
DISEÑO
ESTRUCTURADOS
DE
ALGORITMOS

OBJETIVO GENERAL

AL FINAL DEL CURSO, EL PARTICIPANTE DISEÑARÁ ALGORITMOS MEDIANTE EJERCICIOS PRÁCTICOS CON AYUDA DE LAS DIFERENTES TÉCNICAS ALGORÍTMICAS, CON LA FINALIDAD DE FORMARSE UNA MENTALIDAD DE PROGRAMADOR.

ÍNDICE

INTRODUCCIÓN GENERAL

Todos tenemos conciencia de que el éxito de una empresa depende de la rapidez, calidad, control de los recursos, exactitud y otros muchos factores.

Hace tiempo, las empresas ya sean grandes o pequeñas, tenían que hacer sus procesos manualmente o con ayuda de maquinas. Pero a raíz de la aparición de las primeras computadoras, las macroempresas obtuvieron unas de estas y comenzaron a tener mayor ventaja sobre las demás organizaciones. Con el paso del tiempo, se crearon computadoras más pequeñas, de menos costo, más rápidas, lo cual ha provocado que cualquier persona o empresa pueda adquirir una o más de estas computadoras.

En la actualidad, muchas empresas realizan sus operaciones por medio de computadoras, por ejemplo en las fábricas ensambladoras de autos se utilizan robots programados, los cuales se encargan de montar y soldar las partes que forman el carro; en los supermercados, se utilizan las computadoras junto con un programa para registrar rápidamente las compras de los clientes, además de que les ayuda para llevar el control de su inventario y de sus ingresos entre otras cosas; en los hospitales, se están utilizando pequeños robots programados, los cuales se introducen en el cuerpo del paciente para realizar incisiones, cauterizar, saturar, etc.; este manual, fue elaborado en un editor de textos llamado Microsoft Word, el cual es un programa de aplicación diseñado específicamente para poder crear y dar formato a documentos de texto. En fin, podríamos continuar enumerando en donde se utilizan las computadoras y nunca terminaríamos.

Sin embargo y afortunadamente, no todas las empresas cuentan con programas o sistemas para llevar el control de sus actividades y aunque todas las compañías ya contaran con sistemas informáticos, estas necesitan quien se encargue de darles mantenimiento, lo cual nos da un amplio campo de trabajo a nosotros que pretendemos ser programadores. El que dicho sea de paso, es un empleo muy bien remunerado.

Este manual, tiene la finalidad de formarte una mentalidad de programador, mediante la elaboración de algoritmos utilizando diferentes técnicas algorítmicas. Ya que un programador es decir, la persona que diseña sistemas computacionales, antes de comenzar a interactuar con la computadora tiene que tener una manera de pensar diferente a las demás personas para poder analizar y resolver problemas basados en computadoras los cuales debe plasmar en papel.

Para cumplir este objetivo, el manual esta dividido en 6 temas, los cuales tienen una relación progresiva:

Tema 1. *Conceptos Básicos Y Metodología Para La Creación De Sistemas Computacionales.* En este tema se dan las bases mediante conceptos de que es un sistema computacional, un programador, etc.; además, se da a conocer los pasos que se deben realizar para poder implantar un sistema computacional en una organización.

Tema 2. *Operaciones Con Los Datos.* Todo programa, requiere que se realicen cálculos y operaciones matemáticas, por lo cual en este tema se enseñan los diferentes tipos de datos, las operaciones que se pueden realizar con estos y la manera en que la computadora los trata.

Tema 3. *Técnicas Algorítmicas Para La Solución De Problemas.* En este tema se dan a conocer tres técnicas algorítmicas, en las cuales los programadores se basan para posteriormente escribir los sistemas informáticos; basta decir que una de estas tres técnicas es gráfica y el la más usada.

Tema 4. *Estructuras De Control.* Este tema, se puede considerar como el más importante de todos, ya que se aprenderá a escribir en papel sistemas computacionales inteligentes, es decir que estos podrán tomar decisiones propias. Este módulo es el más largo de todos, ya que se ve con mucho detalle, además de que se plantean varios problemas a resolver.

Tema 5. *Arreglos Y Estructuras.* Este tema nos plantea métodos de almacenamiento de datos más complejos, los cuales son muy comunes en la actualidad; Al momento de llegar a este módulo, se debe de manejar completamente las estructuras de control cíclicas.

Tema 6. *Manejo de módulos.* En este tema se trata el manejo de la modulación, lo cual es básicamente el dividir nuestros algoritmos en pequeñas partes independientes que realizan una o más tareas específicas.

Recalcando lo ya anteriormente mencionado, no cualquier persona puede ser programador, esto no quiere decir que sea muy difícil, es verdad que tiene su grado de complejidad, pero este manual esta diseñado para que le sea fácil a cualquier persona que tenga el interés, tiempo y ganas. Para posteriormente ser programador, solo bastará buscar un lenguaje de programación y adaptar nuestros conocimientos a este.

CONVENCIONES USADAS EN ESTE MANUAL

Nota.	Es una especificación adicional a la información que se esta planteando, con un contenido importante.
 **Ejemplo.**	Es la resolución de un problema paso a paso con el fin de comprender como se realiza un procedimiento.
 ***Ejercicios.***	Son problemas prácticos a resolver, que se plantean para adquirir un mejor desempeño y lógica de programador.
Cita Textual.	En la parte inferior de la página se visualizan el autor, libro, editorial y país del que se tomó textualmente un concepto.
 ***Sugerencia.***	Es una recomendación que se hace para tener un mejor desempeño al momento de realizar los algoritmos.
 ***Aspecto Crítico.***	Es un punto muy importante que se debe de tomar en cuenta para evitar dañar el equipo cuando los algoritmos se conviertan en programas.
 ***Evaluación.***	Indica que es momento de realizar una evaluación para comprobar el nivel de avance al momento. Estas se realizan solo al final de cada tema, además de una evaluación diagnostica y una evaluación final.

TEMA I.

CONCEPTOS BÁSICOS Y METODOLOGÍA PARA LA CREACIÓN DE SISTEMAS COMPUTACIONALES

I. CONCEPTOS BÁSICOS Y METODOLOGÍA PARA LA CREACIÓN DE SISTEMAS COMPUTACIONALES

OBJETIVO	Al finalizar el tema, el participante entenderá mediante la exposición del instructor, la lectura y su opinión, los conceptos básicos y la metodología para la solución de problemas por medio de computadoras con la finalidad de que posteriormente los aplique en el diseño de algoritmos.

CONTENIDO

INTRODUCCIÓN

Tal y como se mencionó en la introducción general, se espera que este manual nos ayude a formarnos una mentalidad y lógica de programadores, pero para lograr esto hay que tener una bases sólidas, por lo cual la importancia de este tema, el cual es muy sencillo pero no sin importancia.

Este tema esta desarrollado de una manera tan sencilla, que esperamos comprendas y te aprendas cada uno de los conceptos que se te exponen, ya que sin estos es un poco difícil la comprensión de los temas subsecuentes.

Para que el objetivo del tema se cumpla, se dividió en dos subtemas, en el primero se te dan los conceptos de programador, sistema de información, computadora, entre otros. En el siguiente subtema, se te dan a conocer todos los pasos que debe realizar un programador para poder implantar un sistema computacional en una empresa.

Cuando termines con este tema, realiza la evaluación incluida, la cual es tu punto de comparación para saber si continúas avanzando o repasas, que de antemano estamos seguros no habrá necesidad.

1.1 Conceptos Básicos Para La Solución De Problemas Por Medio De Computadoras

Cuando nosotros terminemos este curso, seremos capaces de diseñar sistemas computacionales, en el lenguaje de programación que nosotros deseemos aprender. Para lo cual debemos de tener muy en claro los siguientes conceptos.

❶ **Sistema.** Un sistema es un conjunto de componentes que interactúan entre sí para lograr un objetivo común [1].

❷ **Sistema Computacional o Sistema de Información.** Es un conjunto de componentes, por el cual los datos de una persona o departamento de una organización fluyen hacia otros [2].

Es un sistema, debido a que el programa que se pueda diseñar por si mismo no realizará nada, sino que tiene que interactuar con la computadora y los usuarios.

1 SENN, James A., Análisis y diseño de sistemas de información.2da Edición, Ed. McGraw Hill, México.
2 ITEM.

❸ ***Programa.*** Es el conjunto de instrucciones escritas de algún lenguaje de programación y que ejecutadas secuencialmente resuelven un problema especifico [3].

❹ **Lenguaje de Programación.** Es cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora [4].

Los lenguajes de programación pueden ser de 3 tipos:

Programas escritos en Lenguaje maquina. (0 y 1)

Programas escritos en Lenguaje ensamblador. (uso de abreviaturas similares al ingles)

Programas escritos en Lenguajes de alto nivel. (enunciados muy similares a los que se utilizan en ingles para comunicarse).

Para que la computadora entienda lo que se le indica que haga, se utilizan traductores, los cuales convierten las instrucciones en cadenas de ceros y unos (lenguaje maquina), dichos traductores se llaman compiladores o interpretes.

❺ ***Computadora.*** Es un dispositivo electrónico-mecánico capaz de ejecutar cálculos y tomar decisiones lógicas a velocidades de millones y a veces miles de millones de instrucciones por segundo [5]. Toda computadora, tiene los siguientes elementos:

3 JOYANES Aguilar Luis;"Fundamentos de Programación, Algoritmos y Estructura de Datos",Ed McGraw Hill
4 NORTON Peter, "Introducción A La Computación", Ed. Pearson, México
5 DEITEL H.M. / DEITEL P.J., "Como Programar en C/C++", Ed. Prentice Hall, México

Dispositivos de Entrada: Como su nombre lo indica, sirven para introducir datos (información) en la computadora para su proceso. Los más usados son el teclado, ratón y scanner.

Dispositivos de Salida: Regresan los datos procesados que sirven de información al usuario. Los más comunes son el monitor y la impresora.

La Unidad Central de Procesamiento (CPU). Aunque generalmente al gabinete se le denomina CPU, el CPU es el microprocesador de la computadora y es el encargado de hacer todos los cálculos y operaciones. El CPU a su vez se divide en las siguientes partes:

- ***Unidad de Control:*** Coordina las actividades de la computadora y determina que operaciones se deben realizar y en que orden; así mismo controla todo el proceso de la computadora.

- ***Unidad Aritmético - Lógica:*** Realiza operaciones aritméticas y lógicas, tales como suma, resta, multiplicación, división y comparaciones.

La Memoria. Es una parte de la computadora en donde se almacenan los datos a procesar y la información resultante. Esta puede ser de dos tipos:

- ***Memoria Primaria:*** Es el espacio en que se almacenan los datos a procesar o calcular en este momento.

- ***Memoria Secundaria:*** Es el espacio en el que se almacena la información resultante para su futura consulta o manejo. Por ejemplo: disquetes, discos duros, unidades de almacenamiento magnético (CD).

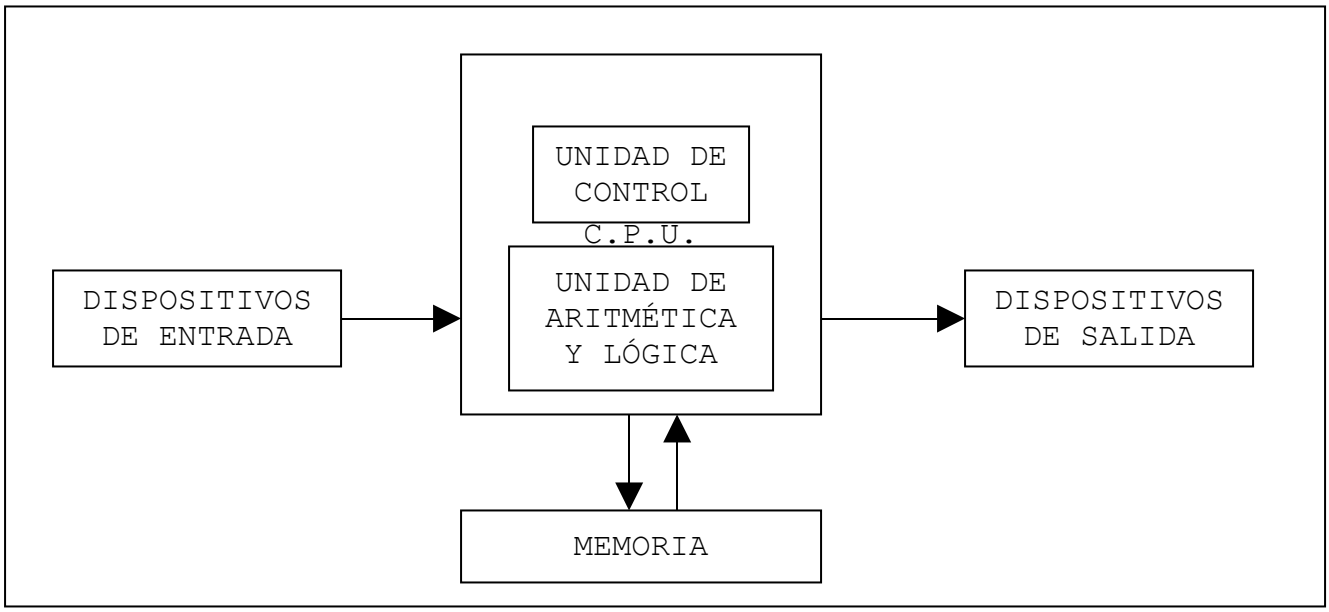


Ilustración 1. Diagrama que representa el funcionamiento de cualquier computadora [6].

Nota. La definición, funcionamiento y partes de una computadora que se están mencionando en este manual son muy básicos, ya que ahondar en este tema nos llevaría todo un curso.

❻ **Programador o analista o diseñador de sistemas.** Es la persona encargada de crear un programa o sistema en un lenguaje de programación específico.

❼ **Usuario**. Es la persona que interactúa con el sistema de información, o mejor dicho con la computadora [7].

6 TANENBAUM Andrew,"Organización De Computadoras, Un Enfoque Estructurado", Ed. Prentice Hall, México
7 SENN, James A., "Análisis y diseño de sistemas de información", Ed. McGraw Hill, México.

Usuario Final Directo. Operan el sistema. Interactúan directamente a través de la computadora, ingresando datos y recibiendo salidas.

Usuario Final Indirecto. Son aquellos que emplean los reportes y otros tipos de información que genera el sistema, pero no operan el equipo.

Dicho y comprendido lo anterior, debemos de conocer el significado de la palabra ALGORITMO, ya que el curso esta diseñado para que aprendamos a realizar estos.

❽ **Algoritmo.** Es la representación en papel de una serie de pasos organizados que describe el camino y las operaciones que se deben seguir para dar solución a un problema específico [8].

La palabra algoritmo se deriva de la degeneración de la palabra árabe Al Jwarizmi, la cual es el pseudónimo de Mohammed Ben Musa, matemático padre del álgebra y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.

Existen diferentes técnicas de representar los algoritmos:

Gráficos: Es la representación del algoritmo por medio de varios símbolos gráficos, donde cada símbolo representa una operación distinta.

No Gráficos: Es la representación del algoritmo por medio de texto el cual es entendible por nosotros.

8 FERREYRA Cortés Gonzalo, "Informática, Para Cursos De Bachillerato", Ed. Alfaomega, México

Híbrido: Es la representación del algoritmo en un modo que combina los 2 métodos anteriores.

Nota. Todo lo referente a las técnicas algorítmicas, se analizará afondo a partir del tercer tema. Estos son solo fundamentos.

1.2 Metodología para la solución de problemas por medio de computadora

Aunque el objetivo de este curso es solo aprender a diseñar algoritmos y no implantar sistemas computacionales, en este subtema se definen brevemente todos los pasos que debe realizar un analista o programador para colocar un sistema de información en una empresa, con la finalidad de que identifique en que parte de esta proceso entra el diseño de los algoritmos.

El ciclo de vida que se debe seguir para implantar un sistema de información en una compañía son los siguientes:

❶ ***Investigación Preliminar.*** Esta comienza cuando se recibe una solicitud para diseñar un sistema y consta de tres partes:

a) **Aclaración De La Solicitud.** En muchas ocasiones las solicitudes no estas formuladas de manera clara. Por consiguiente, la solicitud de proyecto debe examinarse detenidamente para determinar con precisión lo que el solicitante desea y esta debe estar claramente planteada.
b) **Estudio De Factibilidad.** El resultado más importante en la investigación preliminar es el determinar si el sistema es factible; es decir que se pueda hacer o realizar. Existen

tres aspectos relacionados con el estudio de la factibilidad.

1. **Factibilidad Técnica.** El trabajo para el proyecto, ¿puede realizarse con el equipo actual, la tecnología existente de software y el personal disponible? Si se necesita nueva tecnología, ¿cuál es la posibilidad de desarrollarla?

2. **Factibilidad Económica.** Al crear el sistema, ¿los beneficios que se obtienen serán suficientes para aceptar los costos?, ¿los costos asociados con la decisión de NO crear el sistema son tan grandes que se debe aceptar el proyecto?
3. **Factibilidad Operacional.** Si se desarrolla e implanta el sistema, ¿será utilizado?, ¿existirá cierta resistencia al cambio por parte de los usuarios que dé como resultado una disminución de los posibles beneficios de la aplicación?

c) **Aprobación De La Solicitud.** No todas las solicitudes son factibles. Pero cuando se aprueba una solicitud se tiene que estimar su costo, el tiempo para su desarrollo e implantación y las necesidades de personal.

❷ **Análisis Del Sistema.** En esta actividad se tienen que comprender todas las facetas importantes de la parte de la empresa que esta bajo estudio. Se deben estudiar los procesos de una empresa para dar respuesta a las siguientes preguntas claves:

1. ¿Qué es lo que se hace?
2. ¿Cómo se hace?
3. ¿Con qué frecuencia se presenta?

4. ¿Qué tan grande es el volumen de transacciones o de decisiones?
5. ¿Cuál es el grado de eficiencia con el que se efectúan las tareas?
6. ¿Existe algún problema?
7. Si existe un problema, ¿qué tan serio es?
8. Si existe un problema, ¿cuál es la causa que lo origina?

Para contestar estas preguntas, el analista debe entrevistar a varias personas (trabajadores y directivos), así como observar y estudiar su desempeño, para reunir información de cómo se realizan los procesos de la empresa.

Todo esto, mediante el uso de cuestionarios, entrevistas, estudio de manuales y reportes, muestras de formas y documentos y la observación en condiciones reales de trabajo.

Conforme se va reuniendo la información se deben ir identificando las características operacionales tales como controles de procesamiento, tiempos de respuesta y métodos de entrada y salida.

❸ **Diseño Lógico Del Sistema.** Produce los detalles que establecen la forma en la que el sistema cumplirá con los requerimientos identificados en la fase de determinación de requerimientos.

Se comienza el proceso identificando los reportes y demás salidas que debe producir el sistema. Entonces se determina con toda precisión los datos específicos para cada reporte y salida, haciendo bosquejos en formatos de pantalla que se esperan que aparezcan cuando

el sistema este terminado, ya sea en papel o en la pantalla de la computadora.

El diseño de sistema también indica los datos de entrada, aquellos que serán calculados y los que deben ser almacenados. Así mismo se escriben con todo detalle los procedimientos de cálculo y datos individuales. Se tienen que seleccionar las estructuras de archivo y los dispositivos de almacenamiento. Estos procedimientos indican como procesar los datos y producir las salidas.

Todos estos procedimientos que contienen las especificaciones son representados mediante diagramas, tablas, símbolos especiales, etc.; Entonces a partir de estos se comienza la fase de desarrollo de software.

Nota. El diseño de algoritmos se realiza en esta etapa, ayudado por la recolección de información realizada en la segunda etapa.

Nota. El diseño de los algoritmos no es la única tarea que se realiza en esta fase.

❹ **Diseño Físico Del Sistema.** En esta fase se escribe el programa y la base de datos de acuerdo a los documentos recibidos de la actividad anterior.

El programador es responsable de elaborar la documentación de los programas y de proporcionar una explicación de cómo y por que ciertos procedimientos se codifican en determinada forma. La

documentación es esencial para probar el programa y llevar a cabo el mantenimiento una vez que la aplicación se encuentra instalada.

❺ **Prueba De Sistemas.** Durante esta fase, el sistema se emplea de manera experimental para asegurarse de que el software no tenga fallas, es decir que funciona de acuerdo con las especificaciones y en la forma en que los usuarios esperan que lo haga. Se alimentan con entradas de prueba para su procesamiento y después se examinan los resultados. En ocasiones se permite que varios usuarios utilicen el sistema para que se observe como trabajan y como se sienten con él.

Hay que descubrir cualquier error antes de que la organización implante el sistema y dependa de él. Si es que se detecta un error, hay que revisar si este es físico o lógico, es decir, un error físico es que el programa esta mal escrito, pero un error lógico implica regresar a las etapas anteriores para detectar el origen de la falla. Esto provoca que esta sea la etapa más ardua y difícil, ya que es muy probable que tengamos que estar corrigiendo el programa infinidad de veces hasta que no presente problemas.

Es muy probable que esta fase sea realizada por personas ajenas a la empresa para que esta sea objetiva.

❻ **Implantación Y Evaluación.** La implantación es el proceso de instalar el sistema, construir los archivos de datos necesarios y entrenar a los usuarios.

Dependiendo del tamaño de la organización, puede elegirse comenzar la operación del sistema sólo en un área de la empresa (prueba piloto) y con solo unas cuantas personas. Algunas veces se deja que los dos sistemas (viejo y nuevo), trabajen de forma paralela

con la finalidad de comparar resultados; en otras ocasiones simplemente se deja de utilizar el viejo sistema un día y al siguiente día se comienza a utilizar el sistema nuevo.

Estos sistemas generalmente trabajan durante muchos años. Sin embargo las organizaciones y los usuarios cambian con el paso del tiempo. Por consiguiente, es indudable que debe darse mantenimiento, realizar cambios y modificaciones al software, a los archivos o a los procedimientos del sistema. Todo esto con la finalidad de que los sistemas se mantengan al día y no se vuelvan obsoletos. En este sentido la implantación es un proceso de constante evolución.

La evaluación de un sistema se lleva a cabo para identificar puntos débiles y fuertes de este. La evaluación ocurre a lo largo de cualquiera de las siguientes dimensiones:

Evaluación Operacional. Evalúa la forma en que funciona el sistema, incluyendo su facilidad de uso, tiempo de respuesta, lo adecuado de los formatos de información, confiabilidad global y nivel de utilización.

Impacto Organizacional. Identifica y mide los beneficios de la organización en cuanto a costos, ingresos, ganancias, eficiencia operacional e impacto competitivo; desde que fue implantado el sistema.

Opinión De Los Administradores. Evalúa las actitudes de los directivos y administradores dentro de la organización así como de los usuarios finales.

Desempeño De Desarrollo. Se evalúa el desarrollo del sistema en criterios tales como tiempo y esfuerzo de desarrollo, para

ver si concuerdan con los presupuestos y estándares, y otros criterios de administración de proyectos.

CONCLUSIÓN

En este tema, se vieron dos subtemas fundamentales para diseñar sistemas que resuelvan problemas orientados a computadoras.

En el primer subtema, se presentaron conceptos básicos los cuales si no son todos son los más básicos para alguien que piensa dedicarse al diseño de sistemas. Algunas de las definiciones son:

Sistema De información. Conjunto de componentes, por el cual los datos de una persona o departamento de una organización fluyen hacia otros.

Programa. Conjunto de instrucciones escritas de algún lenguaje de programación

Lenguaje De Programación. Lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora

Computadora. Dispositivo electrónico-mecánico capaz de ejecutar cálculos y tomar decisiones lógicas

Programador. Persona encargada de crear un programa o sistema

Algoritmo. Representación en papel de una serie de pasos organizados que describe el camino y las operaciones que se deben seguir para dar solución a un problema específico

En el segundo subtema se dieron a conocer la serie de pasos que se deben de realizar para implantar un sistema informático en una empresa, también conocidos como ciclo de vida de un sistema de información:

Investigación Preliminar. Radica en determinar que es lo que se quiere realizar.

Análisis del sistema. Consiste en estudiar el sistema actual.

Diseño Lógico del Sistema. Fundamenta en poner en papel el nuevo sistema.

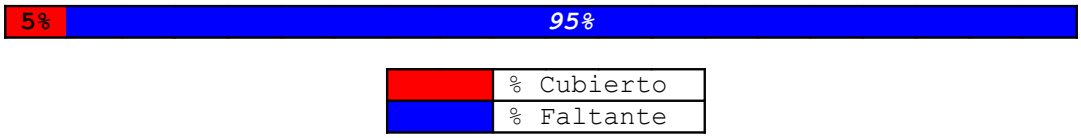
Diseño Físico del Sistema. Gravita en diseñar el nuevo sistema.

Prueba Del Sistema. Consiste en probar el sistema para comprobar que no tiene errores.

Implantación y Evaluación del sistema. Reside en poner a trabajar el sistema en la empresa y comprobar su funcionalidad.

Hasta este momento, con lo aprendido podemos sentirnos satisfechos, pues hemos establecido las bases para cubrir completamente el objetivo del curso. Nos hace falta bastante camino por recorrer, por lo cuál podemos considerar que solo hemos cubierto un 5% del total.

OBJETIVO DEL CURS0



TEMA II.

OPERACIONES CON LOS DATOS

II. OPERACIONES CON LOS DATOS

OBJETIVO	Al terminar el tema, el participante mediante la práctica dominará las operaciones que se realizan con los datos con la finalidad de diseñar algoritmos que calculan y comparan datos.

CONTENIDO

INTRODUCCIÓN

Como ya se ha comentado anteriormente, este curso tiene por objeto enseñarnos a diseñar algoritmos, los cuales en un futuro utilizaremos para escribir programas computacionales.

La importancia de este tema es tan grande debido a que todo sistema de información realiza cálculos con datos para entregar resultados a la empresa, por lo cual debemos saber que los datos que maneja la empresa solamente pueden ser números, letras y números y una respuesta afirmativa o negativa; y los cálculos que el sistema puede realizar sobre estos datos son operaciones como suma, resta, multiplicación y división, además de comparaciones entre dos datos para saber si uno es mayor que el otro, si es menor, si son iguales o diferentes, y establecer un grado de satisfacción entre dos datos en base a las tablas de la verdad (AND, OR y NOT).

Sabiendo todo lo anterior, debemos aprender a expresar los cálculos a realizar por el sistema de una manera que la computadora

pueda comprenderlos y arrojar los resultados correctos mediante una expresión o fórmula que se rige por un conjunto de reglas.

Además de que debemos de aprender a crear los espacios temporales de almacenamiento donde se guardarán tanto los datos como los resultados.

Para cubrir estos puntos, el tema se ha dividido en varios subtemas:

- El primero es para conocer los diferentes tipos de datos que maneja una computadora.
- El segundo esta dedicado a enseñarnos como se redacta una expresión de tal manera que la computadora la entienda.
- El tercero esta diseñado para saber como se crea y se almacena información en un espacio de memoria de la computadora.

Este tema no es difícil de asimilar, pero es fundamental para lograr cumplir el objetivo general del curso, por lo cual se te pide dedicación.

Para ayudarte a especializarte en la creación de expresiones y manejo de operadores, este capítulo cuenta con una buena cantidad de ejercicios lo cuales se te pide que resuelvas. Recuerda que la práctica hace al maestro.

2.1 Tipos De Datos Simples

Cualquier sistema de información por pequeño o sencillo que sea tiene por objetivo procesar diferentes valores para entregar un resultado a la persona indicada, estos valores son conocidos como datos y a los resultados se les denomina información.

Dato. "Es una pequeña parte de información que por si sola no dice nada, pero que en conjunto forma información" [9].

Información. "Es un conjunto de datos estructurados o procesados" [10].

Los datos por sencillos que parezcan, siempre están relacionados a un tipo. ver tabla 1.

TIPOS DE DATOS		
Simples	➢ Numéricos	▪ Enteros
		▪ Reales
	➢ Lógicos	
	➢ Alfanuméricos	
Complejos	➢ Arreglos	▪ Unidimensionales
		▪ Multidimensionales
	➢ Estructuras	

Tabla 1. Clasificación de los datos

9 SENN, James A., Análisis y diseño de sistemas de información.2da Edición, Ed. McGraw Hill, México.
10 ITEM

❶ **Datos Simples.**

***Datos* Numéricos:** Permiten representar valores escalares de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.

- **Enteros.** Son los números que no tienen parte decimal, pueden ser positivos ó negativos, por ejemplo: 10, 0, 1358, -456.

- **Reales.** Son los números que contienen una fracción, es decir, punto decimal y estos al igual que los enteros pueden ser positivos o negativos, por ejemplo: 12.45, 7.0, -157.0001.

Datos Lógicos: Son aquellos que solo pueden tener uno de dos valores posibles (cierto o falso) ya que representan el resultado de una comparación entre otros datos (numéricos o alfanuméricos).

Datos Alfanuméricos: Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva, esto incluye nombres de personas, direcciones, etc. Es posible representar números como alfanuméricos, pero estos pierden su propiedad matemática, es decir no es posible hacer operaciones con ellos. Este tipo de datos se representan encerrados entre comillas.

Nota. Los datos complejos se explican y analizan en el tema V.

2.2 Tipos De Operadores

Cualquier lenguaje de programación tiene la capacidad de realizar a los datos los cálculos más complejos mediante un conjunto de operadores y un grupo de reglas básicas.

Debemos de aprender a utilizar los datos y operadores, pues somos nosotros quien le indicará a la computadora los cálculos a realizar a ciertos datos.

Por ejemplo, si se nos pide un sistema que saque el promedio de un alumno que tiene 5 materias, a la maquina no le podemos decir "saca el promedio del alumno" debido a que es una instrucción que no reconoce, para que despliegue el resultado le tenemos que indicar suma la calificación de la primera materia, la segunda, la tercera, la cuarta y la quinta, y al resultado lo divides entre cinco. Pero aún así no es tan fácil como parece pues tenemos que representar esta instrucción de una manera que la computadora la comprenda. Para lo cual tenemos que elaborar una expresión o fórmula en una sola línea de código, utilizando operadores, operandos y unos criterios de ejecución llamados reglas de precedencia.

Al conjunto de todos los operadores, los podemos dividir en tres grupos:

- **Operadores Aritméticos**
- **Operadores Relaciónales**
- **Operadores Lógicos.**

❶ **Operadores Aritméticos.** Son aquellos con los que podemos realizar operaciones como suma, resta, multiplicación, división, módulo y asignación.

OPERACIÓN	OPERADOR	EXPRESIÓN ALGORÍTMICA
Suma	**+**	a + b 5 + 7 a + 7
Resta	**-**	a - b 5 - 7 a - 7
Multiplicación	*****	a * b 5 * 7 a * 7
División	**/**	a / b 10 / 2 a / 2
Módulo	**%**	a % b 10 % 3 a % 3
Asignación	**=**	a = 8 b = a c = a + b

Tabla 2. Los diferentes operadores aritméticos

Los operadores aritméticos son del tipo binario, es decir; necesitamos de dos operandos, uno a la izquierda y otro a la derecha para realizar una operación.

Con ayuda de estos operadores podemos realizar cualquier cálculo matemático, como elevar al cuadrado, sacar raíces cuadradas, calcular factoriales, etc.

El operador módulo es un operador entero el cual siempre se debe de utilizar con números enteros, y el resultado que envía es el residuo de una división. Por ejemplo, en el caso de 10 % 3 el resultado es 1, debido a que 10 / 3 es igual a 3 y nos sobra 1.

Las expresiones aritméticas se deben escribir en una línea continua y bajo unas ***reglas de precedencia de operadores.*** Las cuales son guías de acción que permiten calcular las expresiones en el orden correcto [11].

1. Se calculan primero las operaciones de multiplicación, división y módulo, los cuales tienen el mismo nivel de precedencia, por lo cual si existen varios de estos en una expresión se comienzan a calcular de izquierda a derecha.

2. Se calculan las operaciones de suma y de resta, los cuales tienen el mismo nivel de precedencia. Si la expresión contiene varias de esta se realizan de izquierda a derecha.

3. Si en la expresión se encuentran paréntesis, esto indica que lo que esta dentro de ellos se debe resolver antes que cualquier cosa siguiendo las reglas de precedencia antes mencionadas, por lo cual los paréntesis son utilizados para obligar a la computadora a evaluar primero ciertas expresiones. En caso de existir paréntesis anidados se evalúa el par más interno.

4. Por último se realiza la asignación, la cual significa que el valor de la derecha es asignado al **identificador** de la izquierda.

Nota. Posteriormente, al ver los otros operadores (lógicos y relacionales), se aplican las mismas reglas de precedencia, con la diferencia de que se aumentaron más operadores.

11 SENN, James A., Análisis y diseño de sistemas de información.2da Edición, Ed. McGraw Hill, México.

Lo anterior se puede resumir en la siguiente tabla.

OPERADOR	PRECEDENCIA
( )	Mayor
*, / , %	
+, -	
=	Menor

Tabla 3. Precedencia de los operadores aritméticos

Ejemplo 1. Supongamos que tenemos la siguiente expresión:

EXPRESIÓN		y = 2 * 5 * 5 + 3 * 5 + 7	
ACTIVIDAD		**OPERACIÓN**	**RESULTADO**
1.	Realiza la multiplicación más a la izquierda	y = 2 * 5 * 5 + 3 * 5 + 7	y = 10 * 5 + 3 * 5 + 7
2.	Realiza la multiplicación más a la izquierda	y = 10 * 5 + 3 * 5 + 7	y = 50 + 3 * 5 + 7
3.	Realiza la multiplicación más a la izquierda	y = 50 + 3 * 5 + 7	y = 50 + 15 + 7
4.	Realiza suma más a la izquierda	y = 50 + 15 + 7	y = 65 + 7
5.	Realiza la suma	y = 65 + 7	**y = 72**

Tabla 4. Ejemplo 1 de precedencia de operadores aritméticos

Ejemplo 2. Supongamos que tenemos la siguiente fórmula:

EXPRESIÓN		Z = 4 * ((2 + 6) * (8 - 10))	
ACTIVIDAD		**OPERACIÓN**	**RESULTADO**
1.	Realiza el paréntesis más interno de la izquierda	Z = 4 * ((2 + 6) * (8 - 10))	Z = 4 * (8 * (8 - 10))
2.	Realiza el paréntesis más interno	Z = 4 * (8 * (8 - 10))	Z = 4 * (8 * -2)
3.	Realiza el paréntesis	Z = 4 * (8 * -2)	Z = 4 * -16
4.	Realiza la multiplicación	Z = 4 * -16	**Z = -64**

Tabla 5. Ejemplo 2 de precedencia de operadores aritméticos

Ejercicios.

I. Resuelve las siguientes operaciones utilizando las reglas de precedencia, donde: W = 5 , X = 7, Y = 3, Z = 9	
A = y - z * x + w / 3	
A = z + w % y	
A = X * (Z – Y)/ W	
A = (4 * Y + Z % W) * X	
A = Z * W – X + Y / Z	
II. Expresa las siguientes formulas para que las entienda la computadora.	
Calcular el perímetro de un círculo.	
Calcular el área de un rectángulo	
Calcular el área de un circulo	
$X = Z^3$	

$X = \frac{4AC + (A + C) - 2AB}{2AB}$	
$X = 5(Y^2 + Z^3 - 5ZW + 3)$	

💬 ***Sugerencia.*** Al momento realizar una expresión, hay que tomar en cuenta que la división de 2 datos del tipo numéricos enteros da como resultado un numero entero, es decir, si el resultado puede tener fracción esta se pierde y nunca se hace un redondeo. Por lo cual se recomienda que una división se haga entre 2 números o al menos 1 del tipo numérico real.

❷ **Operadores Relaciónales.** Los operadores relacionales se usan para determinar la relación de la expresión de la izquierda con la de la derecha (binarios). El resultado de esta evaluación regresa el valor de falso o verdadero, donde falso es igual a cero y verdadero es igual a 1.

OPERADOR	RELACIÓN
= =	Igual
!=	Diferente
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que

Tabla 6. Conjunto de operadores relacionales

NO TODOS los operadores relacionales están al mismo nivel de precedencia entre ellos. Los operadores <, <=, >, >=, tienen mayor precedencia que los operadores de == y !=.

En una operación o fórmula se pueden mezclar tanto operadores aritméticos como relacionales, pero los operadores relacionales tienen menor precedencia que los operadores de suma y resta pero mayor que el operador de asignación.

OPERADOR	PRECEDENCIA
( )	Mayor ↓
*, / , %	
+, -	
<, >, <=, >=	
==, !=	
=	Menor

Tabla 7. Precedencia de los operadores aritméticos y relacionales

Ejemplo 1. Supongamos que tenemos la siguiente fórmula:

EXPRESIÓN		Z = 4 <= 2 == 6 != 8 > 10	
ACTIVIDAD		**OPERACIÓN**	**RESULTADO**
1.	Realiza la comparación de mayor precedencia de la izquierda.	Z = 4 <= 2 == 6 != 8 > 10	Z = 0 == 6 != 8 > 10
2.	Realiza la comparación de mayor precedencia	Z = 0 == 6 != 8 > 10	Z = 0 == 6 != 0
3.	Realiza la comparación de mayor precedencia de la izquierda.	Z = 0 == 6 != 0	Z = 0 != 0
4.	Realiza la comparación	Z = 0 != 0	**Z = 0**

Tabla 8. Ejemplo 1 de cómo se utilizan los operadores relacionales

Ejemplo 2. Supongamos que tenemos la siguiente fórmula:

EXPRESIÓN		Z = 8 == (9 + (1 != 0)) > 3 * 5	
ACTIVIDAD		**OPERACIÓN**	**RESULTADO**
1.	Realiza la operación dentro del paréntesis más interno	Z = 8 == (9 + (1 != 0)) > 3 * 5	Z = 8 == (9 + 1) > 3 * 5
2.	Realiza la operación dentro del paréntesis	Z = 8 == (9 + 1) > 3 * 5	Z = 8 == 10 > 3 * 5
3.	Realiza la multiplicación	Z = 8 == 10 > 3 * 5	Z = 8 == 10 > 15
4.	Realiza la comparación de mayor precedencia	Z = 8 == 10 > 15	Z = 8 == 0
5.	Realiza la comparación	Z = 8 == 0	**Z = 0**

Tabla 9. Ejemplo 2 de cómo se utilizan los operadores relacionales

Ejercicios.

I. Realiza las siguientes operaciones siguiendo las reglas de precedencia, donde:

W = 3, X = 5, Y = 7, Z = 9

A = X == Z	
A = W >= Y	
A = W == X < Y < Z	
A = (W == X) == (Y > Z)	

A = X != (W < Z < Y) == 1	
A = W * Y >= W * Z	
A = Y + W * Z / W != Z + W – Y * X	
A = (Y + W) * Z / W == Y * X – 20 / 4	
A = W * Y >= W * Z == (Y + W) * Z > 0	
A = X > Z * (W + Y)!= W <= X	

❸ **Operadores Lógicos.** Los operadores Lógicos, se usan para soportar las operaciones básicas lógicas AND, OR y NOT de un dato verdadero y un falso, de dos verdaderos o de dos falsos, de acuerdo con las tablas de la verdad correspondientes.

La computadora entiende que falso es igual a 0 y verdadero es cualquier valor diferente a 0. Al regresar los valores asigna un 0 para decir que el resultado de la expresión es falso y un 1 para verdadero.

Las tablas de la verdad AND y OR nos sirven para determinar el grado de satisfacción de acuerdo al valor lógico de dos datos. La tabla del operador NOT solo nos regresa el contrario o negación del valor lógico de un dato. Las tablas se describen a continuación.

a	b	a AND b
0	0	0
0	No 0	0
No 0	0	0
No 0	No 0	1

Tabla 10. Tabla de la verdad del operador lógico AND

a	b	a OR b
0	**0**	**0**
0	**No 0**	**1**
No 0	**0**	**1**
No 0	**No 0**	**1**

Tabla 11. Tabla de la verdad del operador lógico OR

A	NOT
0	**1**
No 0	**0**

Tabla 12. Tabla de la verdad del operador lógico NOT

OPERADOR	OPERACION LÓGICA
&&	**AND**
||	**OR**
!	**NOT**

Tabla 13. Conjunto de Operadores lógicos

Los operadores lógicos NO están al mismo nivel de precedencia entre ellos. El operador NOT es el de mayor, posteriormente se encuentra el AND y por último el OR.

En una operación o fórmula se pueden mezclar los operadores aritméticos, relacionales, y lógicos, aunque resulta más común dividir una expresión de este tipos en dos o más.

Esta es la tabla de precedencia de todos los operadores:

OPERADOR	PRECEDENCIA
( )	Mayor
!	
*, / , %	
+, -	
<, >, <=, >=	
==, !=	
&&	
||	
=	Menor

Tabla 14. Tabla de precedencia de todos los operadores

Ejemplo 1. Supongamos que tenemos la siguiente fórmula:

EXPRESIÓN		Z = 0 || 4 || 2 && ! 8	
ACTIVIDAD		**OPERACIÓN**	**RESULTADO**
1.	Realiza primero la negación	Z = 0 || 4 || 2 && ! 8	Z = 0 || 4 || 2 && 0
2.	Realiza la operación del AND	Z = 0 || 4 || 2 && 0	Z = 0 || 4 || 0
3.	Se realiza la operación OR más a la izquierda	Z = 0 || 4 || 0	Z = 1 || 0
4.	Realiza la comparación del OR	Z = 1 || 0	**Z = 1**

Tabla 15. Ejemplo 2 de cómo se utilizan los operadores relacionales

Nota. Al momento de que la computadora ejecuta la expresión, cuando llega al paso 3 termina la ejecución, debido a que ya sabe que el resultado será 1 y no puede cambiar.

Ejemplo 2. Supongamos que tenemos la siguiente fórmula:

EXPRESIÓN		Z = 1 || (6 * !0 > 5 && 9 < 3 * 4)	
ACTIVIDAD		**OPERACIÓN**	**RESULTADO**
1.	Se realiza todo lo que esta dentro del paréntesis	Z = 1 || (6 * !0 > 5 && 9 < 3 * 4)	
2.	Dentro del paréntesis se realiza primero la negación	Z = 1 || (6 * !0 > 5 && 9 < 3 * 4)	Z = 1 || (6 * 1 > 5 && 9 < 3 * 4)
3.	Dentro del paréntesis se realiza la multiplicación de más a la izquierda	Z = 1 || (6 * 1 > 5 && 9 < 3 * 4)	Z = 1 || (6 > 5 && 9 < 3 * 4)
4.	Dentro del paréntesis se realiza la multiplicación	Z = 1 || (6 > 5 && 9 < 3 * 4)	Z = 1 || (6 > 5 && 9 < 12)
5.	Dentro del paréntesis se realiza la comparación de más a la izquierda	Z = 1 || (6 > 5 && 9 < 12)	Z = 1 || (1 && 9 < 12)
6.	Dentro del paréntesis se realiza la comparación	Z = 1 || (1 && 9 < 12)	Z = 1 || (1 && 1)
7.	Dentro del paréntesis se establece el resultado lógico	Z = 1 || (1 && 1)	Z = 1 || 1
8.	Se establece el resultado lógico	Z = 1 || 1	**Z = 1**

Tabla 16. Ejemplo 2 de cómo se utilizan los operadores relacionales

Ejercicios.

I. Realiza las siguientes operaciones siguiendo las reglas de precedencia, donde:

W = 3, X = 0, Y = 7, Z = 1

A = X && Z	
A = !W || X	
A = W || X || Y && !Z || X && Z	

A = W || X || Y && !(!Z || X && Z)	
A = W == X && Y > Z	
A = X != (W < Z || Y) + 1	
A = W * Y >= W && Z == !(X + Y * W)	
A = (Y + W) || !(Z / W && Z + W - Y * X)	
A = (Y || W) && Z / W == Y * X - 20	
A = W * Y >= W && Z == (Y + W) * Z > 0	
A = X > Z * !(W + Y)!= W || X	
A = W + X && Z * W > W - Z && X - Y	
A = !(3 + W && Z || W * X && 7 > 1)	

2.3 Identificadores

Como ya se vio anteriormente, una computadora puede manejar y manipular ciertos datos. Pero para que la computadora los procese, los datos se pueden guardar temporalmente en una pequeña parte de la memoria de la computadora, a este espacio se le debe decir que tipo de datos puede almacenar (enteros, reales, alfanuméricos, etc.) y como queremos que se le llame para poder localizarlo posteriormente. A este espacio de memoria con un nombre y tipo específico, se le conoce como **identificador**.

¿Porqué usar identificadores?

Si nosotros no creamos un identificador, el dato que deseamos guardar se almacenaría en una posición de memoria la cual esta identificada por un número hexadecimal, y para recuperarla tendríamos que saber esta dirección, por lo cual es más fácil asignarle un nombre. Además, si nosotros no le indicamos un tipo para los datos que se van a almacenar, la computadora no sabrá como tratar a esta información, recordemos que en la computadora solo están almacenados ceros y unos.

FF00h					FF0Ch					Sin el uso de identificadores, tendríamos que saber la dirección de memoria en donde se guardo la información.
FF01h	00101110				FF0Dh					
FF02h					FF0Eh					
FF03h					FF0Fh	10001111				
FF04h					FF10h					
FF05h					FF11h					
FF06h	11011101				FF12h					
FF07h					FF13h					
FF08h					FF14h					
FF09h					FF15h	11110001				
FF0Ah					FF16h					
FF0Bh					FF17h					

Tabla 17. Cómo se almacenarían los datos si no existiesen los identificadores.

Cuando reservemos un espacio de memoria asignándole un **identificador**, solo se tiene dar este nombre para acceder al dato que tiene guardado.

FF00h					FF0Ch					Con el uso de identificadores, solo se tiene que hacer referencia al nombre de este.
Ident1	00101110				FF0Dh					
					FF0Eh					
FF03h					Ident3	10001111				
FF04h										
FF05h					FF11h					
Ident2	11011101				FF12h					
					FF13h					
FF08h					FF14h					
FF09h					Ident4	11110001				
FF0Ah										
FF0Bh					FF17h					

Tabla 18. Cómo se guardan los datos usando identificadores.

Los identificadores se dividen en dos:

❶ **Constantes.** Es aquel en el cual, el dato que tiene dentro es el mismo desde que comienza el programa hasta que termina, y bajo ninguna circunstancia ni procedimiento puede cambiar. Por ejemplo: Pi, ya que siempre es 3.1416.

❷ **Variables.** Es aquel en el cual, el dato que tiene dentro puede cambiar todas las veces necesarias por otro en cualquier parte del programa siempre y cuando sean del tipo especificado anteriormente. Por ejemplo: edad, ya que puede almacenar en determinado momento mi edad, en otro la tuya, etc. A su vez, las variables se pueden clasificar por su uso en:

> ***Variables de Trabajo:*** *Son* aquellas que reciben el resultado de una operación matemática compleja y que se usan normalmente dentro de un programa, pero si es del tipo alfanumérico solo se utiliza para almacenar información. Ejemplo: promedio = (9 + 8 + 7) / 3

Contadores: Se utilizan para llevar el control del número de ocasiones en que se realiza una operación o se cumple una condición. Con los incrementos generalmente de uno en uno. Podríamos utilizarlos cuando necesitamos llevar el conteo del número de personas que votaron por el PAN. Son exclusivamente del tipo entero.

Acumuladores: Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente. Una variable de este tipo podríamos utilizarla para ir sumando poco a poco el monto total de nuestra compra en un supermercado.

Nota. En estas variables (de hecho en todas), solo se actualiza el valor, no se almacenan los valores previos.

Variable indicador o de bandera: Es aquella que recibe uno de dos posibles valores. Se les conoce también como BANDERAS y generalmente son del tipo **boleano**.

Nota. Todas las variables pueden recibir o modificar un valor anterior mediante el signo de asignación, para lo cual deben de estar colocadas al lado izquierdo de este.

Reglas para formar un identificador

Debe comenzar con una letra (A-Z, mayúsculas o minúsculas)

No deben contener espacios en blanco.

Dígitos y caracteres especiales están permitidos después del primer carácter.

La longitud de identificadores puede ser de hasta 8 caracteres.

El nombre del identificador debe ser significativo.

Indicar su tipo (entero, real, alfanumérico, bolean).

Si se desea, se puede indicar su uso, el cual como ya sabemos solo es para las variables.

Si se desea, asignarles un valor de inicio. En los constantes es forzoso este punto.

Ejemplos.

1) Necesitamos un identificador para almacenar el promedio que obtuve en el semestre:

Pro_sem : entera : trabajo = 0

2) Necesitamos un identificador el cual contenga siempre el IVA a calcular:

IVA : real = .15

3) Necesitamos un identificador para llevar la relación de cuantos goles anota Cuauhtemoc Blanco con el Veracruz:

Goles_cua : entera : contador = 0

4) Necesitamos un identificador que almacene el nombre de una persona:

Nombre : alfanumérico : trabajo = “Carlos Augusto”

Nota. Para almacenar cadenas de caracteres hay que utilizar comillas.

Ejercicios.

I. Declara un identificador para cada uno de los siguientes casos e inicialízalos, además especifica si será una variable o una constante		
Dirección de una persona		
Código postal		
Una tonelada en kilos		
Peso de un producto a granel		
Total de tiempo corrido en 20 vueltas a un campo		
Talla de zapatos en EE.UU.		
Número telefónico de una persona		
Un kilómetro en metros		
Estatura de una persona		
Total de las ventas realizadas en un estadio		
Punto de ebullición		
Total de artículos vendidos		
La velocidad de la luz.		
Promedio de un alumno del conalep		
Número de horas trabajadas		
Número de control de un alumno		
Total de ingresos de una familia		
Numero de días del año		
Número de cervezas por cartón		

CONCLUSIÓN

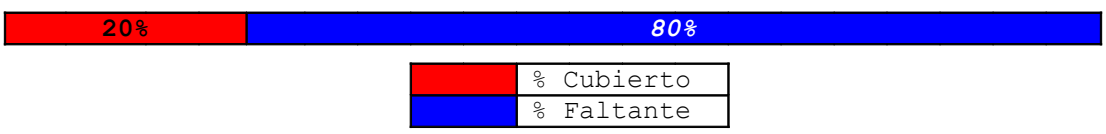
En este tema se abarcaron tres subtemas, en el primero que explicó que un dato por si solo no dice nada pero que en conjunto forma la información, se vio que existen datos simples y datos complejos, y que los datos simples pueden ser numéricos enteros, numéricos reales, alfanuméricos y boleanos.

En el segundo, se dieron a conocer los diferentes tipos de operadores, los cuales pueden ser aritméticos para realizar operaciones como suma, resta, multiplicación, división y módulo; relacionales con los cuales se puede obtener un resultado falso o verdadero al comparar dos valores; Y establecer resultados en base a tablas de verdad con ayuda de los operadores lógicos AND, OR y NOT.

En el tercero se enseño a crear identificadores, es decir, variables o constantes en las cuales se pueden guardar valores para posteriormente ser utilizados al realizar operaciones o ecuaciones con los diferentes operadores.

La compresión absoluta de este tema es muy importante, ya que casi todos los sistemas de información requieren hacer cálculos con los datos para generar información importante para la toma de decisiones dentro de una empresa. Por lo cual se recomienda que si su compresión no es absoluta se vuelva a dar un repaso para proseguir. Debido a esto, se ha dado un gran avance para el logro del objetivo del curso, calculado en un 15% más.

OBJETIVO DEL CURS0



TEMA III.
TÉCNICAS ALGORÍTMICAS PARA LA SOLUCIÓN DE PROBLEMAS

III. TÉCNICAS ALGORÍTMICAS PARA LA SOLUCIÓN DE PROBLEMAS

OBJETIVO	Al terminar este tema, el participante mediante la lectura y la exposición del instructor, comprenderá las diferentes técnicas algorítmicas existentes con la finalidad de resolver problemas orientados a computadoras.

CONTENIDO

INTRODUCCIÓN

Cuando hayamos estudiado y comprendido este tema, habremos dado el primer paso para diseñar algoritmos, ya que sabremos cuales son las 3 diferentes técnicas que existen para crearlos.

Es por lo anterior donde radica la importancia de este módulo, debido que a partir de este momento podremos identificarnos con que técnica algorítmica nos sentimos más a gusto y con cual tendremos mayor facilidad de uso.

Este tema se encuentra dividido en tres secciones, donde cada una aborda a uno de los diferentes métodos.

- El primer subtema nos presenta a la técnica algorítmica NO gráfica llamada **Pseudocódigo**.

- El segundo subtema nos muestra a la técnica gráfica para la resolución de problemas orientados a computadoras llamada **Diagramas De Flujo**, que según mi consideración es la más fácil y entendible de las 3 tácticas.

- El tercer subtema nos exhibe al método híbrido llamado **Diagramas Nassi-Schneiderman** o **Diagramas N-S**.

Este tema junto con el primero son los más fáciles de todo el curso debido a que son teóricos, sin embargo, no por eso habrá que restarles importancia, por lo cual se espera que espera que lo asimiles al 100%.

3.1 Pseudocódigo

El pseudocódigo o pseudolenguaje, son una serie de instrucciones en nuestro lenguaje natural (español, ingles, etc.) y expresiones que representan cada uno de los pasos que resuelven un problema especifico (algoritmo) [12].

Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado. El pseudocódigo utiliza palabras que indican el proceso a realizar, por todo lo anterior es una técnica **NO GRÁFICA**.

Se considera un primer borrador, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. Cabe señalar que el pseudocódigo no puede ser ejecutado por una computadora.

12 DEITEL H.M. / DEITEL P.J., "Como Programar en C/C++", Ed. Prentice Hall, México

La forma en que se escribe un pseudocódigo es la siguiente:

1. Se escribe la palabra **pseudocódigo** seguida de dos puntos y a continuación un nombre que describa de manera general el problema a resolver.

2. En caso de haber **estructuras** se describen en la sección con este nombre, si no hay se pueden omitir.

3. En caso de haber **funciones o módulos** se describen en la sección con este nombre, si no hay se pueden omitir.

4. En caso de haber **constantes** se describen en la sección con este nombre, si no hay se pueden omitir.

5. En caso de haber **variables** se describen en la sección con este nombre, si no hay se pueden omitir.

6. Se colocan en orden las instrucciones y expresiones a ejecutar, las cuales deben de estar enumeradas, donde se debe respetar lo siguiente:

   - La primera instrucción es la palabra **inicio**.
   - La última instrucción es la palabra **fin**.
   - En caso de estar dentro de una sentencia de selección o dentro de una estructura cíclica, utilizar una subnumeración y una sangría.
   - Indicar siempre el final de la estructura de selección o estructura cíclica antes de continuar con la numeración normal.

A continuación tenemos el ejemplo de un pseudocódigo, el cual no realiza nada específico, pero se muestra la estructura que debe de tener.

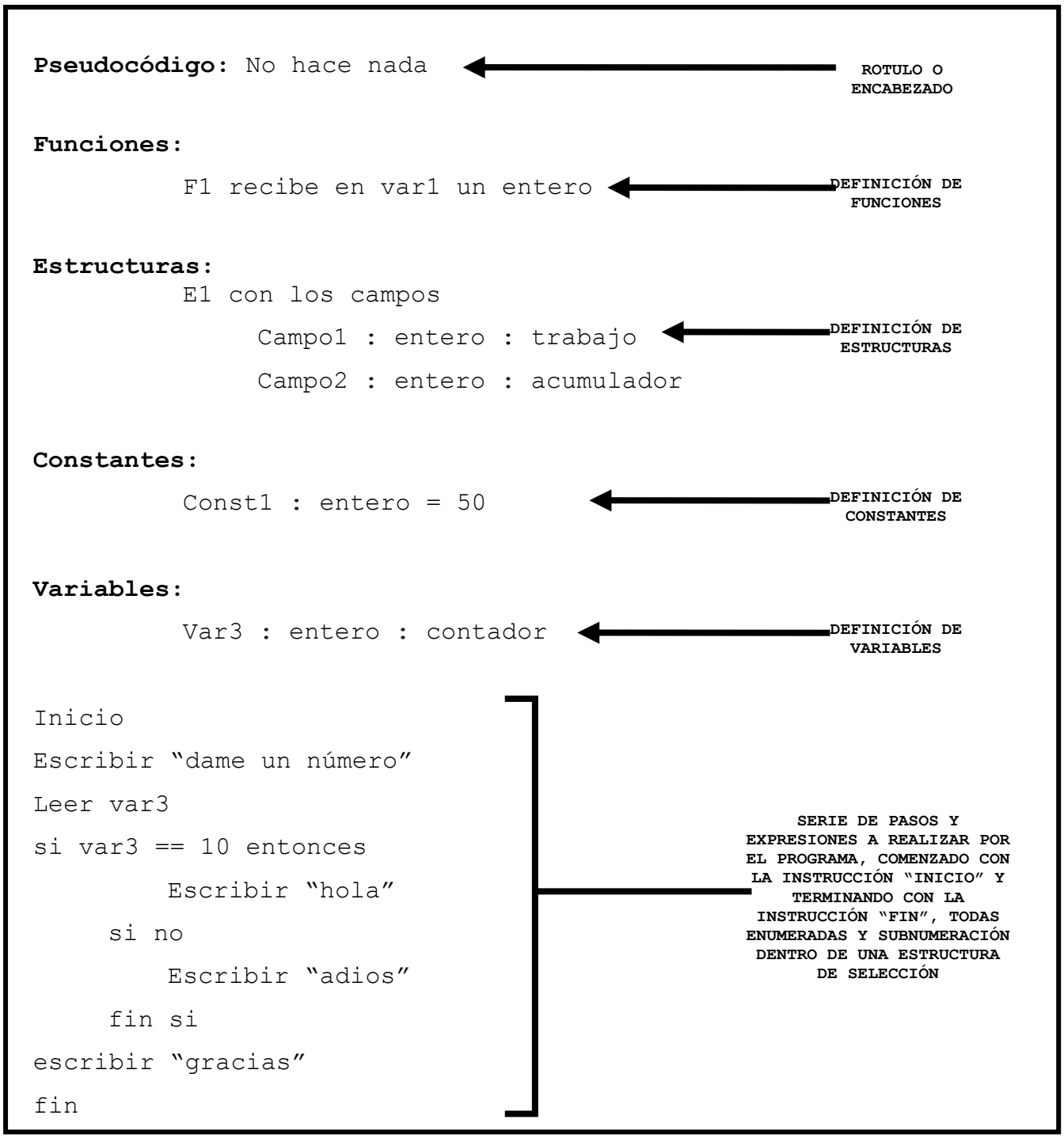


Ilustración 2. Ejemplo de cómo se debe escribir y estructurar un pseudocódigo.

En la sección en la que se colocan los pasos y expresiones a realizar para resolver un problema específico utilizamos para definir una tarea o proceso determinado las siguientes palabras:

Inicio, **Fin**. Indica el comienzo y término del algoritmo.

Escribir. Muestra mensajes e información en el monitor.

Imprimir. Datos y mensaje que son enviados a la impresora.

Leer. Almacena un dato que es capturado desde el teclado en una variable.

Guardar en... Indica el(los) dato(s) a guardar en una ubicación específica de un dispositivo de almacenamiento secundario (disquete, disco duro, CD, etc.).

Recuperar desde... Indica la ubicación específica de un dispositivo de almacenamiento secundario (disquete, disco duro, CD, etc.) desde el cual se va a leer información y en donde se almacenará temporalmente esta.

Llamar a... Indica que se debe de ejecutar a la función o módulo que se esta invocando.

Si ... entonces. Es una pregunta para una estructura de selección, donde si la respuesta es verdad se realizan unas tareas especificas y cuando es falso se pueden realizar otras.

Si no. Indica el comienzo de las instrucciones a realizar cuando la respuesta a la pregunta ***si...entonces*** es falsa.

Fin si. Indica el término de la estructura condicional ***si...entonces***.

Casos para... / **Fin casos**. Indica las acciones a realizar cuando una variable puede tener uno de varios posibles valores.

Hacer mientras... / **fin mientras**. Estructura cíclica la cual indica un conjunto de instrucciones que se deben de repetir mientras que la respuesta a la pregunta ***hacer mientras...*** sea verdadera.

Repetir / **hasta**.. Estructura cíclica la cual indica un conjunto de instrucciones que se deben de repetir mientras que la respuesta a la pregunta ***hasta...*** sea falsa.

Hacer para... hasta ... / **fin para**. Estructura cíclica la cual indica el número exacto de veces que un conjunto de instrucciones que se deben de repetir.

//... Indica que es comentario, el cual solo sirve para documentar nuestra solución puesto que no se ejecuta ninguna instrucción.

Ventajas de utilizar un Pseudocódigo

Ocupa muy poco espacio en una hoja de papel

Permite representar en forma fácil operaciones repetitivas complejas

Es muy fácil pasar de pseudocódigo a un programa en algún lenguaje de programación ya que solo basta con aprender como se maneja cierta instrucción en ese lenguaje.

Si se siguen las reglas se puede observar claramente los niveles que tiene cada operación debido a la numeración, subnumeración y sangrías.

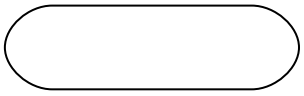
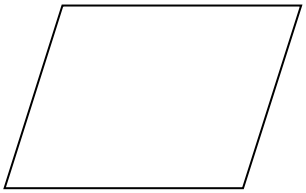
3.2 Diagramas De Flujo

Un diagrama de flujo es la representación gráfica de un algoritmo [13]. También se puede decir que es la representación detallada en **Forma Gráfica** de como deben realizarse los pasos en la computadora para producir resultados.

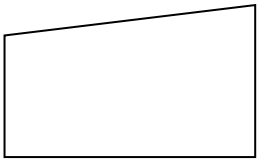
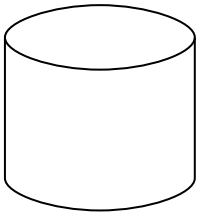
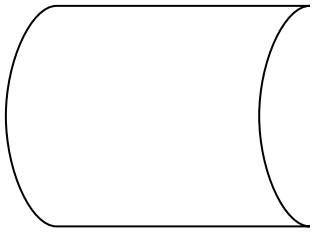
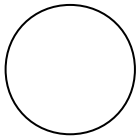
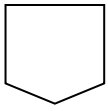
Esta representación gráfica se da cuando varios símbolos (que indican diferentes procesos en la computadora), se relacionan entre si mediante líneas que indican el orden en que se deben ejecutar los procesos.

Nota. Estos procesos son casi los mismos que se detallaron en la técnica no gráfica (pseudocódigo), pero representados con símbolos.

Los símbolos utilizados han sido normalizados por el instituto norteamericano de normalización (ANSI).

SÍMBOLO	NOMBRE	DESCRIPCIÓN
	Terminador	Indica el comienzo o termino de nuestro algoritmo, para eso se debe de identificar con la palabra ***inicio*** ó ***fin***.
	Proceso	Dentro de el se coloca una expresión para que se ejecute.
	Datos	Dentro de este símbolo se declaran las funciones, módulos, estructuras, constantes y variables a utilizar durante el algoritmo.
SÍMBOLO	**NOMBRE**	**DESCRIPCIÓN**

13 FERREYRA Cortés Gonzalo, "Informática, Para Cursos De Bachillerato", Ed. Alfaomega, México

SÍMBOLO	NOMBRE	DESCRIPCIÓN
	Entrada manual	Indica que se recibe un dato desde el teclado y dentro de este se coloca la variable en donde se almacenará.
	Pantalla	Dentro de el se coloca el mensaje y datos que queremos aparezcan en el monitor.
	Impresora o documento	Dentro de el se coloca el mensaje y datos que queremos mandar a la impresora.
	Almacenamiento	Indica el(los) dato(s) a guardar en una ubicación específica de un dispositivo de almacenamiento secundario (disquete, disco duro, CD, etc.).
	Datos almacenados	Indica la ubicación específica de un dispositivo de almacenamiento secundario (disquete, disco duro, CD, etc.) desde el cual se va a leer información y en donde se almacenará temporalmente esta.
	Llamada a función o módulo	Indica que se debe de ejecutar a la función o módulo que esta escrita dentro de él.
	Conector en la misma página	Se utiliza para continuar la secuencia del algoritmo en otra parte de la hoja. El conector debe de estar en ambos lados y con el mismo número.
	Conector con otra página	Se utiliza para continuar la secuencia del algoritmo en otra página. El conector debe de estar en ambos lados y con el mismo número.

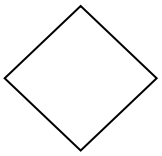
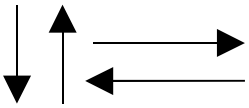
	Decisión	Se utiliza para plantear una pregunta y con la respuesta se optará por avanzar por solo uno de los caminos posibles.
	Flechas	Se usan para indicar el flujo o camino a seguir por el programa.

Tabla 19 Conjunto de símbolos para diseñar diagramas de flujo.

Nota. El símbolo de decisión es utilizado para representar a las estructuras cíclicas y a las estructuras de selección.

Reglas para diseñar un buen diagrama de Flujo

Al no haber un símbolo para colocar el encabezado del diagrama, se recomienda colocarlo en la parte superior como un comentario.

Se debe comenzar el algoritmo con el símbolo inicio, al igual que indicar el término con el símbolo **fin**.

Después del símbolo inicio, se colocan todas las funciones, módulos, estructuras, variables y constantes a usar en el símbolo datos.

Nota. La descripción de la función o módulo se debe de realizar en un diagrama de flujo independiente.

Todas las líneas que conectan a dos símbolos deben de tener una punta de flecha. Una flecha con doble sentido es incorrecta.

Se deben se usar solamente líneas de flujo horizontal y/o vertical.

Se debe evitar el cruce de líneas utilizando los conectores.

Se deben usar conectores solo cuando sea necesario.

No deben quedar líneas de flujo sin conectar.

Se deben trazar los símbolos de manera que se puedan leer de arriba hacia abajo y de izquierda a derecha.

Todo texto escrito dentro de un símbolo deberá ser escrito claramente, evitando el uso de muchas palabras.

Al tomar una decisión, se debe indicar el valor de los caminos posibles, generalmente son falso y verdadero.

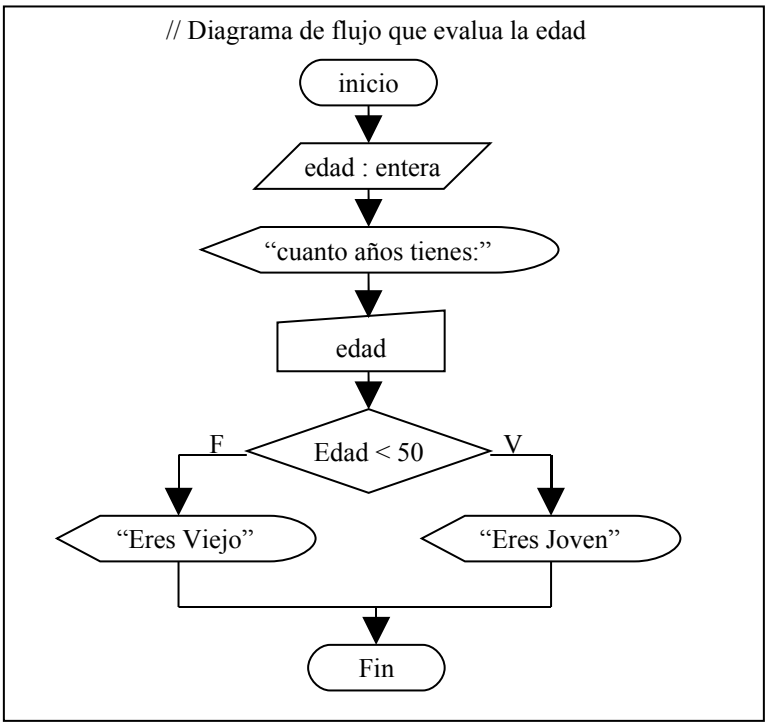


Ilustración 3 Ejemplo de cómo debe estar diseñado un diagrama de flujo

3.3 Diagramas Estructurados (Nassi-Schneiderman)

El diagrama estructurado Nassi-Scheneiderman también conocido como **diagrama de Chapín** es una mezcla de un diagrama de flujo con el pseudocódigo. Este diagrama se denomina así en honor a sus inventores.

Se parece al **diagrama de flujo** ya que de manera visual podemos identificar el camino que se sigue para resolver un algoritmo, pero sin utilizar flechas, ya que todas las acciones se colocan en cajas contiguas. Se parece al **pseudocódigo** debido a que cada acción a realizar se escribe de igual manera pero dentro de las ya mencionadas cajas sin utilizar una numeración la cual en ocasiones es difícil de realizar.

Un buen diagrama N-S debe de cumplir con lo siguiente:

En la primera caja de acción se coloca el encabezado o titulo del algoritmo.

En la segunda la palabra inicio y en la última la instrucción fin.

En la tercera las variables, funciones (solo la declaración), estructuras y constantes.

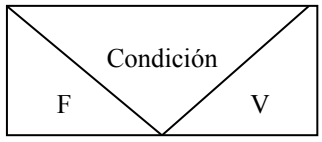
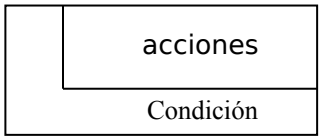
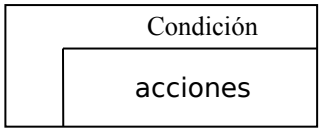
En caso de las estructuras de selección y cíclicas, indicar el valor del camino posible para evitar confusiones.

Debe quedar diseñado completamente en una sola página; nunca se debe continuar un diagrama en otra hoja (salvo un módulo o función), ya que este es considerado como una sola acción.

Sin embargo, presenta los siguientes inconvenientes:

- Difíciles de entender cuando el problema se vuelve muy complejo
- Difíciles de actualizar y cuando se tienen que modificar, la labor se vuelve tediosa pues hay que redibujar las cajas que los componen.
- Generalmente no caben en el ancho de una hoja.
- se necesitan hojas grandes para diseñarlos y aun así es muy probable que no sea el espacio suficiente.

Aunque cada una de las acciones a realizar se coloca dentro de una caja, las estructuras de selección y cíclicas tienen una pequeña variación.

SÍMBOLO	NOMBRE	DESCRIPCIÓN
	Condición	Es una pregunta, donde si la respuesta es verdad se realizan unas tareas especificas y cuando es falso se pueden realizar otras.
	Repetir hasta	Indica el conjunto de acciones que se deben de repetir **hasta** que la respuesta a la condición sea verdadera.
	Hacer mientras	Indica el conjunto de acciones que se deben de realizar **mientras** que la respuesta a la condición sea verdadera.

Condición acciones	Hacer Para	Indica el número exacto de veces que un conjunto de instrucciones se deben de repetir.

Tabla 20 Símbolos para estructuras de Selección y Cíclicas

A continuación tenemos un ejemplo de cómo se debe diseñar un diagrama N-S:

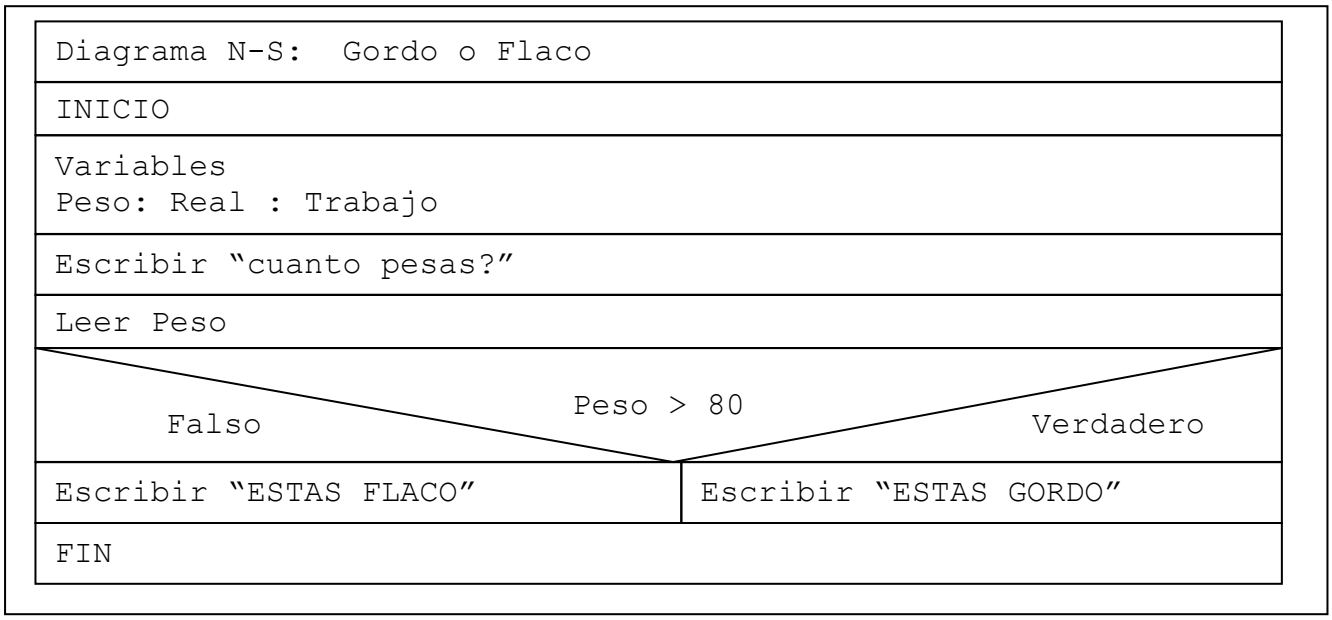


Ilustración 4 Ejemplo de un diagrama N-S

CONCLUSIÓN

El presente tema, nos mostró las tres diferentes técnicas existentes para diseñar algoritmos.

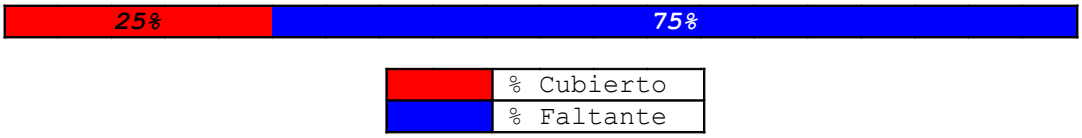
El primer subtema, nos enseñó la estructura básica del Pseudocódigo, el cuál es un lenguaje informal para resolver problemas orientados a computadoras y el cual es del tipo Escrito o No Gráfico.

El segundo subtema nos describió la estructura y los símbolos utilizados para resolver un problema orientado a computadoras por medio de la técnica algorítmica de Diagramas de Flujo, los cuales son del tipo gráfico.

El tercer subtema, nos mostró la técnica algorítmica de los Diagramas Estructurados N-S, la cual es considerada del tipo híbrido ya que mezcla símbolos y texto para poder representar la solución a un problema orientado a computadoras.

En este tema se ha dado un pequeño paso para lograr el objetivo general del curso, pero aún así, es muy importante su total asimilación, ya que aquella persona que no comprenda correctamente la estructura de las tres diferentes técnicas algorítmicas, difícilmente diseñará algoritmos eficaces para la solución de problemas orientados a computadoras, por lo cual se considera haber avanzado otro 5%.

OBJETIVO DEL CURS0



TEMA IV.
ESTRUCTURAS DE CONTROL

IV. ESTRUCTURAS DE CONTROL

OBJETIVO	Al terminar este tema, el participante mediante la elaboración de ejercicios, manejará las diferentes estructuras de control utilizando las tres técnicas algorítmicas con la finalidad de resolver problemas orientados a computadoras.

CONTENIDO

INTRODUCCIÓN

Después de haber conocido en el tema pasado las diferentes técnicas algorítmicas, en el presente tema las vamos a utilizar para resolver problemas enfocados a computadoras. Pero como veremos estos podrán utilizar tres estructuras diferentes.

Estos algoritmos tendrán la cualidad de llevar un orden progresivo, tomar decisiones y si es necesario repetir un bloque de instrucciones un determinado número de veces.

Por lo antes mencionado, este tema requiere de toda nuestra capacidad para comprenderlo en su totalidad, puesto quien lo asimile se puede considerar prácticamente un programador, ya que solo tendrá que adaptar sus algoritmos a un lenguaje de programación. Además de que los temas siguientes solo bastará con adaptarlos a estos conocimientos. Lamentablemente quien no se sienta seguro de lo aprendido, tendrá que repasar nuevamente el módulo.

Este tema se encuentra dividido en tres subtemas, donde cada uno maneja su funcionamiento con las técnicas ya estudiadas.

El primer subtema es el más sencillo ya que habla de las estructuras secuénciales, en las cuales no hay decisiones que tomar y tan solo basta con analizar cuales son los datos de entrada necesarios para producir las salidas deseadas.

El segundo subtema nos guía a diseñar algoritmos inteligentes, es decir que toman decisiones, para lo cual se manejan las estructuras condicionales; de las cuales existen las condicionales sencillas que son aquellas en donde solo existen dos caminos (falso y verdadero) y las de selección múltiple en las que los caminos posibles son inmensos.

En el tercer subtema, crearemos algoritmos en los cuales un bloque de instrucciones se repite dependiendo de la respuesta de una condición, a esto es a lo que comúnmente llamaremos ciclos y existen de manera general 3 diferentes: los que se ejecutan un número exacto de veces, los que se repiten hasta que la respuesta sea verdadera y los que se ciclan mientras la respuesta sea verdadera.

En este tema nos encontramos con ejemplos para asimilar el funcionamiento de las estructura y con decenas de ejercicios, los cuales esperamos que resuelvas ya que mediante la práctica es como se te formará una mentalidad de programador y solo así se podrá alcanzar el objetivo de este tema y el objetivo del curso.

4.1 Estructuras Secuénciales

Los algoritmos más sencillos de realizar son los que no toman decisiones, tan solo se dedican a realizar o ejecutar instrucción tras instrucción en el orden determinado.

Estos algoritmos están representados por las estructuras secuénciales, en las que una acción (instrucción) sigue a otra en **secuencia**. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso.

De manera general un algoritmo con una estructura secuencial se representa de la siguiente forma en las tres diferentes técnicas algorítmicas (el siguiente ejemplo no realiza nada en específico, solo es de carácter ilustrativo):

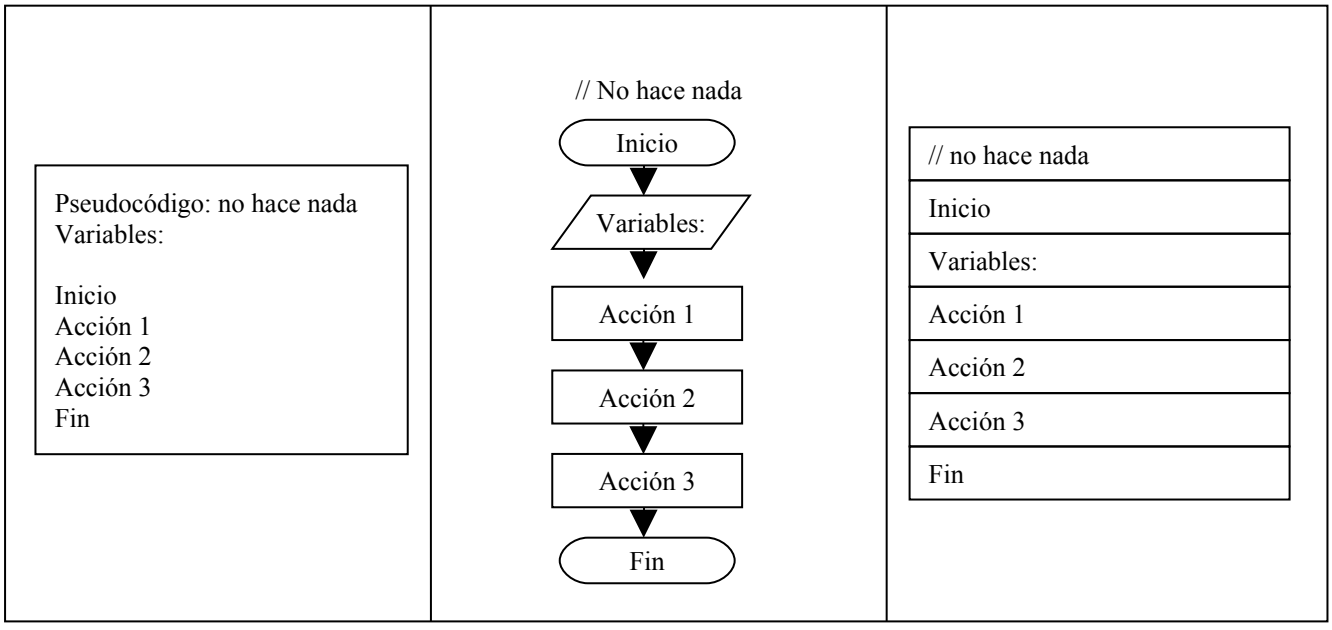


Ilustración 5 Ejemplo de cómo se diseña un algoritmo secuencial.

En las estructuras secuénciales, se encuentran las acciones o instrucciones de **inicio** y **fin**, **escribir** en monitor, **imprimir** en impresora, **leer** desde el teclado, **guardar en** una ubicación

específica, **recuperar desde** una ubicación específica, **llamar** y ejecutar a una función o módulo y la **ejecución de expresiones aritméticas** para obtener un resultado guardándolo en una variable. El uso de estas acciones ya fue explicado en el tema II y III.

A continuación vamos a realizar algunos ejemplos, resolviéndolos en las tres técnicas algorítmicas, para lo cual debemos de recordar que para diseñar un algoritmo, debemos de realizar tres pasos:

1. **Analizar el problema** que se nos esta planteando. En este análisis hay que identificar cuales son los **datos de salida**, es decir, los resultados que debe de arrojar nuestro algoritmo; identificar cuales son los **datos de entrada** necesarios para lograr los resultados esperados, es decir, los datos que nos tiene que dar el usuario; identificar los **procesos** a realizar con los datos de entrada para obtener los datos de salida, en otras palabras las expresiones a calcular; y en caso de ser necesario identificar los datos que permanecen **constantes** durante todo el proceso o algoritmo.

2. **Diseñar el Algoritmo** en alguna de las tres técnicas algorítmicas conocidas, pero en estos casos serán todas.

3. **Probar el algoritmo** para evitar un posible error lógico, para lo cual se hace una corrida de escritorio, lo cual significa dar valores ficticios a las variables y checar los resultados.

Ejemplo ❶	Realizar un algoritmo que calcule la edad de una persona a la cual solo se le solicitará el año en que nació.		
Paso I. Analizar el problema.			
Cada uno de estos datos se debe de expresar en variables y no en frases largas.			
Salidas	**Entrada**	**Constantes**	**Procesos**
▪ Edad	▪ Año_nac		▪ Edad = Año_act – Año_nac

- Año_act

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Edad personal

Variables:
       Edad: entera : trabajo // almacenará la edad del usuario
       Año_nac: entera : trabajo // guardará el año en que nació
       Año_act: entera : trabajo // Contendrá el año en que estamos

   1.  Inicio
   2.  Escribir "En que año naciste?" // muestra el mensaje que esta entre comillas
   3.  Leer Año_nac // guarda el dato que es tecleado por el usuario en la variable
   4.  Escribir "En que año estamos?"
   5.  Leer Año_act
   6.  Edad = Año_act – Año_nac // realiza una operación y almacena el resultado en
                                // la variable de la izquierda de la expresión.
   7.  Escribir "Tu edad actual es:", Edad // Cuando deseamos mostrar el contenido
                                           // de una variable, esta no debe de
                                           // estar entre comillas
   8.  Fin

// Es recomendable poner comentarios en todos nuestros algoritmos, ya que esto los
// hace más entendibles no solo para nosotros sino para cualquier persona.
```

DIAGRAMA DE FLUJO

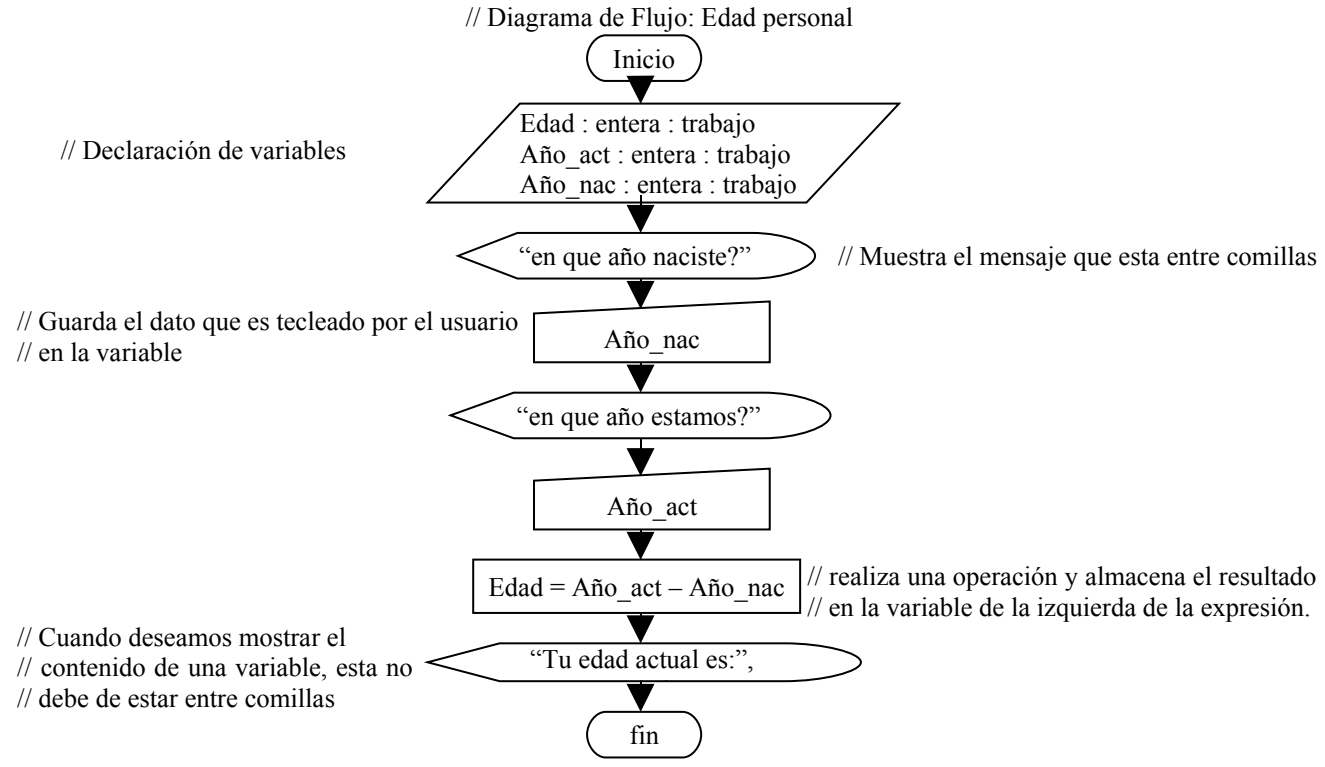


Diagrama N-S

// diagrama N-S : Edad Personal
Inicio
Variables:

Edad: entera : trabajo
Año_nac: entera : trabajo
Año_act: entera : trabajo
	// almacenará la edad del usuario
	// guardará el año en que nació
	// Contendrá el año en que estamos

Escribir "En que año naciste?"	// muestra el mensaje que esta entre comillas
Leer Año_nac	// guarda el dato que es tecleado por el usuario en la variable
Escribir "En que año estamos?"	
Leer Año_act	
Edad = Año_act – Año_nac	// realiza una operación y almacena el resultado en la variable de la // izquierda de la expresión.
Escribir "Tu edad actual es:", Edad	// Cuando deseamos mostrar el contenido de una variable, esta no // debe de estar entre comillas
Fin	

Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
Año_nac = 1977 Año_act = 2004	Edad = Año_act – Año_nac Edad = 2004 - 1977	**Edad = 27**

Tabla 21 Ejemplo 1 de una estructura secuencial

Nota. Con el paso del tiempo y con la práctica, no será necesario escribir los pasos I y II, ya que estos se pueden realizar mentalmente o en un pedazo de papel, pero no de manera formal.

Nota. Con la práctica será posible solo declarar la variable y su tipo, sin necesidad de indicar su uso.

Ejemplo ❷	Supongamos que en una tortillería se necesita un sistema que calcule y le muestre el total a pagar por cada cliente, si sabemos que cada kilo de tortilla cuesta $4.50.

Paso I. Analizar el problema.

Salidas	Entrada	Constantes	Procesos
▪ Total	▪ Kilos	▪ P_kilo = 4.5	▪ Total = kilos * P_kilos

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: total a pagar

Constantes:
          P_kilo: real = 4.5

Variables:
          Total : real : trabajo
          Kilos : real : trabajo
   1. Inicio
   2. Escribir "Cuantos Kilos quieres?"
   3. Leer kilos
   4. Total = Kilos * P_kilos
   5. Escribir "el total a pagar es:", Total
```

6. Fin

DIAGRAMA DE FLUJO

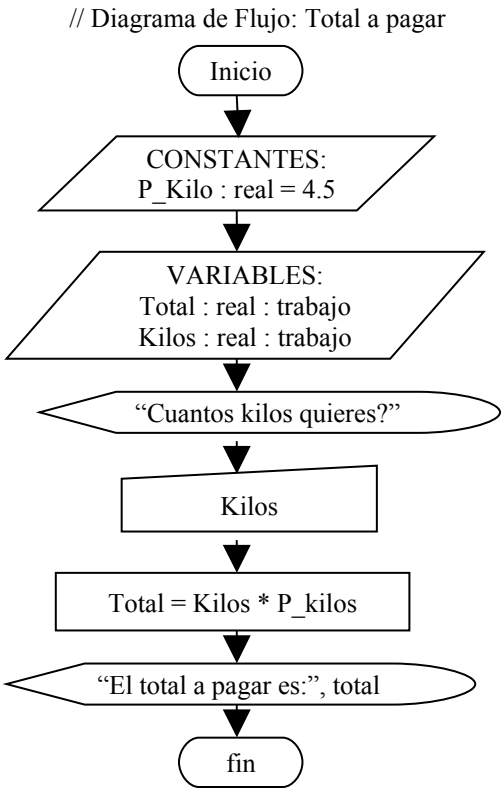


Diagrama N-S

```
// diagrama N-S : total a pagar
Inicio
Constantes:
          P_kilo: real = 4.5
Variables:
          Total : real : trabajo
          Kilos : real : trabajo
Escribir "Cuantos Kilos quieres?"
Leer kilos
Total = Kilos * P_kilos
Escribir "el total a pagar es:", Total
Fin
```

Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
Kilos = 3.5	Total = Kilos * P_kilos Total = 3.5 * 4.5	**Total = 15.75**

Tabla 22 Ejemplo 2 de una estructura secuencial

✍**Ejemplo ❸**	Suponga que un individuo desea invertir su capital en un banco y desea saber cuanto dinero ganará después de un año si el banco paga a razón de 2% mensual.

Paso I. Analizar el problema.

Salidas	Entrada	Constantes	Procesos

▪ Ganancia	▪ Capital	▪ Interes = 0.02 ▪ Año = 12	Ganancia =(Capital * Interes) * Año

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Ganancias Anuales

Variables:
          Ganancia : real : trabajo
          Capital : real : trabajo

Constantes:
          Interes : real = 0.02
          Año : entero = 12

   1. Inicio
   2. Escribir "cuanto dinero piensas invertir?"
   3. Leer Capital
   4. Ganancia = ( Capital * Interes ) * Año
   5. Escribir "Tu ganancia será de:", Ganancia
   6. Fin
```

DIAGRAMA DE FLUJO

// Diagrama de Flujo: Ganancias anuales

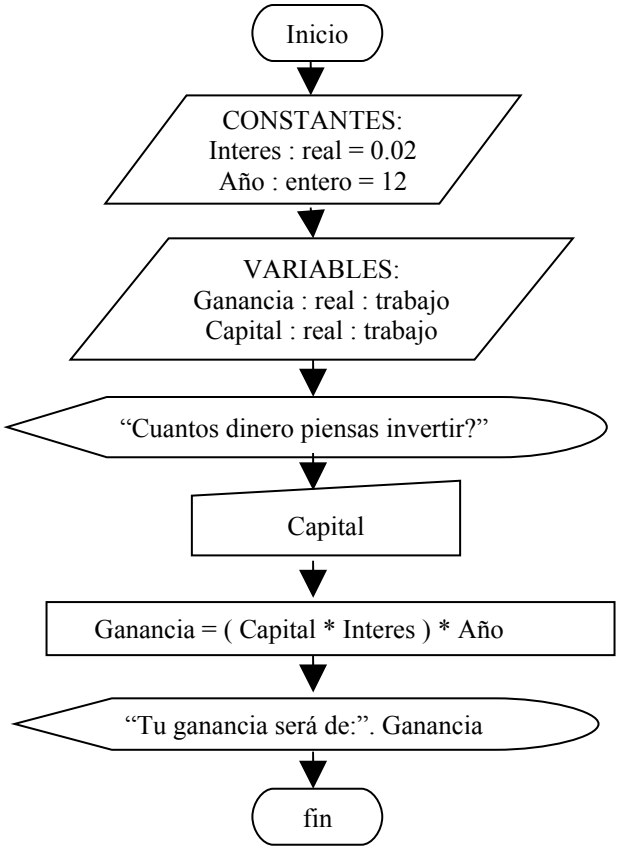


Diagrama N-S

// diagrama N-S : Ganancias Anuales
Inicio
Variables:

```
            Ganancia : real : trabajo
            Capital : real : trabajo
Constantes:
            Interes : real = 0.02
            Año : entero = 12
Escribir "cuanto dinero piensas invertir?"
Leer Capital
Ganancia = ( Capital * Interes ) * Año
Escribir "Tu ganancia será de:", Ganancia
Fin
```

Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
Capital = 10000	Ganancia = (Capital * Interes) * Año Ganancia = (10000 * 0.02) * 12	**Ganancia = 2400**

Tabla 23 Ejemplo 3 de una estructura secuencial.

Ejercicios. Resuelve lo que se te pide.

I. Diseña un algoritmo para cada uno de los problemas que se te plantean, utilizando las tres técnicas algorítmicas.

1. Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuanto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.
2. Una tienda ofrece un descuento del 15% sobre el total de la compra y un cliente desea saber cuanto deberá pagar finalmente por su compra.
3. Un alumno desea saber cual será su calificación final en la materia de Algoritmos. Dicha calificación se compone de tres exámenes parciales.
4. Un maestro desea saber que porcentaje de hombres y que porcentaje de mujeres hay en un grupo de estudiantes.
5. Dada una cantidad en pesos, obtener la equivalencia en dólares, asumiendo que la unidad cambiaria es un dato desconocido.
6. Calcular el nuevo salario de un obrero si obtuvo un incremento del 25% sobre su salario anterior.
7. Calcular el área de un círculo.
8. Convertir una distancia en metros a pies y pulgadas.
9. Elevar al cubo un número.
10. Desplegar el peso dado en kilos de una persona en gramos, libras y toneladas.

4.2 Estructuras Condicionales

Las estructuras condicionales comparan una variable contra otro(s) valor(es), para que en base al resultado de esta comparación, se siga un curso de acción dentro del programa.

Estas estructuras son las que nos dan la capacidad de crear sistemas inteligentes, es decir, que toman decisiones.

Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite. Existen dos tipos básicos, las simples y las múltiples.

Condiciones Simples. Son aquellas en que solamente se puede escoger uno de dos caminos posibles y al seleccionar se ejecutarán las instrucciones que se encuentren dentro de este. Esto es similar a la situación que sufrimos cuando nos encontramos en la punta de una cuchilla, solamente se puede ir por un camino ya que es imposible cruzar por ambos a la vez.

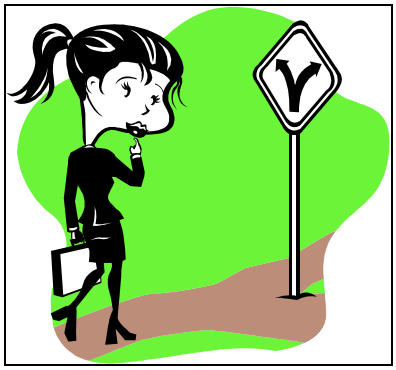


Ilustración 6 Condición simple

Condiciones Múltiples. Son aquellas en que solamente se puede escoger uno de **n** caminos posibles, y al seleccionar se ejecutarán las instrucciones que se encuentren dentro de este. Esto es similar a la situación que sufrimos cuando nos encontramos en un cruce de caminos, solamente se puede ir por un camino ya que es imposible cruzar por todos a la vez.

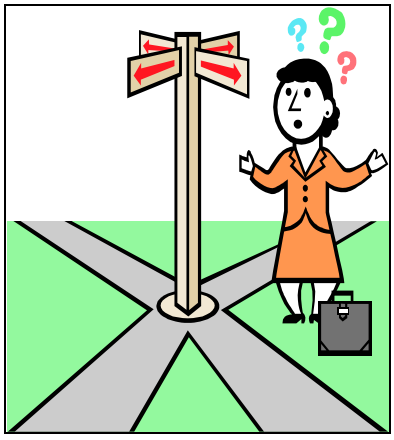


Ilustración 7 Condición Múltiple

En este momento analizaremos a las condiciones simples con las tres técnicas algorítmicas ya conocidas.

En pseudocódigo se utiliza la instrucción **si ... entonces**, donde en lugar de los puntos suspensivos se coloca la expresión a evaluar (en esta parte es donde nos sirven los operadores lógicos y relacionales), donde si el resultado de esta evaluación es verdadero se ejecutan las instrucciones que se encuentran entre esta instrucción y las palabras **si no**; Pero si el resultado es falso, se ejecutan las instrucciones que se encuentran después de las palabras **si no** y las palabras **fin si**. Por lo cual podemos decir que los delimitadores de esta estructura son las palabra **si ... entonces** y **fin si**.

Las instrucciones que se encuentran dentro de la condición **si...entonces** pueden ser estructuras secuénciales y en este caso las acciones llevan una **subnumeración secuencial**, menos las palabras **si no** y **fin si**.

Sugerencia. Colocar una sangría a las acciones internas a la condición para tener una mejor legibilidad de nuestro algoritmo.

Al momento de diseñar un algoritmo con esta estructura se puede omitir el lado falso, es decir las instrucciones dentro del **si no** y el **fin si**; En caso de no desear hacer nada en esta parte.

Sugerencia. En caso de no desear hacer nada en el lado falso de la condición, es recomendable poner dentro de las palabras **si no** y **fin si** el siguiente comentario: ***// no hace nada***

```
1. Inicio
2. Acción 1
3. Acción 2
4. Si Variable operador  valor entonces
      4.1 Acción 3
```

```
        4.2 Acción 4
    Si no
        4.3 Acción 5
        4.4 Acción 6
    Fin si
5.  Acción 7
6.  Acción 8
7.  Fin
```

Ilustración 8 Estructura condicional simple en pseudocódigo.

En diagrama de flujo esta representada por el símbolo **decisión**, donde, dentro de este se coloca la expresión a evaluar, y del símbolo salen dos flujos o flechas donde cada una debe de tener la leyenda del camino posible (falso o verdadero), estos flujos indican el conjunto de acciones o instrucciones a realizar dependiendo de la respuesta a la condición. El final de la estructura se indica uniendo nuevamente los dos flujos en uno solo, en caso de no desear realizar acciones dentro del lado falso, se debe de sacar la forzosamente la flecha para tener una indicación de donde termina la estructura.

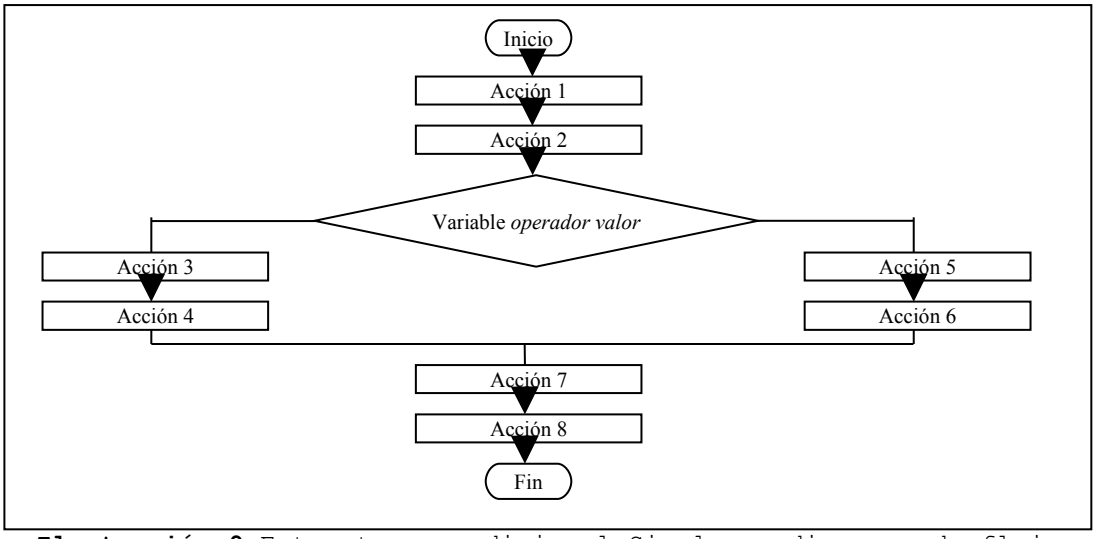


Ilustración 9 Estructura condicional Simple en diagrama de flujo

En diagrama N-S se utiliza para representar la estructura al símbolo **Condición**, donde en el triangulo de en medio se coloca la expresión a evaluar, en los triángulos izquierdo y derecho se

colocan las leyendas verdadero y falso respectivamente y debajo de estos las cajas que contienen cada una de las instrucciones a realizar. La estructura condicional simple termina colocando una caja con una instrucción que abarca todo el ancho de ambos caminos.

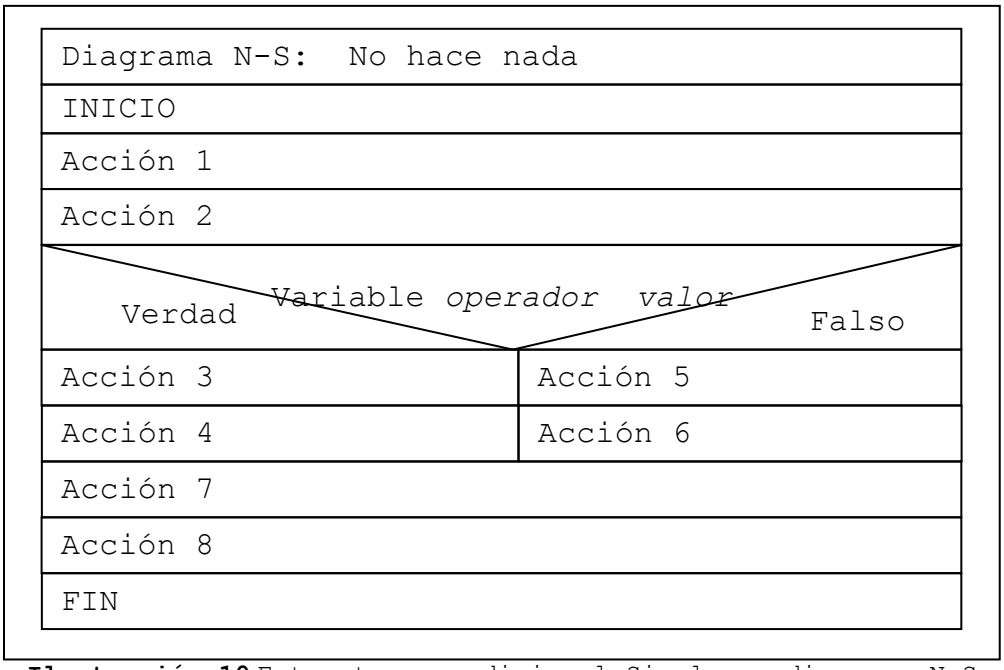


Ilustración 10 Estructura condicional Simple en diagrama N-S

A continuación realizamos unos ejemplos de estructuras condicionales simples, utilizando las tres técnicas algorítmicas. Pero además veremos otros ejemplos utilizando el concepto de **anidación**.

La anidación es el proceso de colocar dentro de una estructura ya sea condicional o cíclica otra estructura condicional o cíclica.

✍Ejemplo ❶	Se necesita un sistema para un supermercado, el cual dará un 10% de descuento a las personas que compren más de $1000, al cliente se le debe de dar el total a pagar.		
Paso I. Analizar el problema.			
Salidas	**Entrada**	**Constantes**	**Procesos**
▪ Total	▪ Subtotal		Cuando subtotal > 1000

	▪ Descuento		Descuento = Subtotal * 0.10 Total = Subtotal – Descuento Cuando Subtotal <= 1000 Total = Subtotal

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Supermercado

Variables:
          Total : real : trabajo
          Subtotal : real : trabajo
          Descuento : real : trabajo

   1.  Inicio
   2.  Escribir "Cuanto compró el cliente?"
   3.  Leer Subtotal
   4.  Si Subtotal > 1000 entonces
          4.1 Descuento = Subtotal * 0.10
          4.2 Total = Subtotal – Descuento

       si no
          4.3 Total = Subtotal
       fin si

   5.  Escribir "el total a pagar es:", Total
   6.  Fin
```

DIAGRAMA DE FLUJO

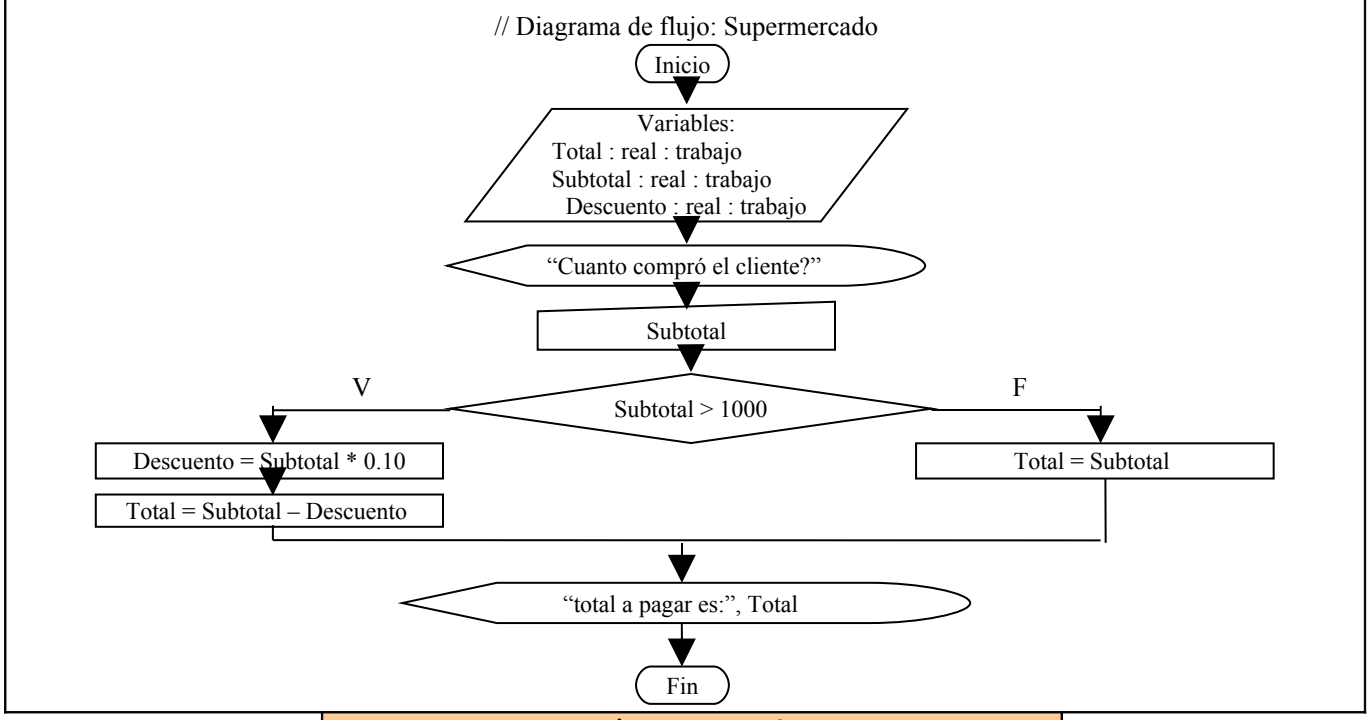
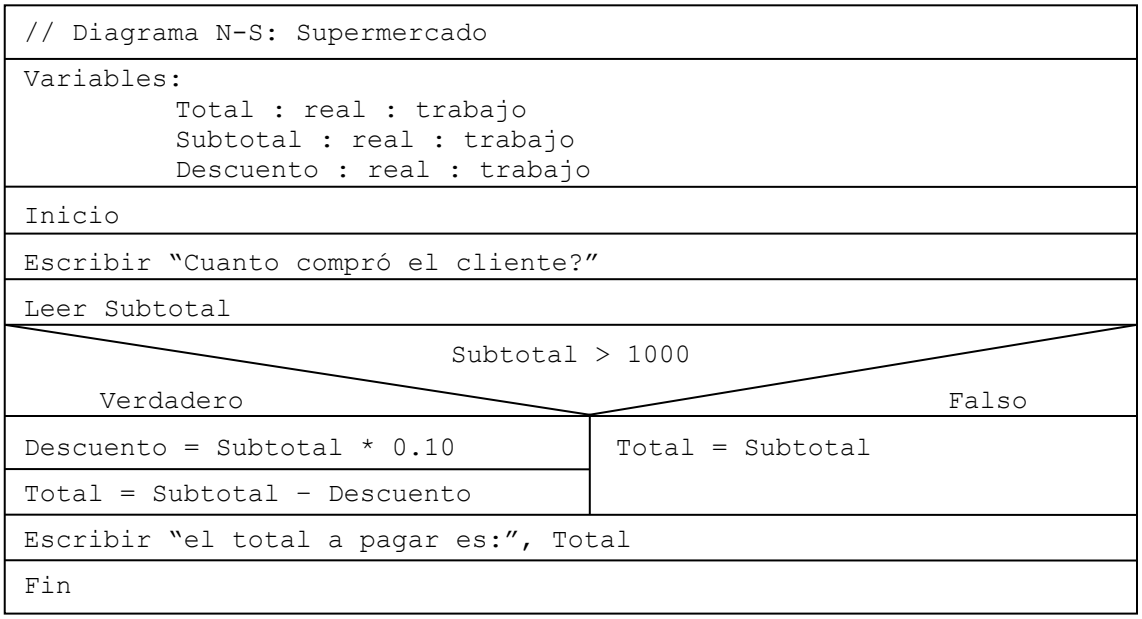


Diagrama N-S



Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
Subtotal = 750.80	Subtotal > 1000 750. 80 > 1000 → NO Total = Subtotal Total = 750.80	**Total = 750.80**
Subtotal = 1300	Subtotal > 1000 1300 > 1000 → SI Descuento = Subtotal * 0.10 Descuento = 1300 * 0.10 Descuento = 130 Total = Subtotal - Descuento Total = 1300 - 130 Total = 1170	**Total = 1170**

Tabla 24 Ejemplo 1 de una Estructura Condicional simple.

En este ejemplo, se dieron diferentes valores a subtotal para ver que es lo que pasa cuando se va por cualquier camino.

Aspecto Crítico. En este ejemplo, reciben el descuento del 10% solo aquellos clientes que su compra es **mayor** a $1000, los que compraron $1000 exactamente no reciben descuento. Por lo cual si se quiere que esta cifra tenga descuento, en la expresión debemos de colocar el operador **mayor o igual**.

✍**Ejemplo ❷**	Se necesita un sistema que reciba tres calificaciones parciales de un alumno y en base a estas darle su promedio donde si el promedio es menor a 6 se le dirá que esta **reprobado**, en caso contrario el mensaje será **aprobado**

Paso I. Analizar el problema.

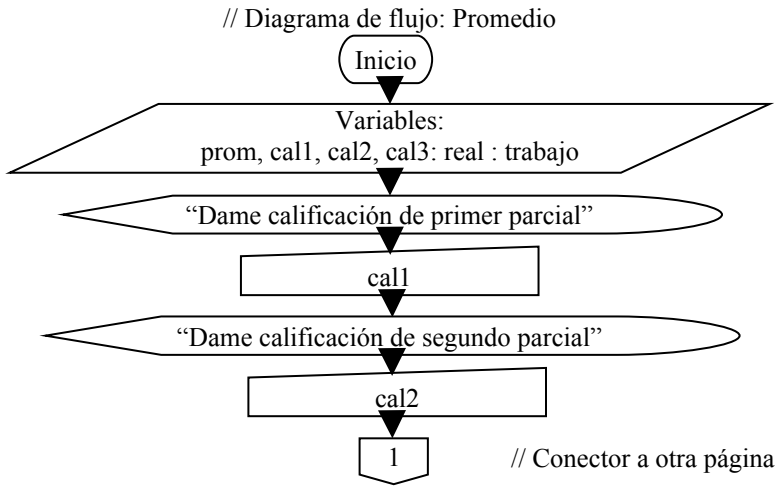
Salidas	Entrada	Constantes	Procesos
▪ Prom ▪ Un mensaje (Aprobado o Reprobado)	▪ Cal1 ▪ Cal2 ▪ Cal3		Prom = (cal1 + cal2 + cal3) / 3 Cuando Prom < 6 "REPROBADO" Cuando Prom >= 6 "APROBADO"

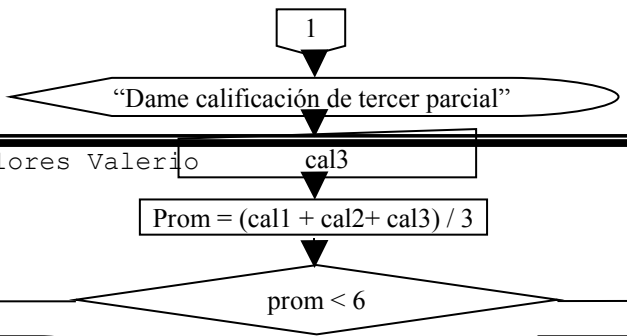
Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Promedio alumno

Variables:

prom, cal1, cal2, cal3 : real : trabajo

// La declaración de varias variables y constantes del mismo tipo se puede realizar
// en el mismo renglón siempre y cuando el nombre de cada una este separado por una
// coma.

   1. Inicio
   2. Escribir "dame calificación de primer parcial:"
   3. leer cal1
   4. Escribir "dame calificación de segundo parcial:"
   5. leer cal2
   6. Escribir "dame calificación de tercer parcial:"
   7. leer cal3
   8. prom = (cal1 + cal2 + cal3) / 3
   9. Si prom < 6 entonces
         9.1 Escribir "Tu promedio es:", prom, "y estas REPROBADO"

      Si no
         9.2 Escribir "Tu promedio es:", prom, "y estas APROBADO"
      Fin si

  10. Fin
```

DIAGRAMA DE FLUJO





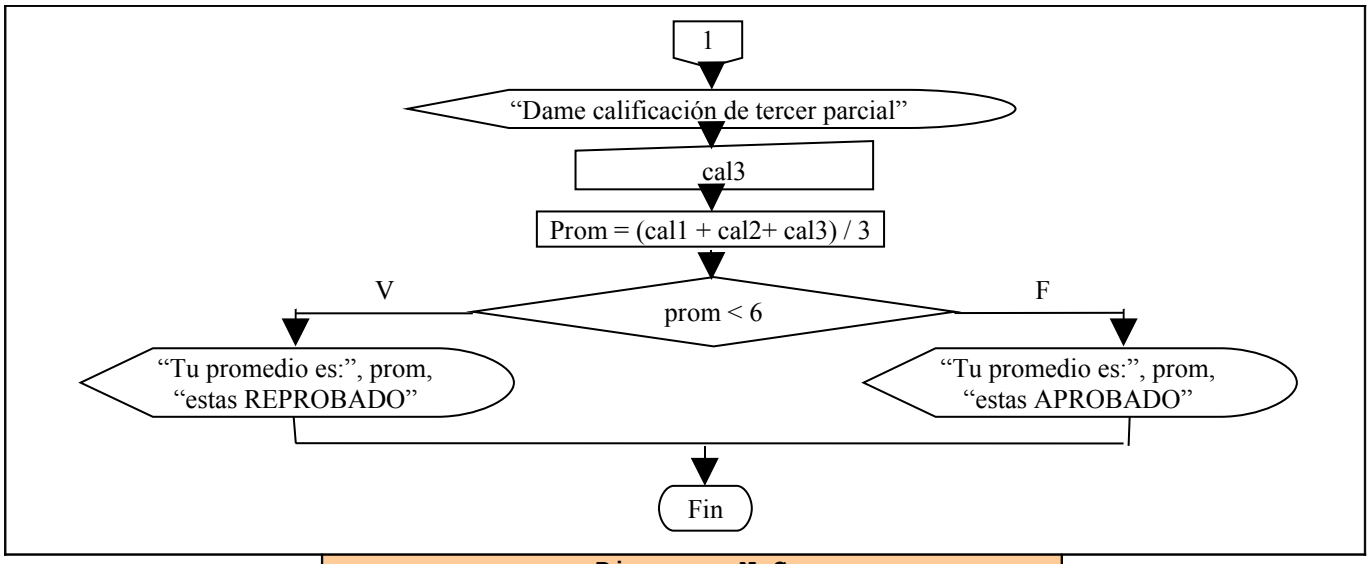


Diagrama N-S

Diagrama N-S: Promedio alumno	
Inicio	
Variables: Prom, cal1, cal2, cal3 : real : trabajo	
Escribir "dame calificación de primer parcial:"	
Leer cal1	
Escribir "dame calificación de segundo parcial:"	
Leer cal2	
Escribir "dame calificación de tercer parcial:"	
Leer cal3	
Prom = (cal1 + cal2 + cal3) / 3	
prom < 6	
Verdadero	Falso
Escribir "tu promedio es:", prom, "y estas REPROBADO"	Escribir "tu promedio es:", prom, "y estas APROBADO"
Fin	

Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
cal1 = 8.5 cal2 = 9.2 cal3 = 6.8	prom = (cal1 + cal2 + cal3) / 3 prom = (8.5 + 9.2 + 6.8) / 3 prom = 24.5 / 3 prom = 8.16 prom < 6 8.16 < 6 → NO	**Prom = 8.16** **"APROBADO"**
cal1 = 4.5 cal2 = 6.2 cal3 = 5.3	prom = (cal1 + cal2 + cal3) / 3 prom = (4.5 + 6.2 + 5.3) / 3 prom = 16 / 3 prom = 5.33 prom < 6 5.33 < 6 → SI	**Prom = 5.33** **"REPROBADO"**

Tabla 25 Ejemplo 2 de una estructura condicional simple

✍Ejemplo ❸	Se necesita un sistema para un supermercado, en el cual si el monto de la compra del cliente es mayor de $5000 se le hará un descuento del 30%, si es menor o igual a $5000 pero mayor que $3000 será del 20%, si no rebasa los $3000 pero si los $1000 la rebaja efectiva es del 10% y en caso de que no rebase los $1000 no tendrá beneficio.

Paso I. Analizar el problema.

Salidas	Entrada	Constantes	Procesos
▪ Total	▪ subtotal ▪ descuento		Cuando subtotal > 5000 descuento = subtotal * 0.30 total = subtotal – descuento Cuando subtotal > 3000 pero <= 5000 descuento = subtotal * 0.20 total = subtotal – descuento Cuando subtotal > 1000 pero <= 3000 descuento = subtotal * 0.10 total = subtotal – descuento Cuando subtotal <= 1000 total = subtotal

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Descuentos

Variables:

total, subtotal, descuento : real : trabajo = 0

// se inicializan todas las variables con el valor de cero

   1.  Inicio
   2.  Escribir "Cuanto compró el cliente?"
   3.  Leer subtotal
   4.  Si subtotal > 5000 entonces  // inicio de primera condición
            4.1 descuento = subtotal * 0.30

       si no  // lado falso de primera condición
            4.2 si subtotal > 3000 entonces // inicio de segunda condición
                     4.2.1 descuento = subtotal * 0.20

                 si no  // lado falso de segunda condición
                      4.2.2 si subtotal > 1000 entonces // inicio tercera cond.
                               4.2.2.1 descuento = subtotal * 0.10

                            si no // lado falso tercera condición
                                     // no hace nada

                            fin si  // fin de tercera condición

                 fin si  // fin de segunda condición

       fin si // fin de primera condición

   5.  total = subtotal - descuento
   6.  Escribir "el total a pagar es:", Total
   7.  Fin
```

DIAGRAMA DE FLUJO

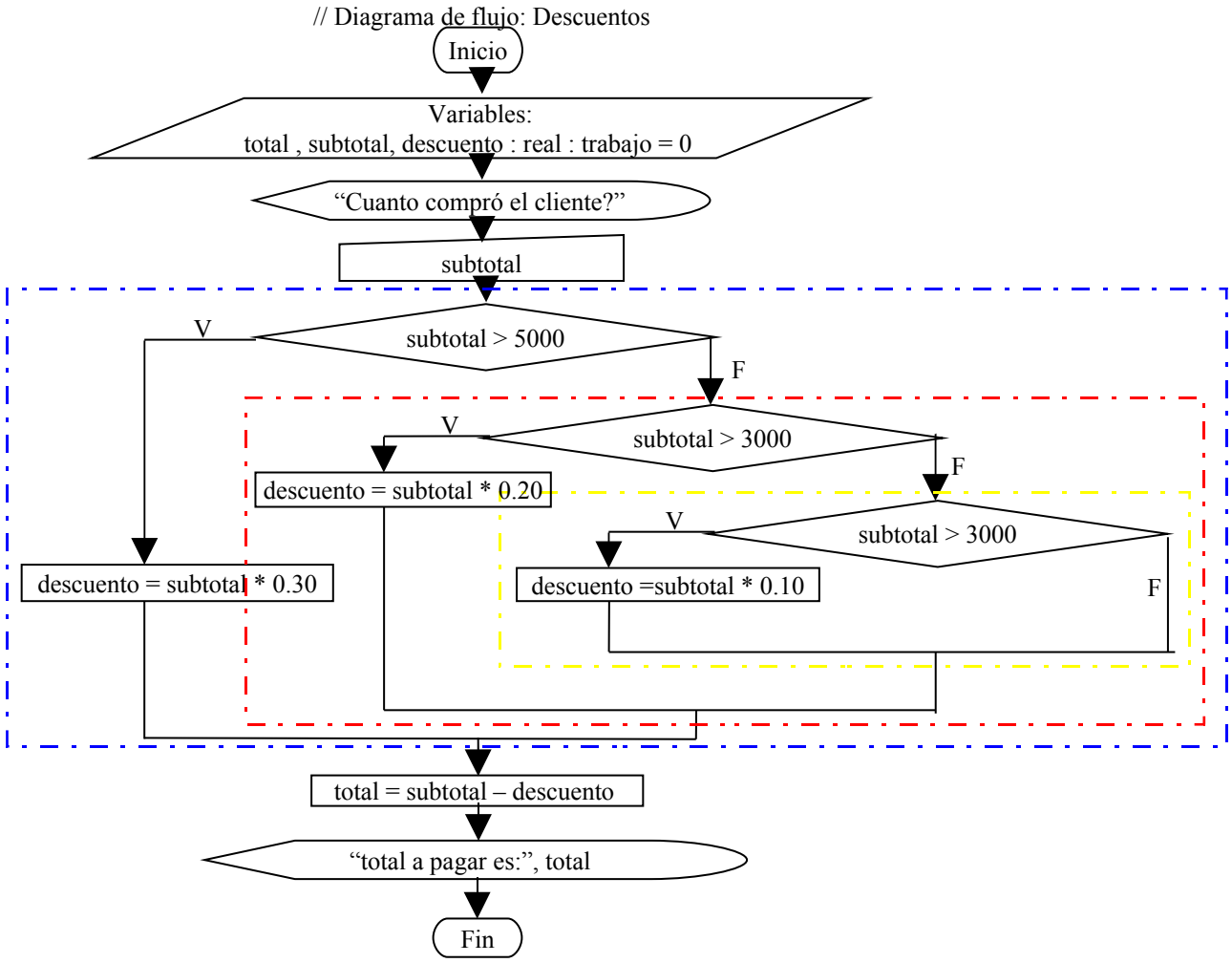
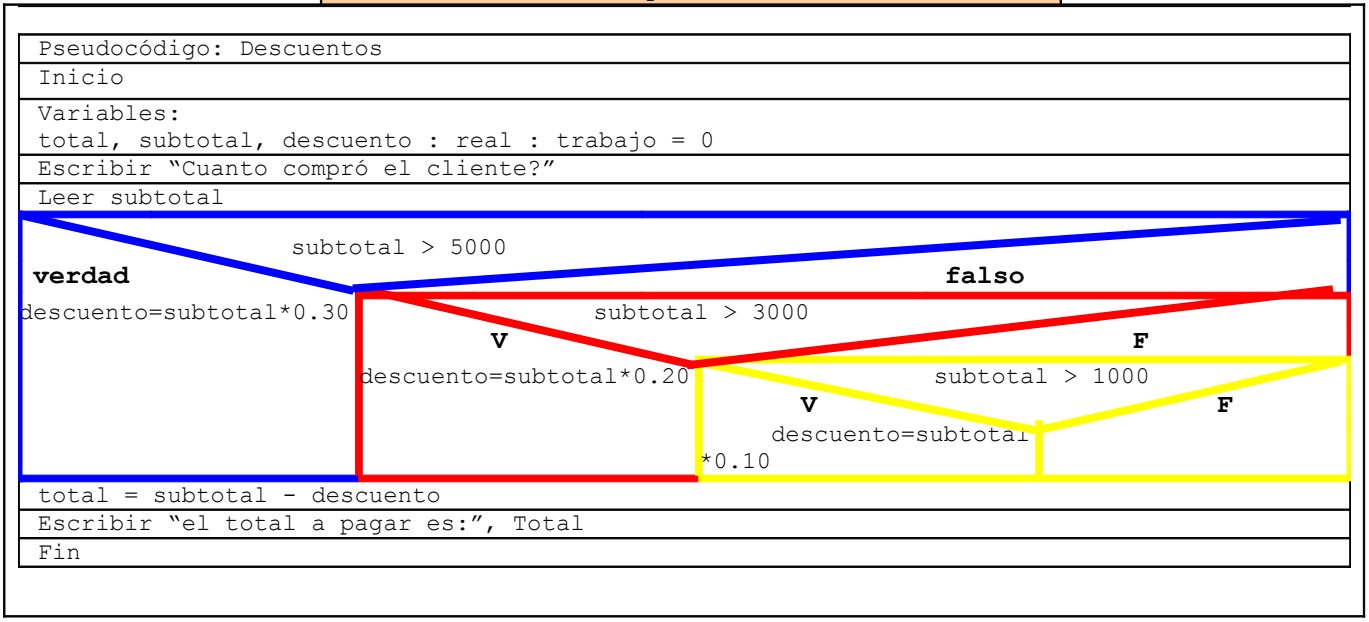


Diagrama N-S



Paso III. Prueba Del Algoritmo.		
Valores a entradas	**Procesos**	**Resultados**
subtotal = 5300	subtotal > 5000 5300 > 5000 → **SI** descuento = subtotal * 0.30 descuento = 5300 * 0.30 descuento = 1590 total = subtotal – descuento total = 5300 – 1590 total = 3710	**Total = 3710**
subtotal = 4100	subtotal > 5000 4100 > 5000 → **NO** subtotal > 3000 4100 > 3000 → **SI** descuento = subtotal * 0.20 descuento = 4100 * 0.20 descuento = 820 total = subtotal – descuento total = 4100 – 820 total = 3280	**Total = 3280**
subtotal = 1850	subtotal > 5000 1850 > 5000 → **NO** subtotal > 3000 1850 > 3000 → **NO** subtotal > 1000 1850 > 3000 → **SI** descuento = subtotal * 0.10 descuento = 1850 * 0.10 descuento = 185 total = subtotal – descuento total = 1850 – 185 total = 1665	**Total = 1665**
subtotal = 700	subtotal > 5000 700 > 5000 → **NO** subtotal > 3000 700 > 3000 → **NO** subtotal > 1000 700 > 3000 → **NO** total = subtotal – descuento total = 700 – 0 total = 700	**Total = 700**

Tabla 26 Ejemplo 3 de una estructura condicional simple anidada

En este ejemplo, se inicializaron las variables a cero, con lo cual se pudo resolver la expresión **total = subtotal – descuento**, ya que descuento podía tener cualquier valor, además gracias a esto pudimos colocar la expresión fuera de toda estructura.

Ejemplo ❹	Se necesita un sistema que le muestre a un alumno su calificación en letra y su promedio, el promedio se saca en base a 3 parciales, donde si el promedio es menor a 6 su letra es **NA**, si es mayor o igual a 6 y cuando mucho 8 le corresponde **S**, si sobrepasa el 8 pero menor o igual a 9 debe tener **B**, todo lo demás es una **E**.

Paso I. Analizar el problema.

Salidas	Entrada	Constantes	Procesos
▪ Mensaje (NA, S, B, E)	▪ Cal1 ▪ Cal2 ▪ Cal3		prom = (cal1 + cal2 + cal3) / 3 cuando prom < 6 **NA** Cuando prom >= 6 pero <= 8 **S** Cuando prom > 8 pero <= 9 **B** Cuando prom > 9 **E**

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Promedio en letra

Variables:
     cal1, cal2, cal3, prom : real : trabajo

   1.  Inicio
   2.  Escribir "dame calificación de primer parcial:"
   3.  leer cal1
   4.  Escribir "dame calificación de segundo parcial:"
   5.  leer cal2
   6.  Escribir "dame calificación de tercer parcial:"
   7.  leer cal3
   8.  prom = (cal1 + cal2 + cal3) / 3
   9.  Si prom < 6 entonces
            9.1 Escribir "Tu calificación con letra es NA"
       Si no
            9.2 Si prom <= 8 entonces
                     9.2.1 Escribir "Tu calificación con letra es S"
                Si no
                     9.2.2 Si prom <= 9 entonces
                              9.2.2.1 Escribir "Tu calificación con letra es B"
                           Si no
                              9.2.2.2 Escribir "Tu calificación con letra es E"
                           Fin si
                Fin si
       Fin si
   10. Escribir "ya que tu promedio es:", prom
   11. fin
```

DIAGRAMA DE FLUJO

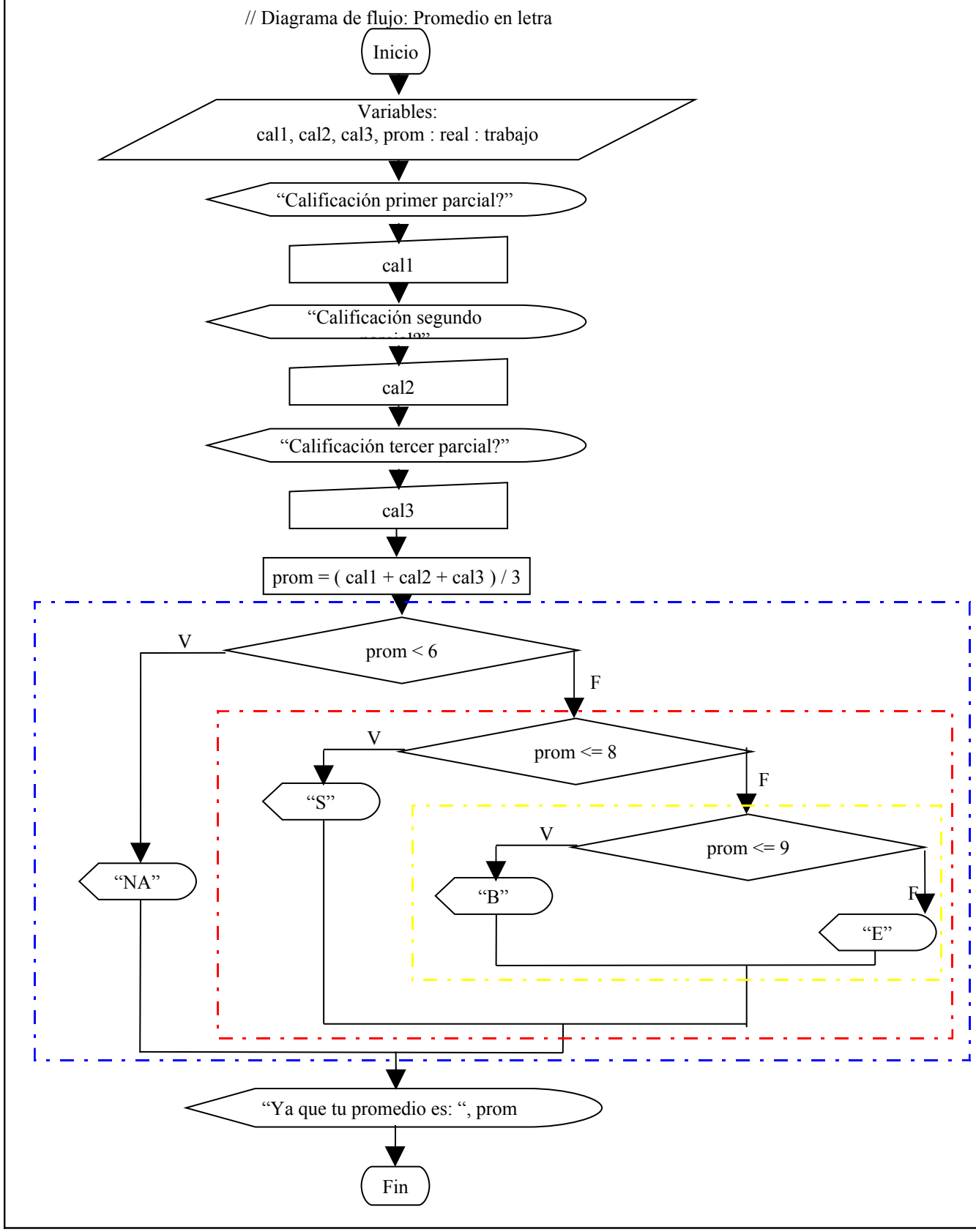
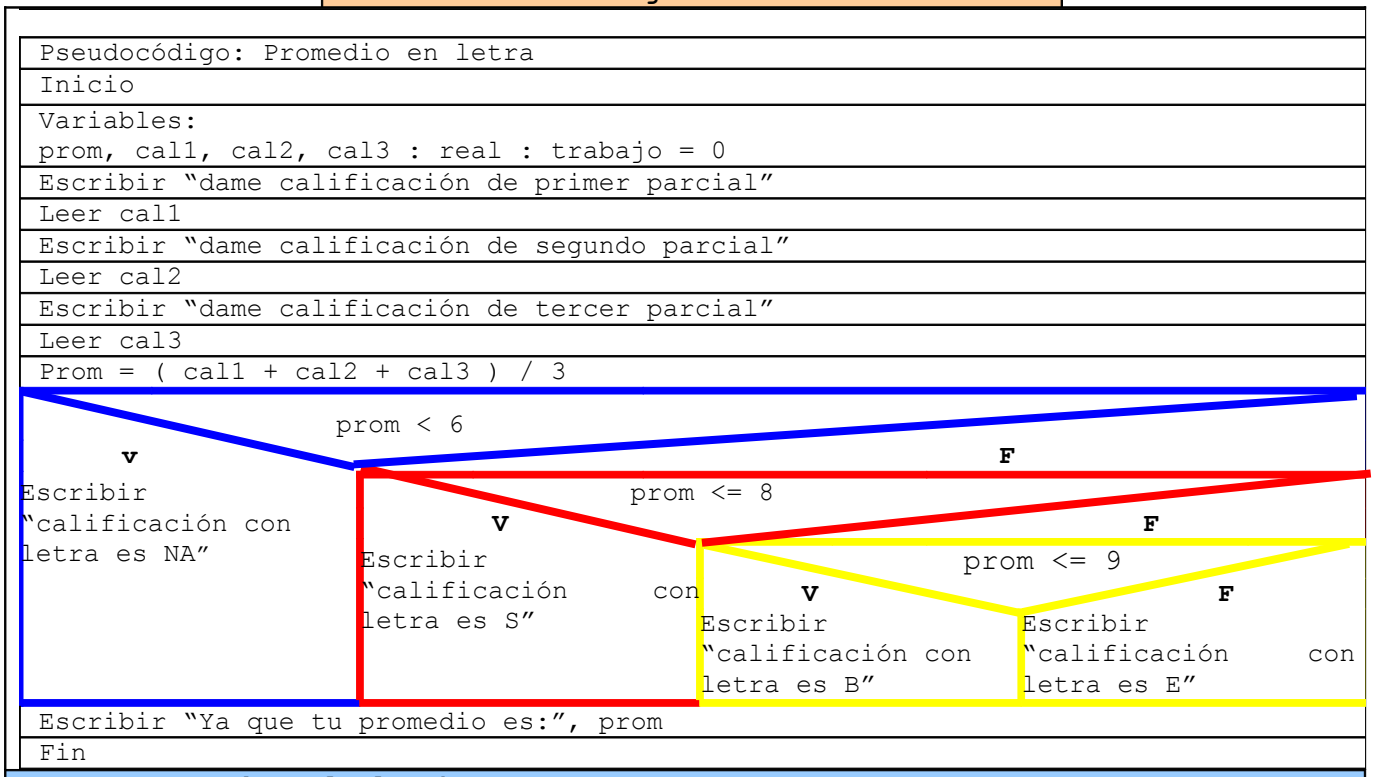


Diagrama N-S



Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
cal1 = 4 cal2 = 5 cal3 = 3	prom = (cal1 + cal2 + cal3) / 3 prom = (4 + 5 + 3) / 3 prom = 4 prom < 6 4 < 6 → **SI** "calificación con letra es NA" "ya que tu promedio es: 4"	**NA** **prom = 4**
cal1 = 8 cal2 = 6 cal3 = 7	prom = (cal1 + cal2 + cal3) / 3 prom = (8 + 6 + 7) / 3 prom = 7 prom < 6 7 < 6 → **NO** prom <= 8 7 <= 8 → **SI** "calificación con letra es S" "ya que tu promedio es: 7"	**S** **prom = 7**

cal1 = 9 cal2 = 8 cal3 = 9	prom = (cal1 + cal2 + cal3) / 3 prom = (8 + 6 + 7) / 3 prom = 8.7 prom < 6 8.7 < 6 → **NO** prom <= 8 8.7 <= 8 → **NO** prom <= 9 8.7 <= 9 → **SI** "calificación con letra es B" "ya que tu promedio es: 8.7"	**B** **prom = 8.7**
cal1 = 10 cal2 = 9 cal3 = 10	prom = (cal1 + cal2 + cal3) / 3 prom = (8 + 6 + 7) / 3 prom = 9.7 prom < 6 9.7 < 6 → **NO** prom <= 8 9.7 <= 8 → **NO** prom <= 9 9.7 <= 9 → **NO** "calificación con letra es E" "ya que tu promedio es: 8.7"	**E** **prom = 9.7**

Tabla 27 Ejemplo 4 de una estructura condicional simple anidada

Ejercicios. Realiza los siguientes ejercicios:
I. Escribe un algoritmo en pseudocódigo, diagrama de flujo y diagrama N-S para cada una de las situaciones siguientes:
1. Necesitamos saber si una persona es "joven" o "vieja" basándonos en su edad. Joven es aquella menor de 45 años.
2. Necesitamos saber si el usuario es alto o chaparro. Chaparro es aquel que mide cuando mucho 1.65 mts.
3. Necesitamos verificar que la contraseña que escribe el usuario es igual a "solrac". Dependiendo de lo ingresado desplegar el mensaje correspondiente.
4. Que lea dos números y los imprima en forma ascendente
5. Leer 2 números; si son iguales que los multiplique, si el primero es mayor que el segundo que los reste y si no, que los sume.

6. Leer tres números diferentes e imprimir el número mayor.

7. El IMSS requiere clasificar a las personas que se jubilaran en el año 2004. Existen tres tipos de jubilaciones: por edad, por antigüedad joven y por antigüedad adulta. Las personas adscritas a la jubilación por edad deben tener 60 años o mas y una antigüedad en su empleo de menos de 25 años. Las personas adscritas a la jubilación por antigüedad joven deben tener menos de 60 años y una antigüedad en su empleo de 25 años o más. Las personas adscritas a la jubilación por antigüedad adulta deben tener 60 años o mas y una antigüedad en su empleo de 25 años o mas.

8. Calcular la utilidad que un trabajador recibe en el reparto anual de utilidades si a este se le asigna un porcentaje de su salario mensual que depende de su antigüedad en la empresa de acuerdo con la siguiente tabla:

Tiempo	**Utilidad**
Menos de 1 año	5 % del salario
1 año o mas y menos de 2 años	7% del salario
2 años o mas y menos de 5 años	10% del salario
5 años o mas y menos de 10 años	15% del salario
10 años o mas	20% del salario

9. Un obrero necesita calcular su salario semanal, el cual se obtiene de la sig. manera:
   - Si trabaja 40 horas o menos se le paga $16 por hora
   - Si trabaja más de 40 horas se le paga $16 por cada una de las primeras 40 horas y $20 por cada hora extra.

10. Una empresa quiere hacer una compra de varias piezas de la misma clase a una fábrica de refacciones. La empresa, dependiendo del monto total de la compra, decidirá que hacer para pagar al fabricante.
    - Si el monto total de la compra excede de $500 000 la empresa tendrá la capacidad de invertir de su propio dinero un 55% del monto de la compra, pedir prestado al banco un 30% y el resto lo pagara solicitando un crédito al fabricante.
    - Si el monto total de la compra no excede de $500 000 la empresa tendrá capacidad de invertir de su propio dinero un 70% y el restante 30% lo pagara solicitando crédito al fabricante.
    - El fabricante cobra por concepto de intereses un 20% sobre la cantidad que se le pague a crédito.

11. Determinar la cantidad de dinero que recibirá un trabajador por concepto de las horas extras trabajadas en una empresa, sabiendo que cuando las horas de trabajo exceden de 40, el resto se consideran horas extras y que estas se pagan al doble de una hora normal cuando no exceden de 8; si las horas extras exceden de 8 se pagan las primeras 8 al doble de lo que se pagan las horas normales y el resto al triple.

Las estructuras condicionales múltiples se analizan a continuación en las tres técnicas algorítmicas.

Pseudocódigo. Para representar estas estructuras, se debe de utilizar la instrucción **casos para...**, donde en lugar de los puntos suspensivos se coloca la variable a evaluar. Para saber que instrucciones se van a ejecutar cuando la variable tenga un valor específico, se coloca una etiqueta **cuando es igual a ...:** por cada uno de estos, en la cual en lugar de los puntos suspensivos hay que colocar el valor que puede tener la variable. En caso de que se quiera realizar un conjunto de instrucciones para todos los demás valores que no han sido tomados en cuenta, se puede utilizar la etiqueta **para todos los demás valores:**. Para saber que se ha terminado la estructura, se coloca la instrucción **fin casos**.

```
Pseudocódigo: No hace nada
Variables
        resp : entera : trabajo
  1. Inicio
  2. Escribir "Escribe un número [1/2]"
  3. Leer resp
  4. Casos para resp

        Cuando es igual a 1:
             4.1  Escribir " escribiste un 1"

        Cuando es igual a 2:
             4.2  Escribir " escribiste un 2"

        Para todos los demás valores:
             4.3  Escribir " No escribiste ni un 1 ni un 2"

     Fin casos
```

```
5. Fin
```

Ilustración 11 Diseño que debe tener una estructura condicional múltiple.

Aspecto Crítico. Los valores que puede tener una variable a evaluar en una estructura condicional múltiple, solo pueden ser valores enteros, por lo cual se debe declarar la variable como tal.

Diagrama de flujo. Para representar un estructura de selección múltiple, se sigue usando el símbolo de decisión, pero a diferencia de las estructuras de selección sencilla, ahora no sale una flecha por el lado izquierdo y otra por el derecho, sale un solo camino del cual se desprenden todos los demás, y dentro del símbolo no se coloca una expresión, solamente se coloca la variable a evaluar. Para saber que instrucciones se van a ejecutar, en cada uno de los caminos se coloca una etiqueta con el valor, al igual que en pseudocódigo se puede utilizar una etiqueta para todos los **demás valores** que no fueron tomados en cuenta. El final de la estructura se indica uniendo todos los caminos en uno solo nuevamente.

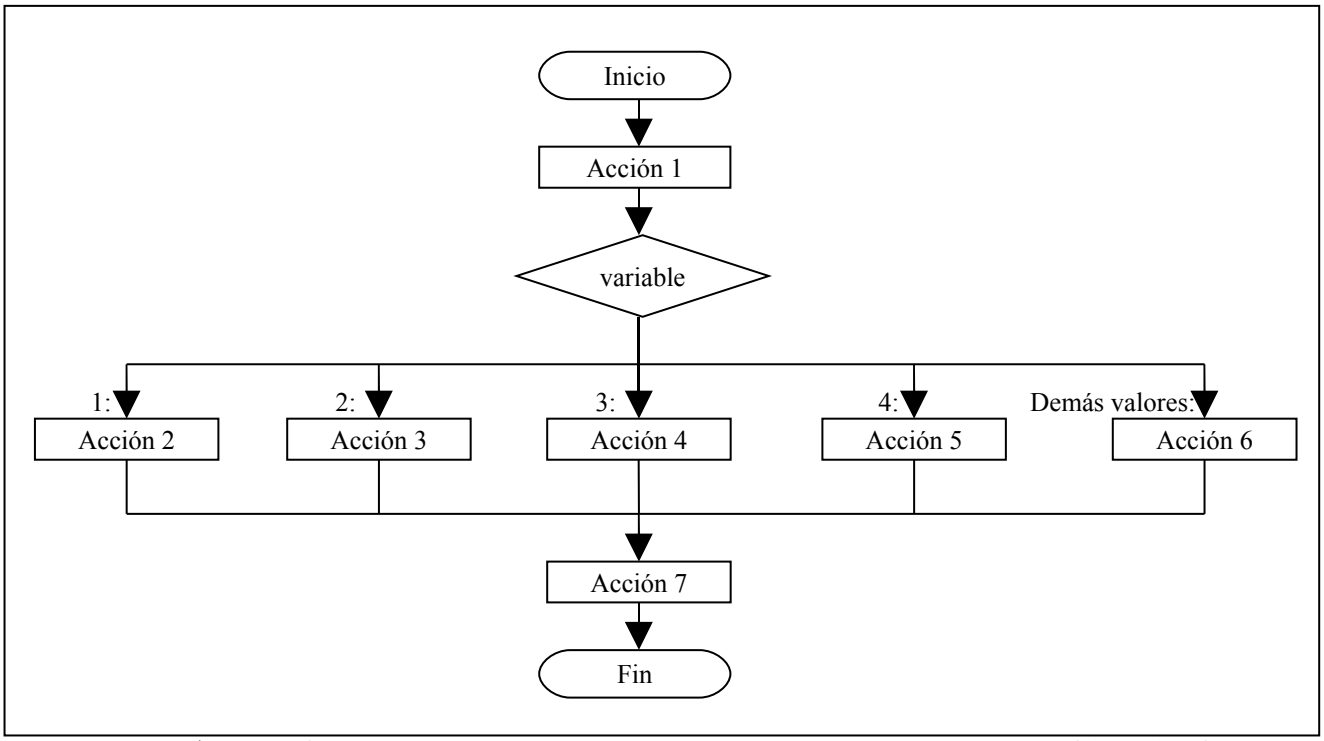


Ilustración 12 Diseño que debe de tener una estructura de selección múltiple.

Sugerencia. Es muy probable que al diseñar un diagrama de flujo que utiliza estructuras de selección múltiple con varios posibles valores, no quepan todos en la misma hoja, por lo cual se deben colocar conectores a otra hoja.

Diagrama N-S. Para representar una estructura condicional múltiple se sigue utilizando la caja de decisión, pero esta se debe de dividir en todos los posibles valores. Al igual que en la técnicas algorítmicas anteriores, existe un posible camino para todos los valores no tomados en cuenta. Esta estructura ha terminado cuando todas las columnas en que se dividió se convierten en una sola.

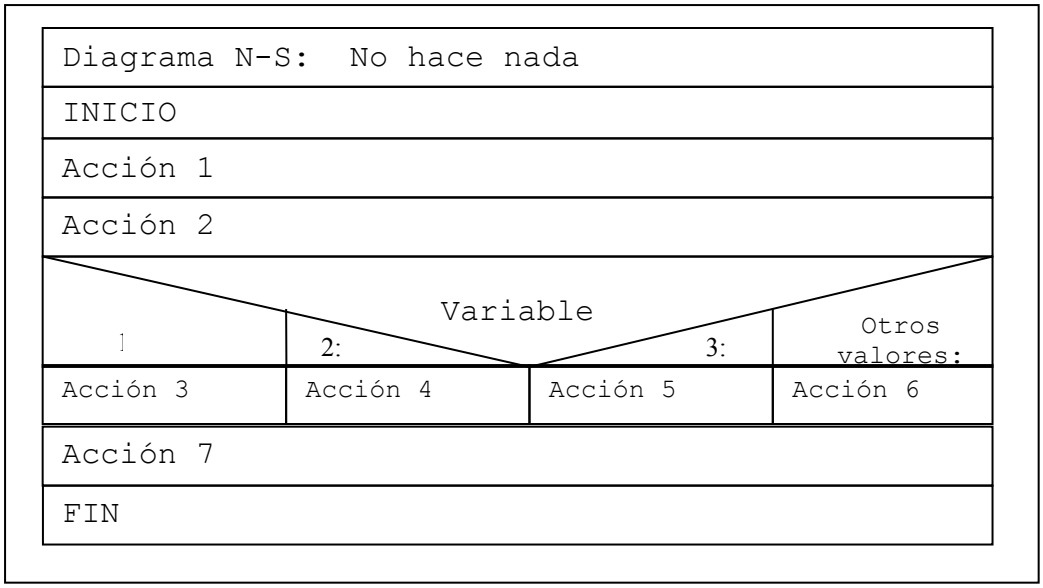


Ilustración 13 Diseño de una estructura de selección múltiple.

A continuación realizamos un par de ejemplos, pero antes debemos de entender que el principal uso de estas estructuras es para el manejo de menús.

Nota. Al igual que todas las estructuras de cualquier tipo, estas se pueden anidar.

En ocasiones se querrá ejecutar un mismo conjunto de instrucciones para diferentes posibles valores. En este tipo de estructuras es posible determinar un rango de posibles valores utilizando lo siguiente: **valor_inicial...valor_final:.**

✍**Ejemplo ❶**	Se necesita un sistema que tenga tres opciones, si se selecciona la primera se calcula el perímetro de un cuadrado, si la opción es la dos se calcula el perímetro de un triangulo equilátero, y cuando se elija la tres se calcula el perímetro de un círculo, además de que mandara un mensaje de "error" en caso de presionar cualquier otro número.		
Paso I. Analizar el problema.			
Salidas	**Entrada**	**Constantes**	**Procesos**
▪ perim	▪ opc ▪ lado		Cuando opc == 1 perim = lado * 4 Cuando opc == 2

			perim = lado * 3 Cuando opc == 3 perim = lado * 3.1416 Cuando opc tenga otro valor "ERROR"

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: menú perímetros

Variables:

       opc : entera : trabajo
       perim, lado : reales : trabajo = 0

   1.  Inicio
   2.  Escribir "Menu de Perímetros"
   3.  Escribir "1. Cuadrado"
   4.  Escribir "2. Triangulo"
   5.  Escribir "3. Circulo"
   6.  Escribir "cual eliges?:"
   7.  Leer opc
   8.  Casos para opc

          cuando es igual a 1:
                8.1 Escribir "dame el valor de un lado del cuadrado:"
                8.2 Leer lado
                8.3perim = lado * 4

          cuando es igual a 2:
                8.4 Escribir "dame el valor de un lado del triangulo:"
                8.5Leer lado
                8.6perim = lado * 3

          cuando es igual a 3:
                8.7 Escribir "dame el valor del diámetro:"
                8.8Leer lado
                8.9 perim = lado * 3.1416

          para todos los demás valores:
                8.10Escribir "ERROR"

       fin casos

   9.  Escribir "el resultado es:", perim
   10. Fin
   // En este programa no es necesario declarar tantas variables ya que solo se irá
   // por un solo camino. Esto nos ahorra al momento de programar memoria.
```

DIAGRAMA DE FLUJO

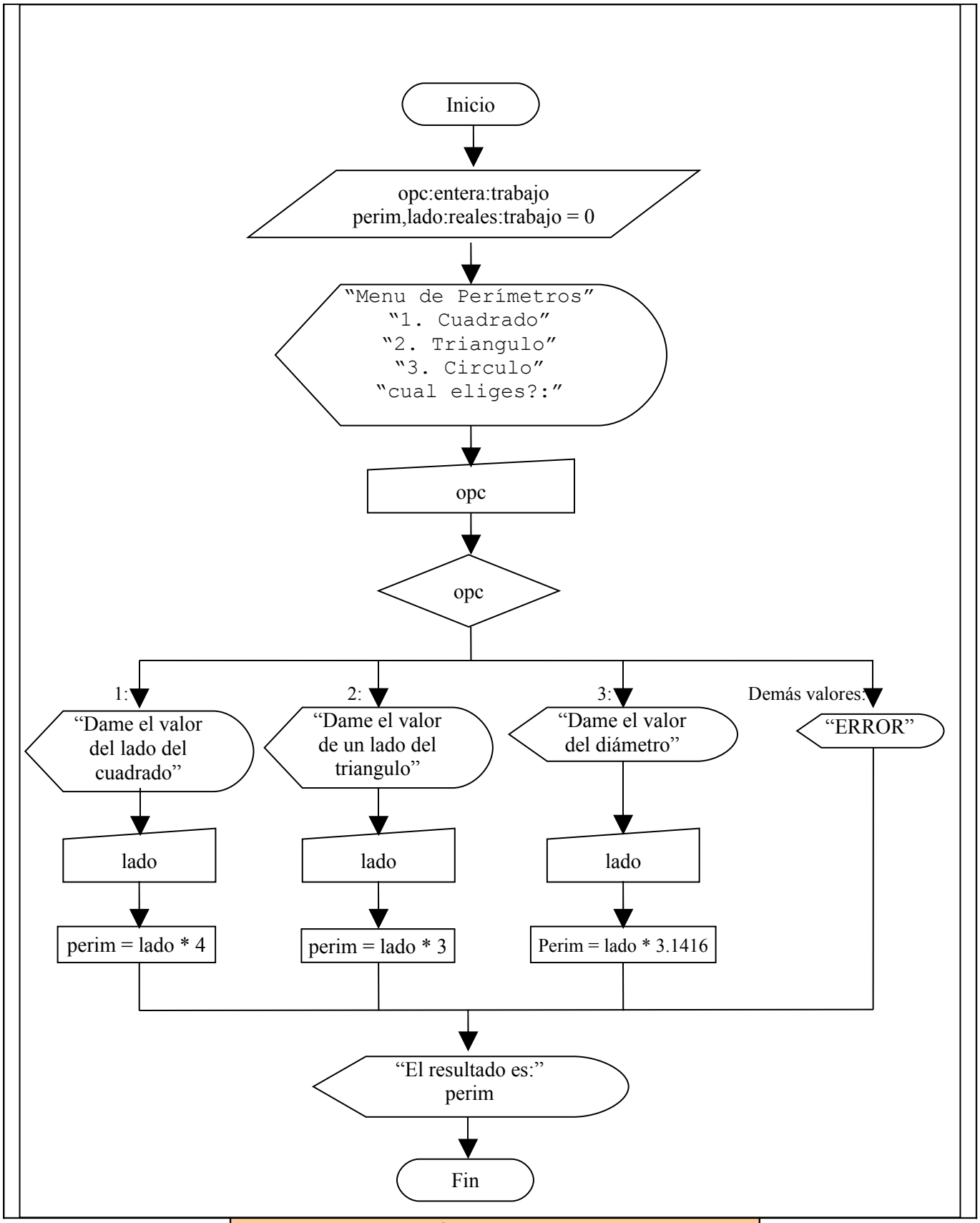

Inicio
opc:entera:trabajo
perim,lado:reales:trabajo = 0
"Menu de Perímetros"
"1. Cuadrado"
"2. Triangulo"
"3. Circulo"
"cual eliges?:"
opc
opc
1:
2:
3:
Demás valores:
"Dame el valor del lado del cuadrado"
"Dame el valor de un lado del triangulo"
"Dame el valor del diámetro"
"ERROR"
lado
lado
lado
perim = lado * 4
perim = lado * 3
Perim = lado * 3.1416
"El resultado es:"
perim
Fin

Diagrama N-S

// diagrama N-S : Menú perímetros			
Inicio			
Variables: opc:entera:trabajo perim,lado:reales:trabajo = 0			
Escribir “Menú perímetros”, “1. Cuadrado”, “2. Triangulo”, “3. Circulo”, “Cual eliges?:”			
Leer opc			
opc			
1:	2:	3:	Demás valores:
Escribir “Dame el valor del lado del cuadrado”	Escribir “Dame el valor del lado del triangulo”	Escribir “Dame el valor del diámetro”	Escribir “ERROR”
	Leer lado	Leer lado	
Escribir “El resultado es:”, perim			
Fin			

Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
opc = 1 lado = 5	opc == 1 1 == 1 → **SI** perim = lado * 4 perim = 5 * 4 perim = 20	**perim = 20**
opc = 2 lado = 10	opc == 1 2 == 1 → **NO** opc == 2 2 == 2 → **SI** perim = lado * 3 perim = 10 * 3 perim = 30	**perim = 30**
opc = 3 lado = 2	opc == 1 3 == 1 → **NO** opc == 2 3 == 2 → **NO** opc == 3 3 == 3 → **SI** perim = lado * 3.1416 perim = 2 * 3.1416 perim = 6.2832	**perim = 6.2832**
opc = 8	opc == 1 8 == 1 → **NO** opc == 2 8 == 2 → **NO** opc == 3 8 == 3 → **NO** opc es otro valor → **SI**	**perim = 0**

Tabla 28 Ejemplo 1 de una estructura de selección múltiple.

Ejemplo ❷

Un supermercado realiza una tómbola solo con aquellos clientes que realizan una compra superior a $500, en la cual tienen que sacar de una canasta una bolita la cual tiene un número grabado, los premios se dan bajo la siguiente tabla:

# bolita	Premio
1	1 shampoo CAPRICE
2	1 paquete(3) de jabones ROSA VENUS
3	1 pasta de dientes COLGATE
4	1 bolsa de detergente MAS COLOR
5	1 caja de cereal ZUCARITAS

Paso I. Analizar el problema.

Salidas	Entrada	Procesos
▪ MENSAJE	▪ compra ▪ n_bol	Cuando compra > 500 Cuando n_bol == 1 "1 shampoo CAPRICE" Cuando n_bol == 2 "1 paquete(3) de jabones ROSA VENUS" Cuando n_bol == 3 "1 pasta de dientes COLGATE" Cuando n_bol == 4 "1 bolsa de detergente MAS COLOR" Cuando n_bol == 5 "1 caja de cereal ZUCARITAS"

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Tómbola

Variables:
          compra : real : trabajo
          n_bol : entera : trabajo
   1.  Inicio
   2.  Escribir "Total de compra:"
   3.  Leer compra

   4.  si compra > 500 entonces
          4.1 Escribir "Número de bolita que sacaste:"
          4.2 Leer n_bol

          4.3 Casos para n_bol
                    Cuando es igual a 1:
                           4.3.1 Escribir "1 shampoo CAPRICE"
                    Cuando es igual a 2:
                           4.3.2 Escribir "1 paquete(3) de jabones ROSA VENUS"
                    Cuando es igual a 3:
                           4.3.3 Escribir "1 pasta de dientes COLGATE"
                    Cuando es igual a 4:
                           4.3.4 Escribir "1 bolsa de detergente MAS COLOR"
                    Cuando es igual a 5:
                           4.3.5 Escribir "1 caja de cereal ZUCARITAS"
                Fin casos

       si no
          // no hace nada ya que no tiene derecho a sacar bolita
       fin si
   5.  Fin
```

DIAGRAMA DE FLUJO

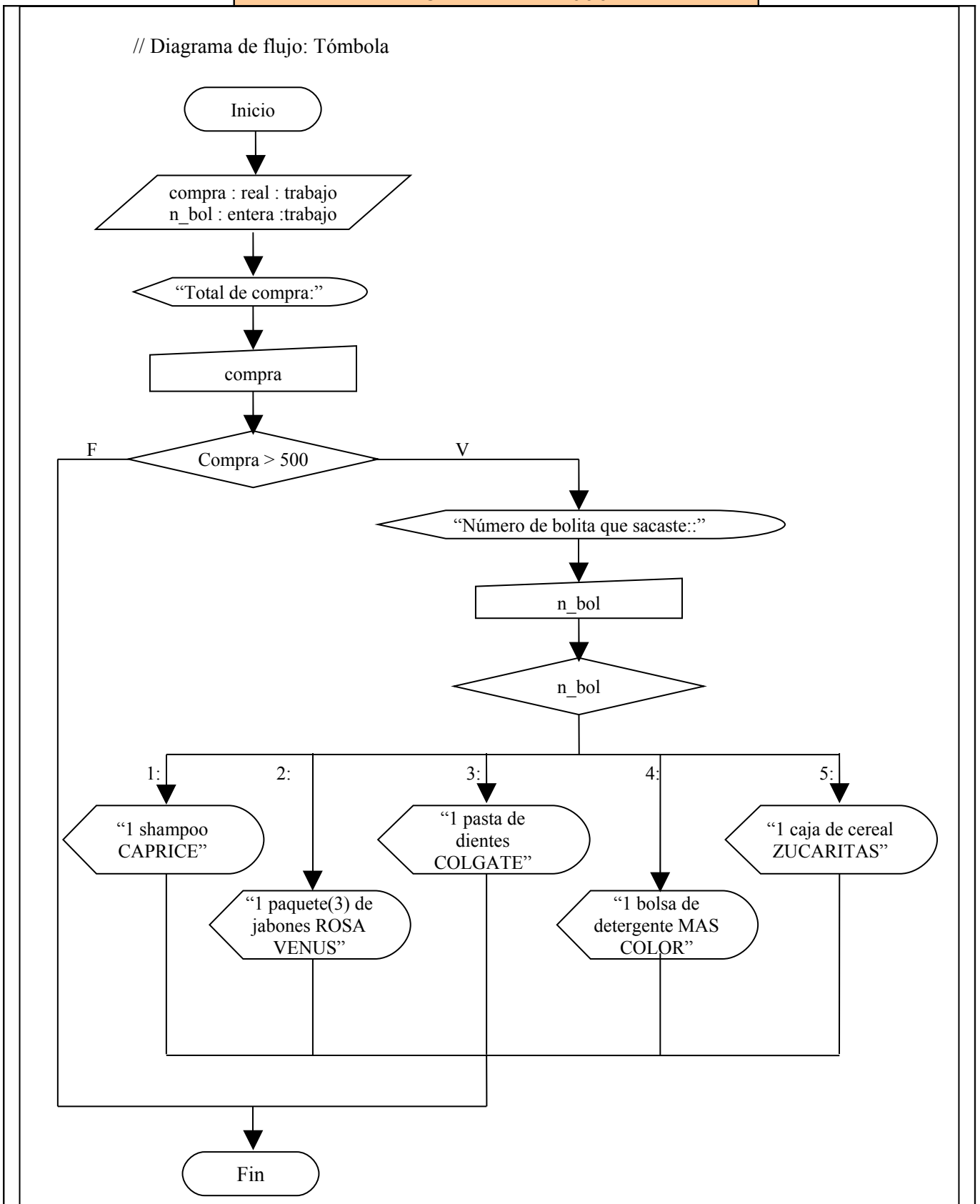


Diagrama N-S

// diagrama N-S : Tómbola					
Inicio					
Variables: compra : real : trabajo n_bol : entera : trabajo					
Escribir "Total de compra:"					
Leer compra					
compra > 500 F / V					
	Escribir "Número de bolita que sacaste:"				
	Leer n_bol				
	n_bol 3:				
	Escribir "1 shampoo CAPRICE"	Escribir "1 paquete (3) jabones ROSA VENUS"	Escribir "1 pasta de dientes COLGATE"	Escribir "1 bolsa de detergente MAS COLOR"	Escribir "1 caja de cereal ZUCARITAS"
Fin					

Paso III. Prueba Del Algoritmo.

Valores a entradas	Procesos	Resultados
compra = 350	compra > 500 350 > 500 → **NO**	--
compra = 550 n_bol = 1	compra > 500 550 > 500 → **SI** n_bol == 1 1 == 1 → **SI**	"1 shampoo CAPRICE"
compra = 800 n_bol = 2	compra > 500 800 > 500 → **SI** n_bol == 1 2 == 1 → **NO** n_bol == 2 2 == 2 → **SI**	"1 paquete(3) jabones ROSA VENUS"
compra = 630 n_bol = 3	compra > 500 630 > 500 → **SI** n_bol == 1 3 == 1 → **NO** n_bol == 2 3 == 2 → **NO** n_bol == 3	"1 pasta de dientes COLGATE"

	3 == 3 → **SI**	
compra = 920 n_bol = 4	compra > 500 920 > 500 → **SI** n_bol == 1 4 == 1 → **NO** n_bol == 2 4 == 2 → **NO** n_bol == 3 4 == 3 → **NO** n_bol == 4 4 == 4 → **SI**	**"1 bolsa de detergente MAS COLOR"**
compra = 501 n_bol = 5	compra > 500 501 > 500 → **SI** n_bol == 1 5 == 1 → **NO** n_bol == 2 5 == 2 → **NO** n_bol == 3 5 == 3 → **NO** n_bol == 4 5 == 4 → **NO** n_bol == 5 5 == 5 → **SI**	**"1 caja de cereal ZUCARITAS"**
compra = 500.01 n_bol = 8	compra > 500 500.01 > 500 → **SI** n_bol == 1 8 == 1 → **NO** n_bol == 2 8 == 2 → **NO** n_bol == 3 8 == 3 → **NO** n_bol == 4 8 == 4 → **NO** n_bol == 5 8 == 5 → **NO**	--

Tabla 29 Ejemplo 2 de una estructura de selección múltiple.

Se puede anidar cualquier estructura dentro de otra como en este ejemplo.

Ejercicios.

I. Escribe un algoritmo en las tres técnicas manejadas para cada uno de los problemas siguientes:

1. Necesitamos visualizar un menú del conalep, en el cual hay que elegir que semestre esta cursando un alumno. Dependiendo la opción elegida, que se despliegue un mensaje en el que se diga en que semestre va.

2. Necesitamos un menú del conalep en el que se visualicen las cuatro carreras que se imparten y dentro de cada una de estas opciones que se visualice un menú con los 6 semestres. Al seleccionarlo, que se despliegue un mensaje de la carrera y semestre que cursa el alumno.

3. Necesitamos un menú del conalep en el que se visualicen las cuatro carreras que se imparten y dentro de cada una de estas opciones que se visualice un menú con los 6 semestres, y dentro de cada semestre hay que elegir entre el turno matutino y el vespertino. Al seleccionarlo, que se despliegue un mensaje de la carrera, semestre y turno que cursa el alumno.

4. Necesitamos un menú del conalep en el que se visualicen las cuatro carreras que se imparten; dentro de cada una de estas opciones que se visualice un menú con los 6 semestres; dentro de cada semestre hay que elegir entre el turno matutino y el vespertino; Por último hay que elegir si al alumno se le da de alta o de baja. Al seleccionarlo, que se despliegue un mensaje de la carrera, semestre, turno y condición (baja o alta).

5. Un supermercado realiza una tómbola con todos los clientes, si son hombres tienen que sacar de una canasta una bolita la cual tiene un número grabado y si son mujeres lo mismo pero de otra canasta, los premios se dan bajo la siguiente tabla:

HOMBRES		MUJERES	
# bolita	Premio	# bolita	Premio
1	Desodorante	1	Loción
2	SixPack de cerveza	2	Bikini
3	Boxer	3	Crema p/ la cara
4	Rasuradora	4	Plancha
5	Sudadera	5	Barniz de uñas

6. Una empresa automotriz necesita un sistema para seleccionar el tipo de carro (auto, camioneta o vagoneta) lo cual debe de aparecer en un menú, y el color (negro, blanco o rojo) en otro menú. Al final se necesita que despliegue la selección realizada.

Nota. Debe de anidarse una estructura de selección múltiple dentro de otra.

4.3 Estructuras Cíclicas

Este tipo de estructuras, son las que nos permiten ejecutar varias veces un conjunto determinado de instrucciones, a esta repetición se le conoce con el nombre de ciclos.

De manera general existen 3 tipos de estructuras cíclicas, las cuales manejaremos a continuación, para lo cual se explican, se desarrollan ejemplos y se resuelven ejercicios.

❶ **HACER MIENTRAS...** Estructura cíclica la cual indica un conjunto de instrucciones que se deben de repetir mientras que la respuesta a la a la expresión que se coloca en lugar de los puntos suspensivos sea **verdadera**. Es decir, que cuando la respuesta a la condición sea **falsa** se continúa con la siguiente instrucción después de la etiqueta **fin mientras**. El conjunto de instrucciones a ejecutar se encuentra entre las instrucciones **hacer mientras...** y **fin mientras**.

Debido a su estructura es muy posible que nunca se ejecute el ciclo debido a varias circunstancias:

- La variable a evaluar no tiene ningún valor almacenado.
- Nunca se le pidió al usuario que almacenará un dato en la variable.
- El usuario decidió no ingresar a la estructura.

💬***Sugerencia.*** Se recomienda que la variable a ser evaluada sea inicializada con un valor que permita no ingresar a la estructura para evitar lo que llamamos un ciclo infinito.

Aspecto Crítico. Siempre solicite al usuario un dato para la variable a evaluar antes de la instrucción **hacer mientras...**, ya que probablemente no desea ingresar al ciclo.

Aspecto Crítico. Siempre coloque dentro de la estructura **hacer mientras...** las instrucciones que permitan al usuario o al sistema almacenar un nuevo valor en la variable a evaluar para evitar un ciclo infinito.

A continuación vamos a esquematizar el diseño básico de esta estructura en las tres técnicas algorítmicas.

Pseudocódigo. En pseudocódigo se utilizan las instrucciones que hemos estado mencionando.

```
Pseudocódigo: ciclo hacer mientras
// Imprime HOLA tantas veces como el usuario presione "s"
Variables:
        Resp : alfanumérica : trabajo = "n"
  1. Inicio
  2. Escribir "deseas ingresar al ciclo:"
  3. Leer Resp
  4. Hacer mientras Resp == "s"
        4.1  Escribir "hola"
        4.2  Escribir "deseas ingresar nuevamente al ciclo:"
        4.3  Leer Resp
     Fin mientras
  5. Escribir "Gracias por usar este sistema"
  6. FIN
```

Ilustración 14 Pseudocódigo básico del ciclo **hacer mientras...**

Diagrama de Flujo. En diagrama de flujo, se utiliza el **símbolo de decisión** para representar a la estructura, del cual salen dos caminos posibles: el verdadero y el falso. En la ruta del lado verdadero se colocan todas las instrucciones que deseamos se repitan, después de la ultima instrucción una flecha debe de regresar y

conectar justo entre el símbolo de decisión y el símbolo que se encuentra antes. Del camino falso sale una flecha que conecta con la siguiente instrucción a ejecutar cuando se salga del ciclo.

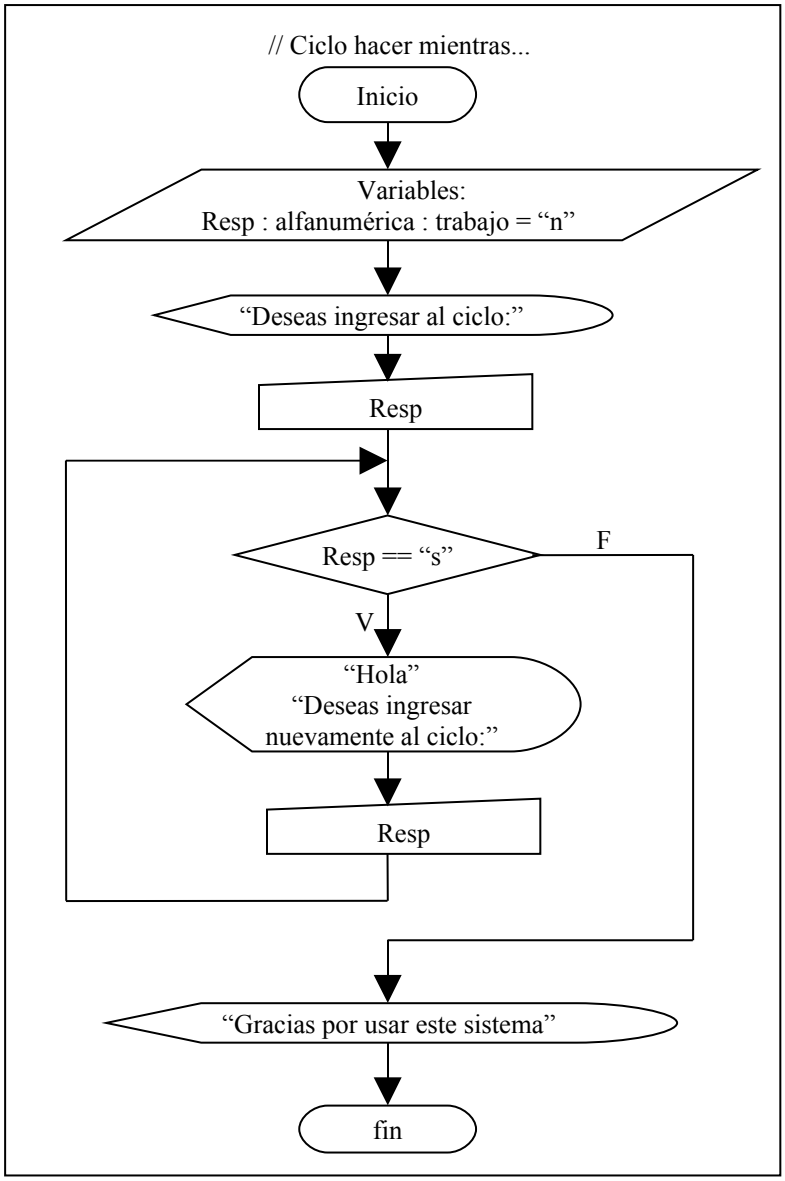


Ilustración 15 Diagrama de Flujo básico del ciclo **hacer mientras...**

Diagramas N-S. Para representar esta estructura existe el símbolo especial **hacer mientras...,** el cual es una sola caja con dos partes: la escuadra que parece un **siete** (7) pero apuntando al lado

contrario, es en donde se coloca la expresión a evaluar; la otra parte es la caja que se encuentra entre las dos paredes de la escuadra, es donde se colocan las instrucciones a realizar cuando el ciclo se ejecute.

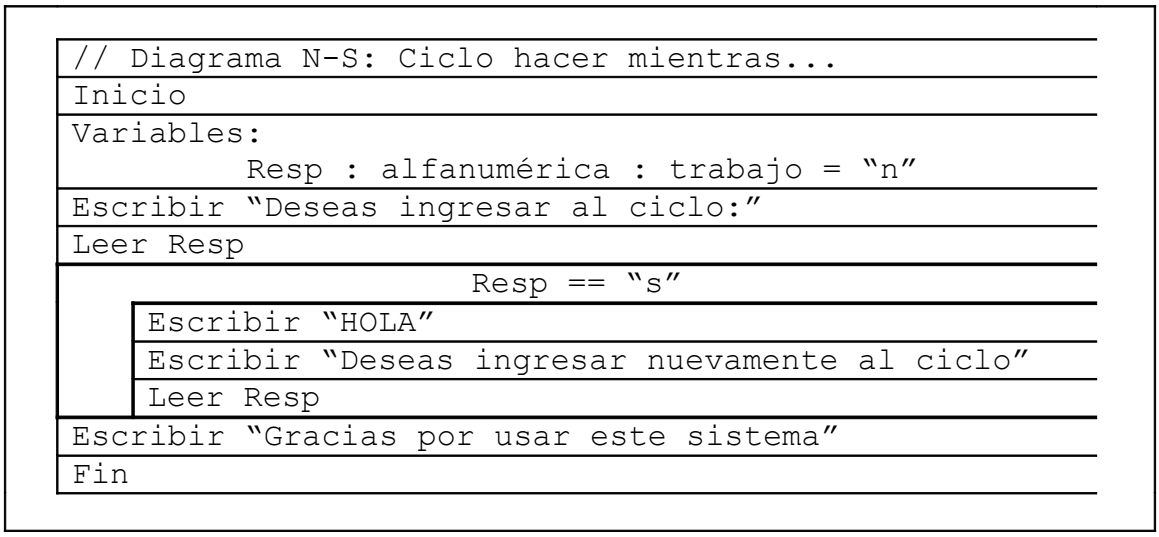


Ilustración 16 Diagrama N-S básico del ciclo **hacer mientras...**

A continuación se presentan un par de ejercicios con las tres técnicas algorítmicas los cuales manejan esta estructura.

Ejemplo ❶	Un maestro necesita un sistema para capturar las calificaciones de 3 parciales de sus alumnos, después recapturarlas necesita que se despliegue el promedio, cuando ya no quiera capturar más alumnos, necesita que se despliegue el promedio general de todos los alumnos capturados.

Paso I. Analizar el problema.

Salidas	Entrada	Procesos
▪ prom_alum ▪ prom_gen	▪ par1 ▪ par2 ▪ par3 ▪ resp	Cuando resp == 's' prom_alu = (par1 + par2 par3) / 3 acum_prom = acum_prom + prom_alu total_alum = total_alum + 1 prom_gen = acum_prom / total_alum

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: calificaciones

Variables:
          par1, par2, par3, prom_alum, prom_gen : reales : trabajo = 0
          acum_prom : real : acumulador = 0
          total_alum : entera : contador = 0
          resp : alfanumérica = "n"
   1. Inicio
   2. Escribir "deseas capturar calificaciones de un alumno:"
```

```
3.  Leer resp
4.  Hacer mientras resp == "s"
       5.1 Escribir "primer parcial:"
       5.2 Leer par1
       5.3 Escribir "segundo parcial:"
       5.4 Leer par2
       5.5 Escribir "tercer parcial:"
       5.6 Leer par3
       5.7 prom_alu = ( par1 + par2  par3 ) / 3
       5.8  Escribir "el promedio de este alumno es", prom_alum
       5.9 acum_prom = acum_prom + prom_alu // acumula los promedios
       5.10total_alum = total_alum + 1 // lleva el conteo del total de alumnos
       5.11Escribir "deseas capturar otro alumno:"
       5.12Leer resp
    fin mientras
5.  prom_gen = acum_prom / total_alum
6.  Escribir "Promedio general de los alumnos es:", prom_gen
7.  Fin
```

DIAGRAMA DE FLUJO

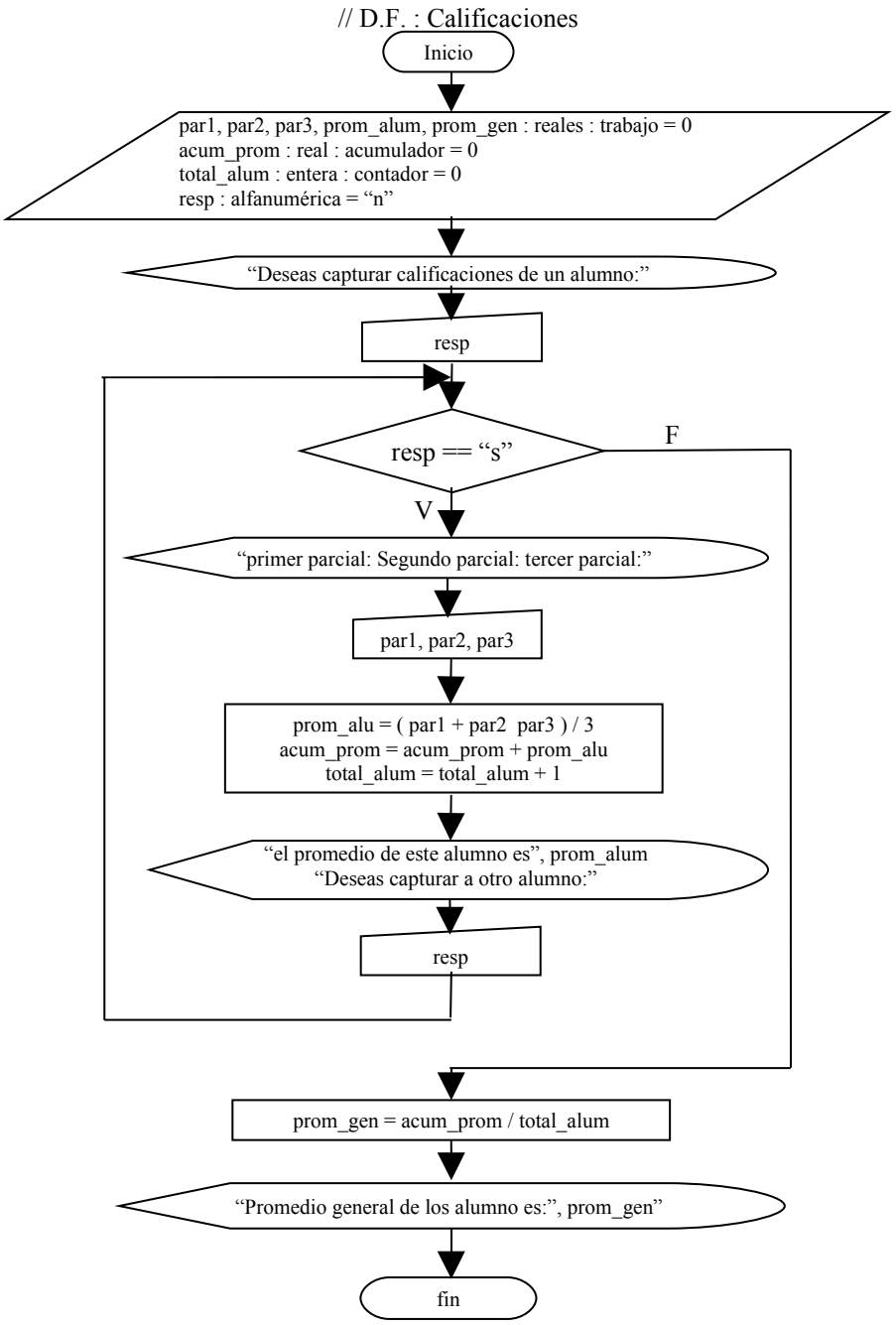


Diagrama N-S

```
// diagrama N-S :  Calificaciones
Inicio
Variables:
            par1, par2, par3, prom_alum, prom_gen : reales : trabajo = 0
            acum_prom : real : acumulador = 0
            total_alum : entera : contador = 0
            resp : alfanumérica = "n"
Escribir "deseas capturar calificaciones de un alumno:"
Leer resp
                                    resp == "s"
```

```
        Escribir "primer parcial:"
        Leer par1
        Escribir "segundo parcial:"
        Leer par2
        Escribir "tercer parcial:"
        Leer par3
        prom_alu = ( par1 + par2  par3 ) / 3
        Escribir "el promedio de este alumno es", prom_alum
        acum_prom = acum_prom + prom_alu
        total_alum = total_alum + 1
        Escribir "deseas capturar otro alumno:"
        Leer resp
prom_gen = acum_prom / total_alum
Escribir "Promedio general de los alumnos es:", prom_gen
Fin
```

Paso III. Prueba Del Algoritmo.

PRIMERA CORRIDA

resp = "n"

resp == "s"
n == "s" → **NO**

prom_gen = acum_prom / total_alum
prom_gen = 0 / 0
prom_gen = 0

SEGUNDA CORRIDA

resp = "s"

resp == "s"
"s" == "s" → **SI**

par1 = 9
par2 = 8
par3 = 10
prom_alu = (par1 + par2 par3) / 3
prom_alu = (9 + 8 + 10) / 3
prom_alu = 9
acum_prom = acum_prom + prom_alu
acum_prom = 0 + 9
acum_prom = 9
total_alum = total_alum + 1
total_alum = 0 + 1
total_alum = 1

SEGUNDA CORRIDA (CONTINUACIÓN...)

```
resp = "s"

resp == "s"
"s" == "s" → SI

          par1 = 7
          par2 = 8
          par3 = 9
          prom_alu = ( par1 + par2  par3 ) / 3
          prom_alu = ( 7 + 8 + 9 ) / 3
          prom_alu = 8
          acum_prom = acum_prom + prom_alu
          acum_prom = 9 + 8
          acum_prom = 17
          total_alum = total_alum + 1
          total_alum = 1 + 1
          total_alum = 2

resp = "s"

resp == "s"
"s" == "s" → SI

          par1 = 10
          par2 = 7
          par3 = 10
          prom_alu = ( par1 + par2  par3 ) / 3
          prom_alu = ( 10 + 7 + 10 ) / 3
          prom_alu = 9
          acum_prom = acum_prom + prom_alu
          acum_prom = 17 + 9
          acum_prom = 26
          total_alum = total_alum + 1
          total_alum = 2 + 1
          total_alum = 3

resp = "n"

resp == "s"
"n" == "s" → NO

prom_gen = acum_prom / total_alum
prom_gen = 26 / 3

prom_gen = 8.67
```

Tabla 30 Ejemplo 1 de la estructura cíclica **Hacer mientras ...**

En el ejemplo anterior se utiliza un ciclo, debido a que se necesita repetir un mismo conjunto de instrucciones por cada alumno, además se utiliza la estructura hacer mientras debido a que el total de estudiantes a capturar es indefinido. Además se utilizaron variables del tipo acumulador y contador. Acum_prom, acumula los promedios y total_alum, lleva el conteo del número de alumnos.

✍Ejemplo ❷	Un supermercado dará un descuento del 10% a **todos** los clientes que el total de su compra supere los $1000, además se necesita saber a cuanto ascendieron los ingresos del día.

Paso I. Analizar el problema.

Salidas	Entrada	Procesos
▪ subtotal ▪ ingresos	▪ subtotal	mientras haya clientes capturar subtotal cuando subtotal > 1000 descuento = subtotal * .10 total = subtotal – descuento en caso contrario total = subtotal ingresos = ingresos + total

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
pseudocódigo: supermercado

variables:
        total, subtotal, descuento, ingresos : reales = 0
        resp : alfanumérico = "n"
// si ya las sabe manejar no es necesario colocar el uso

   1. Inicio
   2. Escribir "hay clientes en la tienda"
   3. Leer resp
   4. Hacer mientras resp == "s"
          4.1 Escribir "cuanto compró el cliente?"
          4.2 Leer subtotal
          4.3 Si subtotal > 1000 entonces
                    4.3.1 descuento = subtotal * 0.10
                    4.3.2 total = subtotal – descuento
              si no
                    4.3.3 total = subtotal
              fin si
          4.4ingresos = ingresos + total
          4.5Escribir "el total a pagar es:", total
          4.6Escribir "Hay más clientes en la tienda:"
          4.7Leer resp
       fin mientras
   5. Escribir "ingresos:", ingresos
   6. Fin
```

DIAGRAMA DE FLUJO

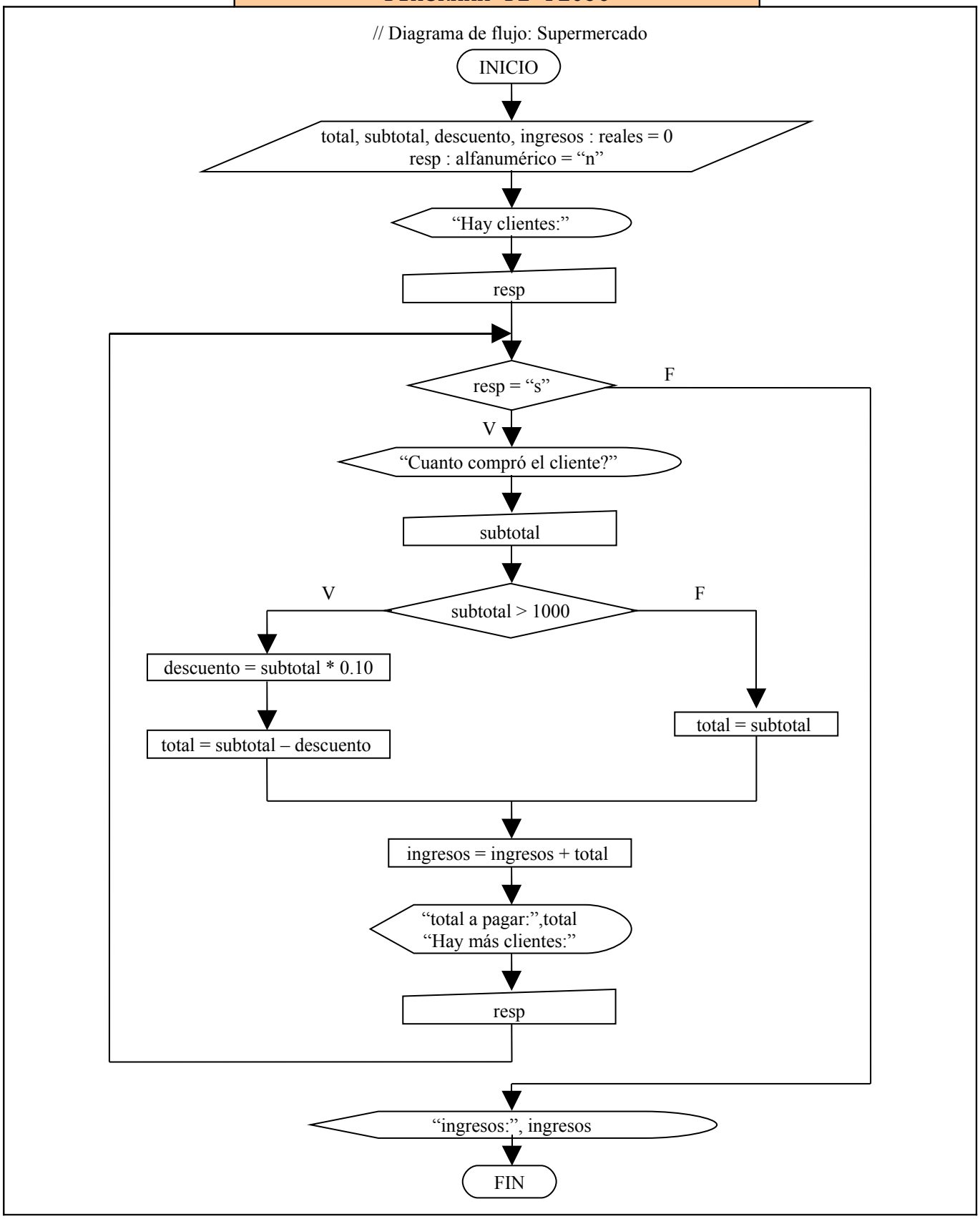


Diagrama N-S

```
// Diagrama N-S: Supermercado
Inicio
variables:
          total, subtotal, descuento, ingresos : reales = 0
          resp : alfanumérico = "n"
Escribir "hay clientes en la tienda"
Leer resp
                                   resp == "s"
    Escribir "cuanto compró el cliente?"
    Leer subtotal
                                   subtotal > 1000
    VERDADERO                                          FALSO
    descuento = subtotal * 0.10  |  total = subtotal
    total = subtotal - descuento |
    ingresos = ingresos + total
    Escribir "el total a pagar es:", total
    Escribir "Hay más clientes en la tienda:"
    Leer resp
Escribir "ingresos:", ingresos
FIN
```

Paso III. Prueba Del Algoritmo.

PRIMERA CORRIDA DE ESCRITORIO

resp = "n"

resp == "s"
"n" == "s" → **NO**

ingresos = 0

SEGUNDA CORRIDA DE ESCRITORIO

resp = "s"

resp == "s"
"s" == "s" → **SI**

subtotal = 1230

subtotal > 1000
1230 > 1000 → **SI**

descuento = subtotal * .10
descuento = 1230 * .10
descuento = 123
total = subtotal - descuento
total = 1230 - 123
total = 1107

ingresos = ingresos + total
ingresos = 0 + 1107
ingresos = 1107

resp = "s"

resp == "s"
"s" == "s" → **SI**

```
       subtotal = 800

       subtotal > 1000
       800 > 1000 → NO

              total = subtotal
              total = 800

       ingresos = ingresos + total
       ingresos = 1107 + 800
       ingresos = 1907

resp = "s"

resp == "s"
"s" == "s" → SI

       subtotal = 4500

       subtotal > 1000
       4500 > 1000 → SI

              descuento = subtotal * .10
              descuento = 4500 * .10
              descuento = 450
              total = subtotal – descuento
              total = 4500 - 450
              total = 4050

       ingresos = ingresos + total
       ingresos = 1907 + 4050
       ingresos = 5957

resp = "s"

resp == "s"
"s" == "s" → SI

       subtotal = 100

       subtotal > 1000
       100 > 1000 → NO

              total = subtotal
              total = 100

       ingresos = ingresos + total
       ingresos = 5957 + 100
       ingresos = 6057

resp = "n"

resp == "s"
"n" == "s" → NO

ingresos = 6057
```

Tabla 31 Ejemplo 2 de estructura cíclica hacer mientras

A continuación se proponen unos cuantos ejercicios, los cuales al resolverlos reforzaran los conocimientos adquiridos. Estos puede ser que necesiten estructuras anidadas.

Ejercicios.

I. Diseña un algoritmo utilizando las tres diferentes técnicas para cada uno de los problemas que se te plantean.
1. Se necesita un sistema que lea los votos obtenidos por tres candidatos a presidente municipal en la ciudad de Orizaba y calcule e imprima al ganador, junto con el porcentaje obtenido de votos.
2. Se necesita un programa para calcular el factorial de un número dado, que corresponda a la fórmula: N! = N*(N-1)*(N-2)* ... *(N-(N-1))
3. Se necesita un sistema que despliegue un menú con 4 opciones, si se presiona la opción 1, se calculará el área de un triangulo; si se presiona la opción 2, se calculará el área de un cuadrado; si se presiona la opción 3, se calculará el área de un circulo; si se presiona la opción 4, será la única forma de salir del sistema.
4. Se necesita un sistema que pide una contraseña. Si la contraseña es igual a "ábrete sésamo", se terminará el programa, de otra manera se seguirá solicitando la contraseña.
5. Se necesita que sistema que calcula perímetros y áreas, para lo cual aparece un menú con tres opciones (1. Perímetros, 2. Áreas, 3. Salir) dentro de las primeras 2 opciones aparece otro menú con 4 opciones (1. Triangulo, 2. Cuadrado, 3. Circulo, 4. Regresar). Dentro del cual solo se puede volver al menú principal presionando la opción 4.

❷ **REPETIR / HASTA...** Estructura cíclica la cual indica un conjunto de instrucciones que se deben de repetir mientras que la respuesta a la condición colocada en lugar de los puntos suspensivos sea falsa. Es decir, que si la respuesta es verdadera se termina de ejecutar el ciclo.

A diferencia de la estructura hacer mientras..., esta estructura se ejecuta siempre al menos una vez, debido a que las instrucciones a ejecutar se encuentran entre las etiquetas **repetir** y **hasta ...**, y la expresión a evaluar se ejecuta después de la última instrucción dentro del ciclo. Aún así es muy probable que la estructura se ejecute infinidad de veces debido a las siguientes causas:

- La variable a evaluar no tiene ningún valor almacenado.
- Nunca se le pidió al usuario que almacenará un dato en la variable.
- El usuario decidió ingresar nuevamente a la estructura.

Sugerencia. Se recomienda que la variable a ser evaluada sea inicializada con un valor que permita romper la estructura, para evitar tener un ciclo infinito.

Aspecto Crítico. Siempre coloque dentro de la estructura **repetir / hasta ...** las instrucciones que permitan al usuario o al sistema almacenar un nuevo valor en la variable a evaluar para evitar un ciclo infinito.

A continuación vamos a esquematizar el diseño básico de esta estructura en las tres técnicas algorítmicas.

Pseudocódigo. En pseudocódigo se utilizan las instrucciones que hemos estado mencionando.

```
Pseudocódigo: ciclo repetir hasta
// Imprime HOLA al menos una vez y todas las veces que el
// usuario presione algo diferente a "n"
Variables:
        Resp : alfanumérica : trabajo = "n"
  1. Inicio
  2. Repetir
        2.1  Escribir "hola"
        2.2  Escribir "deseas ingresar nuevamente al ciclo:"
        2.3  Leer Resp
     Hasta Resp == "n"
  3. Escribir "Gracias por usar este sistema"
  4. FIN
```

Ilustración 17 Pseudocódigo básico del ciclo **Repetir / Hasta...**

Diagrama de Flujo. En diagrama de flujo, se utiliza el **símbolo de decisión** para representarla, pero el símbolo se coloca antes de la siguiente instrucción a ejecutar después de terminada la estructura. Del símbolo salen los dos caminos posibles: el verdadero y el falso. La ruta del lado verdadero conecta con la siguiente instrucción a ejecutar cuando se salga del ciclo. Del camino falso sale una flecha que conecta justo antes de la primera de las instrucciones que deseamos se repitan y de la última instrucción antes del ciclo.

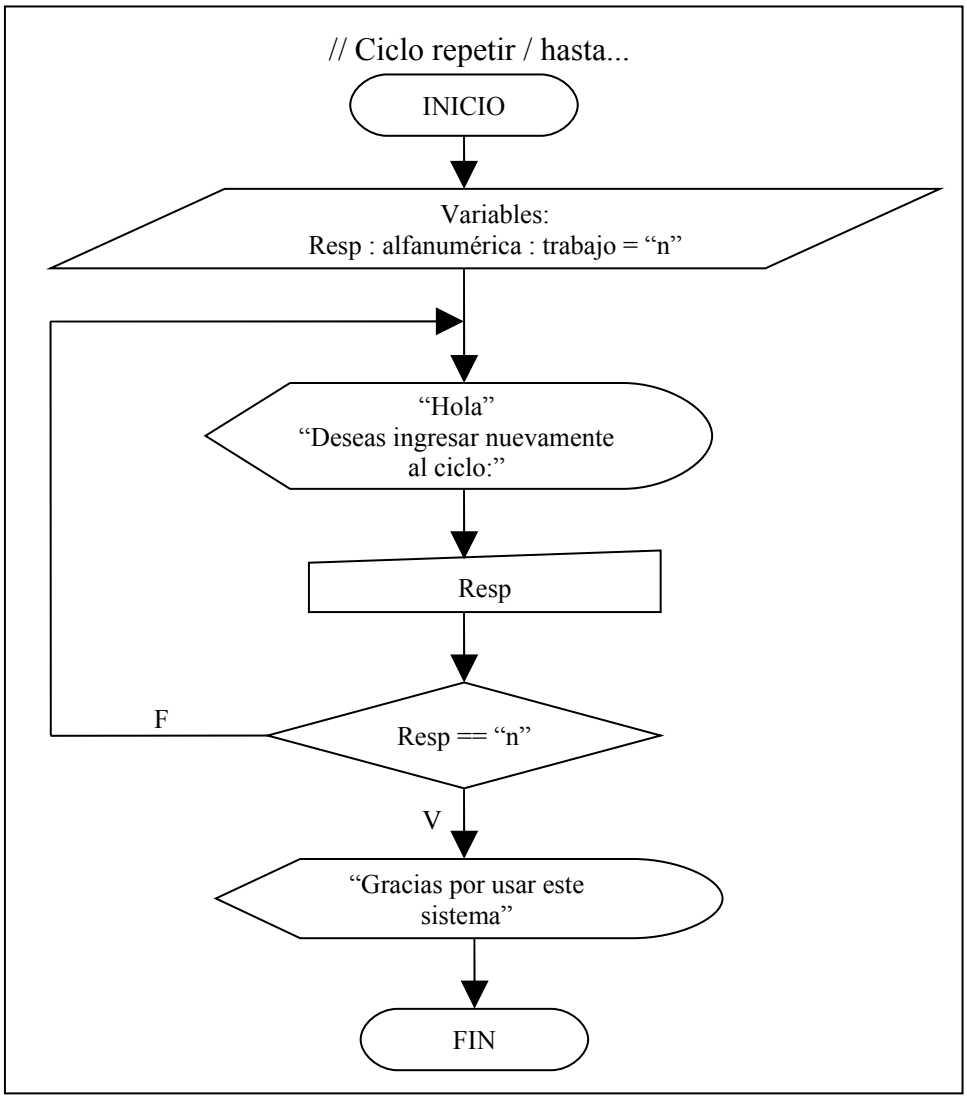


Ilustración 18 Diagrama de flujo del ciclo **Repetir / hasta ...**

Diagramas N-S. Para representar esta estructura existe el símbolo especial **repetir / hasta ...**, el cual es una sola caja con dos partes: la escuadra que parece una **ele mayúscula** (L), es en donde se coloca la expresión a evaluar; la otra parte es la caja que se encuentra entre las dos paredes de la escuadra, es donde se colocan las instrucciones a realizar cuando el ciclo se ejecute.

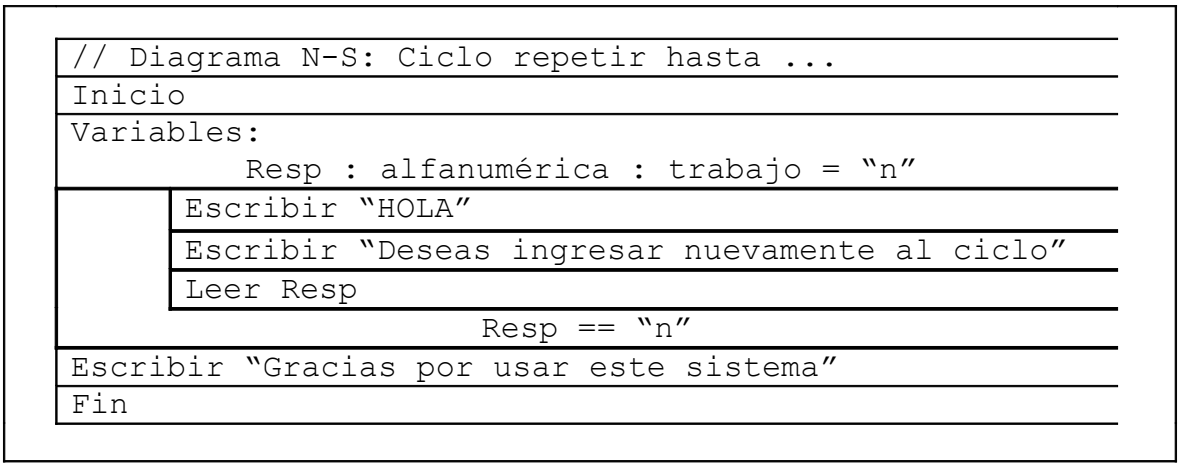


Ilustración 19 Diagrama N-S básico del ciclo **repetir / hasta ...**

Ejemplo ❶	Se necesita un sistema que muestra el cuadrado de los números que introduce el usuario.	
Paso I. Analizar el problema.		
Salidas	**Entrada**	**Procesos**
▪ num_elevado	▪ número ▪ resp	cuando resp != "n" num_elevado = número * número
Paso II. Diseñar El algoritmo		

PSEUDOCÓDIGO

```
Pseudocódigo: Cuadrados

Variables:
        resp : alfanumérico == "s"
        número, num_elevado : entero = 0

  1. Inicio
  2. Repetir
        2.1Escribir "Número que quieres elevar al cuadrado:"
        2.2Leer número
        2.3num_elevado = número * número
        2.4Escribir número, "al cuadrado es:", num_elevado
        2.5Escribir "Deseas calcular otro número:"
        2.6Leer resp
     Hasta resp == "n"
  3. Fin
```

DIAGRAMA DE FLUJO

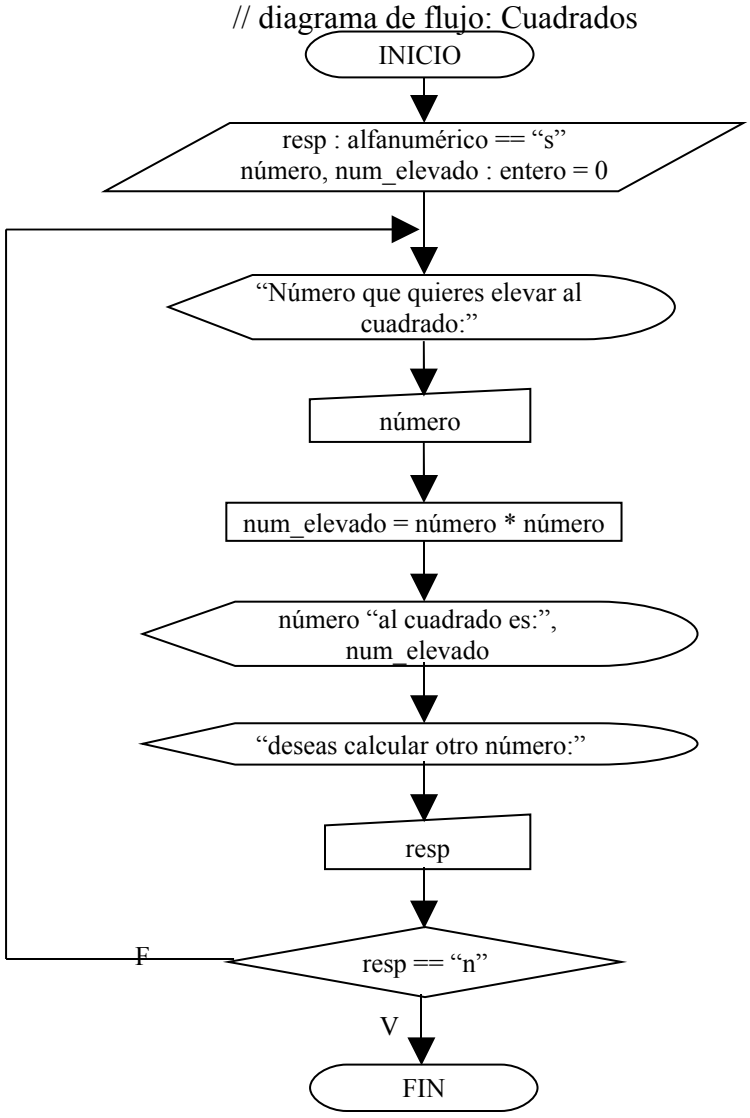


Diagrama N-S

```
// diagrama N-S : Cuadrados
Inicio
Variables:
          resp : alfanumérico == "s"
          número, num_elevado : entero = 0
          Escribir "Número que quieres elevar al cuadrado:"
          Leer número
          num_elevado = número * número
          Escribir número, "al cuadrado es:", num_elevado
          Escribir "Deseas calcular otro número:"
          Leer resp
                                  resp == "n"
Fin
```

Paso III. Prueba Del Algoritmo.

PRIMERA CORRIDA DE ESCRITORIO

```
     número = 5
     num_elevado = número * número
     num_elevado = 5 * 5
     num_elevado = 25
     resp = "n"

resp == "n"
"n" == "n" → SI
```

SEGUNDA CORRIDA DE ESCRITORIO

```
     número = 3
     num_elevado = número * número
     num_elevado = 3 * 3
     num_elevado = 9
     resp = "s"

resp == "s"
"s" == "n" → NO

     número = 7
     num_elevado = número * número
     num_elevado = 7 * 7
     num_elevado = 49
     resp = "s"

resp == "s"
"s" == "n" → NO

     número = 10
     num_elevado = número * número
     num_elevado = 10 * 10
     num_elevado = 100
     resp = "s"

resp == "s"
"s" == "n" → NO

     número = 8
     num_elevado = número * número
     num_elevado = 8 * 8
     num_elevado = 64
     resp = "s"

resp == "n"
"n" == "n" → SI
```

Tabla 32 Ejemplo 1 que usa la estructura cíclica **Repetir Hasta...**

En este ejemplo, como pudimos observar en la primera corrida de escritorio se ejecutó una vez aún cuando la respuesta a la

condición fue verdadera, cosa que no ocurre con el ciclo **hacer mientras...**

Ejemplo ❷	Se necesita un sistema que calcule el salario mensual de n trabajadores, el cual se obtiene de la siguiente forma: ➢ Si trabaja 40 horas o menos se le paga $16 por hora ➢ Si trabaja más de 40 horas se le paga $16 por cada una de las primeras 40 horas y $20 por cada hora extra.

Paso I. Analizar el problema.

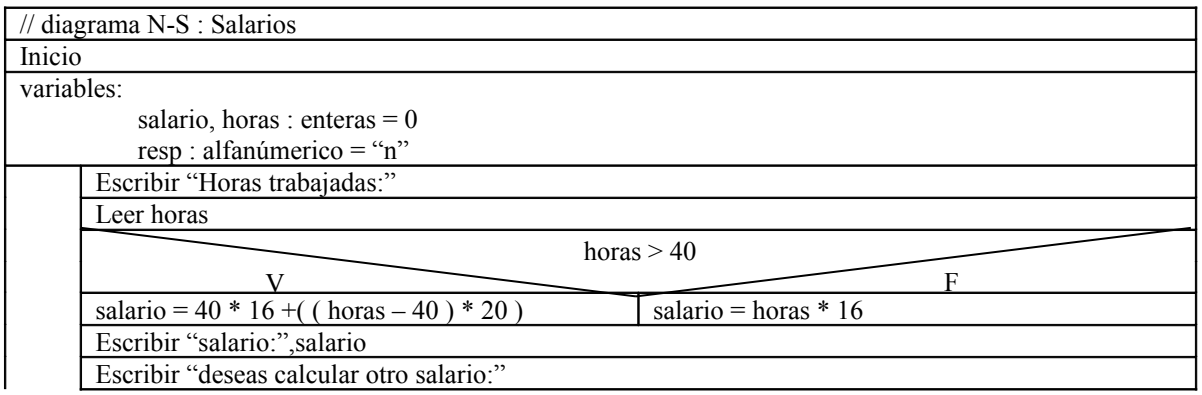
Salidas	Entrada	Procesos
▪ salario	▪ horas ▪ resp	cuando resp != "n" si horas > 40 salario = 40 * 16 +((horas – 40) * 20) si no salario = horas * 16

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Salarios
Variables:
            Salario, horas : enteras = 0
            Resp : alfanúmerico = "n"

   1.  Inicio
   2.  repetir
         2.1Escribir "Horas trabajadas:"
         2.2Leer horas
         2.3si horas > 40 entonces
                  2.3.1 salario = 40 * 16 +( ( horas – 40 ) * 20 )
              si no
                  2.3.2 salario = horas * 16
              fin si
         2.4Escribir "salario:",salario
         2.5Escribir "deseas calcular otro salario:"
         2.6Leer resp
       Hasta resp == "n"
   3.  Fin
```

Diagrama N-S

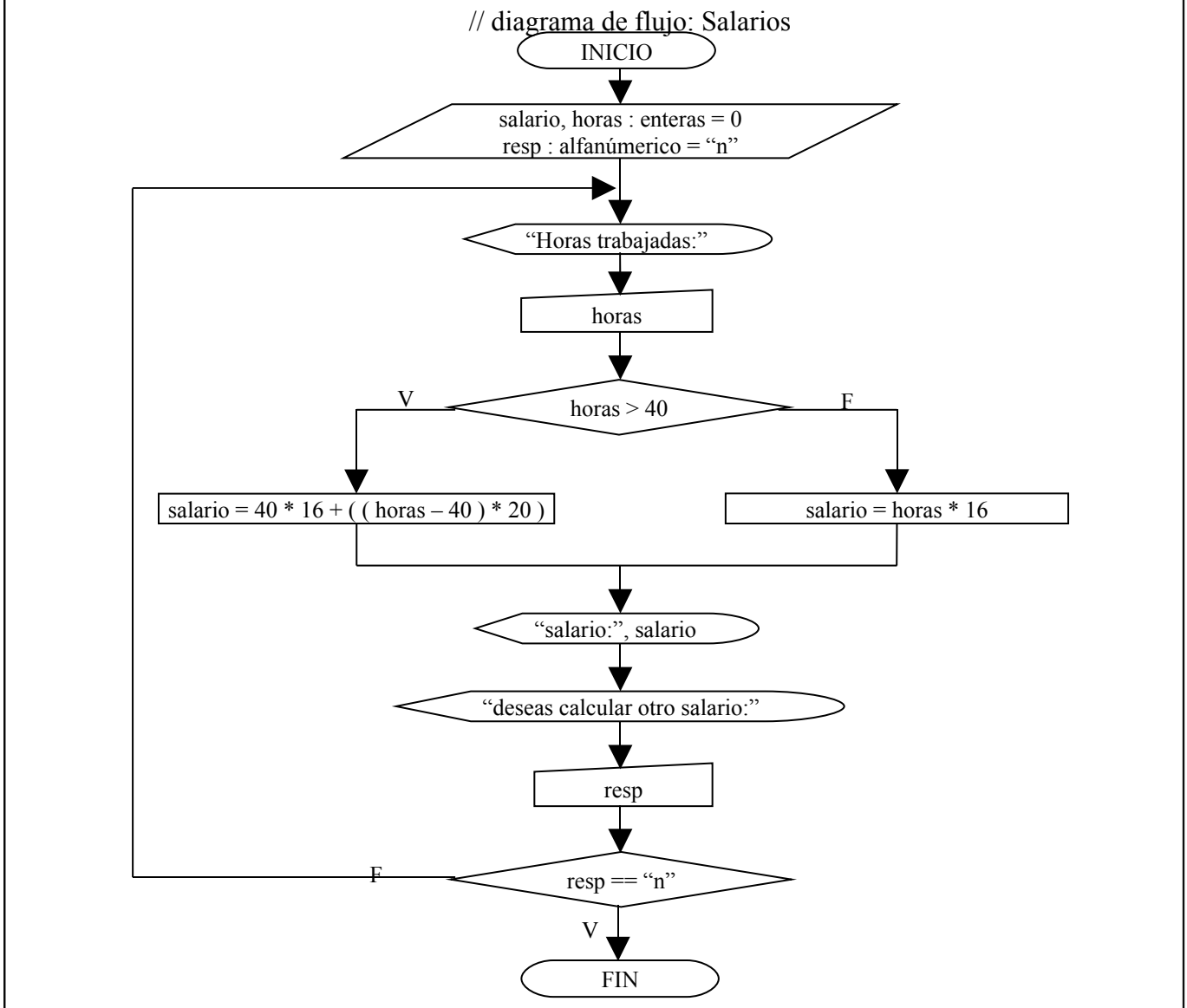


Leer resp

resp == “n”

Fin

DIAGRAMA DE FLUJO



Paso III. Prueba Del Algoritmo.

PRIMERA CORRIDA DE ESCRITORIO

```
Horas = 50

Horas > 40
50 > 40 → SI

          salario = 40 * 16 + ( ( horas – 40 ) * 20 )
          salario = 640 + ( ( 50 – 40 ) * 20 )
          salario = 640 + ( 10 * 20 )
          salario = 640 + 200
          salario = 840

resp = "n"

resp == "n"
"n" == "n" → SI
```

SEGUNDA CORRIDA DE ESCRITORIO

```
Horas = 30
Horas > 40
30 > 40 → NO

          salario = horas * 16
          salario = 30 * 16
          salario = 480

resp = "s"
resp == "n"
"s" == "n" → NO

Horas = 41
Horas > 40
41 > 40 → SI

          salario = 40 * 16 + ( ( horas – 40 ) * 20 )
          salario = 640 + ( ( 41 – 40 ) * 20 )
          salario = 640 + ( 1 * 20 )
          salario = 640 + 20
          salario = 660

resp = "n"
resp == "n"
"n" == "n" → SI
```

Tabla 33 Ejemplo 2 que usa la estructura cíclica **Repetir Hasta...**

Ejercicios.

I. Diseña un algoritmo utilizando las tres diferentes técnicas para cada uno de los problemas que se te plantean.

1. Se necesita un sistema que solicita dos números, los cuales son un rango, de los cuales queremos que imprima el total de la suma de todos los números que se encuentran dentro de este rango
2. Se necesita un sistema que calcula el salario semanal de n trabajadores, el cual depende de su puesto (licenciado, técnico y obrero), del turno (primero, segundo y tercero) y las horas trabajadas. Donde los del primer turno ganan $200 adicionales a su salario, los del segundo $100 y los del tercero $300; El

obrero gana $30 por hora, el técnico $50 y el licenciado $100.
3. Se necesita un sistema que lea los votos obtenidos por tres candidatos a presidente municipal en la ciudad de Orizaba y calcule e imprima al ganador, junto con el porcentaje obtenido de votos.
4. Se necesita un programa para calcular el factorial de un número , que corresponda a la fórmula: N!=N*(N-1)*(N-2)*...*(N-(N-1))
5. Se necesita un sistema que despliegue un menú con 4 opciones, si se presiona la opción 1, se calculará el área de un triangulo; si se presiona la opción 2, se calculará el área de un cuadrado; si se presiona la opción 3, se calculará el área de un circulo; si se presiona la opción 4, será la única forma de salir del sistema.

❸ **HACER PARA... HASTA ...** Estructura cíclica la cual se indica el rango de valores exacto que debe de tener una variable para que un conjunto de instrucciones se repitan. En donde el valor de inicio de la variable se ***asigna*** en lugar de los primeros puntos suspensivos y el último valor de la variable se ***compara*** en lugar de los segundos puntos suspensivos.

Las instrucciones a ejecutar se encuentran entre las instrucciones **hacer para ... hasta ...** y **fin para**, y estas se ejecutarán mientras la respuesta a la expresión colocada en los segundos puntos suspensivos sea falsa, en el momento que la respuesta sea verdadera el ciclo se termina.

Aun así, el ciclo se puede ejecutar infinidad de veces debido a la falta de una instrucción que permita incrementar o decrementar el valor de la variable a evaluar.

También es posible que nunca se ejecute el ciclo debido a que la asignación de la variable a evaluar colocada en lugar de los primeros puntos suspensivos sea un valor que de cómo resultado

verdadero a la condición colocada en lugar de los segundos puntos suspensivos.

Aspecto Crítico. Siempre coloque las instrucciones que permiten incrementar o decrementar el valor de la variable a evaluar dentro de la estructura **hacer para ... hasta ...** para evitar un ciclo infinito.

Aspecto Crítico. Siempre en el lugar de los primeros puntos suspensivos de la estructura **hacer para ... hasta ...,** asígnele un valor a la variable que de cómo resultado falso a la condición colocada en lugar de los segundos puntos suspensivos para que pueda ingresar al ciclo al menos una vez.

A continuación vamos a esquematizar el diseño básico de la estructura cíclica hacer para... en las tres técnicas algorítmicas manejadas.

Pseudocódigo. En pseudocódigo se utilizan las instrucciones que hemos estado mencionando.

```
Pseudocódigo: ciclo hacer para
// imprime hola 10 veces
Variables:
        Cont : numérica : contador
  1. Inicio
  2. Hacer para Cont = 1 hasta cont > 10
        6.1  Escribir "hola"
        6.2  Cont = Cont + 1 // incrementar en 1 la variable
     Fin para
  3. Escribir "Gracias por usar este sistema"
  4. FIN
```

Ilustración 20 Diseño básico del ciclo **hacer para ... hasta ...**

Diagrama de Flujo. En diagrama de flujo, se utiliza un símbolo especial para representarla, el cual es un rectángulo dividido en tres partes, en la primera se realiza la asignación del valor inicial de la variable, en la segunda se coloca la expresión a evaluar, en la tercera se coloca la instrucción para realizar el incremento de la variable. Del símbolo en la parte de la condición salen los dos caminos posibles: el verdadero y el falso. La ruta del lado verdadero conecta con la siguiente instrucción a ejecutar cuando se salga del ciclo. Del camino falso sale una flecha que conecta con las instrucciones a ejecutar por el ciclo y de la última instrucción sale una flecha que conecta nuevamente con este símbolo en la parte del incremento. La instrucción antes del ciclo debe de conectar con el símbolo en la parte de asignación.

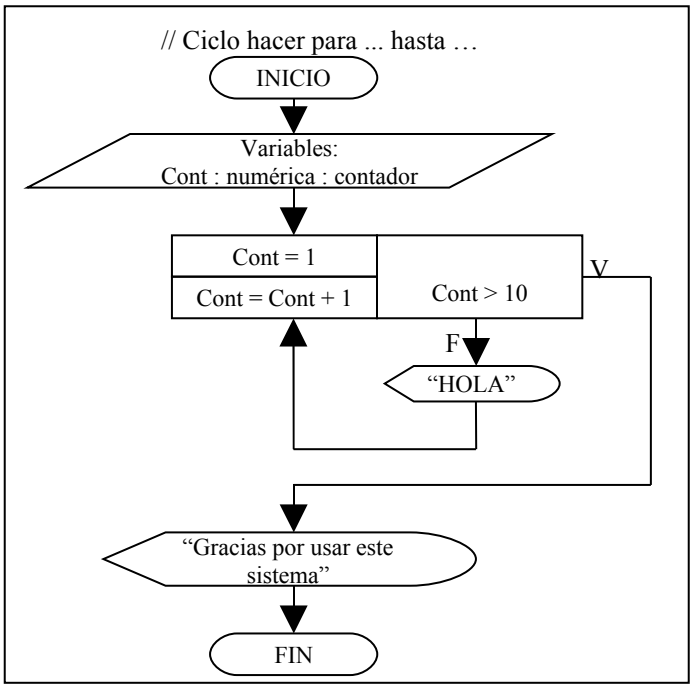


Ilustración 21 Diagrama de Flujo del ciclo **hacer para ... hasta ...**

Diagramas N-S. Para representar esta estructura existe el símbolo especial **hacer para ... hasta ...**, el cual es una sola caja con dos partes: la parte que parece una **herradura** o una "**C**", es en donde se coloca el valor de inicio de la variable (parte izquierda de

la herradura), la expresión a evaluar(parte superior de la herradura) y la instrucción que realiza el incremento o decremento de la variable(parte inferior de la herradura); la otra parte es la caja que se encuentra entre las tres paredes de la herradura, es donde se colocan las instrucciones a realizar cuando el ciclo se ejecute.

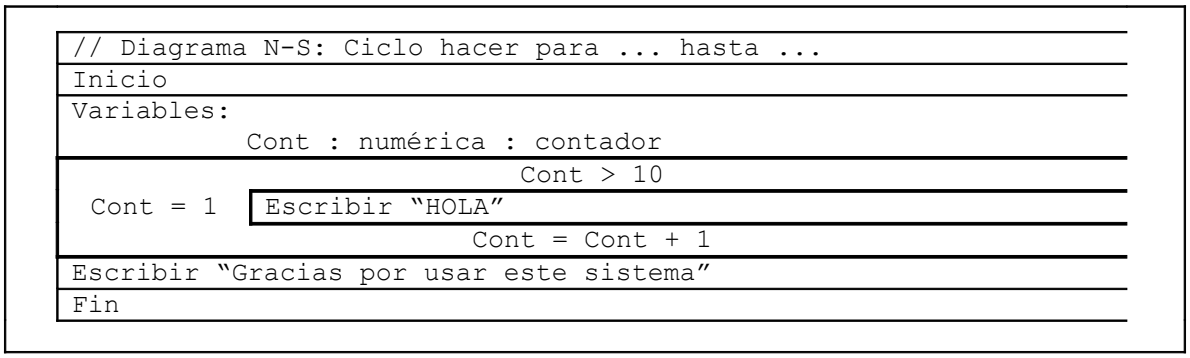


Ilustración 22 Diagrama N-S básico del ciclo **hacer para ... hasta ...**

A continuación se presentan un par de ejercicios con las tres técnicas algorítmicas los cuales manejan esta estructura.

Ejemplo ❶	Se necesita un sistema que despliega una tabla de multiplicar de un número dado por el usuario.	
Paso I. Analizar el problema.		
Salidas	**Entrada**	**Procesos**
▪ resultado	▪ tabla	mientras contador <= 10 resultado = tabla * contador contador = contador + 1
Paso II. Diseñar El algoritmo		

PSEUDOCÓDIGO

```
Pseudocódigo: Tabla
Variables:
        tabla, contador, resultado : enteras = 0

1. Inicio
2. Escribir "tabla que deseas visualizar"
3. Leer tabla
4. Hacer para contador = 1 hasta contador > 10
        4.1 resultado = tabla * contador
        4.2 Escribir tabla, "*", contador, "=", resultado
        4.3 contador = contador + 1
    Fin para
5. Fin
```

DIAGRAMA DE FLUJO

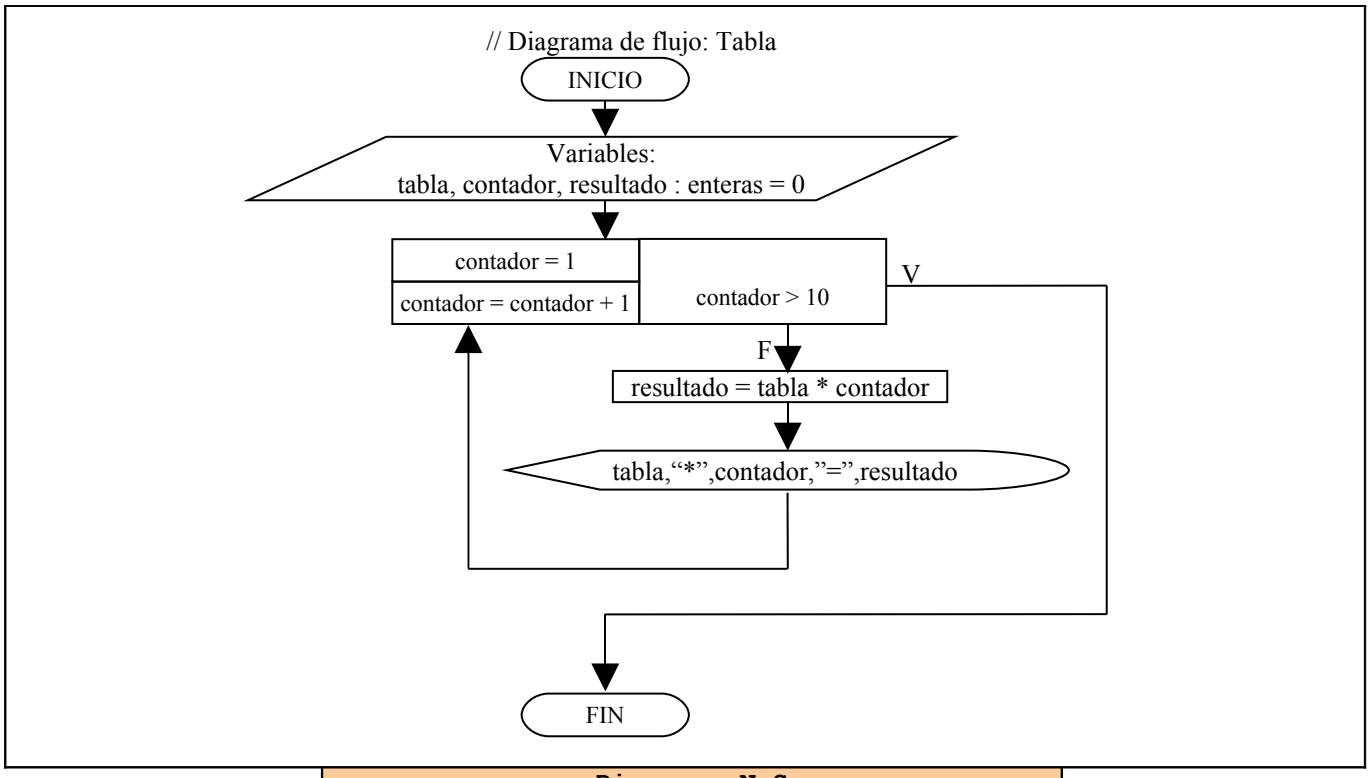


Diagrama N-S

```
// diagrama N-S : Tabla
Inicio
Variables:
          tabla, contador, resultado : enteras = 0
Escribir "tabla que deseas visualizar"
Leer tabla
                                  contador > 10
contador = 1 | resultado = tabla * contador
             | Escribir tabla, "*", contador, "=", resultado
                                  contador = contador + 1
Fin
```

Paso III. Prueba Del Algoritmo.

ÚNICA CORRIDA DE ESCRITORIO

```
tabla = 5

contador = 1

contador > 10
1 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 1
              resultado = 5
contador = contador + 1
contador = 1 + 1
contador = 2

contador > 10
2 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 2
              resultado = 10
contador = contador + 1
contador = 2 + 1
contador = 3

contador > 10
3 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 3
              resultado = 15
contador = contador + 1
contador = 3 + 1
contador = 4

contador > 10
4 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 4
              resultado = 20
contador = contador + 1
contador = 4 + 1
contador = 5

contador > 10
5 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 5
              resultado = 25
contador = contador + 1
contador = 5 + 1
contador = 6

contador > 10
6 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 6
              resultado = 20
contador = contador + 1
contador = 6 + 1
contador = 7

contador > 10
7 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 7
              resultado = 35
contador = contador + 1
contador = 7 + 1
contador = 8

contador > 10
8 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 8
              resultado = 40
contador = contador + 1
contador = 8 + 1
contador = 9

contador > 10
9 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 9
              resultado = 45
contador = contador + 1
contador = 9 + 1
contador = 10

contador > 10
10 > 10 → no
              resultado = tabla * contador
              resultado = 5 * 10
              resultado = 50
contador = contador + 1
contador = 10 + 1
contador = 11

contador > 10
11 > 10 → si
```

Tabla 34 Ejemplo 1 de la estructura cíclica **hacer para... hasta...**

Ejemplo ❷	Se necesita un sistema que despliega las tablas de multiplicar del uno al tres(cada tabla del 1 al 5).
Paso I. Analizar el problema.	
Salidas	**Procesos**

▪ resultado	multiplicador = 1
	mientras multiplicador <= 3
	  multiplicando = 1
	  mientras multiplicando <= 5
	  resultado = multiplicador * multiplicando
	  multiplicando = multiplicando + 1
	  multiplicador = multiplicador + 1

Paso II. Diseñar El algoritmo	
PSEUDOCÓDIGO	

```
pseudocódigo: tablas

variables:
```

```
        multiplicador, multiplicando, resultado : enteras = 0

1.  Inicio
2.  Hacer para multiplicador = 1 hasta multiplicador > 3
     2.1 Hacer para multiplicando = 1 hasta multiplicando > 5
          2.1.1 resultado = multiplicador * multiplicando
          2.1.2 escribir multiplicador, "*", multiplicando, "=", resultado
          2.1.3 multiplicando = multiplicando + 1
          Fin para
     2.2multiplicador = multiplicador + 1
     Fin para
3.  Fin
```

Diagrama N-S

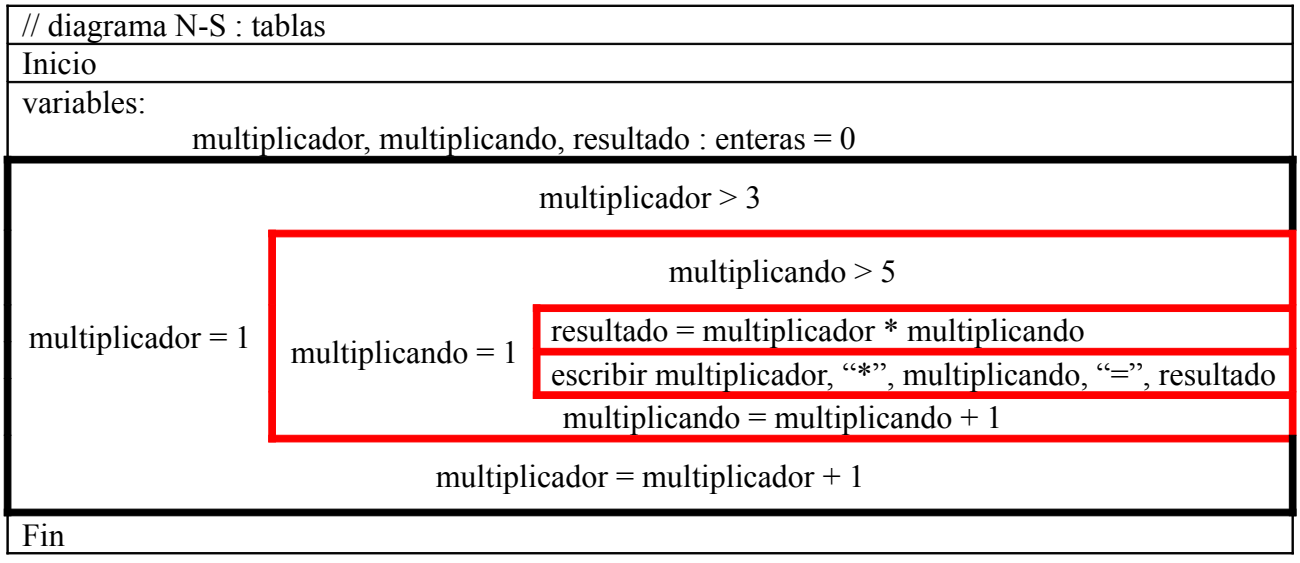
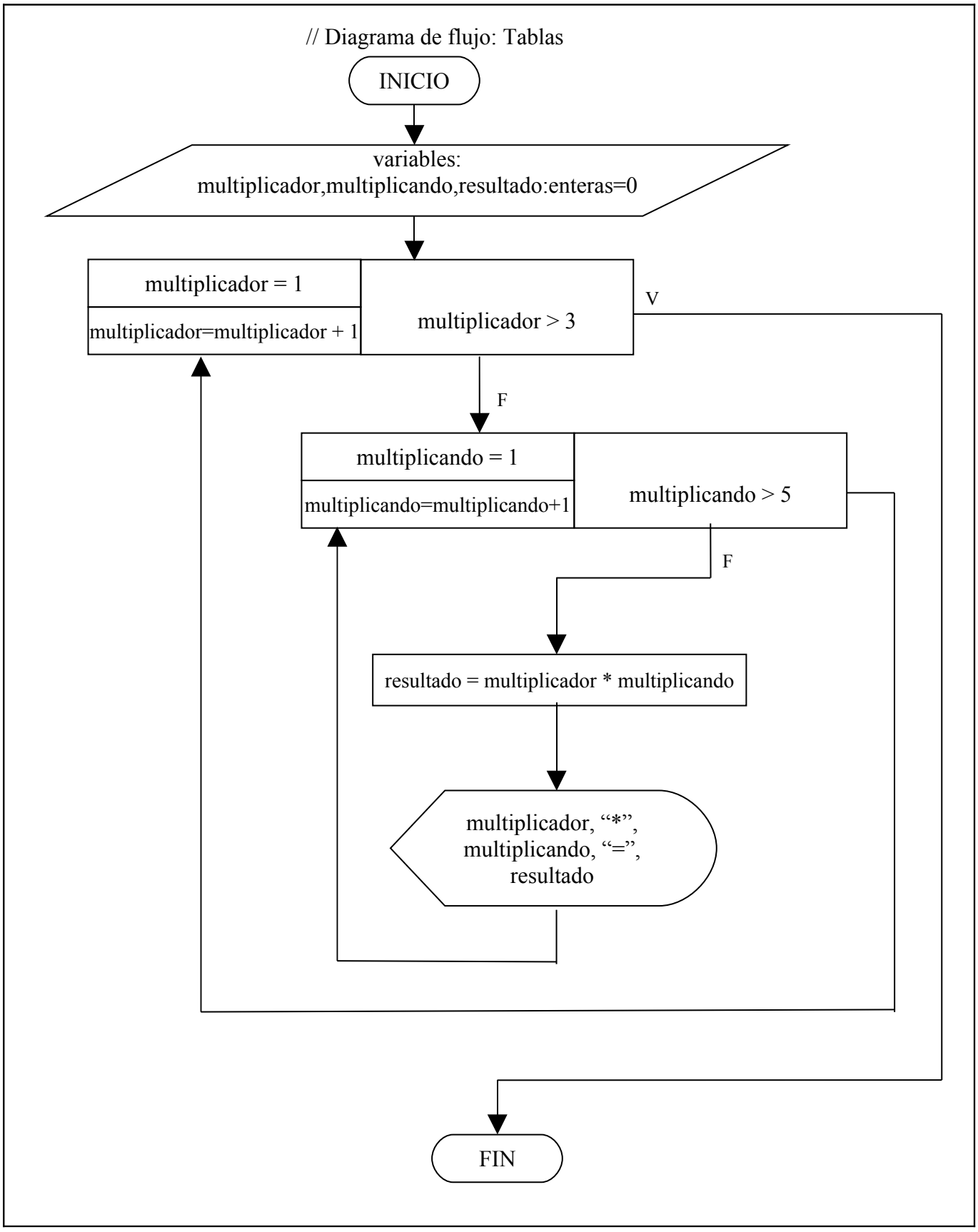


DIAGRAMA DE FLUJO


// Diagrama de flujo: Tablas
INICIO
variables:
multiplicador,multiplicando,resultado:enteras=0
multiplicador = 1
multiplicador=multiplicador + 1
multiplicador > 3
V
F
multiplicando = 1
multiplicando=multiplicando+1
multiplicando > 5
F
resultado = multiplicador * multiplicando
multiplicador, “*”,
multiplicando, “=”,
resultado
FIN

Paso III. Prueba Del Algoritmo.

ÚNICA CORRIDA DE ESCRITORIO

multiplicador = 1

multiplicador > 3
1 > 3 → **NO**

multiplicando = 1

multiplicando > 5
1 > 5 → **NO**

resultado = multiplicador * multiplicando
resultado = 1 * 1
resultado = 1

multiplicando = multiplicando + 1
multiplicando = 1 + 1
multiplicando = 2
multiplicando > 5
2 > 5 → **NO**

resultado = multiplicador * multiplicando
resultado = 1 * 2
resultado = 2

multiplicando = multiplicando + 1
multiplicando = 2 + 1
multiplicando = 3
multiplicando > 5
3 > 5 → **NO**

resultado = multiplicador * multiplicando
resultado = 1 * 3
resultado = 3

multiplicando = multiplicando + 1
multiplicando = 3 + 1
multiplicando = 4
multiplicando > 5
4 > 5 → **NO**

resultado = multiplicador * multiplicando
resultado = 1 * 4
resultado = 4

multiplicando = multiplicando + 1
multiplicando = 4 + 1
multiplicando = 5
multiplicando > 5
5 > 5 → **NO**

resultado = multiplicador * multiplicando
resultado = 1 * 5
resultado = 5

multiplicando = multiplicando + 1
multiplicando = 5 + 1
multiplicando = 6
multiplicando > 5
6 > 5 → **SI**

multiplicador = multiplicador + 1
multiplicador = 1 + 1
multiplicador = 2

multiplicador > 3
2 > 3 → **NO**

multiplicando = 1

multiplicando > 5
1 > 5 → **NO**

resultado = multiplicador * multiplicando
resultado = 2 * 1
resultado = 2

multiplicando = multiplicando + 1
multiplicando = 1 + 1
multiplicando = 2

multiplicando > 5
2 > 5 → **NO**
resultado = multiplicador * multiplicando
resultado = 2 * 2
resultado = 4
multiplicando = multiplicando + 1
multiplicando = 2 + 1
multiplicando = 3
multiplicando > 5
3 > 5 → **NO**
resultado = multiplicador * multiplicando
resultado = 2 * 3
resultado = 6
multiplicando = multiplicando + 1
multiplicando = 3 + 1
multiplicando = 4
multiplicando > 5
4 > 5 → **NO**
resultado = multiplicador * multiplicando
resultado = 2 * 4
resultado = 8
multiplicando = multiplicando + 1
multiplicando = 4 + 1
multiplicando = 5
multiplicando > 5
5 > 5 → **NO**
resultado = multiplicador * multiplicando
resultado = 2 * 5
resultado = 10
multiplicando = multiplicando + 1
multiplicando = 5 + 1
multiplicando = 6
multiplicando > 5
6 > 5 → **SI**

multiplicador = multiplicador + 1
multiplicador = 2 + 1
multiplicador = 3

multiplicador > 3
3 > 3 → **NO**

multiplicando = 1

multiplicando > 5
1 > 5 → **NO**
resultado = multiplicador * multiplicando
resultado = 3 * 1
resultado = 3
multiplicando = multiplicando + 1
multiplicando = 1 + 1
multiplicando = 2
multiplicando > 5
2 > 5 → **NO**
resultado = multiplicador * multiplicando
resultado = 3 * 2
resultado = 6
multiplicando = multiplicando + 1
multiplicando = 2 + 1
multiplicando = 3
multiplicando > 5
3 > 5 → **NO**
resultado = multiplicador * multiplicando
resultado = 3 * 3
resultado = 9
multiplicando = multiplicando + 1
multiplicando = 3 + 1
multiplicando = 4
multiplicando > 5
4 > 5 → **NO**

```
                                    resultado = multiplicador * multiplicando
                                    resultado = 3 * 4
                                    resultado = 12
            multiplicando = multiplicando + 1
            multiplicando = 4 + 1
            multiplicando = 5

            multiplicando > 5
            5 > 5 → NO
                                    resultado = multiplicador * multiplicando
                                    resultado = 3 * 5
                                    resultado = 15
            multiplicando = multiplicando + 1
            multiplicando = 5 + 1
            multiplicando = 6

            multiplicando > 5
            6 > 5 → SI

multiplicador = multiplicador + 1
multiplicador = 3 + 1
multiplicador = 4

multiplicador > 3
4 > 3 → SI
```

Tabla 35 Ejemplo 2 de la estructura cíclica **Hacer Para... hasta...**

Ejercicios.

I. Diseña un algoritmo con la estructura **Hacer para ... hasta ...** utilizando las tres diferentes técnicas para cada uno de los problemas que se te plantean.
1. Calcular el promedio de un alumno que tiene 7 calificaciones en la materia de Diseño Estructurado de Algoritmos
2. Calcular el promedio de 10 alumnos los cuales tienen 7 calificaciones cada uno en la materia Diseño Estructurado de Algoritmos.
3. Leer 10 números y obtener su cuadrado y cubo.
4. Leer 10 números e imprimir solamente los números positivos
5. Leer 20 números e imprimir cuantos son positivos, cuantos negativos y cuantos neutros.
6. Leer 15 números negativos y convertirlos a positivos e imprimir dichos números.
7. Suponga que se tiene un conjunto de calificaciones de un grupo de 40 alumnos. Realizar un algoritmo para mostrar la calificación más alta y la calificación mas baja de todo el grupo.
8. Simular el comportamiento de un reloj digital, imprimiendo la hora, minutos y segundos de un día desde las 0:00:00 horas hasta las 23:59:59 horas

CONCLUSIÓN

En este tema, se han visto tres subtemas, cada una más importante que el anterior:

En el primero se dio a conocer las estructuras secuénciales para diseñar algoritmos, lo cual no es otra cosa más que colocar las instrucciones en una forma ordenada de tal manera que una sentencia se ejecute después de otra. Esta estructura se aplico utilizando las tres diferentes técnicas algorítmicas.

En el segundo subtema, se trabajó con las estructuras condicionales, las cuales nos sirven para darle capacidad de "razonamiento" a nuestros sistemas gracias al manejo de los operadores relacionales, ya que al comparar dos valores el sistema devuelve una respuesta (falso o verdadero) y con esta elegir un camino por el cual avanzar. Al mismo tiempo utilizamos dos tipos de estructuras condicionales: las simples, en el cual solo se puede ir por uno de dos caminos posibles, representadas por la instrucción *si ... entonces*; las múltiples, representadas con la instrucción *casos para ...*en las que solo se puede avanzar por una de varias rutas posibles.

En el tercer subtema, se aprendió a diseñar sistemas que utilizan las estructuras cíclicas las cuales permiten repetir un conjunto específico de instrucciones. Dentro de estas encontramos tres diferentes: la estructura *hacer mientras...* es la que nos brinda la posibilidad ciclar mientras la respuesta a la comparación hecha sea verdadera, ya que cuando esta sea falsa se romperá bucle, tomando en cuenta que es posible nunca entrar en este; La estructura *repetir / hasta...* es aquella que utilizamos para que al menos una vez se ejecuten las instrucciones que se encuentran dentro del ciclo,

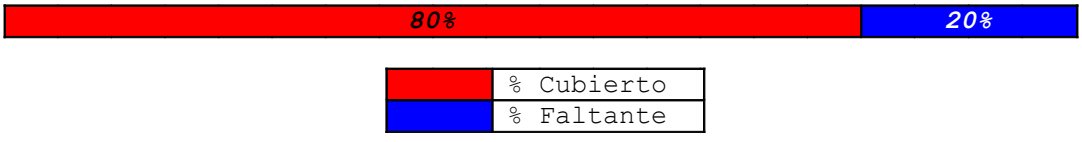
debido a que la comparación se encuentra al final de la estructura y en donde si la respuesta a esta es verdadero se sale del bucle y si es falsa las sentencias se volverán a ejecutar; la estructura *hacer para ... hasta ...*, es aquella que utilizamos para ejecutar un conjunto de instrucciones un número exacto de veces, ya que en base a una variable de uso contador se le asignan valores de inicio y final, y cuando se exceda se acabará la repetición.

Dentro de este mismo tema, se manejo el concepto de anidación, el cual consiste en colocar una estructura dentro de otra, respetando la regla de que para terminar una estructura externa primero se debe concluir una interna.

En este tema, se ha logrado prácticamente el objetivo del curso, ya que con los conocimientos adquiridos en el presente se puede considerar que hemos adquirido una mentalidad de programador, ya que si se nos presentar cualquier problema nosotros tendremos la capacidad de poderlo resolver por medio de un sistema de información computacional.

Por lo mismo, si no se esta seguro de haber asimilado al 100% el tema, se recomienda hacer todos los repasos pertinentes ya que de otra manera no podremos resolver problemas orientados a computadoras, pues se considera que hemos llegado al 80% del objetivo general.

OBJETIVO DEL CURS0



TEMA V.
ARREGLOS
Y
ESTRUCTURAS

V. ARREGLOS Y ESTRUCTURAS

OBJETIVO	Al finalizar el tema el participante mediante la elaboración de ejercicios, manejará los arreglos y estructuras utilizando las tres diferentes técnicas algorítmicas con la finalidad de almacenar información.

CONTENIDO

INTRODUCCIÓN

Este tema es muy sencillo de comprender, siempre y cuando se tengan bien afianzados los temas anteriores, ya que los arreglos y las estructuras son una especie de variables un poco más complejas.

Nosotros utilizaremos las estructuras y arreglos para almacenar datos, a diferencia de que en una variable solo se puede almacenar un dato. Sin embargo nos daremos cuenta de que estos no son iguales, ya que en una estructura podremos almacenar varios datos de diferentes tipos y en un arreglo se pueden guardar varios datos pero del mismo tipo.

Este tema para su absoluta comprensión esta dividido en dos subtemas, donde en el primero se estudian y trabaja con los arreglos y en el segundo con las estructuras. Cabe mencionar que dentro de los arreglos estudiaremos dos tipos: los arreglos unidimensionales y los multidimensionales.

5.1 ARREGLOS

Para explicar que es un arreglo basta con decir lo mismo que dicen las empresas constructoras de casas habitación, ***"si ya no hay espacio en el piso, hay mucho espacio hacia arriba"***, y en realidad han ganado demasiado dinero al comenzar a construir los famosos condominios, los cuales son exactamente iguales en cada uno de sus niveles, solo se identifican por el piso en que se encuentran.

Una variable de un tipo específico es como una casa habitación común y corriente, pero si nosotros le construimos más pisos a nuestra variable esta se convierte en un arreglo, al cual se le puede almacenar información de un mismo tipo en el piso deseado.

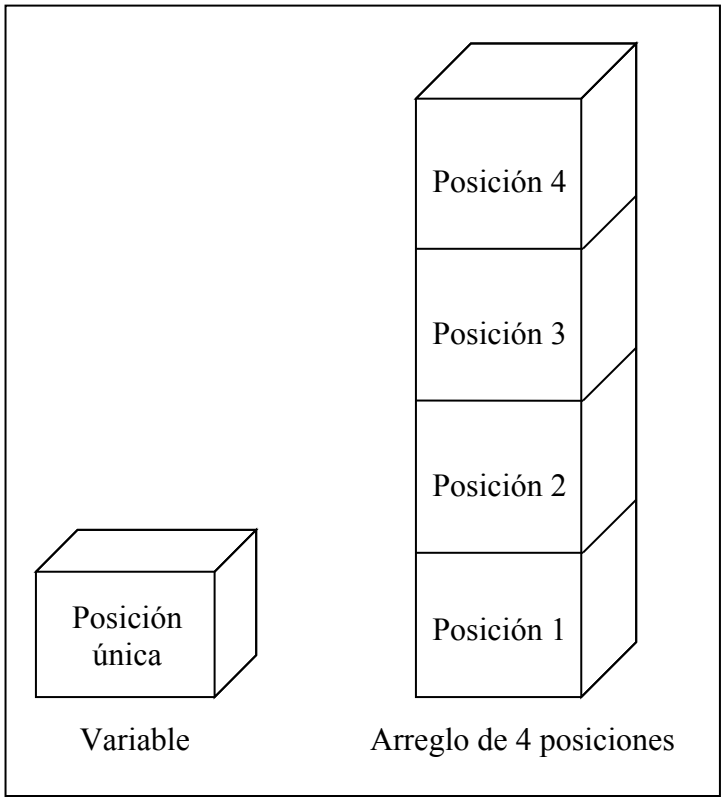


Ilustración 23 Representación y comparación de una variable y un arreglo

Las operaciones que se pueden realizar sobre los arreglos son exactamente las mismas que a las variables: declaración, desplegar, almacenar, asignar, inicializar y comparar. Pero a diferencia de las variables, los arreglos se pueden ordenar ya sea de manera ascendente o descendente.

La declaración de los arreglos no desvaría mucho de la declaración de variables, con la excepción de que se tiene que definir el tamaño del arreglo, el cual se tiene que poner entre paréntesis cuadrados o corchetes, aunque para menor confusión los declaramos en una sección llamada arreglos.

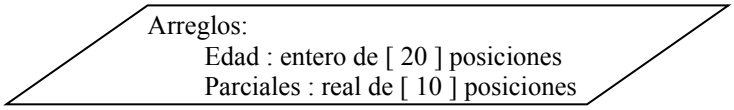
PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
Arreglos: Edad : entero de [20] posiciones Parciales : real de [10] posiciones	

Ilustración 24 Forma en que se declaran los arreglos

Nota. El número entre corchetes se llama subíndice o índice.

Al declarar un arreglo, se le pueden **dar valores de inicio** para cada una de sus posiciones, pera lo cual después de indicar el tamaño de este se coloca un signo de igual y entre llaves "{}", los valores de cada posición separados por comas.

PSEUDOCÓDIGO Y DIAGRAMA N-S

```
Arreglos:
    Edad : entero de [ 5 ] posiciones = {25, 9, 18, 20, 31}
    Parciales : real de [ 10 ] posiciones = {0}
    // si se coloca solo un cero todas las posiciones toman este valor
```

DIAGRAMA DE FLUJO

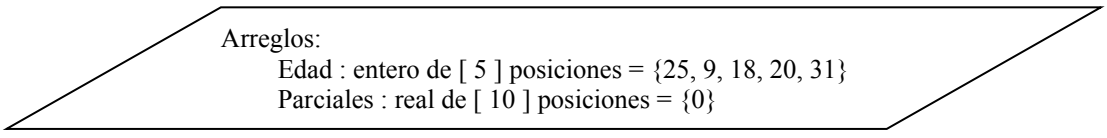


Ilustración 25 Forma en que se inicializan los arreglos

Para **escribir** lo que contiene un arreglo debemos de indicar el nombre del arreglo y la posición entre corchetes que deseamos visualizar.

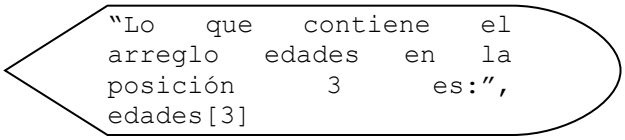
PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
Escribir "lo que contiene el arreglo edades en la posición 3 es:", edades[3]	

Ilustración 26 Forma en que se despliega información desde un arreglo.

Para **almacenar** un dato en una posición específica de un arreglo debemos de indicar el nombre del arreglo y la posición entre corchetes en que deseamos guardar.

PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO

Leer edades[3]
// almacena una dato en la posición 3
// del arreglo edades
	

Ilustración 27 Forma en que se almacena información a un arreglo.

Para **asignar** el resultado de una operación en una posición específica de un arreglo debemos de indicar el nombre del arreglo y la posición entre corchetes en que deseamos colocar el resultado de la expresión.

PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO

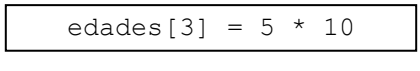
edades[3] = 5 * 10
// almacena una dato en la posición 3
// del arreglo edades
	

Ilustración 28 Forma en que se asignan datos a un arreglo

La comparación se realiza de la misma manera en que se realizan las operaciones ya vistas: colocando el nombre y posición del arreglo que se quiere comparar, el operador relacional y el valor o variable contra quien se coteja, teniendo en cuenta que podría ser otra posición de otro arreglo. No se ejemplifican debido a que se tendría que desarrollar la estructura condicional o cíclica.

A continuación realizamos el primer ejemplo de un algoritmo que utiliza arreglos. Dentro de este problema utilizamos una variable contador llamada **subíndice**, la cual es la que se encarga de apuntar a una posición específica del arreglo, la cual es aumentada dentro de una estructura cíclica **hacer para**.

Ejemplo ❶	Se necesita de un sistema que utiliza un arreglo de seis posiciones para almacenar los 5 parciales de un alumno y sacar su promedio, el cual se guardará en la última localidad. Mostrar todas las calificaciones y el promedio.

Paso I. Analizar el problema.

Salidas	Entrada	Procesos
▪ calif[1] ▪ calif[2] ▪ calif[3] ▪ calif[4] ▪ calif[5] ▪ calif[6]	calif[subíndice]	subíndice = 1 mientras subíndice <= 6 si subíndice == 6 calif[subíndice] acum_calif = acum_calif + calif[subíndice] en caso contrario calif[subindice] = acum_calif / 5

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: calificaciones

Arreglos:
          calif : real de [6] posiciones

Variables:
          subíndice : entera = 0
          acum_calif : real = 0

   1.  Inicio

   2.  Hacer para subíndice = 1 hasta subíndice > 6
        2.1 Si subíndice != 6
                    2.1.1 Escribir "dame calificación de parcial ", subíndice, ":"
                    2.1.2 Leer calif[subíndice]
                    2.1.3 acum_calif = acum_calif + calif[subíndice]
               Si no
                    2.1.4 calif[subindice] = acum_calif / 5
               Fin si

        2.2 subíndice = subíndice + 1
       Fin para

   3.  Hacer para subíndice = 1 hasta subíndice > 6
        3.1 Si subíndice == 6 entonces
                    3.1.1 Escribir "Promedio : ", calif[subíndice]
               Si no
                    3.1.2  Escribir "Parcial ", subíndice, ":", calif[subíndice]
               Fin si
        3.2 subíndice = subíndice + 1
       Fin para

   4.  Fin
```

DIAGRAMA DE FLUJO

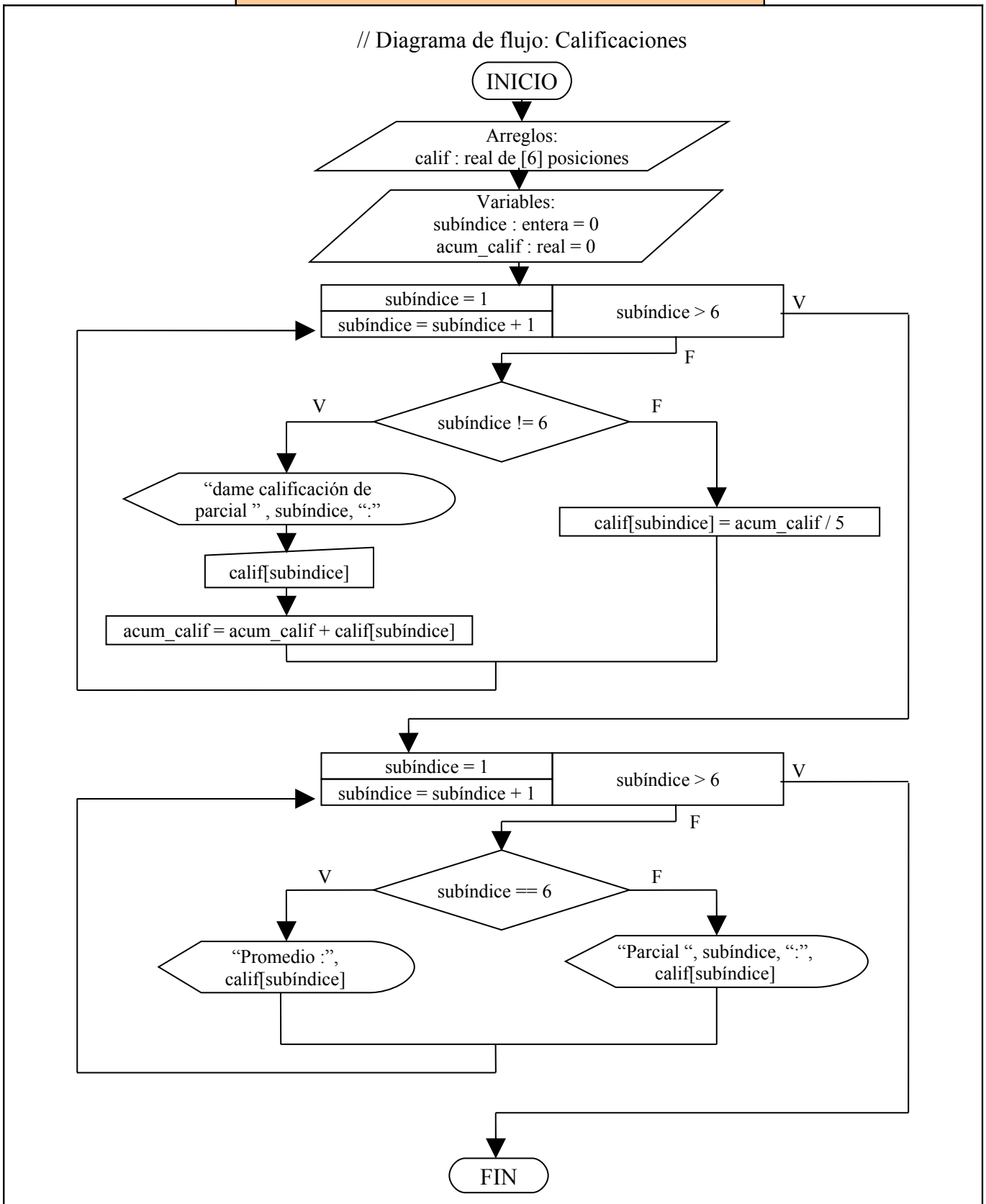
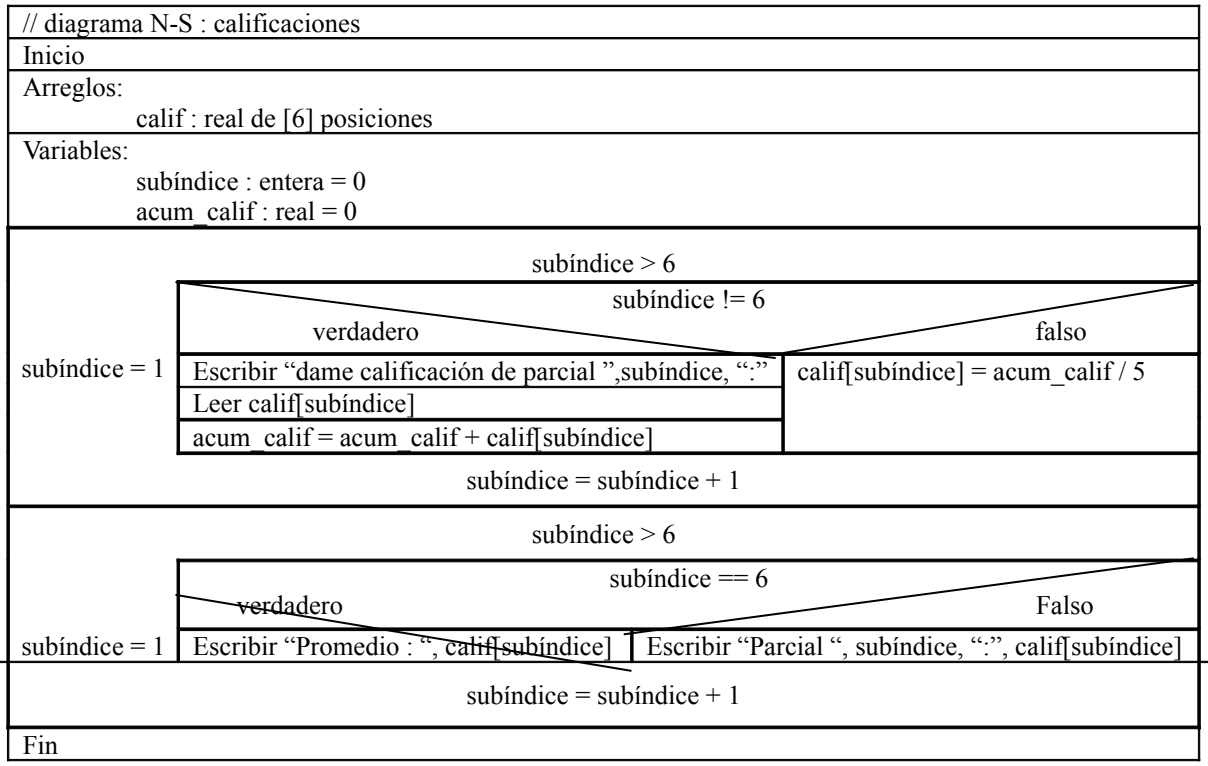


Diagrama N-S



Paso III. Prueba Del Algoritmo.

ÚNICA CORRIDA DE ESCRITORIO

subíndice = 1

subíndice > 6
1 > 6 → **NO**

subíndice != 6
1 != 6 → **SI**

calif[subíndice] = 8
calif[1] = 8
acum_calif = acum_calif + calif[subíndice]
acum_calif = 0 + calif[1]
acum_calif = 0 + 8
acum_calif = 8

subíndice = subíndice + 1
subíndice = 1 + 1
subíndice = 2

subíndice > 6
2 > 6 → **NO**

subíndice != 6
2 != 6 → **SI**

calif[subíndice] = 6
calif[2] = 6
acum_calif = acum_calif + calif[subíndice]
acum_calif = 8 + calif[2]
acum_calif = 8 + 6
acum_calif = 14

subíndice = subíndice + 1
subíndice = 2 + 1
subíndice = 3

subíndice > 6
3 > 6 → **NO**

subíndice != 6
3 != 6 → **SI**

calif[subíndice] = 10
calif[3] = 10
acum_calif = acum_calif + calif[subíndice]
acum_calif = 14 + calif[3]
acum_calif = 14 + 10
acum_calif = 24

subíndice = subíndice + 1
subíndice = 3 + 1
subíndice = 4

subíndice > 6
4 > 6 → **NO**

subíndice != 6
4 != 6 → **SI**

calif[subíndice] = 5
calif[4] = 5
acum_calif = acum_calif + calif[subíndice]
acum_calif = 24 + calif[4]
acum_calif = 24 + 5
acum_calif = 29

subíndice = subíndice + 1
subíndice = 4 + 1
subíndice = 5

subíndice > 6
5 > 6 → **NO**

subíndice != 6
5 != 6 → **SI**

calif[subíndice] = 9
calif[5] = 9
acum_calif = acum_calif + calif[subíndice]
acum_calif = 29 + calif[5]
acum_calif = 29 + 9
acum_calif = 38

subíndice = subíndice + 1
subíndice = 5 + 1
subíndice = 6

subíndice > 6
6 > 6 → **NO**

subíndice != 6
6 != 6 → **NO**

calif[subíndice] = acum_calif / 5
calif[6] = 38 / 5
calif[6] = 7.6

subíndice = subíndice + 1
subíndice = 6 + 1
subíndice = 7

subíndice > 6
7 > 6 → **SI**

Tabla 36 Ejemplo 1 del manejo de arreglos

A continuación vamos a ver en el siguiente ejemplo que con un mismo subíndice se puede acceder a varios arreglos.

✍Ejemplo ❷	Se necesita un sistema que utiliza 3 arreglos, en los dos primeros se colocan los promedios de dos grupos de 5 alumnos cada uno y el tercer arreglo almacenará el promedio más alto de cada posición. Imprimir los promedios más altos

Paso I. Analizar el problema.

Salidas	Entrada	Procesos
▪ prom[índice]	▪ grupo1[índice] ▪ grupo2[índice]	mientras índice <= 5 si grupo1[índice] > grupo2[índice] prom[índice] = grupo1[índice] en caso contrario prom[índice] = grupo2[índice]

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
pseudocódigo: calificaciones de 2 grupos

Arreglos:
      grupo1, grupo2, prom : real de [5] posiciones

Variables:
      índice : entero = 0

   1.  Inicio
   2.  Hacer para índice = 1 hasta índice > 5
          II.1Escribir "dame promedio ", índice, "del primer grupo:"
          II.2Leer grupo1[índice]
          II.3Escribir "dame promedio ", índice, "del segundo grupo:"
          II.4Leer grupo2[índice]
          II.5Si grupo1[índice] > grupo2[índice] entonces
                     II.5.1 prom[indice] = grupo1[índice]
                Si no
                     II.5.2   prom[indice] = grupo2[índice]
                Fin si
          II.6índice = índice + 1
       Fin para
   3.  Hacer para índice = 1 hasta índice > 5
          3.1   Escribir "promedio mayor", índice, ":", prom[índice]
          3.2 índice = índice + 1
       Fin para
   4.  Fin
```

Diagrama N-S

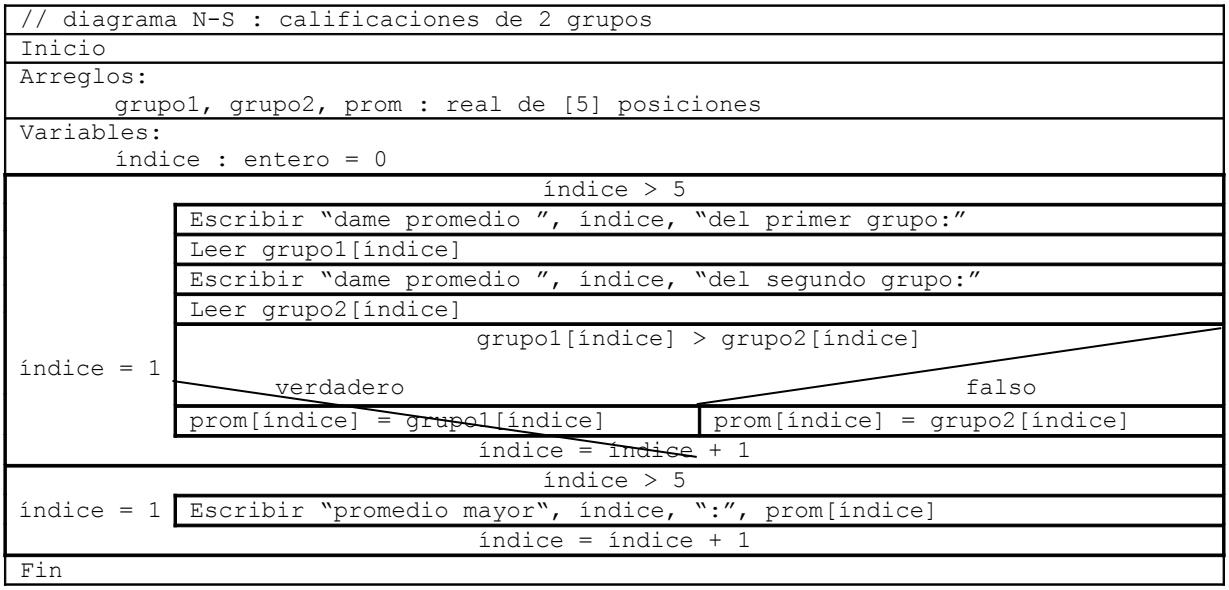
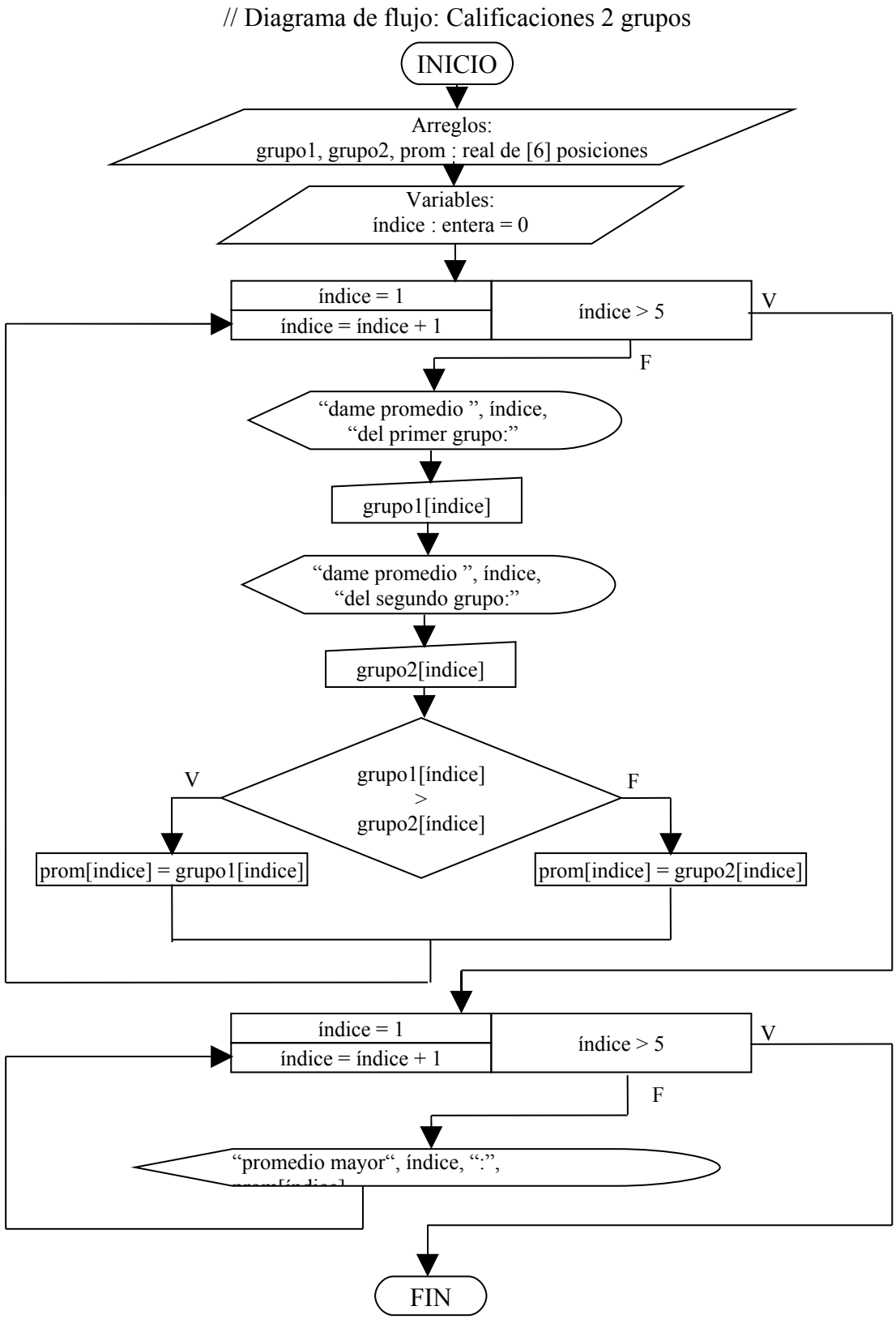


DIAGRAMA DE FLUJO



Paso III. Prueba Del Algoritmo.

ÚNICA CORRIDA DE ESCRITORIO

índice = 1

índice > 5
1 > 5 → **NO**

grupo1[índice] = 8.5
grupo1[1] = 8.5
grupo2[índice] = 6.9
grupo2[1] = 6.9

grupo1[índice] > grupo2[índice]
grupo1[1] > grupo2[1]
8.5 > 6.9 → **SI**

prom[índice] = grupo1[índice]
prom[1] = grupo1[1]
prom[1] = 8.5

índice = índice + 1
índice = 1 + 1
índice = 2

índice > 5
2 > 5 → **NO**

grupo1[índice] = 7
grupo1[2] = 7
grupo2[índice] = 8.7
grupo2[2] = 8.7

grupo1[índice] > grupo2[índice]
grupo1[2] > grupo2[2]
7 > 8.7 → **NO**

prom[índice] = grupo2[índice]
prom[2] = grupo2[2]
prom[2] = 8.7

índice = índice + 1
índice = 2+ 1
índice = 3

índice > 5
3 > 5 → **NO**

grupo1[índice] = 6.8
grupo1[3] = 6.8
grupo2[índice] = 9.5
grupo2[3] = 9.5

grupo1[índice] > grupo2[índice]
grupo1[3] > grupo2[3]
6.8 > 9.5 → **NO**

prom[índice] = grupo2[índice]
prom[3] = grupo2[3]
prom[3] = 9.5

índice = índice + 1
índice = 3 + 1
índice = 4

índice > 5
4 > 5 → **NO**

grupo1[índice] = 9.7
grupo1[4] = 9.7
grupo2[índice] = 9.3
grupo2[4] = 9.3

grupo1[índice] > grupo2[índice]
grupo1[4] > grupo2[4]
9.7 > 9.3 → **SI**

prom[índice] = grupo1[índice]
prom[4] = grupo1[4]
prom[4] = 9.7

índice = índice + 1
índice = 4 + 1
índice = 5

índice > 5
5 > 5 → **NO**

grupo1[índice] = 6
grupo1[5] = 6
grupo2[índice] = 6
grupo2[5] = 6

grupo1[índice] > grupo2[índice]
grupo1[5] > grupo2[5]
6 > 6 → **NO**

prom[índice] = grupo2[índice]
prom[5] = grupo2[5]
prom[5] = 6

índice = índice + 1
índice = 5 + 1
índice = 6

índice > 5
6 > 5 → **SI**

Tabla 37 Ejemplo 2 del manejo de arreglos

Ejercicios.

I. Realiza un algoritmo que maneja arreglos utilizando las tres diferentes técnicas para cada uno de los problemas que se plantean.
1. Un supermercado necesita un sistema en donde almacenar sus ingresos, los cuales son la sumatoria de todas las ventas realizadas a los clientes (100 clientes).
2. Se necesita un sistema que utiliza 2 arreglos para almacenar 20 números, en el primero se almacenan los números tal y como son capturados y en el segundo se almacenan sus inversos (5, -5).
3. Necesitamos un sistema que capture 20 números y después de capturarlos que haga la revisión de estos para indicarnos cuantos son pares y cuantos son impares.
4. Se necesita un sistema que almacena 20 números en tres diferentes arreglos, en el primero se almacena el número tal cual se tecleo, en el segundo se almacena el cuadrado de dicho número y en el tercero su cubo.
5. Se necesita un sistema que almacena automáticamente todos los números primos desde el uno hasta el mil uno; recordando que un número primo es aquel que solamente es divisible entre uno y si mismo.

Ordenar Arreglos (Ordenamiento Tipo Burbuja).

En varias ocasiones cuando desarrollemos un sistema, nos vamos a encontrar con la necesidad de ordenar la información de una manera específica, generalmente en manera ascendente o descendente.

Nosotros vamos a utilizar la manera más sencilla, conocida como ***ordenamiento tipo burbuja***. Este método es llamado así ya que los elementos del vector (arreglo) mayores tienden a irse hasta el fondo del arreglo y los menores comienzan a flotar hacía arriba del mismo, como lo hacen las burbujas de aire en el agua.

Supongamos que hemos declarado un arreglo del tipo flotante llamado edades con 5 posiciones. Dicho vector fue llenado con diversos datos pero queremos que sea ordenado de manera ascendente, quedando de la siguiente manera:

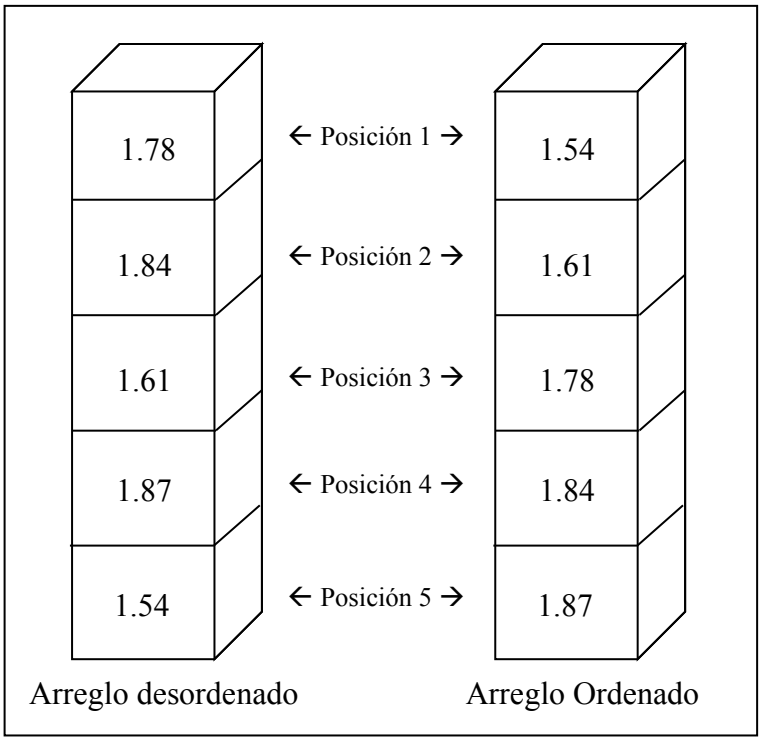


Ilustración 29 Vista de un arreglo llamado edades ordenado

Naturalmente para nosotros resulta muy fácil ejecutar este proceso mental, pero para que la computadora realice este proceso se lleva un buen número de instrucciones, los cuales consisten en ir ordenando poco a poco el arreglo. Las instrucciones son las siguientes (solo las mostramos en pseudocódigo):

```
Hacer para N_pasadas = 1 hasta N_pasadas >= TAM_ARREGLO
     Hacer para posición = 1 hasta posición == TAM_ARREGLO
        Si edades[ posición ] > edades[ posición + 1 ] entonces
           temporal = edades[ posición ]
           edades[ posición ] = edades[ posición + 1 ]
           edades[ posición + 1 ] =  temporal
        fin si
        posición = posición + 1
     Fin para
     N_pasadas = N_pasadas + 1
Fin para
```

Tabla 38 Instrucciones para ordenar el arreglo edades

En este programa la variable ***N_pasadas*** es del tipo entero y es la que lleva el conteo de cuantas veces se va a revisar todo el arreglo, el número de pasadas tiene que ser menor al tamaño del arreglo el cual esta almacenado en la variable ***TAM_ARREGLO***, ya que cuando la variable ***N_pasadas*** ya no sea menor que la variable ***TAM_ARREGLO*** este ya estará completamente ordenado.

La variable ***posición*** es la que nos dirá que localidad del arreglo revisar. Y el secreto de este método radica en comparar dos posiciones contiguas del arreglo (***edades[posición]*** contra ***edades[posición + 1]***), si la primera es mayor que la segunda, lo que se encuentra en ***edades[posición]*** es guardado en la variable ***temporal***, entonces el valor que se encuentra en ***edades[posición+1]*** es asignado en ***edades[posición]***, y por ultimo el dato que esta almacenado en la variable ***temporal*** es dado a ***edades[posición+1]***. En caso de que ***edades[posición]*** no sea mayor que ***edades[posición+1]***, los valores de cada elemento se dejan igual. A continuación se realiza paso a paso el ordenamiento del arreglo edades, en el cual se enumeran todas las comparaciones.

C1. Como el resultado de la primera comparación es falso, el arreglo continúa igual.

N_pasadas=	1
TAM_ARREGLO=	5
Posición=	1
edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[1]	1.78
edades[2]	1.84
edades[3]	1.61
edades[4]	1.87
edades[5]	1.54

➔

edades[1]	1.78
edades[2]	1.84
edades[3]	1.61
edades[4]	1.87
edades[5]	1.54

C2. Como el resultado de la segunda comparación es verdadero, los valores de estas posiciones se intercambian.

N_pasadas=	1
TAM_ARREGLO=	5
Posición=	2
edades[posición] > edades[posición + 1]	V
Temporal=	1.84
Edades[posición] =	1.61
Edades[posición + 1] =	1.84

edades[1]	1.78
edades[2]	1.84
edades[3]	1.61
edades[4]	1.87
edades[5]	1.54

➔

edades[1]	1.78
edades[2]	1.61
edades[3]	1.84
edades[4]	1.87
edades[5]	1.54

C3. Como el resultado de la tercera comparación es falso, el arreglo continúa igual.

N_pasadas=	1
TAM_ARREGLO=	5
Posición=	3
edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[1]	1.78
edades[2]	1.61
edades[3]	1.84
edades[4]	1.87
edades[5]	1.54

➔

edades[1]	1.78
edades[2]	1.61
edades[3]	1.84
edades[4]	1.87
edades[5]	1.54

C4. Como el resultado de la cuarta comparación es verdadero, los valores de estas posiciones se intercambian.

N_pasadas=	1
TAM_ARREGLO=	5
Posición=	4
edades[posición] > edades[posición + 1]	V
Temporal=	1.87
Edades[posición] =	1.54
Edades[posición + 1] =	1.87

edades[1]	1.78
edades[2]	1.61
edades[3]	1.84
edades[4]	1.87
edades[5]	1.54

➔

edades[1]	1.78
edades[2]	1.61
edades[3]	1.84
edades[4]	1.54
edades[5]	1.87

Con esto hemos terminado la primera pasada, ya que al incrementar la posición ahora vale 5 y si realizamos otra comparación tendría que ser lo que esta en la posición 5 contra lo que esta en la posición 6 y esta no existe, por lo cual se debe de terminar el ciclo cuando el valor de posición es igual al tamaño del arreglo (posición == TAM_ARREGLO). Aun así el arreglo todavía no se encuentra completamente ordenado, ya que en la primera pasada solo se garantiza que valor mayor pase a la última posición del arreglo, en la siguiente pasada el siguiente dato mayor pasa a la penúltima posición, en la siguiente el tercer valor mayor pasa a la antepenúltima localidad del arreglo y así sucesivamente.

C5. Como el resultado de la quinta comparación es verdadero, los valores de estas posiciones se intercambian.

N_pasadas=	2
TAM_ARREGLO=	5
Posición=	1
edades[posición] > edades[posición + 1]	V
Temporal=	1.78
Edades[posición] =	1.61
Edades[posición + 1] =	1.78

edades[1]	1.78
edades[2]	1.61
edades[3]	1.84
edades[4]	1.54
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.78
edades[3]	1.84
edades[4]	1.54
edades[5]	1.87

C6. Como el resultado de la sexta comparación es falso, el arreglo continúa igual.

N_pasadas=	2
TAM_ARREGLO=	5
Posición=	2
edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[1]	1.61
edades[2]	1.78
edades[3]	1.84
edades[4]	1.54
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.78
edades[3]	1.84
edades[4]	1.54
edades[5]	1.87

C7. Como el resultado de la séptima comparación es verdadero, los valores de estas posiciones se intercambian.

N_pasadas=	2
TAM_ARREGLO=	5
Posición=	3
edades[posición] > edades[posición + 1]	V
Temporal=	1.84
Edades[posición] =	1.54
Edades[posición + 1] =	1.84

edades[1]	1.61
edades[2]	1.78
edades[3]	1.84
edades[4]	1.54
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.78
edades[3]	1.54
edades[4]	1.84
edades[5]	1.87

C8. Como el resultado de la octava comparación es falso, el arreglo continúa igual.

N_pasadas=	2
TAM_ARREGLO=	5
Posición=	4
edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[1]	1.61
edades[2]	1.78
edades[3]	1.54
edades[4]	1.84
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.78
edades[3]	1.54
edades[4]	1.84
edades[5]	1.87

Con esto terminamos la segunda pasada, pero si somos observadores notaremos que al irse colocando con cada pasada los valores mayores al fondo del arreglo, se vuelve innecesario realizar las evaluaciones correspondientes, esto produce exceso de tiempo.

Sugerencia. Para evitar esta perdida o exceso de tiempo se recomienda hacer una mejora al programa, la cual consiste en que después de cada pasada decrementar en uno la variable que contiene el tamaño del arreglo (TAM_ARREGLO).

Nota. Existe otra posible mejora, pero se menciona la final.

C9. Como el resultado de la novena comparación es falso, el arreglo continúa igual.

N_pasadas=	3
TAM_ARREGLO=	5
Posición=	1
edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[1]	1.61
edades[2]	1.78
edades[3]	1.54
edades[4]	1.84
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.78
edades[3]	1.54
edades[4]	1.84
edades[5]	1.87

C10. Como el resultado de la décima comparación es verdadero, los valores de estas posiciones se intercambian.

N_pasadas=	3
TAM_ARREGLO=	5
Posición=	2
edades[posición] > edades[posición + 1]	V
Temporal=	1.78
Edades[posición] =	1.54
Edades[posición + 1] =	1.78

edades[1]	1.61
edades[2]	1.78
edades[3]	1.54
edades[4]	1.84
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.54
edades[3]	1.78
edades[4]	1.84
edades[5]	1.87

C11. Como el resultado de la onceava comparación es falso, el arreglo continúa igual

N_pasadas=	3
TAM_ARREGLO=	5
Posición=	3
edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[1]	1.61
edades[2]	1.54
edades[3]	1.78
edades[4]	1.84
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.54
edades[3]	1.78
edades[4]	1.84
edades[5]	1.87

C12. Como el resultado de la doceava comparación es falso, el arreglo continúa igual.

N_pasadas=	3
TAM_ARREGLO=	5
Posición=	4
edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[1]	1.61
edades[2]	1.54
edades[3]	1.78
edades[4]	1.84
edades[5]	1.87

➔

edades[1]	1.61
edades[2]	1.54
edades[3]	1.78
edades[4]	1.84
edades[5]	1.87

Comenzamos la cuarta pasada.

C13. Como el resultado de la treceava comparación es verdadero, los valores de estas posiciones se intercambian

N_pasadas=	4
TAM_ARREGLO=	5
Posición=	1
edades[posición] > edades[posición + 1]	V
Temporal=	1.61
Edades[posición] =	1.54
Edades[posición + 1] =	1.61

edades[1]	1.61
edades[2]	1.54
edades[3]	1.78
edades[4]	1.84
edades[5]	1.87

➔

edades[1]	1.54
edades[2]	1.61
edades[3]	1.78
edades[4]	1.84
edades[5]	1.87

En este momento nuestro arreglo ya se encuentra ordenado, pero como el programa no lo sabe, él continua hasta que el valor que asume N_pasadas sea igual a TAM_ARREGLO.

Sugerencia. Para salirse antes de un arreglo ya ordenado, se recomienda utilizar una variable entera del tipo contador que se incrementa dentro de la condición **si entonces**, la cual con cada pasada es reiniciada en cero, pero si al término de una pasada esta sigue en cero quiere decir que no hubo intercambio de valores entre posiciones por lo cual el arreglo ya esta ordenado.

Como nos podemos dar cuenta es un método muy sencillo y de fácil manejo, pero es ineficaz para cuando se tienen arreglos de grandes dimensiones, ya que se consumen grandes cantidades de tiempo máquina. En este ejemplo al ser un arreglo de 5 posiciones se realiza 16 veces la comparación. Si fuera un arreglo de 10 posiciones se hubieran realizado 91 comparaciones. Para sacar el total de comparaciones al total de posiciones se le resta 1 y el total se eleva al cuadrado. Entonces en una empresa que probablemente utiliza arreglos de 1000 posiciones, el programa tendrá que ejecutar 999^2 = 998,001 comparaciones, aunque estas pueden disminuir con las mejoras ya comentadas. Por lo cual existen métodos de ordenación más eficaces y eficientes pero no se ven dentro de este curso.

Arreglos Bidimensionales (Matrices).

Hasta este momento solo se han utilizado arreglos de una sola dimensión (1 columna) y en cualquier lenguaje de programación se pueden crear arreglos de múltiples dimensiones, pero la más común es la de dos dimensiones, la cual significa que es un arreglo conformado por varios renglones y varias columnas.

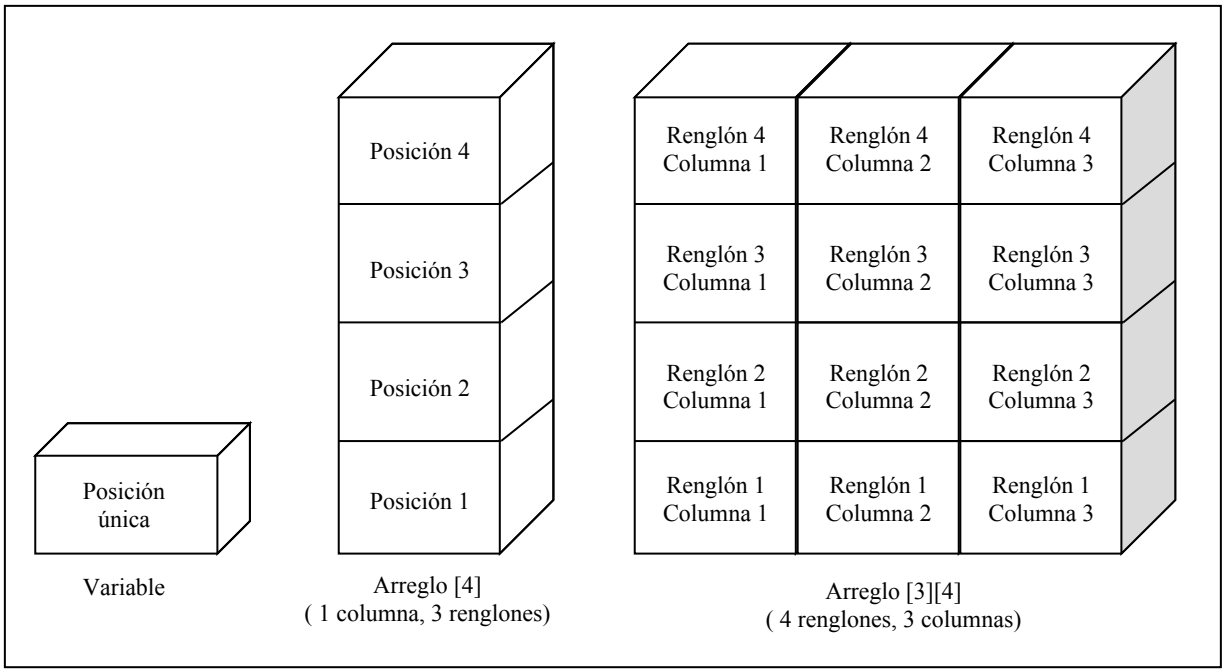


Ilustración 30 Comparación entre una variable, un arreglo unidimensional y un arreglo de dos dimensiones.

Para **declarar** una matriz, el tamaño de cada una de sus dimensiones se coloca en su propio paréntesis cuadrado o corchetes.

PSEUDOCÓDIGO Y DIAGRAMA N-S

```
Arreglos:
    Edad : entero de [ 3 ] renglones [4] columnas
    Parciales : real de [ 5 ] renglones [8] columnas
```

DIAGRAMA DE FLUJO

```
Arreglos:
    Edad : entero de [ 3 ] renglones [4] columnas
    Parciales : real de [ 5 ] renglones [8] columnas
```

Ilustración 31 Forma en que se declaran matrices

Al declarar una matriz, se le pueden **dar valores de inicio** para cada una de sus posiciones, pera lo cual después de indicar el tamaño de este se coloca un signo de igual y entre llaves "{}", los valores de cada posición separados por comas, recordando que el primer valor corresponde al renglón 1 columna 1, el segundo valor a renglón 1 columna 2 y así sucesivamente.

Sugerencia. Se recomienda que al inicializar matrices, se coloquen en una misma línea los valores para el primer renglón del arreglo en otra los del segundo y así sucesivamente, de forma que visualicemos la distribución física de nuestra matriz.

PSEUDOCÓDIGO Y DIAGRAMA N-S

```
Arreglos:
    Edad : entero de [3] renglones [4] columnas = {25, 9, 18, 20, 31
                                                   4, 13, 28, 15, 54
                                                   29, 7, 12, 11, 23}
    Parciales : real de [5] renglones [8] columnas = {0}
    // si se coloca solo un cero todas las posiciones toman este valor
```

DIAGRAMA DE FLUJO

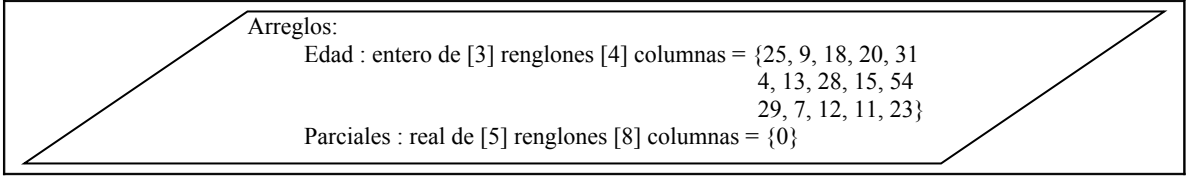


Ilustración 32 Forma en que se inicializan matrices

Para **escribir** lo que contiene una matriz debemos de indicar el nombre del arreglo y la posición del renglón entre corchetes y columna entre corchetes que deseamos visualizar.

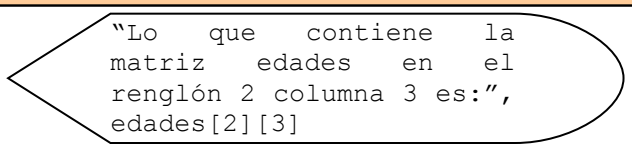
PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
Escribir "lo que contiene la matriz edades en el renglón 2 columna 3 es:", edades[2][3]	

Ilustración 33 Forma en que se despliega información desde una matriz.

Para **almacenar** un dato en una posición específica de una matriz debemos de indicar el nombre del arreglo y la posición del renglón entre corchetes y la columna entre corchetes en que deseamos guardar el dato dado por el usuario.

PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO

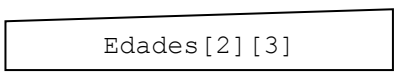
Leer edades[2][3]
// almacena una dato en el renglón 2
// columna 3 del arreglo edades
	

Ilustración 34 Forma en que se almacena información a una matriz.

Para **asignar** el resultado de una operación en una posición específica de una matriz debemos de indicar el nombre del arreglo y la posición del renglón ente corchetes y columna entre corchetes en que deseamos colocar el resultado de la expresión.

PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO

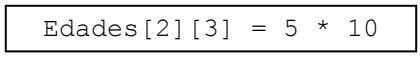
Edades[2][3] = 5 * 10
// almacena una dato en el renglón 2
// columna 3 del arreglo edades
	

Ilustración 35 Forma en que se asignan datos a una matriz.

La comparación se realiza de la misma manera en que se realizan las operaciones ya vistas: colocando el nombre y posición del renglón entre corchetes y la posición de la columna entre corchetes de la matriz que se quiere comparar, el operador relacional y el valor o variable contra quien se coteja, teniendo en cuenta que podría ser otra posición de otra matriz. No se ejemplifican debido a que se tendría que desarrollar la estructura condicional o cíclica.

A continuación realizamos el primer ejemplo de un algoritmo en el cual utilizamos una matriz de 5 renglones por cuatro columnas, donde los renglones hacen referencia a un número de alumno mediante la variable **num_alum** la cual es del tipo contador al igual que la variable **parcial** la cual es la que se encarga de apuntar a las columnas que a su vez indican el número de parcial de cada alumno, es decir que si nos encontramos con la coordenada [3][2] estamos apuntando al parcial 2 del tercer alumno. Cada variable es incrementada en una estructura cíclica **hacer para ... hasta ...**, donde la que controla el parcial esta anidada dentro de la que controla al numero de alumno.

A partir de este momento omitiremos el análisis del sistema ya lo podemos realizar mentalmente al igual que la prueba.

✍Ejemplo ❶	Se necesita de un sistema que utiliza un arreglo de 5 renglones y cuatro columnas, para almacenar los 3 parciales y su promedio de 5 alumnos.

Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: alumnos

Arreglos:
            calificaciones : real de [5] renglones [4] columnas

Variables:
            num_alum, parcial : entero = 0
            acum_cal : real = 0

 1.  Inicio

 2.  Hacer para num_alum = 1 hasta num_alum > 5

      2.1acum_cal = 0

      2.2 Hacer para parcial = 1 hasta parcial > 3

           2.2.1   Escribir "Calificación del alumno ",num_alum,"en parcial:", parcial
           2.2.2 Leer calificaciones[num_alum][parcial]
           2.2.3 acum_cal = acum_cal + calificaciones[num_alum][parcial]
           2.2.4 parcial = parcial + 1

           Fin para

      2.3calificaciones[num_alum][parcial] = acum_cal / 3
      2.4num_alum = num_alum + 1

     Fin para

 3.  Fin
```

Diagrama N-S

// diagrama N-S : Alumnos		
Inicio		
Arreglos: calificaciones : real de [5] renglones [4] columnas		
Variables: num_alum, parcial : entero = 0 acum_cal : real = 0		
	num_alum > 5	
num_alum=1	acum_cal = 0	
	parcial > 3	
parcial=1	Escribir "Calificación del alumno ",num_alum,"en parcial:", parcial	
	Leer calificaciones[num_alum][parcial]	
	acum_cal = acum_cal + calificaciones[num_alum][parcial]	
	parcial = parcial + 1	
	calificaciones[num_alum][parcial] = acum_cal / 3	
	num_alum = num_alum + 1	
Fin		

DIAGRAMA DE FLUJO

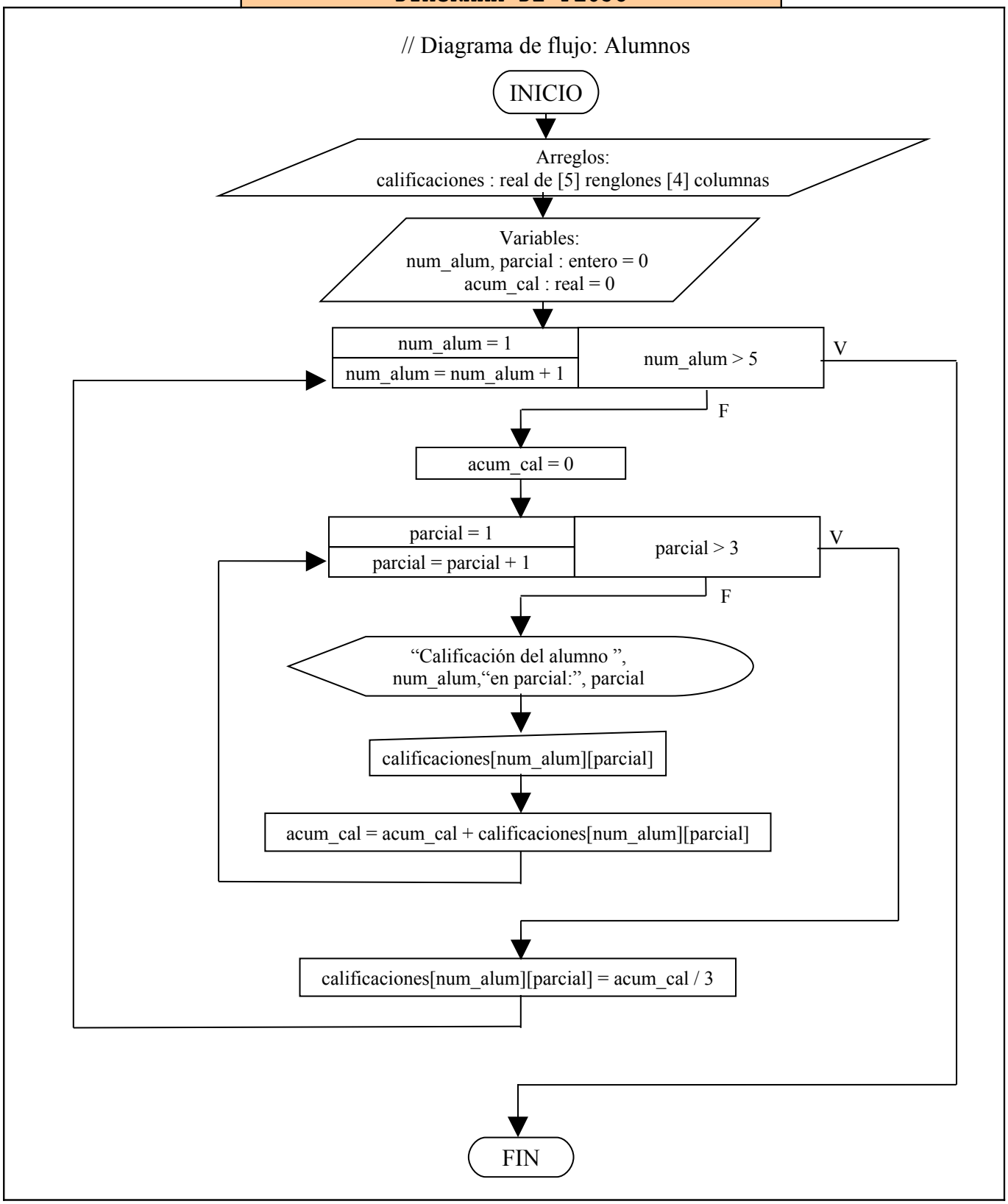


Tabla 39 Ejemplo 1 de arreglos bidimensionales

✍Ejemplo ❷	Se necesita un sistema que utiliza una matriz de 10 renglones y 3 columnas. En las dos primeras columnas se colocan los promedios de los 10 alumnos de dos grupos (A y B) y en la tercera columna se almacenará el promedio más alto de cada posición.

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: alumnos de 2 grupos

Arreglos:
            grupos : real de [10] renglones [3] columnas

Variables:
            alumno : entero = 0

 1.  Inicio

 2.  Hacer para alumno = 1 hasta alumno > 10

       2.1 Escribir "Promedio del alumno ",alumno," del primer grupo:"
       2.2 Leer grupos[alumno][1]
       2.3 Escribir "Promedio del alumno ",alumno," del segundo grupo:"
       2.4 Leer grupos[alumno][2]

       2.5Si grupos[alumno][1] > grupos[alumno][2] entonces

                2.5.1   grupos[alumno][3] = grupos[alumno][1]

             Si no

                2.5.2   grupos[alumno][3] = grupos[alumno][2]

             Fin si

       2.6 alumno = alumno + 1

     Fin para

 3.  Fin
```

Diagrama N-S

// diagrama N-S : alumnos de 2 grupos		
Inicio		
Arreglos: grupos : real de [10] renglones [3] columnas		
Variables: alumno : entero = 0		
alumno > 10		
alumno = 1	Escribir "Promedio del alumno ",alumno," del primer grupo:"	
	Leer grupos[alumno][1]	
	Escribir "Promedio del alumno ",alumno," del segundo grupo:"	
	Leer grupos[alumno][2]	
	grupos[alumno][1] > grupos[alumno][2] verdadero	falso
	grupos[alumno][3] = grupos[alumno][1]	grupos[alumno][3] = grupos[alumno][2]
alumno = alumno + 1		
Fin		

DIAGRAMA DE FLUJO

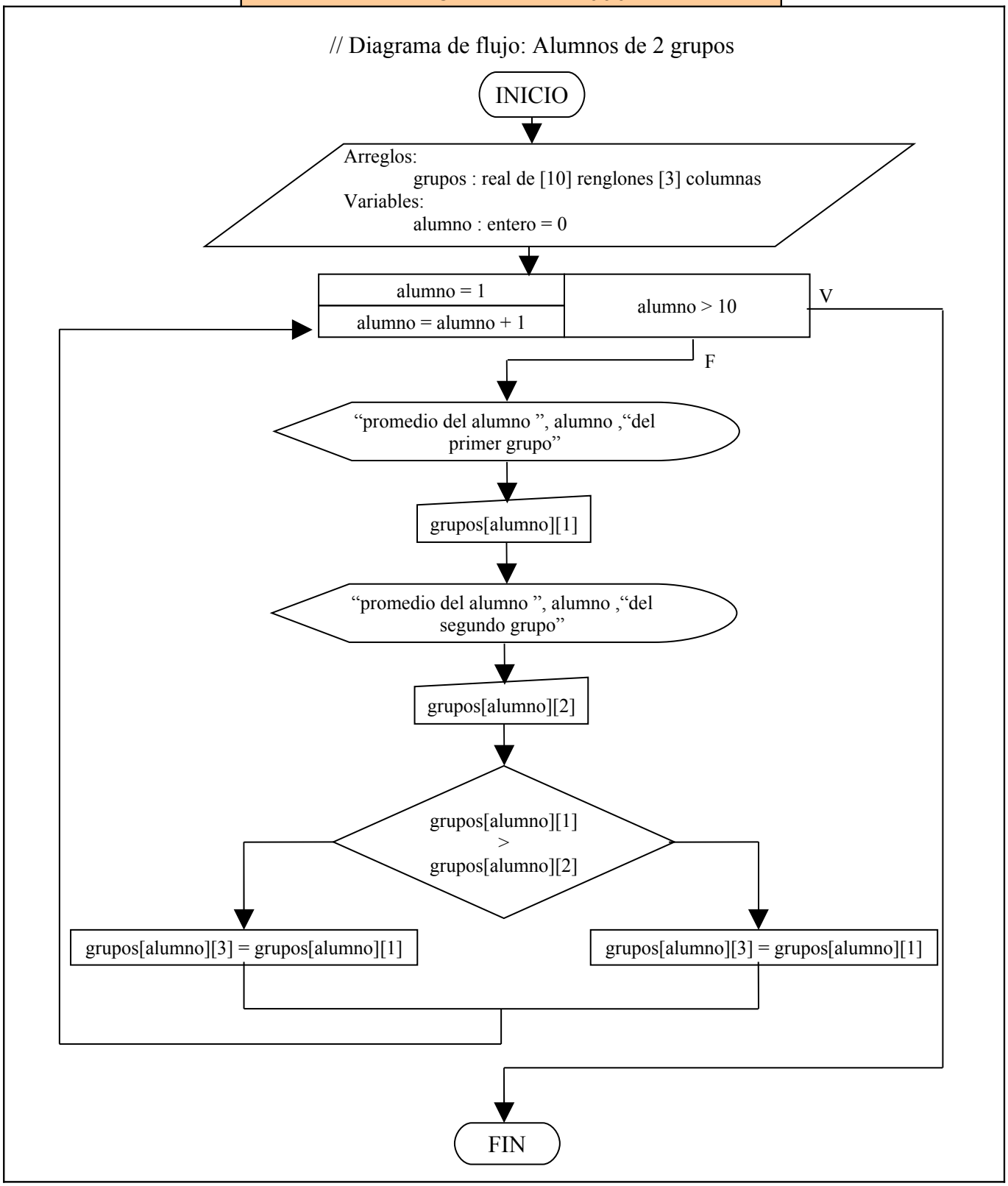


Tabla 40 Ejemplo 2 del manejo de arreglos

Ejercicios.

I. Realiza un algoritmo con las tres diferentes técnicas utilizando matrices para cada uno de los siguientes problemas.
1. Sistema que almacena la estatura, peso y talla de zapatos de hasta 100 personas, preguntando si se desea almacenar los datos de otra persona.
2. Sistema que tiene cuatro opciones: suma, resta, multiplicación y salir, en el cual según la opción que se seleccione muestra las tablas correspondientes o sale del sistema.
3. Sistema que permite almacenar, consultar y modificar el nombre, dirección y teléfono de hasta 10 personas.
4. Sistema que captura y posteriormente ordena alfabéticamente los datos de 10 personas ya sea por nombre, apellido paterno o apellido materno
5. Sistema que almacena los tres parciales y promedios de 10 alumnos, de las cuales necesitamos saber cuantos sacaron de promedio menos de 6, cuantos entre 6 y 8, cuantos entre 8 y 9 y cuantos más de 9 ; además que despliegue los parciales de todos aquellos que tienen promedio de 9 o más.

5.2 ESTRUCTURAS

Este subtema en realidad es muy sencillo, ya que nosotros ya utilizamos estructuras desde los primeros temas:

- **Una variable**, es en realidad la estructura más sencilla a manejar, la cual consiste en almacenar solo un dato de un tipo especifico.
- **Un arreglo**, es una estructura la cual almacena n datos del mismo tipo. Cuando declaramos un arreglo del tipo entero de 3 posiciones llamado arreglo1 (arreglo1[3] : entero), en realidad estamos creando una estructura de tres variables enteras (arreglo1[0], arreglo1[1] y arreglo1[2]).

Pues una estructura no es muy diferente a un arreglo, ya que una estructura es un conjunto n variables, las cuales pueden ser de diferentes tipos.

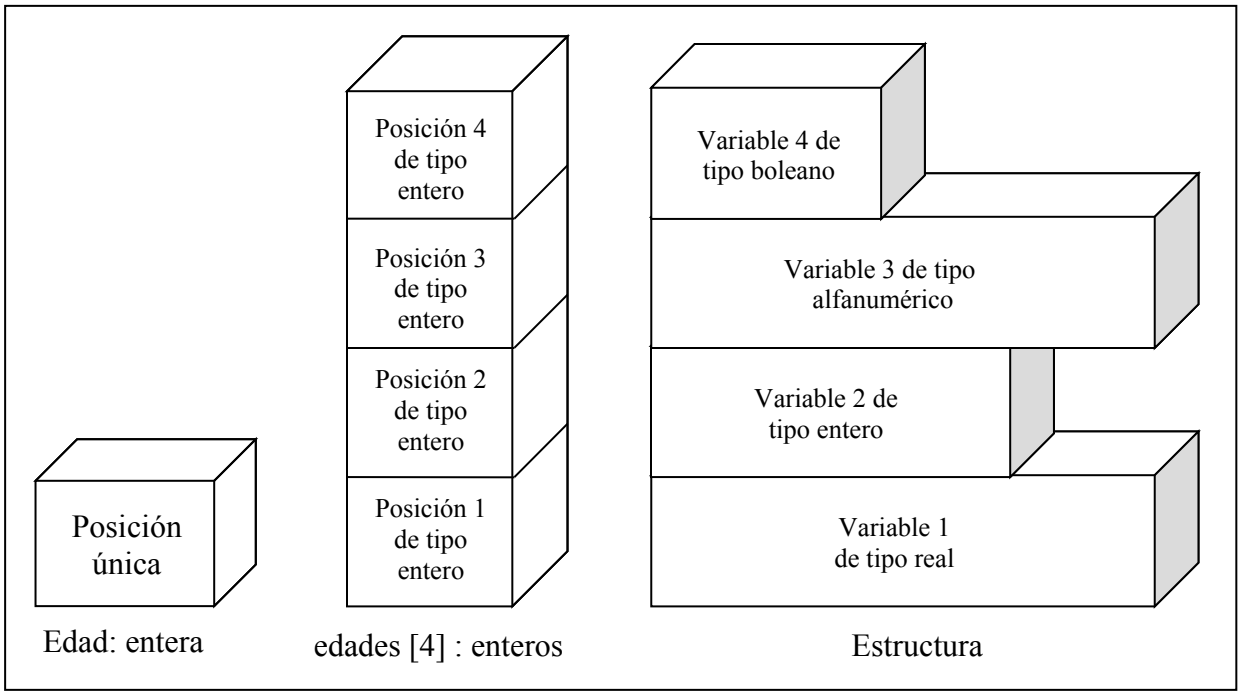


Ilustración 36 Diferencia entre variables, arreglos y estructuras.

Hablando en términos de base datos, una estructura se asemeja a un registro, el cual es un conjunto de campos relacionados entre si.

En un algoritmo antes de utilizar a una estructura para realizarle cualquier operación, se deben de indicar las variables que contendrá la estructura y el nombre de esta. Este proceso se conoce como **definición de la estructura** y se realiza en la sección estructuras.

PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
Estructuras: **Alumno** con los campos: Nombre : alfanumérico N_control : entero Semestre : entero Grupo : alfanumérico Promedio_final : real	Estructuras: **Alumno** con los campos: Nombre : alfanumérico N_control : entero Semestre : entero Grupo : alfanumérico Promedio_final : real

Ilustración 37 Forma en que se define una estructura.

Las operaciones que se pueden realizar sobre las estructuras son exactamente las mismas que a las variables: declaración, desplegar, almacenar, asignar, inicializar y comparar.

La **declaración** no desvaría mucho de la declaración de variables incluso se realiza en la sección de variables, ya que hay que colocar el nombre de la estructura y el tipo de dato, con la excepción de que el tipo de dato ya no es entero, real, alfanumérico o boleano, sino que debe ser el nombre de una estructura ya definida; ya que estamos declarando una variable con un nombre específico que debe tener los campos que se definieron en la estructura. A esto es a lo que llamamos: **declarar una variable del tipo estructura predefinida**.

PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO

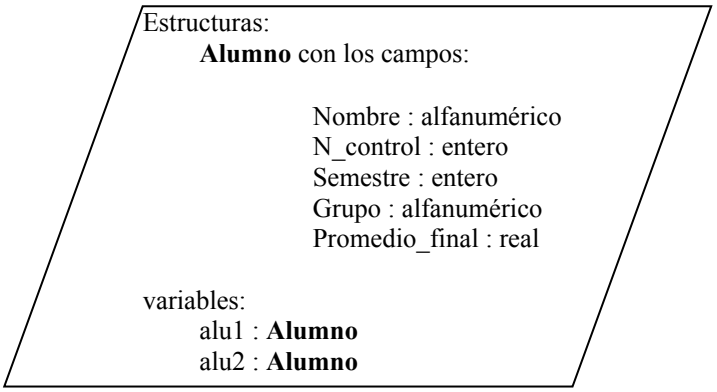
Estructuras:
Alumno con los campos:
 	
Nombre : alfanumérico
N_control : entero
Semestre : entero
Grupo : alfanumérico
Promedio_final : real
 	
variables:
alu1 : **Alumno**
alu2 : **Alumno**
 	
// se están declarando dos variables del
// tipo Alumno
	

Ilustración 38 declaración de variables del tipo de una estructura predefinida

Al declarar una variable del tipo de una estructura predefinida, se le pueden **dar valores de inicio** para cada una de sus variables internas, para lo cual después de declararla se coloca un signo de igual y entre llaves "{}", los valores de cada campo separados por comas, los cuales se introducen en el orden en que esta definida la estructura.

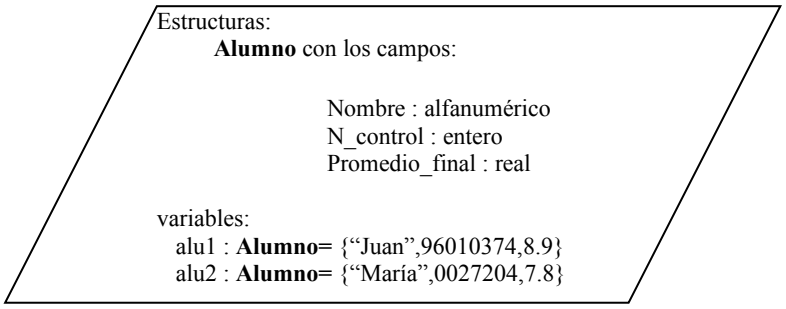
PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
Estructuras: **Alumno** con los campos: Nombre : alfanumérico N_control : entero Promedio_final : real variables: alu1 : **Alumno** = {“Juan”,96010374,8.9} alu2 : **Alumno** = {“María”,0027204,7.8}	

Ilustración 39 Inicialización de variables del tipo estructura predefinida

Para **escribir** lo que contiene una variable de un tipo de estructura predefinido en un campo específico, debemos de indicar el nombre de la variable, colocar un punto “.” y el campo que deseamos visualizar.

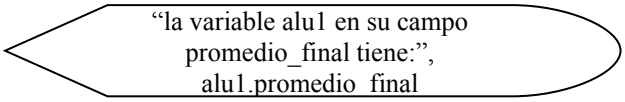
PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
Escribir “la variable alu1 en su campo promedio_final tiene:”, alu1.promedio_final	

Ilustración 40 Desplegar contenido de un campo de una variable del tipo estructura.

Para **almacenar** algo dentro de una variable de un tipo de estructura predefinido en un campo específico, debemos de indicar el nombre de la variable, colocar un punto “.” y el campo que deseamos visualizar.

PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
Leer alu1.promedio_final // almacena un dato en el campo promedio_final de alu1	

Ilustración 41 Almacenar información en campo de una variable del tipo estructura.

Para **asignar** el resultado de una operación dentro de una variable de un tipo de estructura predefinido en un campo específico, debemos de indicar el nombre de la variable, colocar un punto “.” y el campo que deseamos visualizar.

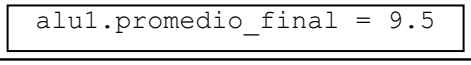
PSEUDOCÓDIGO Y DIAGRAMA N-S	DIAGRAMA DE FLUJO
alu1.promedio_final = 9.5 // almacena un dato en el campo promedio_final alu1	

Ilustración 42 Asignar datos en campo de una variable del tipo estructura.

A continuación realizamos un primer ejercicio que utiliza estructuras, dentro de la definición de la estructura se declara un arreglo de 4 posiciones para guardar las calificaciones del alumno.

✍Ejemplo ❶	Se necesita un sistema que captura el nombre, numero de control, calificación de primer parcial, calificación de segundo parcial, calificación de tercer parcial y promedio final de 2 alumnos, el cual nos debe de decir que alumno salio con respecto a su promedio final.

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

Pseudocódigo: Comparar alumnos
Estructuras:

alumno con los campos:

nombre : alfanumérico
num_control : entero

Arreglos:
parciales : real de [4] posiciones
// el arreglo parciales esta dentro de la estructura alumno

Variables:
alumno1 : alumno
alumno2 : alumno

1. Inicio
2. Escribir “Dame el nombre del primer alumno:”
3. Leer alumno1.nombre
4. Escribir “Dame el número de control del primer alumno:”
5. Leer alumno1.num_control
6. Escribir “Dame calificación de 1er. parcial del primer alumno:”
7. Leer alumno1.parciales[1]
8. Escribir “Dame calificación de 2do. parcial del primer alumno:”
9. Leer alumno1.parciales[2]
10. Escribir “Dame calificación de 3er. parcial del primer alumno:”
11. Leer alumno1.parciales[3]
12. alumno1.parciales[4] = (alumno1.parciales[1] + alumno1.parciales[2] + alumno1.parciales[3]) / 3
13. Escribir “Dame el nombre del segundo alumno:”
14. Leer alumno2.nombre
15. Escribir “Dame el número de control del segundo alumno:”
16. Leer alumno2.num_control
17. Escribir “Dame calificación de 1er. parcial del segundo alumno:”
18. Leer alumno2.parciales[1]
19. Escribir “Dame calificación de 2do. parcial del segundo alumno:”
20. Leer alumno2.parciales[2]
21. Escribir “Dame calificación de 3er. parcial del segundo alumno:”
22. Leer alumno2.parciales[3]
23. alumno2.parciales[4] = (alumno2.parciales[1] + alumno2.parciales[2] + alumno2.parciales[3]) / 3
24. Si alumno1.parciales[4] > alumno2.parciales[4] entonces
    24.1 Escribir alumno1.nombre, “ Salio mejor en promedio final que “,alumno2.nombre
    Si no
    24.2 Escribir alumno2.nombre, “ Salio mejor en promedio final que “,alumno1.nombre
    Fin si
25. Fin

DIAGRAMA DE FLUJO

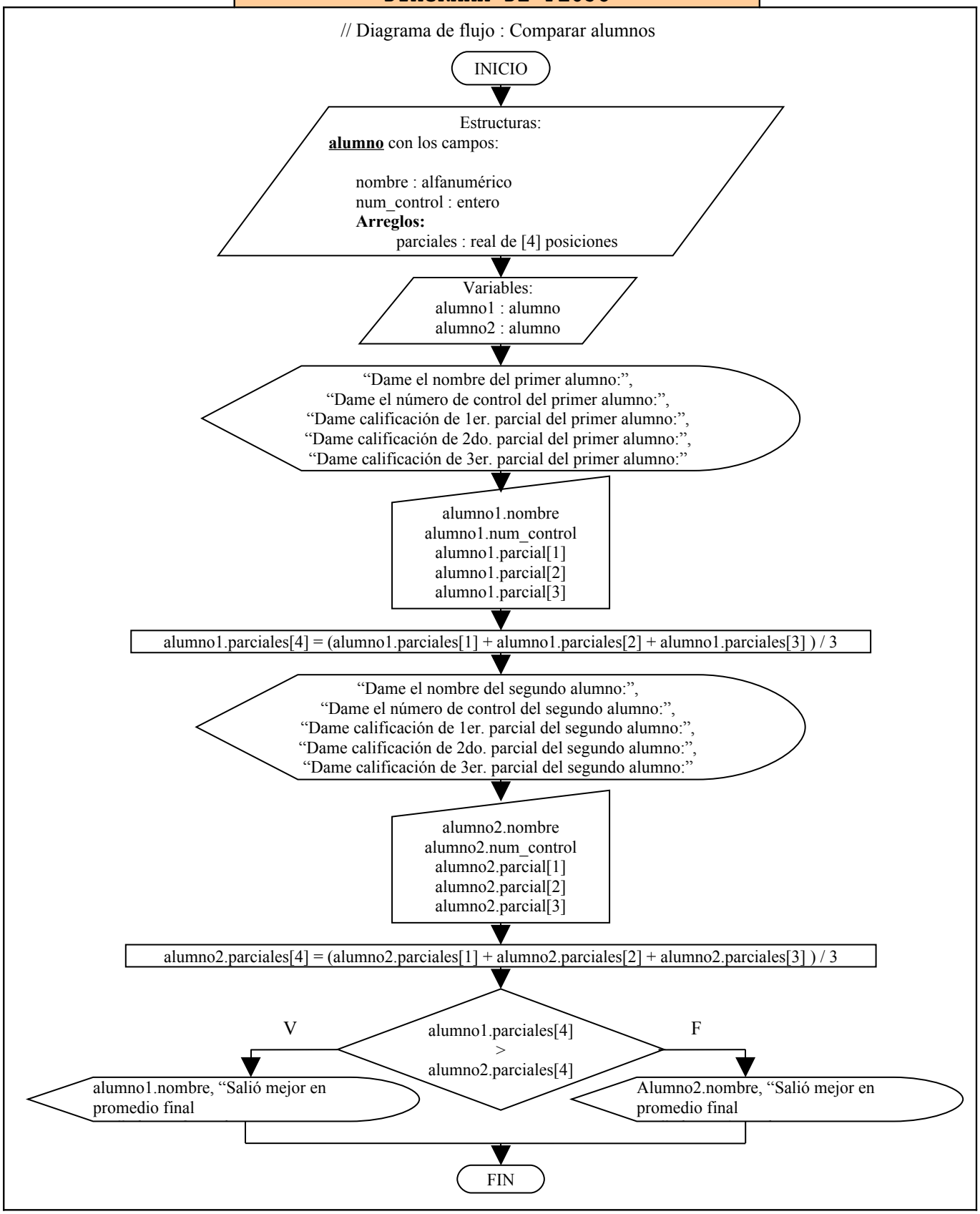


Diagrama N-S

// diagrama N-S : Comparar alumnos
Inicio
Estructuras: **alumno** con los campos: nombre : alfanumérico num_control : entero **Arreglos:** parciales : real de [4] posiciones
Variables: alumno1 : alumno alumno2 : alumno
Escribir “Dame el nombre del primer alumno:”
Leer alumno1.nombre
Escribir “Dame el número de control del primer alumno:”
Leer alumno1.num_control
Escribir “Dame calificación de 1er. parcial del primer alumno:”
Leer alumno1.parciales[1]
Escribir “Dame calificación de 2do. parcial del primer alumno:”
Leer alumno1.parciales[2]
Escribir “Dame calificación de 3er. parcial del primer alumno:”
Leer alumno1.parciales[3]
alumno1.parciales[4] = (alumno1.parciales[1] + alumno1.parciales[2] + alumno1.parciales[3]) / 3
Escribir “Dame el nombre del segundo alumno:”
Leer alumno2.nombre
Escribir “Dame el número de control del segundo alumno:”
Leer alumno2.num_control
Escribir “Dame calificación de 1er. parcial del segundo alumno:”
Leer alumno2.parciales[1]
Escribir “Dame calificación de 2do. parcial del segundo alumno:”
Leer alumno2.parciales[2]
Escribir “Dame calificación de 3er. parcial del segundo alumno:”
Leer alumno2.parciales[3]
alumno2.parciales[4] = (alumno2.parciales[1] + alumno2.parciales[2] + alumno2.parciales[3]) / 3
alumno1.parciales[4] > alumno2.parciales[4]

Verdadero	Falso
Escribir alumno1.nombre, “ Salio mejor en promedio final que “,alumno2.nombre	Escribir alumno2.nombre, “ Salio mejor en promedio final que “,alumno1.nombre

Fin

Tabla 41 Ejemplo 1 del manejo de estructuras

En este primer ejemplo nos damos cuenta que para introducir un dato dentro de un campo de una estructura el cual es a su vez parte de un arreglo, se tiene que indicar en el lado del campo la posición en la que se debe de guardar el dato. Pero ahora veremos como introducir datos a un arreglo del tipo estructura predefinida.

Ejemplo ❷	Se necesita un sistema que almacena la clave, descripción, precio de compra, precio de menudeo y precio de mayoreo de 10 productos.

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo: Supermercado
Estructuras:

                producto con los campos:
                        clave : entera
                        descripción : alfanumérico
                        Arreglos :
                                precios : real, de [3] posiciones
Arreglos:
        productos : producto, de [10] posiciones
Variables:
        índice : entera

 1.  Inicio

 2.  hacer para índice = 1 hasta índice > 10

       2.1  Escribir "Clave del producto ", índice, ":"
       2.2  Leer productos[índice].clave
       2.3  Escribir "Descripción del producto ", índice, ":"
       2.4  Leer productos[índice].descripción
       2.5  Escribir "Precio de compra del producto ", índice, ":"
       2.6  Leer productos[índice].precios[1]
       2.7  Escribir "Precio de menudeo del producto ", índice, ":"
       2.8  Leer productos[índice].precios[2]
       2.9  Escribir "Precio de mayoreo del producto ", índice, ":"
       2.10 Leer productos[índice].precios[3]
       2.11 índice = índice + 1

     Fin Para

 3.  Fin
```

Diagrama N-S

// diagrama N-S : Supermercado	
Inicio	
Estructuras: **producto** con los campos: clave : entera descripción : alfanumérico *Arreglos :* precios : real, de [3] posiciones	
Arreglos: productos : **producto**, de [10] posiciones	
Variables: índice : entera	
índice = 1	índice > 10
	Escribir "Clave del producto ", índice, ":"
	Leer productos[índice].clave
	Escribir "Descripción del producto ", índice, ":"
	Leer productos[índice].descripción
	Escribir "Precio de compra del producto ", índice, ":"
	Leer productos[índice].precios[1]
	Escribir "Precio de menudeo del producto ", índice, ":"
	Leer productos[índice].precios[2]
	Escribir "Precio de mayoreo del producto ", índice, ":"
	Leer productos[índice].precios[3]
	índice = índice + 1
Fin	

DIAGRAMA DE FLUJO

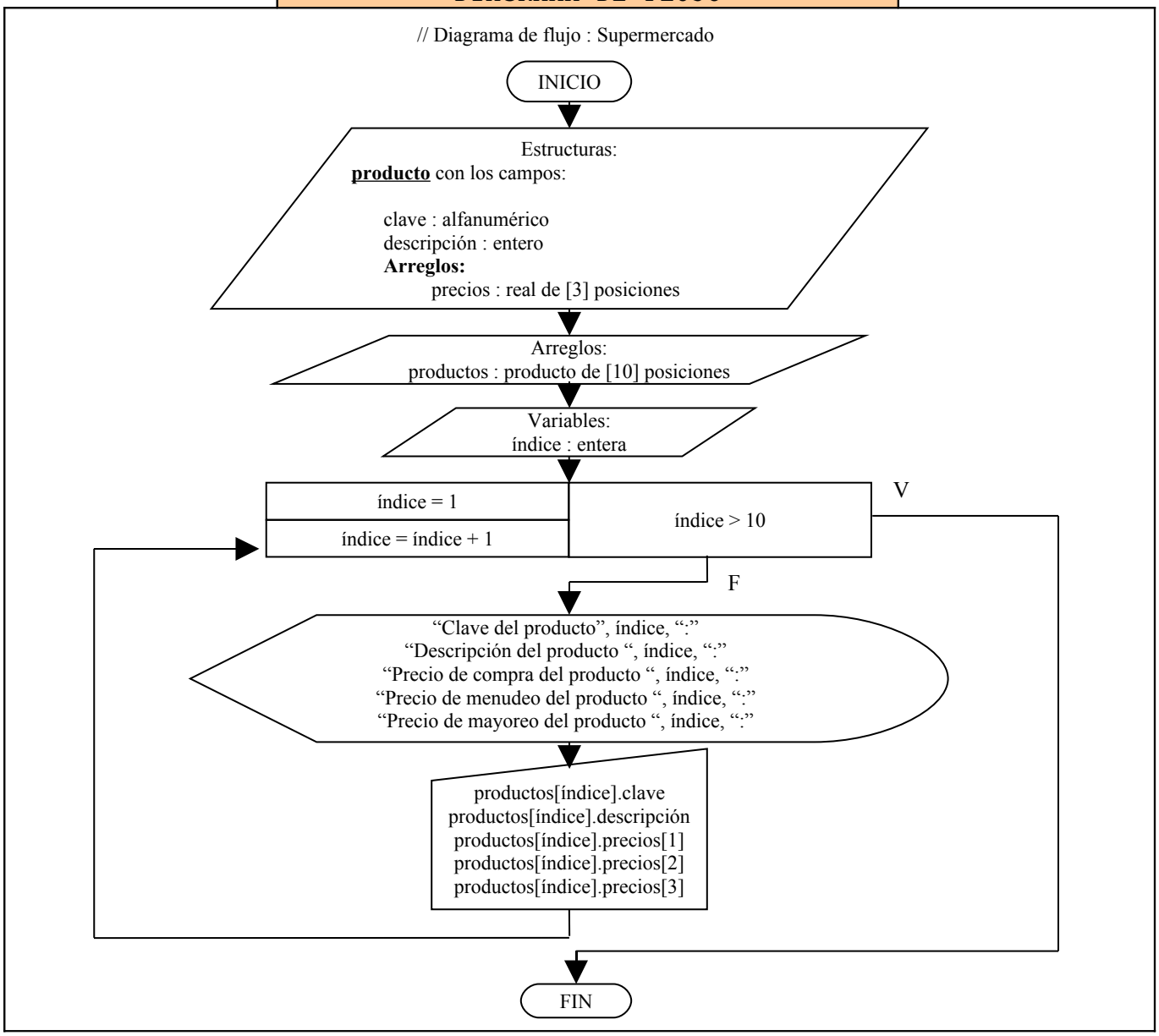


Tabla 42 Ejemplo 2 de manejo de estructuras

Ejercicios.

I. Realiza un algoritmo que utilice estructuras mediante las tres técnicas para cada uno de los puntos siguientes.
1. Se necesita un sistema para una escuela el cual almacene el nombre, dirección, teléfono, semestre, grupo y matricula de 100 alumnos.
2. Se necesita un sistema para una escuela el cual permite almacenar, borrar, buscar y ordenar hasta un máximo de 100 alumnos con los mismos datos del ejemplo anterior.

CONCLUSIÓN

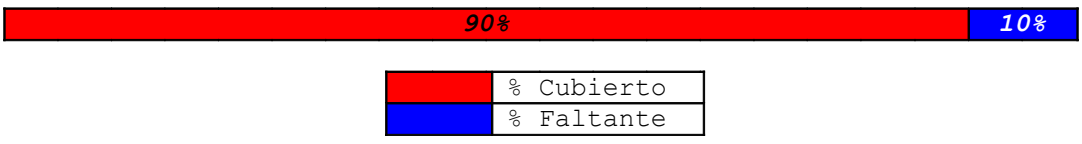
En este tema se presentaron 2 subtemas, donde cada uno tiene una gran importancia para tener una mejor mentalidad de programador, ya que ambos están enfocados a un mejor almacenamiento de datos.

El primero, abordó a los arreglos los cuales nos evitan la necesidad de declarar bastantes variables del mismo tipo, ya que solo bastará con declarar una sola con varias posiciones, donde cada posición puede almacenar un dato del tipo del arreglo. Vimos que a cada posición se puede acceder por medio de un índice, el cual debe de colocarse dentro de unos corchetes después del nombre del arreglo. También se estudió que los arreglos pueden ser de varias dimensiones, donde los más comunes son los de una y dos, a los primeros se les llama vectores y a los segundos matrices.

En el segundo se estudió a las estructuras, las cuales a diferencia de los arreglo se utilizan para almacenar datos de diferentes tipos. Vimos que las estructuras se deben de declarar antes de crear variables de este tipo predefinido. Aprendimos que para almacenar valores dentro de una variable del tipo estructura, debemos de indicar el nombre de la variable, colocar un punto y el nombre de la variable interna en la que se desea guardar el dato.

Con este tema hemos cubierto casi todo nuestro curso, por lo cual si se ha logrado la comprensión hasta este punto podemos decir que estamos a un 90% de lograr el objetivo general.

OBJETIVO DEL CURS0



TEMA VI. MÓDULOS

VI. MANEJO DE MÓDULOS

OBJETIVO	Al terminar el tema el participante mediante la elaboración de ejercicios, diseñará módulos para fragmentar sus algoritmos, con la finalidad de que estos sean más fáciles de desarrollar.

CONTENIDO

Introducción

Un excelente programador se distingue de los otros cuando fragmenta sus programas en otros más pequeños, lo cual provoca que esto sean más sencillos, cortos y funcionales.

En este tema se adquirirán los conocimientos para crear módulos, las cuales como se verá, no son otra cosa más que pequeños programas o algoritmos que se escriben fuera del programa o algoritmo principal. Veremos que la creación de un módulo lleva 2 pasos; la definición del módulo y la llamada del módulo; donde la definición no es otra cosa más que desarrollar los pasos a realizar cuando el módulo se ejecute, y la llamada del módulo es mandar a traerlo para que se reproduzca. Además haremos que nuestros módulos puedan o no recibir ciertos datos desde el algoritmo principal que le ayuden a realizar sus tareas y que al término de su ejecución devuelvan o no un dato al programa principal.

Con este tema el cual es muy fácil se concluye este curso así que: “animo falta poco”.

6.1 Módulos

La programación modular es el proceso que consiste en dividir un gran programa en varios más pequeños, los cuales realizan una tarea específica, siendo estos más fáciles de depurar y de mantener.

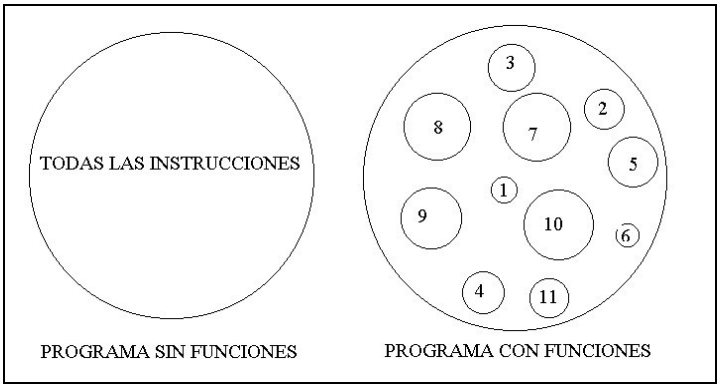


Ilustración 43 Ejemplo de lo que significa modular

Un módulo, función o procedimiento es un programa **reutilizable**, el cual realiza una tarea(s) específica(s).

Para manejar un módulo hay que realizar 2 procesos:

❶ **Definición del módulo**. Es el proceso de escribir todas las instrucciones que se van a ejecutar cuando se haga una llamada a este, incluyendo los datos que puede recibir desde el programa principal y el dato que puede devolver al algoritmo principal.

Para definir una función en pseudocódigo y en diagramas N-S, esta se debe de realizar en la sección con este nombre o en otra hoja independiente, dentro de la cual se coloca el nombre de la función, las variables, constantes, arreglos y estructuras que vaya a manejar, y el conjunto de instrucciones a ejecutar comenzando con la instrucción **Inicio Módulo** y terminando con la instrucción **Fin Modulo**.

Para definir un procedimiento en diagrama de flujo, se debe de crear un diagrama de flujo independiente con el nombre del módulo, en el cual se ilustran todas las instrucciones y datos a utilizar por la función, donde la primera instrucción es **Inicio Módulo** y la última **Fin Módulo**.

❷ **Llamada al módulo**. Es el proceso de mandar a traer al procedimiento para que ejecute su(s) tarea(s).

Este proceso se realiza en pseudocódigo y en diagramas N-S escribiendo la instrucción **Ejecutar**, seguida del nombre del módulo, en caso de que la función requiera ciertos datos desde el programa principal estos se deben de colocar entre paréntesis separados por comas, y si el procedimiento devuelve un dato al algoritmo principal cuando termina de ejecutarse se debe de colocar una flecha (→) apuntando a la variable que recibirá el dato.

En diagrama de flujo, este proceso se realiza utilizando el símbolo **llamada a función o módulo**, el cual es un rectángulo dividido en tres secciones: en la del centro se coloca el nombre del módulo que se quiere ejecutar, en la de la derecha se escriben los datos que requiere la función se pasen desde el programa principal, y en el de la izquierda se coloca la variable que recibirá el dato que devuelve el procedimiento al terminar de ejecutarse.

A continuación veremos un primer ejemplo que utiliza módulos, en el cual veremos la propiedad de ser **reutilizable** con lo cual nos damos cuenta de que nuestro algoritmo principal no es tan extenso ya que las instrucciones que se tendrían que repetir varias veces en un programa sin funciones solo se tienen que escribir una vez dentro del procedimiento y estas se ejecutan cuantas veces se desee.

✍**Ejemplo ❶**	Se necesita un sistema que despliegue el cuadrado de un número dado por el usuario.

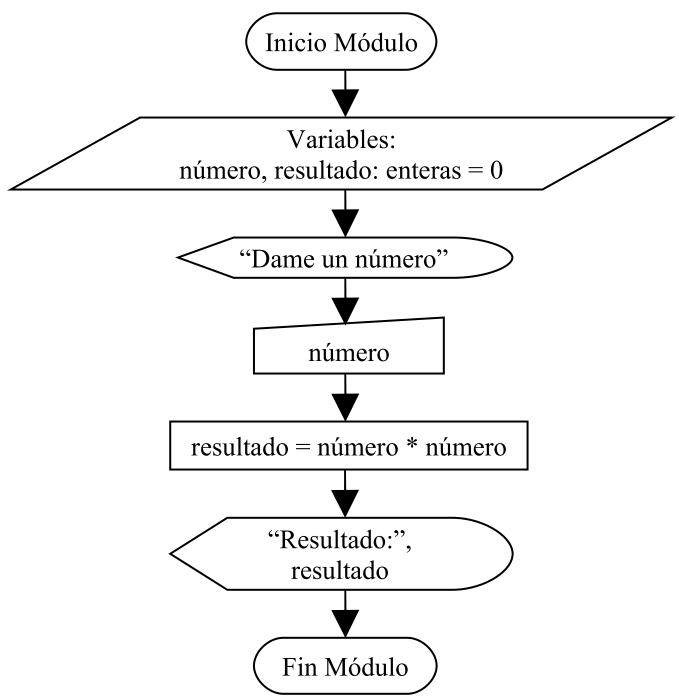
Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo del módulo: CUADRADO

        Variables:
                  número, resultado: enteras
        // estas variables solo pueden ser utilizadas por el módulo o función

     1. Inicio módulo
     2. Escribir "Dame un número"
     3. Leer número
     4. resultado = número * número
     5. Escribir "Resultado :", resultado
     6. Fin módulo

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o

Pseudocódigo: Cuadrado de varios números

Variables: // sin variables

 1.  Inicio
 2.  Escribir "primera llamada al módulo CUADRADO"
 3.  Ejecutar CUADRADO
 4.  Escribir "segunda llamada al módulo CUADRADO"
 5.  Ejecutar CUADRADO
 6.  Escribir "tercera llamada al módulo CUADRADO"
 7.  Ejecutar CUADRADO
 8.  Fin
```

DIAGRAMA DE FLUJO

//Diagrama de flujo del módulo: **CUADRADO**



//Diagrama de flujo: Cuadrado de varios Números

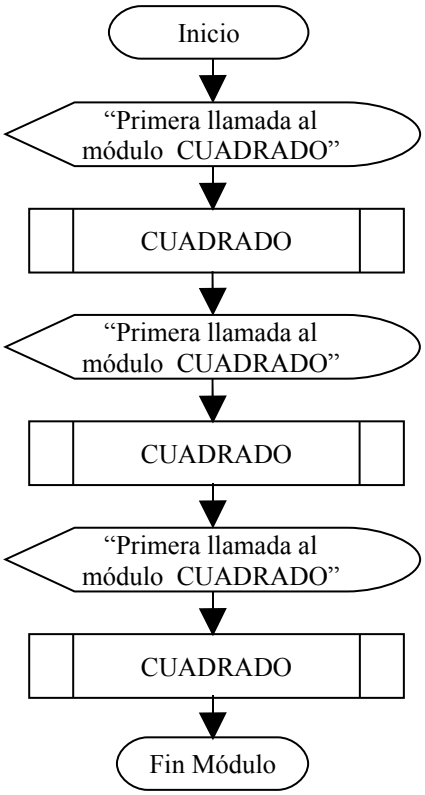


Diagrama N-S

```
// diagrama N-S del módulo: CUADRADO
Inicio Módulo
Variables:
          número, resultado: enteras
Escribir "Dame un número"
Leer número
resultado = número * número
Escribir "Resultado :", resultado
Fin Módulo
```

```
// Diagrama N-S: Cuadrado de varios números
Inicio
Escribir "primera llamada al módulo CUADRADO"
Ejecutar CUADRADO
Escribir "segunda llamada al módulo CUADRADO"
Ejecutar CUADRADO
Escribir "tercera llamada al módulo CUADRADO"
Ejecutar CUADRADO
Fin
```

Tabla 43 Ejemplo 1 del manejo de Módulos o funciones

✍Ejemplo ❷	Se necesita un sistema que llame a un módulo llamado **ÁREA** el cual calcula el área de un rectángulo y para hacerlo debe de recibir dos datos desde el algoritmo principal, uno es la base y otro la altura.

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo del Módulo: ÁREA

Parámetros:
             base : entero
             altura : entero
// los datos que son recibidos cuando es efectuada la llamada se llaman parámetros

Variables:
          resultado : entero

    1. Inicio Módulo
    2. resultado = base * altura
    3. Escribir "Área del rectángulo :", resultado
    4. Fin Módulo

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

Pseudocódigo: Área de varios rectángulos

Variables:
            dato1, dato2 : enteros
            resp : alfanumérico = "n"

   1. Inicio

   2. Repetir

              2.1Escribir "Dame base del rectángulo:"
              2.2Leer dato1
              2.3Escribir "Dame altura del rectángulo:"
              2.4Leer dato2
              2.5Ejecutar ÁREA(dato1, dato2)

// Se ejecuta el módulo área y el valor de dato1 es asignado al parámetro base y
// el valor de dato2 es asignado al parámetro altura.

              2.6Escribir "deseas calcular otra área?:"
              2.7Leer resp

      Hasta resp == "n"

   3. Fin
```

Diagrama N-S

// diagrama N-S del Módulo: **ÁREA**
Inicio Módulo
Parámetros: base : entero altura : entero
Variables: resultado : entero
resultado = base * altura
Escribir "Área del rectángulo :", resultado
Fin Módulo

```
// Diagrama N-S: Área de varios rectángulos
Inicio
Variables:
            dato1, dato2 : enteros
            resp : alfanumérico = "n"
    Escribir "Dame base del rectángulo:"
    Leer dato1
    Escribir "Dame altura del rectángulo:"
    Leer dato2
    Ejecutar ÁREA(dato1, dato2)
    Escribir "deseas calcular otra área?:"
    Leer resp
                              resp == "n"
Fin
```

DIAGRAMA DE FLUJO

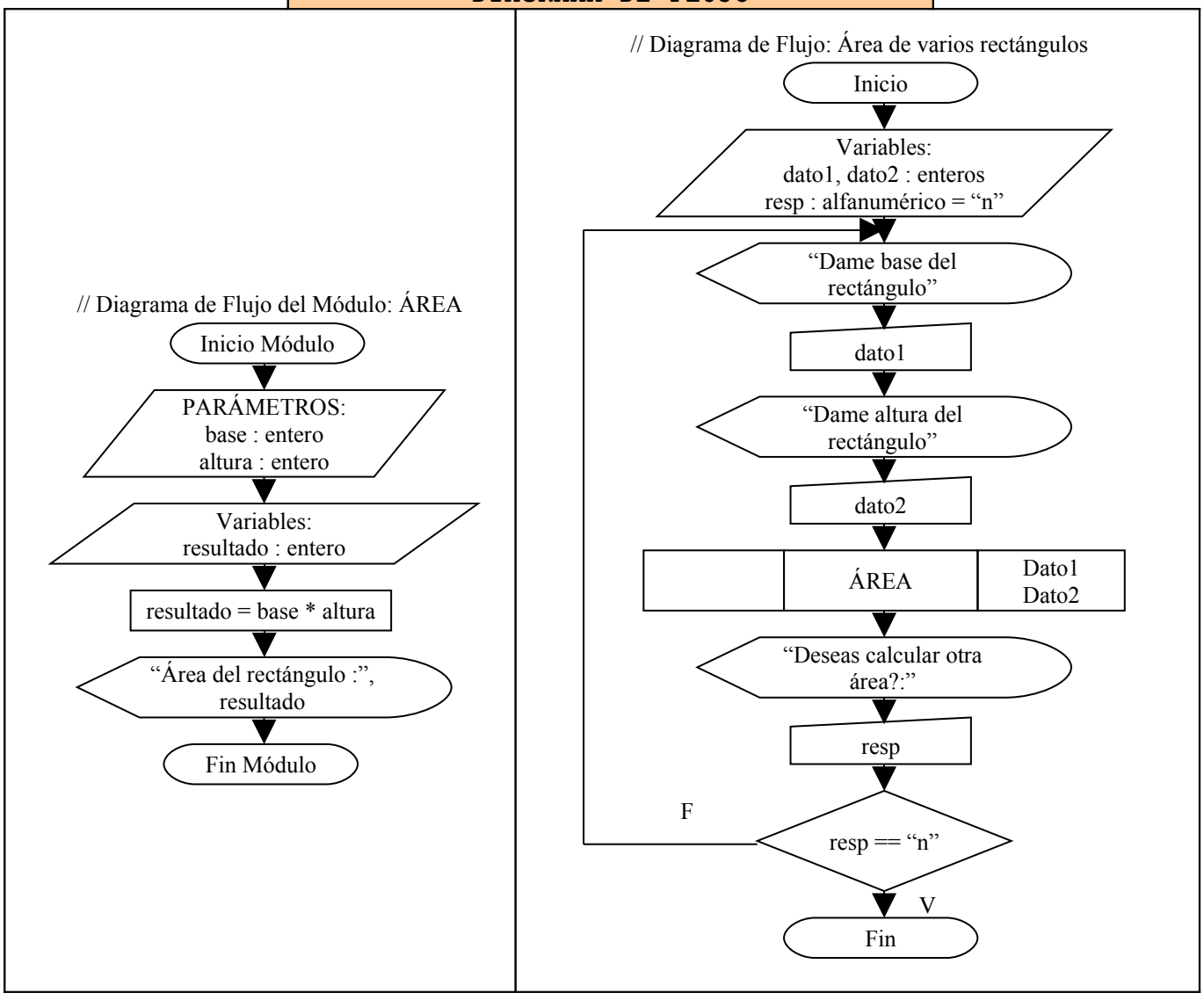


Tabla 44 Ejemplo 2 del manejo de Módulos o Funciones

✍Ejemplo ❸	Se necesita un sistema que mande a llamar a un módulo llamado PRECIO_NETO, el cual para funcionar correctamente debe de recibir desde el algoritmo principal el precio y el descuento deseado; al terminar de ejecutarse se debe regresar el nuevo precio al algoritmo principal.

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
Pseudocódigo del módulo: PRECIO_NETO

Parámetros:
            precio : real
            descuento : real

Dato a devolver:
            nuevo_precio : real

     1. Inicio Módulo
     2. nuevo_precio = precio – (precio * descuento)
     3. Fin Módulo
// Al terminar de ejecutarse el módulo se regresa el valor que tiene la variable
// nuevo_precio al algoritmo principal

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

Pseudocódigo: Tienda
Variables:
            p_neto, prec, desc : reales
            resp : alfanumerico = "s"

1.  Inicio

2. Hacer mientras resp == "s"

        2.1 Escribir "Precio del producto:"
        2.2 Leer prec
        2.3 Escribir "Descuento a realizar:"
        2.4 Leer desc
        2.5 Ejecutar PRECIO_NETO(prec, desc) → p_neto

// El dato que devuelve la función o módulo es asignado a la variable p_neto, la
// cual puede ser utilizada dentro del algoritmo principal

        2.6 Escribir "El precio neto del producto es:", p_neto
        2.7 Escribir "Deseas calcular otro producto?:"
        2.8 Leer resp

     Fin mientras

3. Fin
```

Diagrama N-S

// diagrama N-S para el Módulo : PRECIO_NETO
Inicio Módulo
Parámetros: precio : real descuento : real
Dato a devolver: nuevo_precio : real
nuevo_precio = precio – (precio * descuento)
Fin Módulo

```
// diagrama N-S: TIENDA
Inicio
Variables:
            p_neto, prec, desc : reales
            resp : alfanumerico = "s"
                                    resp == "s"
        Escribir "Precio del producto:"
        Leer prec
        Escribir "Descuento a realizar:"
        Leer desc
        Ejecutar PRECIO_NETO(prec, desc) → p_neto
        Escribir "El precio neto del producto es:", p_neto
        Escribir "Deseas calcular otro producto?:"
        Leer resp
Fin
```

DIAGRAMA DE FLUJO

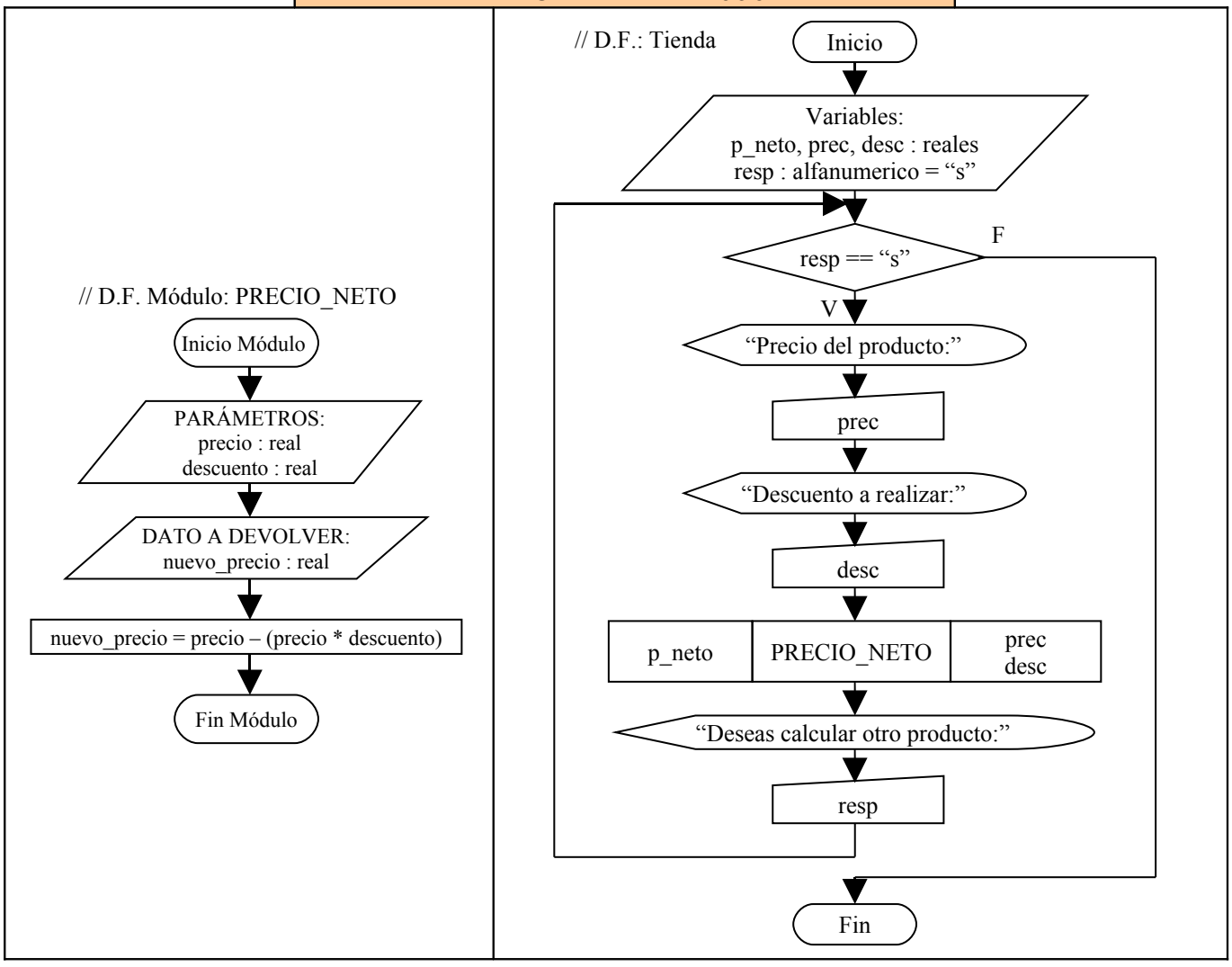


Tabla 45 Ejemplo 3 del manejo de Módulos o Funciones

Ejercicios.

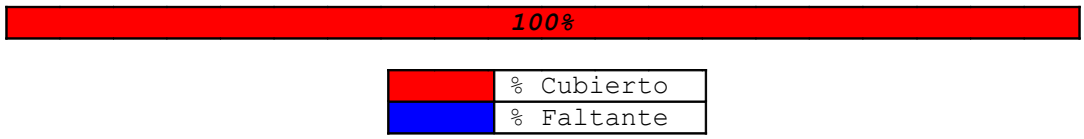
I. Escribe una función y programa para cada uno de los siguientes puntos, utilizando las tres diferentes técnicas.
1. Se necesita un sistema que calcule el factorial de un número dado por el usuario.
2. Se necesita un sistema que calcula el cuadrado de un número dado por el usuario
3. Función que acepta cuatro variables alfanuméricas, las cuales son las opciones del menú.
4. Función que acepta cuatro variables alfanuméricas, las cuales son las opciones del menú, pero además que además devuelva la opción presionada por el usuario.
5. Función que recibe dos números. Si el primero es mayor que el segundo devuelve un 1, si el segundo es mayor que el primero devuelve un -1, pero si son iguales el dato devuelto es un 0.
6. Función que recibe 3 datos numéricos, con los cuales escribe la fecha en la pantalla.
7. Módulo que recibe las horas trabajadas, el precio por hora trabajada y el precio de la hora extra. Con estos calcular y devolver el salario de un trabajador, donde las primeras 40 horas se pagan normal y las restantes se pagan como tiempo extra.
8. Módulo que recibe las tres calificaciones parciales de un alumno, el cual regresa su calificación con letra, donde si su promedio es menor de 6 le corresponde "NA", de 6 a 8 obtiene "S", mayor a 8 y cuando mucho 9 saca "B" y superior a 9 su calificación es "E".

CONCLUSIÓN

En este tema se abarcó un único subtema, el cual tiene como objetivo hacernos unos programadores funcionales, es decir que los sistemas los dividamos en pequeños módulos o funciones, los cuales realizan una tarea específica, teniendo de la capacidad de utilizarse o ejecutarse cuantas veces sea necesario, además de que para ejecutarse cada vez de una manera distinta les podemos enviar ciertos datos llamados parámetros, y también podemos hacer que después de ejecutarse regresen un valor de un tipo específico al algoritmo principal.

Con la comprensión absoluta de este tema hemos cubierto el 10% faltante para lograr el objetivo del curso.

OBJETIVO DEL CURS0



CONCLUSIÓN GENERAL

Si estas leyendo esto ¡FELICIDADES!, ¿Por qué?, por la simple y sencilla razón que has dado el paso más importante para ser un ***gran programador***, ya que para serlo hay que pensar como ellos y tú lo has logrado. Veamos por que:

Si en este momento se te contrata como programador para instalar un sistema de información en una empresa que requiere mostrar los resultados obtenidos en un dispositivo de salida; tu sabes cual es la tarea de un programador, que es un sistema de información, que es un dispositivo de salida, todos los pasos que tendrías que realizar el sistema, y muchos otros conceptos que solo un diseñador de sistemas conoce. Todo esto lo aprendiste en el primer tema.

Si este sistema que se te solicita requiere que se calcule el salario neto de un trabajador deduciendo IMSS, FOVISSTE, ISR, etc. Tú sabes como expresar un cálculo de estos en una sola o varias expresiones utilizando identificadores de diferentes tipos y operadores aritméticos, los cuales manejaste a partir del segundo tema.

Si este proceso que se te esta solicitando requiere que se ejecute para 100 trabajadores y que se den bonificaciones por puntualidad, tu sabrías expresar esta solución tanto en pseudocódigo, diagramas de flujo y diagramas N-S, los cuales manejaste a partir del tercer módulo, además de que podrías tomar la decisión de a quien darle su bono debido a que aplicaste durante el curso las estructuras condicionales, las estructuras cíclicas las aplicarías para hacer que este proceso no solo se le aplique a 100 trabajadores

sino a todos los que sean necesarios. Todo esto lo manejaste y aprendiste en el transcurso del tercer y cuarto tema.

Tu podrías hacer más efectivo tu sistema si decides utilizar estructuras para almacenar los datos del trabajador como nombre, dirección, teléfono, # del IMSS, etc. Además podrías utilizar arreglos para almacenar los datos de 100 o cuantos trabajadores sean necesarios. Esto es parte de los que se abarco en el quinto tema.

Podrías hacer que tus algoritmos estuvieran divididos en otros pequeños, los cuales probablemente se encargarían del descuento del IMSS, del FOVISSTE, etc. Ya que tu sabes crear módulos los cuales manejaste en el sexto tema.

Ahora para ser un excelente programador, solo te basta aprender un lenguaje de programación el cual puedes asimilar en otro curso o simplemente adquiriendo un libro, recordando que la práctica hace al maestro y que no te será difícil ya que ahora tienes las mejores bases.

¡GRACIAS POR TU ESFUERZO!

ATTE.

P.L.I. Carlos Augusto Flores Valerio

BIBLIOGRAFÍA

Fundamentos De Programación, Algoritmos Y Estructura De Datos

JOYANES Aguilar Luis
Ed. McGraw Hill
México

Como Programar En C/C++

DEITEL H.M. / DEITEL P.J.
Ed. Prentice Hall
México

Organización De Computadoras, Un Enfoque Estructurado

TANENBAUM Andrew
Ed. Prentice Hall
México

Análisis Y Diseño De Sistemas De Información

SENN James A.
McGraw Hill
México

Introducción A La Computación

NORTON Peter
McGraw Hill
México

Fundamentos De Programación, Libro De Problemas

JOYANES Aguilar Luis/RODRÍGUEZ Baena Luis/FERNÁNDEZ Azuela Matilde
McGraw Hill
México

Informática II

GONZÁLEZ Osorio Gonzalo
Compañía Editorial Nueva Imagen
México

Informática, Para Cursos De Bachillerato

FERREYRA Cortés Gonzalo
Ed. Alfaomega
México