones alternativas para cada una de ellas.
Aparecieron a finales de los años cincuenta y su utilización fue muy extensa en la década de los sesenta.. Una tabla de decisión, es por tanto:

- Una herramienta para el análisis de problemas.
- Un elemento de comunicación dentro de la jerarquía informática.
- Una representación de problemas que facilita la codificación de los mismos.
- Un instrumento que facilita la detección de errores u omisiones.

Actualmente su uso ha quedado bastante reducido, por lo que solamente mostraremos la estrucutura de una tabla de decisión, por considerar su estudio interesante.

4.1.- Estructura de una tabla de decisión.

Una tabla de decisión es una representación en la que se distinguen cuatro zonas:

- **Condiciones.** Consta de un vector columna donde figuran las condiciones que intervienen en el problema.
- **Acciones.** Consta de un vector columna en el que aparecen las acciones a realizar. Si en algún caso, para un estado determinado de las condiciones, se realizan varias acciones y éstas se tienen que ejecutar en un orden preestablecido, figurarán en ese orden, de arriba abajo.
- **Entrada de condiciones.** Es una matriz de tantas filas como condiciones y columnas como situaciones distintas se pueden presentar.
- **Salida de acciones.** Matriz en la que figuran tantas filas como acciones y columnas como situaciones distintas se pueden presentar.

CONDICIONES	ENTRADA DE CONDICIONES
ACCIONES	SALIDA DE ACCIONES

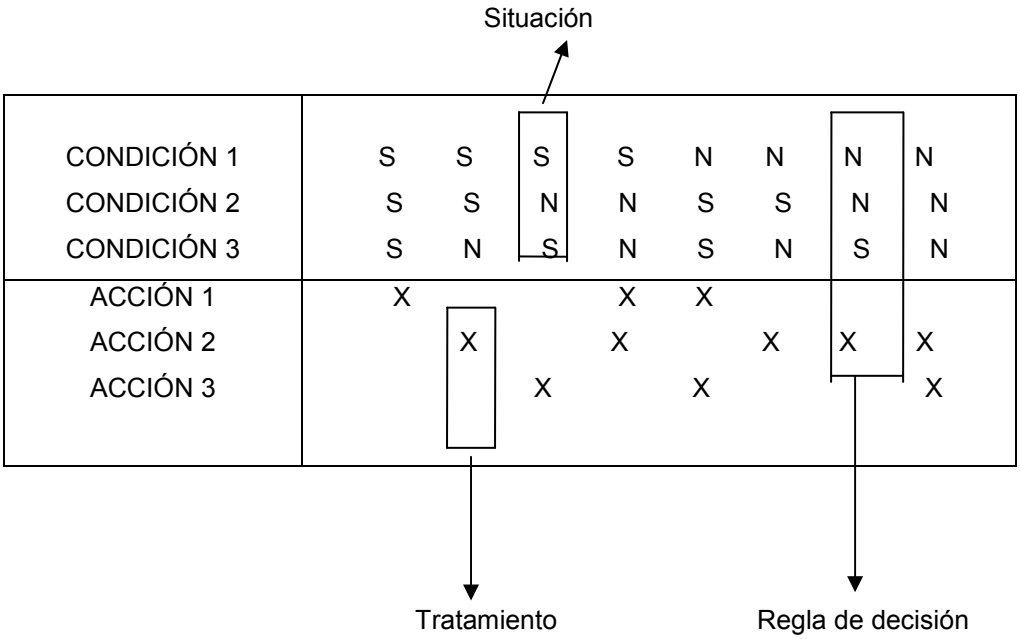
Por ejemplo, supongamos que tenemos tres condiciones que se evalúan con “se cumple” (S) o “no se cumple” (N) y tres acciones que se ejecutarán según la siguiente representación:

CONDICIÓN 1	S	S	S	S	N	N	N	N
CONDICIÓN 2	S	S	N	N	S	S	N	N
CONDICIÓN 3	S	N	S	N	S	N	S	N
ACCIÓN 1	X			X	X			
ACCIÓN 2		X		X		X	X	X
ACCIÓN 3			X		X			X

Cada columna de la matriz que representa la entrada de condiciones, correspondiente a un determinado estado de las mismas, se denomina **situación**.

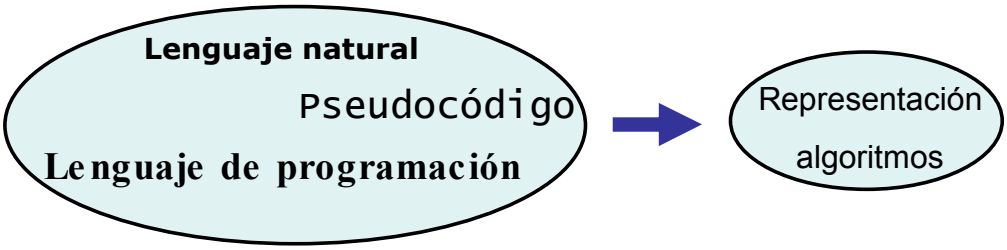
Cada columna de la matriz que representa la salida de acciones se denomina **tratamiento**.

Definimos como **regla de decisión** el conjunto formado por una situación y su tratamiento correspondiente.



5.- Pseudocódigo.

Es una técnica utilizada para la descripción de un algoritmo utilizando un lenguaje intermedio entre el lenguaje natural y el lenguaje de programación. Esta notación se encuentra sujeta a unas determinadas reglas que nos permiten y facilitan el diseño de algoritmos. **Es la técnica que utilizaremos a partir de ahora en todos nuestros algoritmos.**



La escritura o diseño de un algoritmo mediante el uso de esta herramienta, exige la "identación" o "sangria" del texto en el margen izquierdo de las diferentes líneas, lo que facilita el entendimiento y comprensión del diseño realizado.

El pseudocódigo se concibió para superar las dos principales desventajas del diagrama de flujo: lento de crear y difícil de modificar sin un nuevo redibujo. Es una herramienta muy buena para el seguimiento de la lógica de un algoritmo y para transformar con facilidad los algoritmos a programas escritos en un lenguaje de programación específico.

Todo algoritmo representado en pseudocódigo deberá reflejar las siguientes partes:

- Cabecera: Es el área o bloque informativo donde quedará reflejado el nombre del programa y el nombre del algoritmo al que pertenece dicho diseño.
- Cuerpo: Es el resto del diseño, el cual queda dividido en dos bloques, el bloque de datos, donde deberán quedar descritos todos los elementos de trabajo necesarios para la ejecución del programa, y el bloque de acciones que es la zona en la que se deben describir con máxima claridad y detalle todas aquellas acciones que el ordenador deberá realizar durante la ejecución del programa.

Cabecera

Programa: Nombre del programa
Módulo: Nombre del módulo

Cuerpo

Entorno:
Descripción de los datos
Algoritmo:
Inicio
Descripción de las acciones

Fin-programa

En la escritura de algoritmos se hará necesario el uso de alguna de las herramientas de programación. La más adecuada es el pseudocódigo, y debe quedar todo lo más claro posible, de modo que se facilite al máximo su posterior codificación en un lenguaje de programación.

Aunque en Unidad 4 explicaremos con más detalle la estructura general de todo programa, haremos ahora
una primera aproximación a la resolución de algoritmos mediante pseudocódigos. El primer paso para encontrar la solución de un problema es el análisis del mismo. Se debe examinar cuidadosamente, a fin de obtener una idea clara sobre lo que se solicita y determinar los datos necesarios para conseguirlo. El algoritmo deberá ser una secuencia ordenada de pasos, sin ambigüedades, que nos conduzcan a la solución del problema planteado y expresado en lenguaje natural mediante el pseudocódigo. En el algoritmo se deben considerar tres partes:

Entrada: Información dada al algoritmo

Proceso: Operaciones o cálculos necesarios para encontrar la solución del problema.

Salida: Respuestas dadas por el algoritmo o resultados finales de los cálculos.

A modo de modo ejemplo, veamos la siguiente Práctica resuelta y comentada:

Práctica resuelta

Realizar el pseudocódigo de un programa que permita calcular el área de un rectángulo, introduciremos por teclado el valor de la base y de la altura. Etiquetar tanto la entrada de datos como la salida de resultados.

Análisis del problema

Lo primero que se de be hacer es plantearse y contestar a las siguientes preguntas:

Especificaciones de entrada

¿Qué datos son de entrada?
¿Cuántos datos se introducirán?
¿Cuántos son datos de entrada válidos?

Especificaciones de salida

¿Cuáles son los datos de salida?
¿Cuántos datos de salida se producirán?
¿Qué precisión tendrán los resultados?
¿Se debe imprimir una cabecera o etiquetar los datos?

El algoritmo en el primer diseño se podrá representar con los siguientes pasos:

Paso 1.- Entrada desde periférico de entrada, por ejemplo teclado, (así nos lo indica el enunciado) de base y altura.
Paso 2. Cálculo de la superficie, multiplicando base por la altura.
Paso 3. Salida por pantalla de area.

El pseudocódigo propuesto del algoritmo que nos da solución al algoritmo del problema planteado es el que sigue:

Programa Cálculo del Área de un rectángulo

Entorno

base, altura, area: numéricos reales

En el **Entorno** definimos tres variables a las que asignamos los nombres base, altura y área. Las dos primeras son dos datos de entrada que suministramos a través del teclado. La variable area es un dato de salida cuyo valor será el resultado del cálculo del área.

Algoritmo

Inicio

Visualizar "Introduce la base"

Leer base

Visualizar "Introduce la altura "

Leer altura

area ← base * altura

Visualizar "El área es", area

Finprograma

Entre comillas escribimos el texto aclaratorio que queremos que se muestre en pantalla. Éste indica que estamos pidiendo que introduzca la base del rectángulo. El mensaje va acompañado de la instrucción **Visualizar** que luego traduciremos al comando que corresponda según el lenguaje de programación que utilicemos.

Los nombres de las variables en las que recogemos los datos que pedimos como entrada, se escriben tal y como las definimos anteriormente en el **Entorno**. En esta sentencia va acompañada de la instrucción **Leer** que luego traduciremos al comando que corresponda según el lenguaje de programación que utilicemos.

Se ejecuta la operación aritmética base *altura y el resultado se guarda en la variable area

Prácticas propuestas

Práctica 3-5.

Realizar el pseudocódigo de un algoritmo que resuelva un programa que lea dos números por teclado y, nos dé como resultado la suma y la resta en pantalla. Visualizar en pantalla mensajes aclaratorios de los datos que se están pidiendo o mostrando.

Práctica 3-6.

Realizar el pseudocódigo del algoritmo de un programa que permita calcular el cuadrado de un número que se introduce por teclado. Visualizar en pantalla mensajes aclaratorios de los datos que se están pidiendo o mostrando.

Objetivos

La resolución de problemas con el ordenador. Herramientas de programación. El pseudocódigo, tiene como finalidad que el alumno aprenda los conceptos, técnicas y habilidades que le posibiliten desarrollar buenos programas. Los objetivos que se persiguen son lo siguientes:

- Entender, analizar y resolver una amplia variedad de problemas computacionales Aprender una metodología de la programación.
- Dominar los métodos y técnicas que permitan desarrollar buenos programas como resultado de la aplicación de un proceso metódico.
- Conocer el concepto de Algoritmo y su importancia en el mundo de las aplicaciones para ordenadores.
- Utilizar las herramientas de diseños de algoritmos.
- Conseguir la destreza suficiente para la interpretación de problemas y el diseño de algoritmos que los resuelvan.

Enlaces de interés

☞ En el siguiente enlace podrás profundizar en la importancia de saber resolver problemas y aprender a razonar.

Aprender a resolver problemas razonando.
http://educacion.idoneos.com/index.php/345898

☞ Si quieres ver otro motor de búsqueda, en este caso de recuperación de la información para uso experimental y educativo, visita el siguiente enlace:

Recuperación de la información en la educación.
http://www.ub.es/biblio/bid/04figue2.htm

☞ Enlace en el que podamos descargar alguna aplicación gratuita para hacer diagramas de flujo. Puedes encontrar varias opciones para Windows**.**

Programas de para elaborar Diagramas de Flujo.
http://descargas.terra.es/index.phtml?modo=2&n_id=1659

☞ Para entornos Linux tienes otras opciones, entre las que destacamos el enlace a DIA, un generador de diagramas de flujo muy útil. En esta página puedes encontrar una breve descripción y algunos enlaces interesantes:

Diagramas de Flujo en Linux.
http://bulma.net/body.phtml?nIdNoticia=1349

☞ Algunos algoritmos resueltos que pueden ayudarte a entender la resolución de problemas y el diseño de algoritmos. En el siguiente enlace vas a encontrar un resumen del tema de algoritmos, con ejemplos de elaboración y al final unos cuantos ejemplos resueltos.

Teoría de Algoritmos.
http://www.geocities.com/inf135/tutc/Tema02.htm

Glosario de términos

Análisis léxico

En la fase de análisis léxico se leen los caracteres del programa fuente y se agrupan en cadenas que representan los componentes léxicos. Cada componente léxico es una secuencia lógicamente coherente de caracteres relativa a un identificador, una palabra reservada, un operador o un carácter de puntuación. A la secuencia de caracteres que representa un componente léxico se le llama lexema (o con su nombre en inglés token).

Argumentos

Se distingue entre argumentos formales y argumentos actuales. Los argumentos formales son los que aparecen en la definición de la función, mientras que los actuales son los que aparecen en la llamada a la función. Se podría también decir que los argumentos formales son los argumentos vistos desde dentro de la función, y los argumentos actuales son los argumentos vistos desde el programa que llama a la función, esto es desde fuera de la función. Los argumentos formales son siempre variables de la función que recogen los valores que se les pasan desde el exterior. Los argumentos actuales pueden ser variables o expresiones cuyos valores se pasan a los argumentos formales.

Bit

Un bit es la unidad mínima de información empleada en informática. Representa un uno o un cero (abierto o cerrado, verdadero o falso, cualquier sistema de codificación sirve). A través de secuencias de bits, se puede codificar cualquier valor discreto como, por ejemplo, números, palabras e imágenes.

Casting

Es el proceso de convertir o promover un objeto de un tipo a otro. Esta operación es necesaria y frecuente en programación; incluso es realizada infinidad de veces por el compilador de forma automática.

Comentario

Información adicional añadida al programa, usada para una introducción del código y proporcionar información adicional que no está disponible en el propio código. Los comentarios sólo deben tener información que sea relevante para leer y entender el programa.

Evaluar una expresión

Sustituir las variables, constantes y demás identificadores por sus valores, y realizar las operaciones indicadas en la expresión para obtener un valor resultado

Expresiones

Son uniones de literales, junto con identificadores y los operadores que los unen, formando unidades de mayor significado. Las expresiones pueden evaluarse, dando como resultado un valor determinado. La expresión tendrá asociado un tipo.

Fuertemente tipado

Un lenguaje de programación es fuertemente tipado cuando te obliga a indicar siempre el tipo de las variables que se usan en un programa antes de usarlas y además no se puede operar para una variable con datos de distinto tipo al que figura en su declaración. Si la variable es de tipo entero, por ejemplo, no podré asignarle un valor real.

Función

Es una rutina de software independiente que realiza una tarea para el programa en que está escrita o para algún otro programa. La función ejecuta la operación y devuelve el control a la instrucción siguiente a la que la llamó o al programa que la llamó. Los lenguajes de programación proveen un conjunto de funciones estándares y permiten a los programadores definir otras.

Identificadores

Nombre que se le asigna a cada elemento que se define en un programa (variables, constantes, clases, métodos o procedimientos, etc.) y mediante el cual podemos identificar y referirnos a él de forma inequívoca.

Literales

Son valores concretos para un tipo de datos básico del lenguaje.

Operadores

Son tokens especiales porque actúan de separadores, manteniéndose a sí mismos como tokens, con su significado propio. Expresan acciones a realizar con los datos o expresiones que les acompañan.

Palabras reservadas

Son aquellos tokens que tienen asignada una función específica en el lenguaje y que los programadores no pueden utilizar más que para lo que el lenguaje establece.

Recurrente o recursiva

Hace referencia a un proceso que se llama a sí mismo, ejecutándose varias copias simultáneamente (concurrentemente) o bien una tras otra.

Registros del microprocesador

Es una pequeña unidad de almacenamiento destinada a contener cierto tipo de datos. Puede estar en la propia memoria central o en unidades de memoria de acceso rápido.

La misión de estos registros es almacenar las posiciones de memoria que van a experimentar repetidas manipulaciones, ya que los accesos a memoria son mucho más lentos que los accesos a los registros. Además, hay ciertas operaciones que sólo se pueden realizar sobre los registros. No todos los registros sirven para almacenar datos, algunos están especializados en apuntar a las direcciones de memoria. La mecánica básica de funcionamiento de un programa consiste en cargar los registros con datos de la memoria o de un puerto de E/S, procesar los datos y devolver el resultado a la memoria o a otro puerto de E/S. Obviamente, si un dato sólo va a experimentar un cambio, es preferible realizar la operación directamente sobre la memoria, si ello es posible.

Un registro del procesador es algo así como una zona de memoria dentro del procesador donde se puede almacenar información, de forma que el acceso a esta información es instantáneo, ya que no hay que utilizar el bus de datos que conecta el procesador con la memoria para obtener dichos datos.

Sentencia

Es una unión de una serie de expresiones que representa una acción completa a realizar por el programa.

Separador

Carácter, símbolo o palabra escrita en el texto de un programa, que indica al compilador dónde termina una palabra y dónde empieza la siguiente.

Token

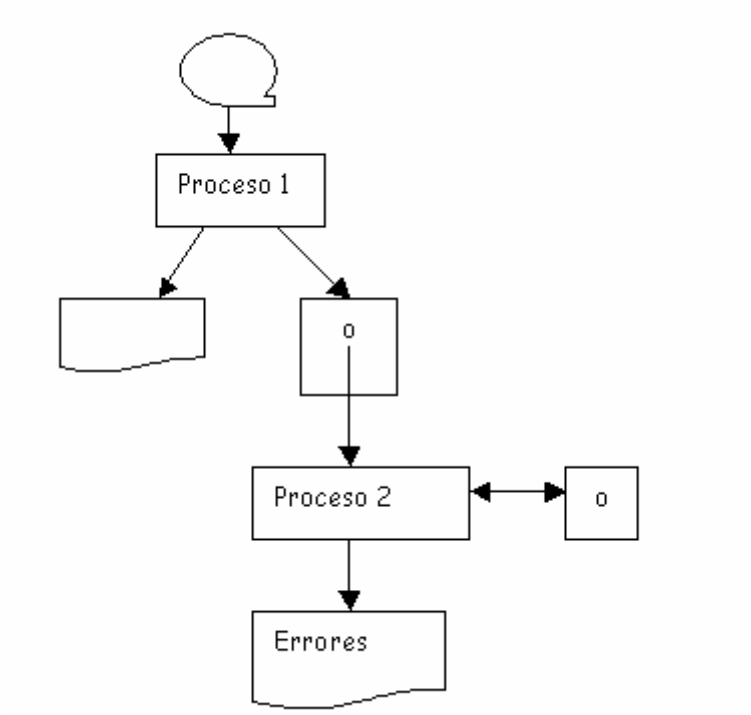
Cada una de las “palabras” (lexemas) que reconoce el compilador.

Valor de verdad

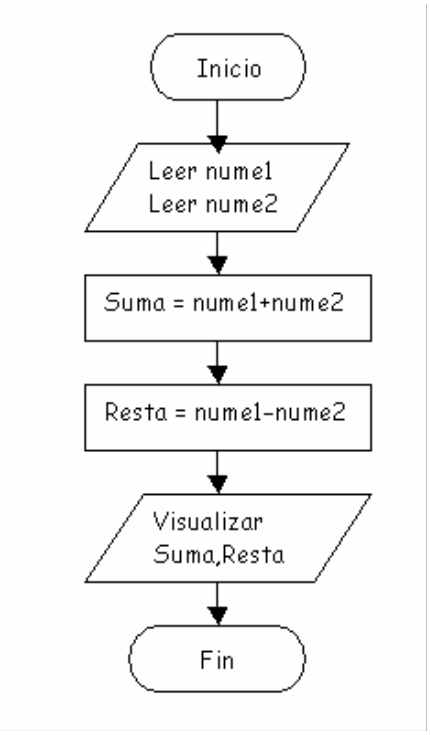
Es un valor que indica en qué medida una declaración es verdad. En informática, los únicos valores de verdad posibles son verdadero y falso.

Soluciones a las prácticas propuestas

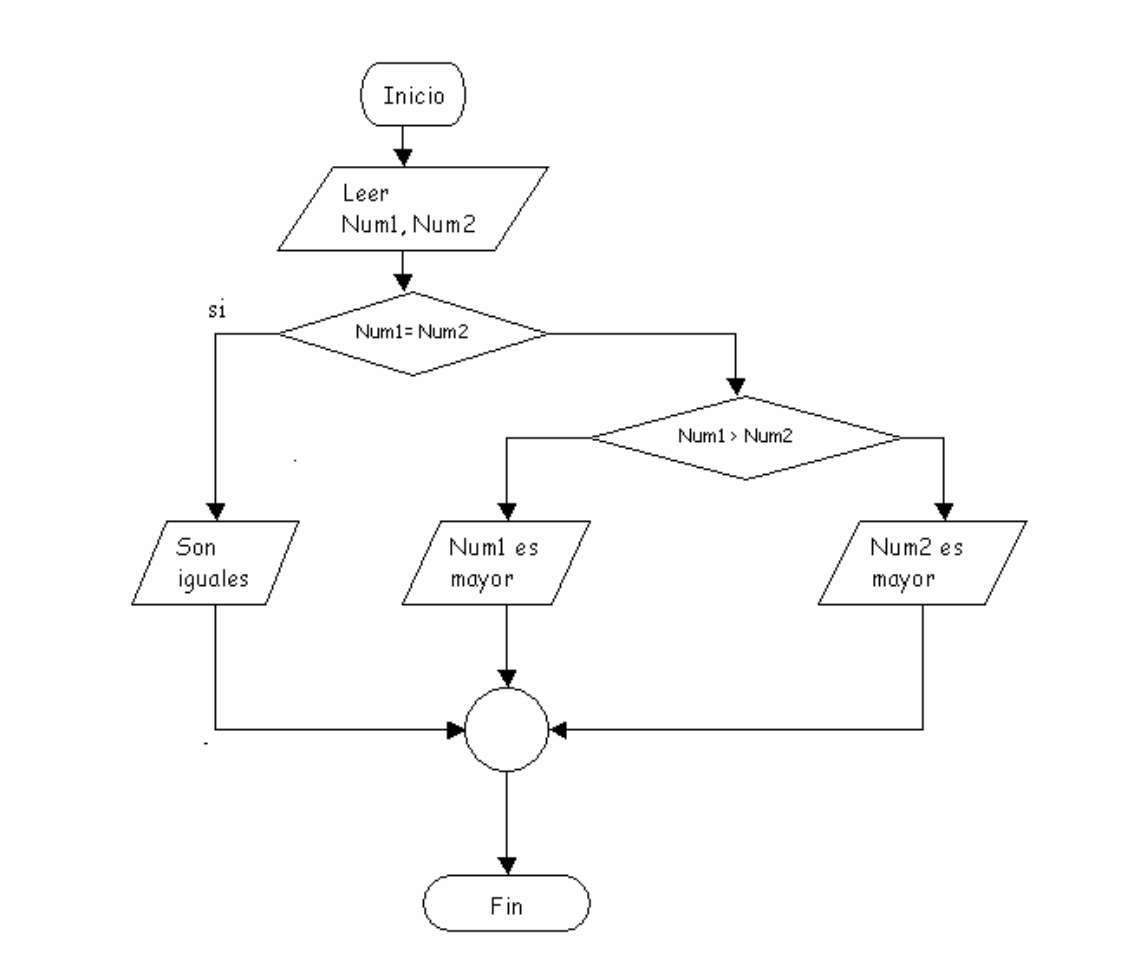
Solucion Práctica 3-1



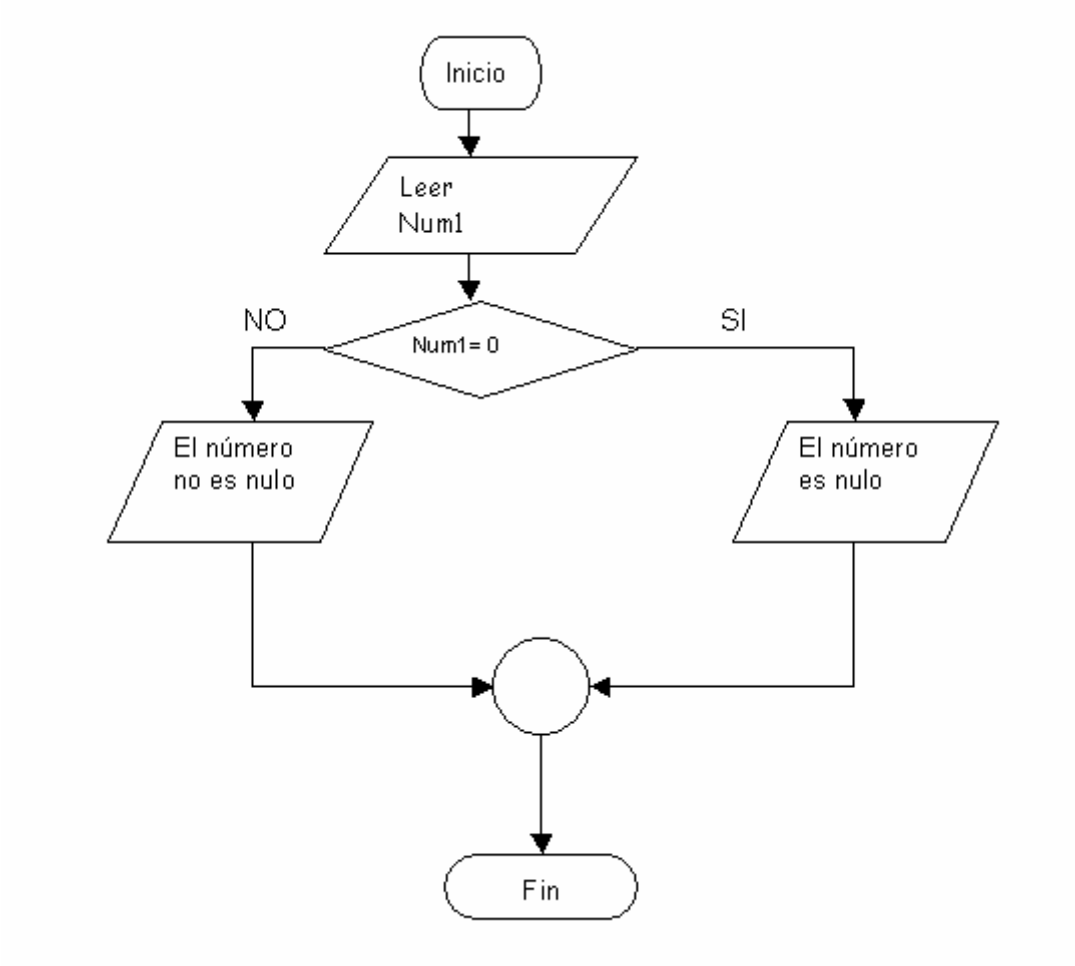
Solución Práctica 3-2.



Solución Práctica 3-3



Solución Práctica 3-4



Solución práctica 3_5

Programa Cálculo del la Suma y de la Resta de dos números

Entorno

num1, num2, suma, resta: numéricos enteros

Algoritmo

Inicio

Visualizar “Introduce el primer número”

Leer num1

Visualizar “Introduce el segundo numero“

Leer num2

suma ← num1 + num2

resta ← num1 – num2

Visualizar “La suma es”, suma

Visualizar “La resta es” , resta

Finprograma

Solución práctica 3_6

Programa Cálculo del cuadrado de un número

Entorno

num1, cuadrado: numéricos enteros

Algoritmo

Inicio

Visualizar “Introduce el número”

Leer num1

cuadrado ← num1 * num1

Visualizar “El cuadrado de ”, num1, “es”, cuadrado

Finprograma