Diseño estructurado de algoritmos aplicados en PSEINT

Pedro Vélez Duque

compAs
Grupo de capacitación e investigación pedagógica

Diseño estructurado de algoritmos aplicados en PSEINT

compAs
Grupo de capacitación e investigación pedagógica

Diseño estructurado de algoritmos aplicados en PSEINT

Pedro Vélez Duque

compAS
Grupo de capacitación e investigación pedagógica

Diseño estructurado de
algoritmos aplicados en PSEINT

Editado en Guayaquil - Ecuador
Primera edición

ISBN: 978-9942-33-463-3

compÁs
Grupo de capacitación e investigación pedagógica

Cita.
Vélez, P. (2021) Diseño estructurado de algoritmos aplicados en PSEINT. Editorial Grupo Compás.

Agradecimiento

Agradezco primero a Dios, por la oportunidad de haber tenido a mi madre, a mi familia por su apoyo incondicional y a la Universidad Agraria del Ecuador por haberme dado la oportunidad de poder hacer esta publicación.

Dedicatoria

Dedico esta obra a mi madre ya que ella siempre fue un pilar fundamental en mi vida, apoyándome irrestrictamente y motivándome para culminar esta obra.

Índice

Resumen

En esta publicación se aborda varios contenidos más triviales en las ciencias informáticas que es el diseño estructurado de algoritmos y el ejercicio profesional de los que siguen esta rama. El diseñó estructurado de algoritmos es una habilidad esencial que debe ser dominada por el educando para que a su vez desarrolle destrezas en cuanto a la resolución de problemas mediante el uso de la computadora. Por esto, surge este proyecto de hacer este libro que contribuya en la formación estudiantes de las carreras de ciencias de computación e informática en cuanto a los conceptos, ejemplos y ejercicios que esta obra presenta en todos los ámbitos que interviene en el desarrollo de aplicaciones.

En el primer capítulo se establece los conceptos básicos para la solución de problemas por medio de computadoras, propósitos para aprender de programar, software, desarrollo de aplicaciones, metodología para la solución de problemas por medio de computadora e introducción a Pseint.

En el segundo capítulo describe las tipos de datos simples, tipos de operadores: aritmético, relacionales, lógicos, identificadores y su uso, elementos básicos de programación de Pseint: constantes, operadores, funciones matemáticas, asignación, entrada y salida.

En el tercer capítulo se detalla las técnicas algorítmicas para la solución de problemas más utilizadas como pseudocodigo y diagrama de flujo: estructura, normas, usos, símbolos y reglas, comenzar a codificar pseudocodigo en Pseint.

En el capítulo cuatro se plantean las estructuras de control: secuenciales, condicionales: simples y compuestas, cíclicas, estructuras de control en Pseint.

En el quinto capítulo habla sobre arreglos: ordenar arreglos y arreglos bidimensionales, estructuras y arreglos en Pseint.

Durante y al final del desarrollo de cada capítulo se presenta actividades teóricas y prácticas para fortalecer los conocimientos impartidos, haciendo

de esta obra, sea una publicación más didáctica y comprensible y además se describen los conceptos de Pseint.

Palabras claves: operadores, técnicas, pseudocodigo, diagramas de flujo, estructuras, interfaces gráficas, diseño de interfaz.

Introducción

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 máquinas. 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..

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 libro, tiene la finalidad de formar 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.

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

1.1 Introducción

Tal y como se mencionó en la introducción general, se espera que este libro 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 está 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.

1.2 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:

1.2.1 Sistema

Un sistema es un conjunto de componentes que interactúan entre sí para lograr un objetivo común (Senn, 2001).

1.2.2 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 (Senn, 2001).

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

1.2.3 Programa

Es el conjunto de instrucciones escritas de algún lenguaje de programación y que ejecutadas secuencialmente resuelven un problema específico (JOYANES, 1996).

1.2.4 Lenguaje de Programación

Es cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora (Norton, 2006).

Los lenguajes de programación pueden ser de 3 tipos:

- Programas escritos en Lenguaje máquina. (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 inglés 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 intérpretes.

1.2.5 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 (DEITEL, 2004). Toda computadora, tiene los siguientes elementos:

- 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).

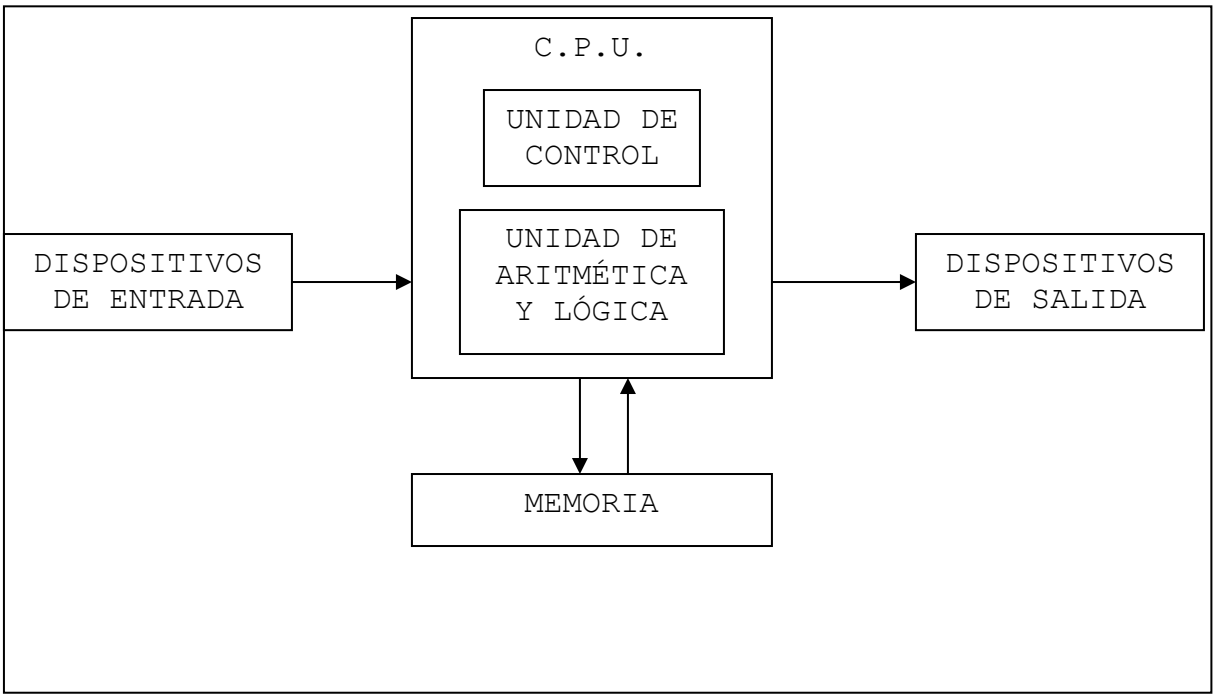


Figura 1 Diagrama que representa el funcionamiento de cualquier computadora

Fuente *(TANNENBAUM, 2000)* **.**

1.2.6 Programador o analista o diseñador de sistemas.

Es la persona encargada de crear un programa o sistema en un lenguaje de programación específico.

1.2.7 Usuario.

Es la persona que interactúa con el sistema de información, o mejor dicho con la computadora (Senn, 2001).

- 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 está diseñado para que aprendamos a realizar estos.

1.2.8 Primeras tareas de un desarrollador

Hasta este punto, hemos visto que la interacción con dispositivos electrónicos se presenta por medio de interfaces. Estas, a su vez, cuentan con un software que traduce nuestras acciones a un lenguaje máquina reconocido por el hardware, con lo cual se obtiene un resultado. Para lograr esto, como desarrolladores es importante que conozcamos la manera de darle al computador, las indicaciones necesarias. Por tal motivo debemos aprender a crearlas por medio del estudio de la lógica de programación, y a plasmarlas en líneas de código de un software específico para diagramar y tipiar.

A continuación, desarrollaremos dos conceptos fundamentales que debemos tener bien en claro durante el desarrollo: algoritmo y lenguajes de programación. Una vez que los dominemos, podremos lograr que el software cumpla con todas nuestras indicaciones.

1.2.9 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 (FERREYRA, 2007).

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.

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

1.3 Propósitos para aprender a programar

Cuando nos proponemos aprender a desarrollar aplicaciones o sistemas, lo hacemos para cobijar determinadas necesidades, ya sean personales o de terceros, y así adquirir un ingreso económico a cambio de nuestro trabajo.

Uno de los pasos fundamentales que debemos desarrollar antes de comenzar es aprender la programación lógica. Esto es significativo porque, si bien los lenguajes de programación tienen sus características, las soluciones lógicas son analizadas de un solo modo. De esta manera, conocer este tema claramente nos permitirá migrar a todos los lenguajes que queramos.

Aprender a desarrollar aplicaciones nos ofrece muchas posibilidades, ya que podremos realizar programas en cualquier plataforma, ya sea para la Web, Windows, Linux o Macintosh; incluso, para móviles, televisión inteligente, etc. (Casale J. , 2012). El propósito principal es tener la base lógica de programación, y luego elegir cuál es el lenguaje en el que deseamos poner nuestro mayor esfuerzo. Puede ser el que esté latente en el mercado, uno específico de un área (como para los trabajos científicos) o, simplemente, aquel en el que nos sintamos más cómodos para trabajar.

Al adquirir estos conocimientos, podremos tomar cualquier modelo de negocio o problema funcional de una organización, y resolverlo mediante la programación de una aplicación.

1.4 Software: conceptos básicos y clasificación

Los programas, conocidos como software, son instrucciones que se le da a la computadora. Sin programas, una computadora es una maquina vacía. Las computadoras no entienden los lenguajes humanos, de modo que se necesita utilizar lenguajes de computadora para comunicarse con ellas.

El software de una computadora es un conjunto de instrucciones de programa detalladas que controlan y coordinan los componentes hardware de una computadora y controlan las operaciones de un sistema informático (Joyanes, 2013). El auge de las computadoras el siglo pasado y en el actual siglo XXI, se debe esencialmente al desarrollo de periódicas generaciones de software potente y cada vez más amistoso, es decir fácil de usar.

Las operaciones que debe realizar el hardware son especificadas por una lista de instrucciones, llamados programas, o software. Un programa de software es un conjunto de sentencias o instrucciones que se le da a la computadora. El proceso de escritura o codificación de un programa se denomina programación y las personas que se especializan en esta actividad se denominan programadores.

Existen dos tipos de importantes de software: software de sistema y software de aplicaciones. Cada tipo realiza una función diferente. Los dos tipos de software están relacionados entre sí, de forma que los usuarios y programadores pueden hacer uso eficiente de la computadora.

El software del sistema es un conjunto generalizado de programas que gestiona los recursos de la computadora, como si fuera el procesador central, enlaces de comunicaciones y periféricos. Los programadores que escriben software del sistema se llaman programadores de sistemas. El software de aplicaciones es el conjunto de programas escritos por empresas o usuarios individuales o en el equipo y que instruyen a la computadora para que ejecuten una tarea específica.

En la figura 1.2 se muestra una visión organizacional de una computadora donde se ven los diferentes tipos de software a modo de capas de la computadora desde su interior (el hardware) hasta su exterior (usuario).

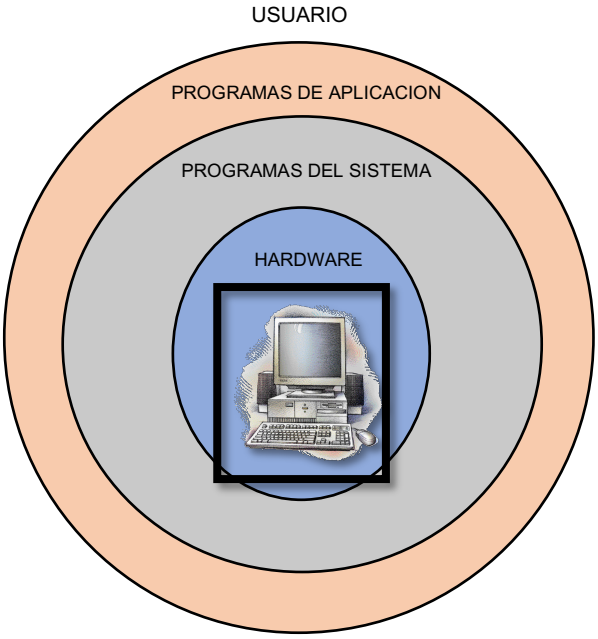


Figura 1. Relación entre programas de aplicación y programas del sistema

***Fuente*:** (Joyanes, 2013)

El software de sistema es el conjunto de programas indispensables para que la maquina funcione; se denomina también programas del sistema. Estos programas son básicamente, el sistema operativo, los editores de textos, los compiladores/interpretes (lenguajes de programación) y los programas de utilidad.

El software de aplicación tiene como función principal asistir y ayudar a un usuario de una computadora para ejecutar tareas específicas. Los usuarios normalmente, compran el software de aplicación en discos CD y DVD o los descargan de la red e instalan el software copiando los programas correspondientes de los discos en el disco duro de la computadora.

El sistema operativo es quizás la parte más importante del software del sistema y es el software que controla y gestiona los recursos de la computadora. En la práctica el sistema operativo es la colección de programas de computadoras que controla la interacción del usuario y el hardware de la computadora. El sistema operativo es el administrador principal de la computadora, y por ello a veces se lo compara con el director

de la orquesta ya que este software es el responsable de dirigir todas las operaciones de la computadora y gestiona todos sus recursos (Joyanes, 2013). El kernel es el núcleo del sistema operativo es el software que contiene los componentes fundamentales dicho sistema operativo.

1.5 Desarrollo de aplicaciones

Es fundamental conocer y comprender los elementos iniciales de los procesos que debemos tener en cuenta para incursionar en el mundo de la programación de aplicaciones. Como desarrolladores, nos propondremos encontrar distintas soluciones posibles para resolver una situación mediante la confección de aplicaciones informáticas (Casale J. , 2012).

En el mundo actual, todos los días nos encontramos con distintos desarrollos de aplicaciones, como, por ejemplo, el programa que controla nuestro teléfono móvil. A su vez, contamos con programas que, en tiempo real, nos permiten traducir diferentes idiomas, conectarnos a Internet, jugar, llevar un listado de lo que compramos en el supermercado registrando su código de barras y estimando el costo total, y muchas alternativas más.

Es posible distinguir que algunas aplicaciones son más básicas, y otras, más complejas. Si bien es posible considerar el teléfono móvil como un aparato complejo, el desarrollo de aplicaciones también impacta en otros elementos de uso cotidiano, tales como las heladeras inteligentes, el programa del microondas, las alarmas, y otros (Casale J. , 2012). El mundo en su totalidad se rige por programas desarrollados mediante algún lenguaje de programación. Todos los elementos electrónicos, en menor o mayor grado, contienen aplicaciones específicas para cumplir su misión.

Una definición que podemos encontrar en primera instancia sobre el desarrollo de una aplicación es: confeccionar, probar y buscar errores de un programa informático. Dicho programa va a soluciona una situación o problema comúnmente llamado “modelo de negocio”, que puede ser, por ejemplo, cuando nuestra organización necesita llevar un control del inventario de productos. Para poder elaborar un programa informático, precisamos manejar un lenguaje de programación que nos permita realizar la prueba o búsqueda de errores (Casale J. , 2012).

Figura 2. Elementos electrónicos como los teléfonos móviles
Fuente: (Casale J. , 2012)

1.6 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:

1.6.1 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 implantar 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.

1.6.2 Análisis Del Sistema.

En esta actividad se tienen que comprender todas las facetas importantes de la parte de la empresa que está 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.

1.6.3 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 esté 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.

1.6.4 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 qué 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.

1.6.5 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 está 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.

1.6.6 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.

1.7 Introducción a Pseint

Pseint (Pseudo interprete) es un entorno de desarrollo integrado (IDE) para Pseudocòdigo, un lenguaje de programación imperativo simple y en castellano. Es decir, Pseint es un editor e intérprete de programas escritos en Pseudocòdigo. Su interfaz gráfica permite crear, almacenar, ejecutar y corregir fácilmente programas en Pseudocòdigo.

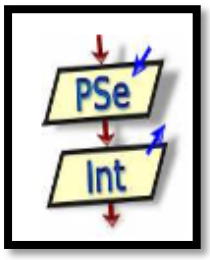


Figura 3 Icono en el escritorio de Pseint
Fuente *(Novara, 2020)*

1.7.1 ¿Para qué sirve Pseint?

Pseint este pensado para ayudar a los estudiantes que se inician en la construcción de algoritmos. El Pseudocòdigo se suele utilizar como primer contacto para introducir conceptos básicos como el uso de estructuras de control, expresiones, variables, etc., sin tener que lidiar con particularidades de la sintaxis de un lenguaje real. Esta aplicación pretende facilitarle al principiante la tarea de escribir algoritmos en este Pseudocòdigo presentado un conjunto de ayudas y asistencias, y brindarles además algunas herramientas adicionales que le ayuden a encontrar y comprender la lógica de los algoritmos.

1.7.2 Instalación de Pseint

Se utilizará como herramienta para el aprendizaje practico la aplicación PseInt versión de fecha 01/05/2020. Esta aplicación está disponible en forma gratuita en el sitio

http://pseint.sourceforge.net/index.php?page=descargas.php cuyo autor, Pablo Novara (zaskar_84@yahoo.com.ar) lo publica bajo licencia GPL.

En el sitio web se indica que esta aplicación está diseñada para asistir a un estudiante en sus primeros pasos en programación. Mediante un simple e intuitivo pseudolenguaje en español (complementado con un editor de diagramas de flujo), le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, quitando las dificultades propias de un lenguaje y proporcionando un espacio de trabajo con numerosas ayudas y recursos didácticos.

Debemos hacer una descarga de la aplicación en su versión más actualizada a partir de la página del autor, se deberá elegir la descarga de acuerdo a nuestro sistema operativo, GNU Linux / Windows / Mac OS. Al finalizar la descarga nos encontraremos con un archivo para instalar en el caso de Windows, y para descomprimir en el caso de utilizar Linux. De todas formas en la misma página de la descarga hay una explicación muy completa que nos ayudará a la instalación.

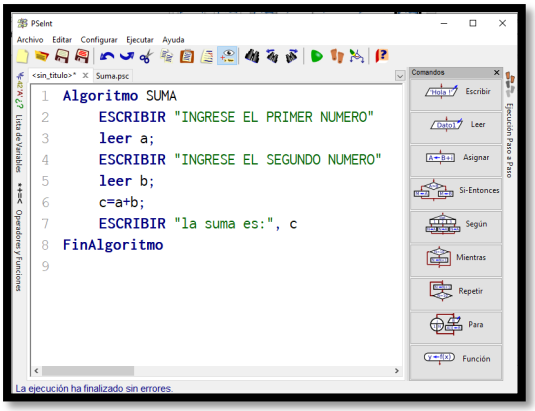


Figura 4 Ventana del programa Pseint
Fuente *(Novara, 2020)*

1.7.3 Características y funcionalidades de Pseint

1. Presenta herramientas de edición básicas para escribir algoritmos en Pseudocòdigo en español.
2. Permite la edición simultánea de múltiples algoritmos.
3. Presentas ayudas en la escritura, autocompletado, ayudas emergentes, plantillas de comandos, coloreado de sintaxis y ejecutar algoritmos escritos
4. Permite ejecutar el algoritmo paso a paso controlando la velocidad e inspeccionando expresiones.
5. Puede confeccionar automáticamente la prueba de escritorio.
6. Determina y marca los errores de sintaxis y en tiempo de ejecución
7. Genera diagramas de flujo a partir del algoritmo escrito.
8. Convierte el algoritmo de Pseudocòdigo a codigo c++
9. Ofrece un sistema de ayuda integrado acerca del Pseudocòdigo.
10. Incluye un conjunto de ejemplos de diferentes niveles de dificultad.
11. Es multiplataforma (probado en Microsoft Windows y GNU/Linux).
12. Es totalmente libre y gratuito (licencia GLP).

1.8 Preguntas de repaso

Marcar VERDADERO (V) o FALSO (F) a los siguiente enunciados

1. La Unidad Aritmético - Lógica es la parte del C.P.U. que realiza operaciones aritméticas y lógicas, tales como suma, resta, multiplicación, división y comparaciones.( )
2. La Memoria es una parte de la computadora en donde se guardan los datos a procesar y la información resultante. ( )
3. El CPU es el microprocesador de la computadora y es el encargado de hacer todos los cálculos y operaciones. ( )
4. Un sistema es un conjunto de componentes que interactúan entre sí para lograr un objetivo común. ( )
5. Un programa es el conjunto de instrucciones escritas de algún lenguaje de programación y que ejecutadas secuencialmente resuelven un problema específico. ( )

6. Un lenguaje de programación es cualquier lenguaje artificial que puede utilizarse para definir una secuencia de instrucciones para su procesamiento por un ordenador o computadora. ( )
7. Un programador es la persona encargada de crear un programa o sistema en un lenguaje de programación específico. ( )
8. El usuario es la persona que interactúa con el sistema de información, o mejor dicho con la computadora. ( )
9. Un 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. ( )
10. El análisis del sistema la actividad que se tienen que comprender todas las facetas importantes de la parte de la empresa que está bajo estudio. ( )

2. OPERACIONES CON LOS DATOS

2.1 Introducción

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

La importancia de este tema es tan grande debido a que todos los sistemas de información realizan cálculos con datos para entregar resultados a las empresas, por lo cual debemos saber que los datos que manejan las empresas solamente pueden ser números, letras y números y una respuesta afirmativa o negativa; y los cálculos que los sistemas pueden 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 está dedicado a enseñarnos como se redacta una expresión de tal manera que la computadora la entienda.

El tercero está diseñado para saber cómo 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.

2.2 Tipos De Datos Simples

Cualquier sistema de información por lo más simple 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. (Senn, 2001).

Información. "Es un conjunto de datos estructurados o procesados". (Senn, 2001).

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

Tabla 1 *Clasificación de los datos*
Fuente: *(Joyanes, 2013)*

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

2.2.1. Datos Simples.

El primer objetivo de toda computadora es el manejo de la información o datos. Los datos describen los objetos con los cuales opera una computadora. (Joyanes, 2013). A continuación se describen los tipos de 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 (Booleanos): 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.

2.3 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 nosotros la indicaremos a la computadora los cálculos a realizar a ciertos datos y además considerar el orden de la precedencia de los operadores.

Por ejemplo, si se le pide a un sistema que obtenga el promedio de un estudiante que tiene 5 materias, a la maquina no le podemos decir "saca el promedio del estudiante" debido a que es una instrucción que no reconoce, para que despliegue el resultado le tenemos que indicar suma de las calificaciones, y al resultado lo divides entre cinco. Pero aun así no es fácil como parece se representa esta instrucción de una manera que la computadora la comprenda. Para lo cual tenemos que elaborar una expresión 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.

2.3.1 Operadores Aritméticos.

Son aquellos con los que podemos realizar operaciones como suma, resta, multiplicación, división, módulo y asignación. En la tabla siguiente se muestran las operaciones indicadas.

Tabla 2 *Los diferentes operadores aritméticos*

OPERACIÓN	OPERADOR	EXPRESIÓN	RESULTADO
Suma	+	5 + 7	12
Resta	-	5 - 7	2
Multiplicación	*	5 * 7	35
División	/	10 / 2	5
Módulo	%	10 % 3	1
Asignación	=	a = 8, b = 2 c = a + b	c=10

Fuente: (Joyanes, 2013)

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. (Senn, 2001).

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 está 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. Lo anterior se puede resumir en la siguiente tabla.

Tabla 3 *Precedencia de los operadores*

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

FUENTE: (Joyanes, 2013)

Ejemplo 1. Supongamos que tenemos la siguiente expresión:

Tabla 4 *Ejemplo 1 de precedencia de operadores aritméticos*

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

Ejemplo 2. Supongamos que tenemos la siguiente fórmula:

Tabla 5. *Ejemplo 2 de precedencia de operadores aritméticos*

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

2.3.2 Operadores Relaciónales.

Los operadores relacionales se usan para determinar la relación de la expresión de la izquierda con la derecha (binarios). El resultado de esta evaluación regresa el valor de falso (0) o verdadero (1).
Lo anterior se puede resumir en la siguiente tabla.

Tabla 6 *Conjunto de operadores relacionales*

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

Fuente: (Joyanes, 2013)

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.

Se puede resumir en la siguiente tabla los operadores aritméticos y relacionales en orden de precedencia.

Tabla 7 *Precedencia de los operadores aritméticos y relacionales*

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

Fuente: (Joyanes, 2013)

Ejemplo 1. Supongamos que tenemos la siguiente fórmula:

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

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

Ejemplo 2. Supongamos que tenemos la siguiente fórmula:

Tabla 9 *Ejemplo 2 de cómo se utilizan los operadores relacionale*

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

2.3.3 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.

Tabla 10 *Tabla de la verdad del operador lógico AND*

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

Fuente: (JOYANES, 1996)

Tabla 11 *Tabla de la verdad del operador lógico OR*

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

Fuente: (JOYANES, 1996)

Tabla 12 . *Tabla de la verdad del operador lógico NOT*

A	NOT
0	1
No 0	0

Fuente: (JOYANES, 1996)

OPERADOR	OPERACION LÓGICA
&&	AND
||	OR
!	NOT

Tabla 13 *Conjunto de Operadores lógicos*
Fuente: (JOYANES, 1996)

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.

A continuación se muestra la siguiente tabla el orden de precedencia de todos los operadores:

Tabla 14 *Tabla de precedencia de todos los operadores*

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

Fuente: (Joyanes, 2013)

Ejemplo 1. Supongamos que tenemos la siguiente fórmula:

Tabla 15 *Ejemplo 1 de cómo se utilizan los operadores relacionales*

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

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:

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

EXPRESIÓN		Z = 1 || (6 * !0 > 5 && 9 < 3 * 4)	
ACTIVIDAD		OPERACIÓN	RESULTADO
1.	Se realiza todo lo que está en el 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

2.4 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.

2.4.1 ¿Por qué usar identificadores?

Si nosotros no creamos un identificador, el dato que deseamos guardar se almacenaría en una posición de memoria la cual está 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 cómo tratar a esta información, recordemos que en la computadora solo están almacenados ceros y unos.

<table>
<tr><td>FF00h</td><td></td><td></td><td></td><td></td><td>FF0Ch</td><td></td><td></td><td></td><td></td><td rowspan="15">Sin el uso de identificadores, tendríamos que saber la dirección de memoria en donde se guardo la información.</td></tr>
<tr><td>FF01h</td><td colspan="4" rowspan="3">00101110</td><td>FF0Dh</td><td></td><td></td><td></td><td></td></tr>
<tr><td>FF02h</td><td>FF0Eh</td><td></td><td></td><td></td><td></td></tr>
<tr><td>FF03h</td><td></td><td></td><td></td><td></td><td>FF0Fh</td><td colspan="4" rowspan="2">10001111</td></tr>
<tr><td>FF04h</td><td></td><td></td><td></td><td></td><td>FF10h</td></tr>
<tr><td>FF05h</td><td></td><td></td><td></td><td></td><td>FF11h</td><td></td><td></td><td></td><td></td></tr>
<tr><td>FF06h</td><td colspan="4" rowspan="2">11011101</td><td>FF12h</td><td></td><td></td><td></td><td></td></tr>
<tr><td>FF07h</td><td>FF13h</td><td></td><td></td><td></td><td></td></tr>
<tr><td>FF08h</td><td></td><td></td><td></td><td></td><td>FF14h</td><td></td><td></td><td></td><td></td></tr>
<tr><td>FF09h</td><td></td><td></td><td></td><td></td><td>FF15h</td><td colspan="4" rowspan="2">11110001</td></tr>
<tr><td>FF0Ah</td><td></td><td></td><td></td><td></td><td>FF16h</td></tr>
<tr><td>FF0Bh</td><td></td><td></td><td></td><td></td><td>FF17h</td><td></td><td></td><td></td><td></td></tr>
</table>

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:

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.

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 booleano.

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, boolean).
- 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.
- No deben de coincidir con palabras reservadas de los lenguajes de programación.

Ejemplos.

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

Pro_sem : entera = 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 = 0

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

Nombre : alfanumérico = “Carlos Augusto”

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

2.5 Elementos Básicos de Programación de Pseint

2.5.1 Constantes e identificadores

- Los identificadores deben constar solo de letras y números, comenzando siempre con una letra. (Novara, 2020).
- Las constantes de tipo carácter se escriben entre comas (').
- En las constantes númericas, el punto (.) es el separador decimal.
- Las constantes lógicas son Verdadero y Falso.

2.5.2 Operadores

Tabla 19 *Operadores Relacionales*

Operador	Significado	Ejemplo
>	Mayor que	3>2
<	Menor que	'Abc'<'abc'
=	Igual que	4=3
<=	Menor o igual que	'a'<='b'
>=	Mayor o igual que	4>=5
<>	Distinto que	var1<>var2

Fuente: (Novara, 2020)

Tabla 20 *Operadores Lógicos*

Operador	Significado	Ejemplo
&	Conjunción (y).	(7>4) & (2=1) // falso
|	Disyunción (o).	(7>4) | (2=1) //Verdadero
~	Negación (no).	~(2<5) // falso

Fuente: (Novara, 2020)

Tabla 21 *Operadores Algebraicos*

Operador	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
^	Potenciación

Fuente: (Novara, 2020)

La jerarquía de los operadores matemáticos es igual a la del algebra, aunque puede alterarse mediante el uso de paréntesis.

2.5.3 Funciones Matemáticas

Tabla 22 *Funciones Matemáticas y su significado*

Función	Significado
RC(X)	Raíz Cuadradade X
ABS(X)	Valor Absoluto de X
LN(X)	Logaritmo Natural de X
EXP(X)	Función Exponencial de X
SEN(X)	Seno de X
COS(X)	Coseno de X
ATAN(X)	Arcontangente de X
TRUNC(X)	Parte entera de X
REDON(X)	Entero más cercano a X

Fuente: (Novara, 2020)

La función raíz cuadrada no debe recibir un argumento negativo. La función exponencial no debe recibir un argumento menor o igual a cero.

2.5.4 Asignación

<Variable> <- <expresión>;

Primero evalúa la expresión de la derecha y luego asigna el resultado a la variable de la izquierda. Deben coincidir en tipo.
Ejemplos:

Var <- 1; // asigna 1 a var

Acumulador<-Acumulador +1; // incrementa en 1 el acumulador

2.5.5 Entrada

Leer <variable1>, <variable2>, ... , <variableN> ;

Lee desde el ambiente (en este caso el teclado) los valores y los asigna a la o las variables respectivamente. Puede leer una o más variables.
Ejemplos:
Leer Cantidad;
Leer Valor1, Valor2, Valor3;

2.5.6 Salida

Escribir <expr1>, <expr2>, ... , <exprN> ;

Devuelve al exterior (en este caso la pantalla) los resultados de las expresiones, o los contenidos de las variables. También puede tener uno o más parámetros separados por comas.

Ejemplos:

Escribir 'Ingrese el nombre:'; Escribir 'Resultado:', x*2;

2.6 Ejercicios de Aplicación

1. 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	
2. Expresa las siguientes fórmulas para que las entienda la computadora.	
Calcular el perímetro de un círculo.	
Calcular el área de un rectángulo	
Calcular el área de un circulo	
$X = Z^3$	
$X = \frac{4AC + (A + C) - 2AB}{2AB}$	

$X = 5(Y^2 + Z^3 - 5ZW + 3)$	

3. 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	

4. 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)	

5. 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 agronomia		
Número de horas trabajadas		
Número de control de un alumno		
Total de ingresos de una familia		
Número de días del año		
Número de cervezas por cartón		

3. TÉCNICAS ALGORÍTMICAS PARA LA SOLUCIÓN DE PROBLEMAS

3.1 Introducción

Una vez que se haya asimilado y entendido este tema, se habrá dado el primer paso para diseñar algoritmos, ya que sabremos cuales son las 2 diferentes técnicas más utilizadas que existen para crearlos.

Es por lo anterior donde radica la importancia de este capitulo, 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 dos secciones, donde cada una aborda a uno de los diferentes métodos.

- El primer subtema nos presenta a la técnica algorítmica NO gráfica llamada Pseudocódigo que según mi consideración es la más fácil y entendible de las 2 tácticas por que daría la importancia a la escritura como si se tratase de la codificación en un lenguaje de programación.

- El segundo subtema nos muestra a la técnica gráfica para la resolución de problemas orientados a computadoras llamada Diagramas De Flujo.

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%.

3.2 Pseudocódigo

Es un lenguaje de especificación de algoritmos, pero muy parecido a cualquier lenguaje de programación, por lo que luego su traducción al lenguaje es muy sencilla, pero con la ventaja de que no se rige por las normas de un lenguaje en particular (Casale J. y., 2014). Nos centramos más en la lógica del problema.

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.

Se comienza escribiendo la palabra Pseudocódigo seguida de dos puntos y a continuación un nombre que describa de manera general el problema a resolver

El pseudocódigo también va a utilizar una serie de palabras clave o palabras especiales que va indicando a continuación lo que significa:

1. Pseudocodigo "Nombre del problema a resolver": es la cabecera.
2. Área de definición de funciones, estructuras, constantes y variables con las cuales se las declara en esta sección del Pseudocodigo.

2.1 Funciones: F1 recibe en var1 un entero
2.2 Estructuras: E1 con los campos
Campo1 : entero : trabajo
Campo2 : entero : acumulador
2.3 Constantes: Const1 : entero = 50
2.4 Variables: Var3 : entero : contador

3. Inicio y Fin: Por donde empieza y acaba el algoritmo.

Begin /end : Pascal.
{ } : En C.

4. Instrucciones de Asignación
5. Instrucciones de lectura (leer)/escritura (escribir)
6. Sí <cond>

Entonces <acc1> → If then else
Sino <acc2>

7. Mientras <cond> /hacer → while do
8. Repetir / hasta → repeat until
9. Desde /hasta → for .. to
10. Según sea → Case
Switch

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.

Nota: A continuación se detalla siguientes puntos que se podría incluir en la escritura de los Pseudocodigo para entenderlos mejor:

1. Los comentarios van encerrados entre llaves.
2. Hay que utilizar la identación.

3.2.1 Estructura del Pseudocòdigo

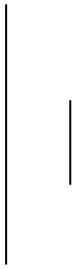
Para poder elaborar los Pseudocòdigo se debe escribir de acuerdo a la siguiente estructura que se apoya con lo antes mencionado en el punto anterior.

```
Pseudocodigo <nombre alg>
Var
        <nombre>: <tipo>
Inicio
        <Instrucciones>
```

Fin

Ejemplo:

```
Pseudocodigo Producto
Var
  P, num: entero
Inicio
  P ← 1
  Leer num
  Mientras num >=0 hacer
    P ← P*num
    Leer num
  Fin mientras
  Escribir p
Fin
```



3.2.2 Normas para el pseudocódigo

Como ya mencionamos, el pseudocódigo es parecido a un lenguaje de programación en su escritura y, como tal, contiene un determinado léxico (Casale J. y., 2014). Se trata de letras o caracteres que serán válidos para escribir las instrucciones que deseamos transmitir. La sintaxis es la especificación de palabras clave en combinación con otras que usaremos para formar las oraciones. Por último, la semántica es el significado que les daremos a dichas frases.

Las ventajas de utilizar un pseudocódigo, es que ocupa menos espacio en la hoja de papel, permite representar fácilmente operaciones repetitivas complejas, simplifica el pasaje de un pseudocódigo a un lenguaje de programación y permite observar con claridad los niveles que tiene cada operación.

Una de las normas generales que encontraremos en la mayoría de los pseudocódigos y codificación en lenguaje de programación es la estructura secuencial. Su definición es una acción o instrucción que sigue a otra en secuencia.

3.2.3 Uso del Pseudocòdigo

Para comprender más sobre el uso de pseudocódigo y la estructura secuencial, a continuación, realizaremos un caso práctico. En este ejemplo, lo representaremos con la puesta en marcha de un automóvil.

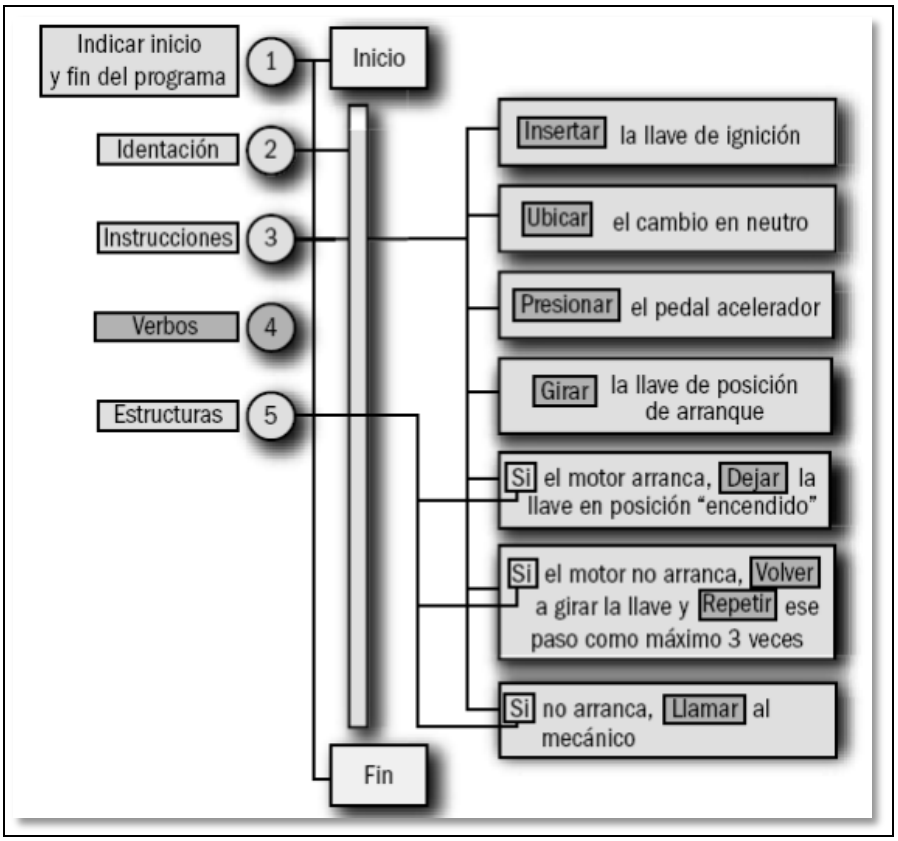


Figura 4. Explicación de pseudocódigo para arrancar un automóvil.
Fuente: (Casale J. y., 2014)

Para resumir algunas de las normas generales que vemos en el ejemplo, podemos decir que: la estructura es secuencial, se indica el INICIO y FIN del programa, en el margen izquierdo se deja un espacio llamado

identación para identificar fácilmente las estructuras, y cada instrucción comienza con un verbo.

3.3 Diagramas de flujo

Es una notación gráfica para implementar algoritmos. Se basa en la utilización de unos símbolos gráficos que denominamos cajas, en las que escribimos las acciones que tiene que realizar el algoritmo.

Las cajas están conectadas entre sí por líneas y eso nos indica el orden en el que tenemos que ejecutar las acciones.

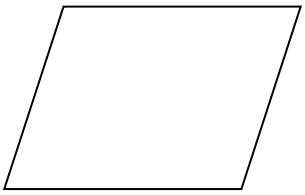
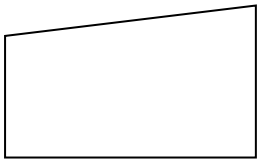
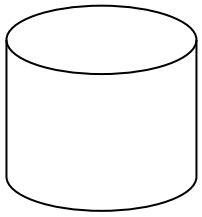
En todo algoritmo siempre habrá una caja de inicio y otra de fin, para el principio y final del algoritmo.

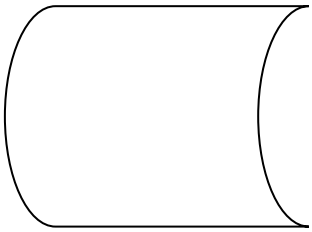
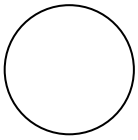
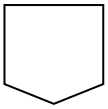
3.3.1 Símbolos que se usan para elaborar diagramas de flujo

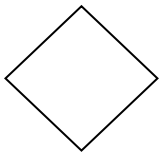
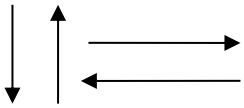
Los símbolos utilizados han sido normalizados por el instituto norteamericano de normalización (ANSI).

Tabla 23 *Conjunto de símbolos para diseñar diagramas de flujo*

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.
	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.

Fuente: (Joyanes, 2013)

3.3.2 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.
- 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.

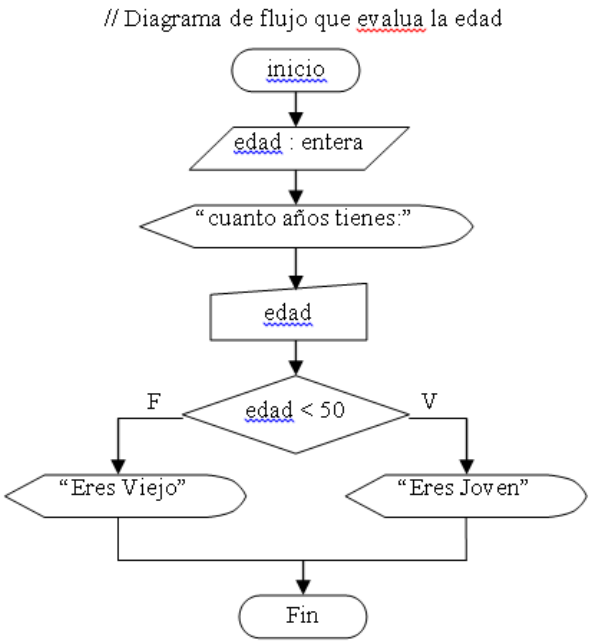


Figura 5 Ejemplo de cómo debe estar diseñado un diagrama de flujo
Fuente: (Joyanes, 2013)

3.4 Comenzar a codificar pseudocodigos en Pseint

Antes de comenzar a codificar, primero vamos a conocer la estructura de un pseudocodigo en Pseint

Forma general de un algoritmo escrito en pseudocodigo:

Proceso <nombre>

<Instrucción 1>

<Instrucción 2>

<Instrucción 3>

....

Fin proceso

Todos los algoritmos deben comenzar con la palabra *Proceso* y a continuación el nombre del mismo y deben finalizar con la palabra *FinProceso*. No puede haber instrucciones fuera del proceso, aunque si comentarios.

Las estructuras no secuenciales pueden anidarse. Es decir, pueden contener otras adentro, pero la estructura contenida debe comenzar y finalizar dentro de la contenedora.

Se pueden introducir comentarios luego de una instrucción o en líneas separadas mediante el uso de la doble barra (/ /). Todo lo que precede a / / no ser a tomado en cuenta al interpretar el algoritmo.

3.4.1 Crear un archivo en Pseint

Una vez instalado el programa Pseint, automáticamente se crea un archivos para iniciar la codificación y para crear los demás, ubíquese en el icono que se encuentra en la barra de herramientas o acceder a la opción Nuevo del menú Archivo y se presentara una imagen como la que se muestra a continuación, en la cual puedes iniciar a escribir el pseudocodigo en el área de trabajo de la ventana.

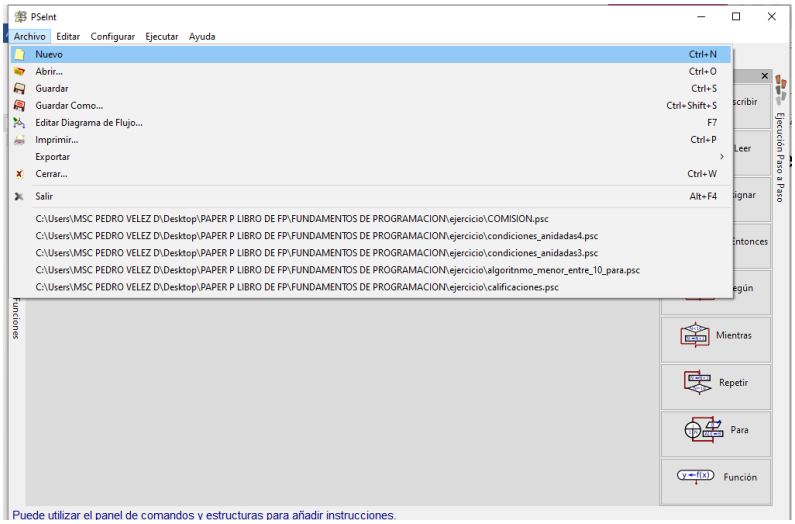


Figura 6 Crear un archivo en pseint
Fuente: (Novara, 2020)

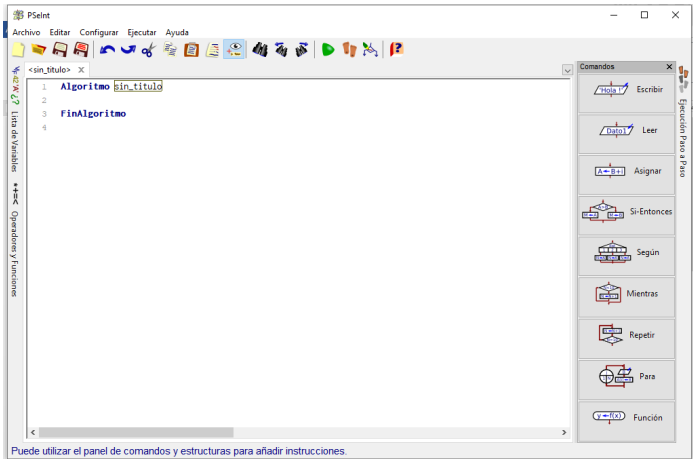


Figura 7 Área de trabajo de Pseint
Fuente: (Novara, 2020)

3.4.2 Ejemplo de codificación del pseudocòdigo en Pseint

Por medio de este ejercicio se expone a continuación la codificación del Pseudocòdigo descrito en páginas anteriores en la aplicación Pseint.

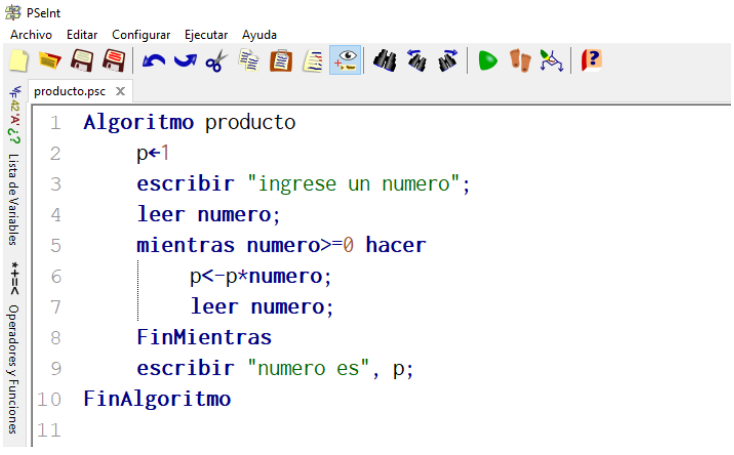


Figura 8 Codificación de un Pseudocòdigo en Pseint
Fuente: (Novara, 2020)

3.4.3 Generar un diagrama de flujo en Pseint

Ya ejecutado el Pseudocòdigo podemos generar el correspondiente diagrama de flujo haciendo clic en el siguiente icono que se encuentra en la barra de herramientas y se presentara una ventana con la imagen del diagrama de flujo.

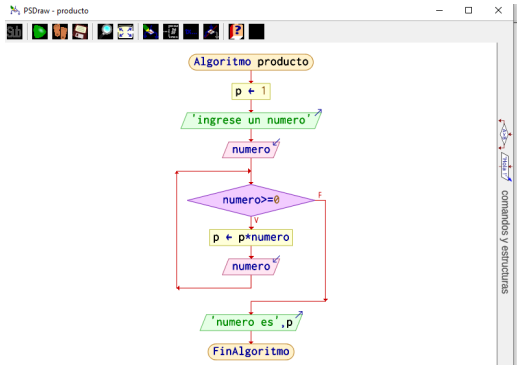


Figura 9 El Pseudocòdigo de ejemplo en diagrama de flujo
Fuente: (Novara, 2020)

El diagrama podrá ser guardado en formato de imagen, dando clic en la misma ventana del diagrama de flujo en el icono guardar y mostrara una ventana en la que se puede elegir el la ubicación y formato con él que se desea guardar dicha imagen.

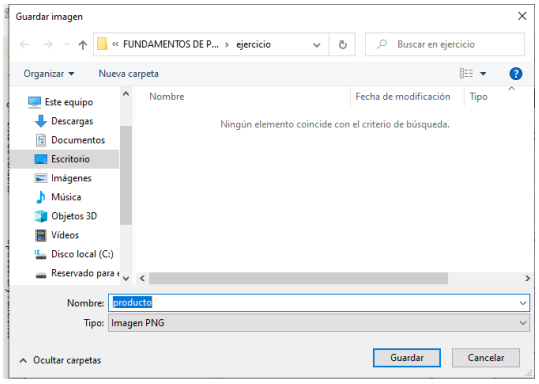


Figura 10 Guardar el diagrama de flujo en forma de imagen
Fuente: (Novara, 2020)

3.4.4 Guardar un archivo en Pseint

Después de haber escrito y ejecutado el Pseudocòdigo se da clic en  cualquiera de los siguientes iconos que se encuentran en la barra de herramientas o acceder al menú de archivo y luego hacer clic en guardar si está editando el Pseudocòdigo o guardar como, si se lo está guardando por primera vez, si esta es la opción aparece una ventana en la que se establecerá la ubicación y a su vez se escribirá el nombre para guardar.

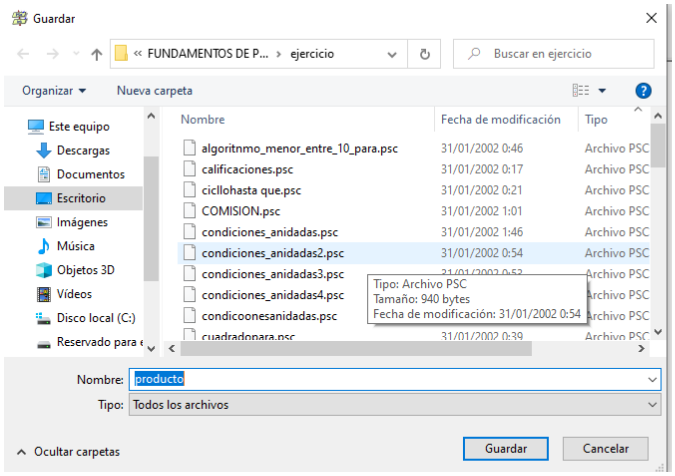


Figura 11 Ventana de guarda el archivo con extensión psc
Fuente: (Novara, 2020)

3.5 Ejercicios propuestos

Dados los siguientes ejercicios, codifíquelos en Pseint, corrija los errores, ejecútelos y genere el diagrama de flujo de cada uno de los pseudocodigos.

1. Dado el siguiente algoritmo que encuentre para que lea los dos números e imprima el resultado de sumar, restar, multiplicar y dividir dichos números.

```
Proceso
Escribir ‘numero 1: ;
Leer n1;
Escribir ‘numero 2: ;
Leer n2;
```

```
Escribir ‘n1’,’+’,’n2’,’=’, n1+n2;
Escribir n1,’-’,n2,’=’, n1-n2;
Escribir n1,’*’,n2,’=’, n1*n2;
Escribir n1,/,n2,’=’, n1/n2;
Finproceso
```

2. Dado el siguiente algoritmo que encuentre para que calcule e imprima el promedio de tres números.

```
Proceso promedio
Escribir ‘Ingrese los tres datos: ;
Leer n1,n2;
suma= n1+n2+n3;
prom<-suma/3;
Escriba El promedio es: ,prom;
Finproceso
```

3. Dado el siguiente algoritmo donde se ingresan las horas trabajadas de una persona y el valor por hora. Calcular su salario e imprimirlo.

```
Proceso
Escribir Numero de horas trabajadas:” ;
Leer nht;
Escribir Precio por hora;
Leer precioh;
salario nht* precioh;
Escribir “Su salario es:”salario;
Finproceso
```

4. ESTRUCTURAS DE CONTROL

4.1 Introducción

Después de haber conocido en el tema pasado las diferentes técnicas algorítmicas, en el presente 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 que se habla de las estructuras secuénciales, en las cuales no hay decisiones y tan solo basta con analizar cuáles son los datos de entrada y operaciones necesarias para producir las salidas deseadas.

El segundo subtema se diseña algoritmos sutiles, 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 conduce a varios 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 tres 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 de esta obra.

4.2 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 (Joyanes, 2013).

Estos algoritmos están representados por las estructuras secuénciales, en las que una acción (instrucción) sigue a otra. 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):

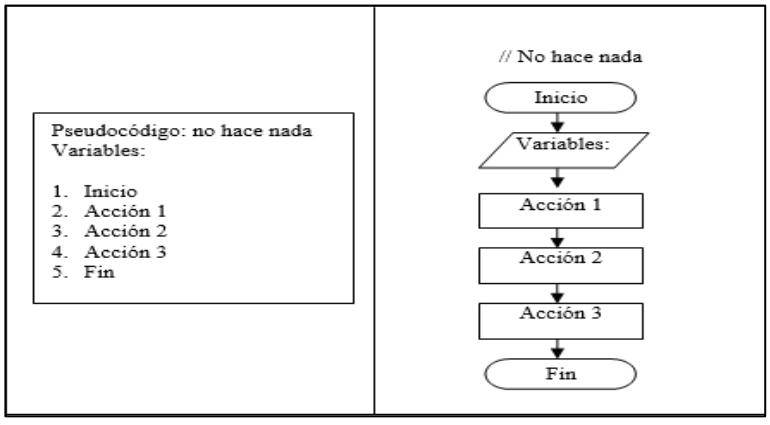


Figura 12 Ejemplo de cómo se diseña un algoritmo secuencial.

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 (Joyanes, 2013). El uso de estas acciones ya fue explicado en capítulos anteriores.

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 está planteando. En este análisis hay que identificar cuáles son los datos de salida, es decir, los resultados que debe de arrojar nuestro algoritmo; identificar 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 (JOYANES, 1996).

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 prueba de escritorio, lo cual significa dar valores simulados a las variables y revisar los resultados.

Ejemplo 1. 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 Año_act		Edad = Año_act – Año_nac

Paso II. Diseñar El algoritmo

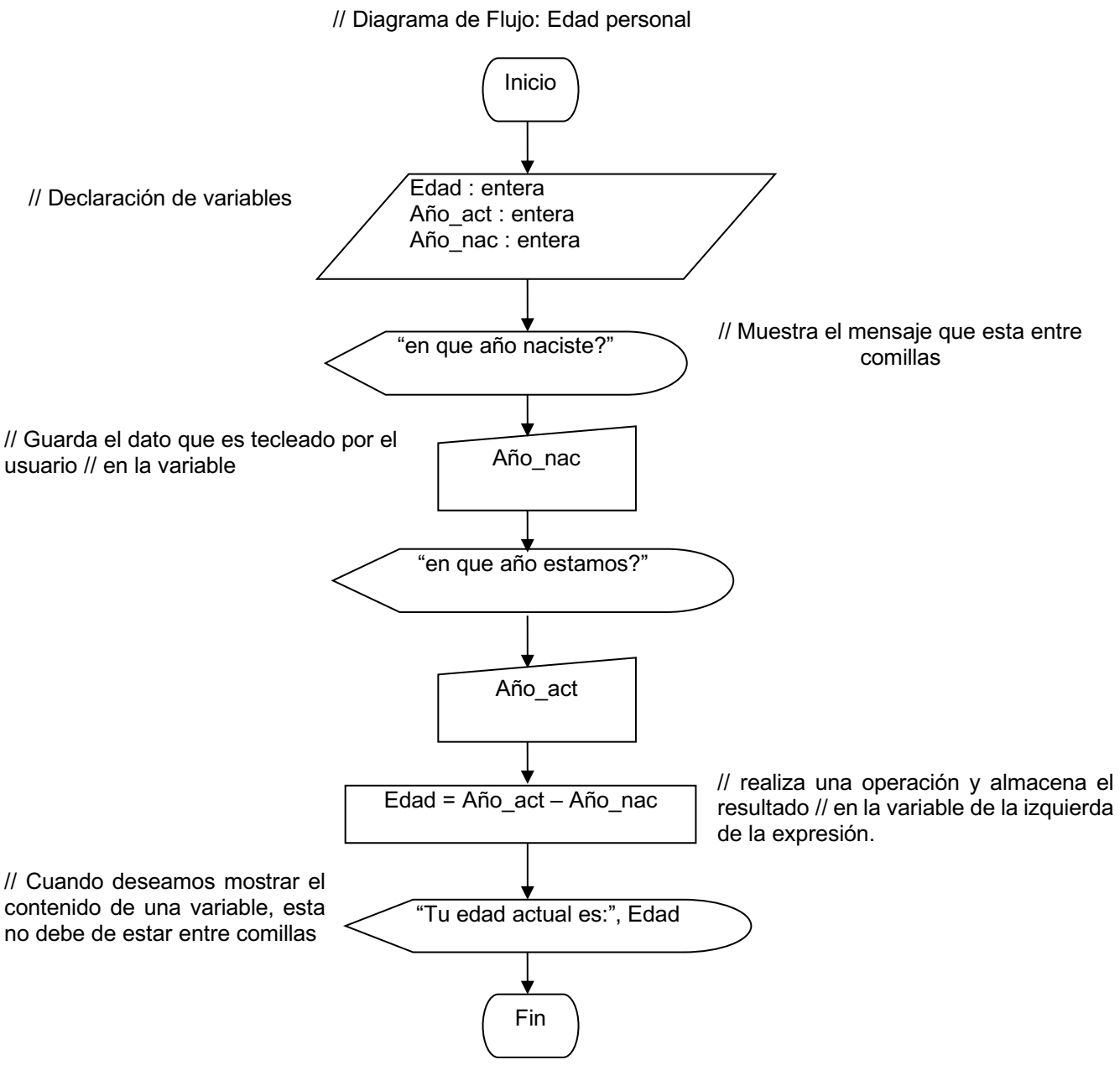
Se procede a realizar el algoritmo luego del análisis que indica la solución, por medio de las dos técnicas conocidas

PSEUDOCÓDIGO

Pseudocódigo: Edad personal

```
Variables:
Edad: entera
Año_nac: entera
Año_act: entera

Inicio
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
```

// 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: Edad personal
Inicio
// Declaración de variables
Edad : entera
Año_act : entera
Año_nac : entera
“en que año naciste?”
// Muestra el mensaje que esta entre comillas
// Guarda el dato que es tecleado por el usuario // en la variable
Año_nac
“en que año estamos?”
Año_act
Edad = Año_act – Año_nac
// realiza una operación y almacena el resultado // en la variable de la izquierda de la expresión.
// Cuando deseamos mostrar el contenido de una variable, esta no debe de estar entre comillas
“Tu edad actual es:”, Edad
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

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

Ejemplo 2

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

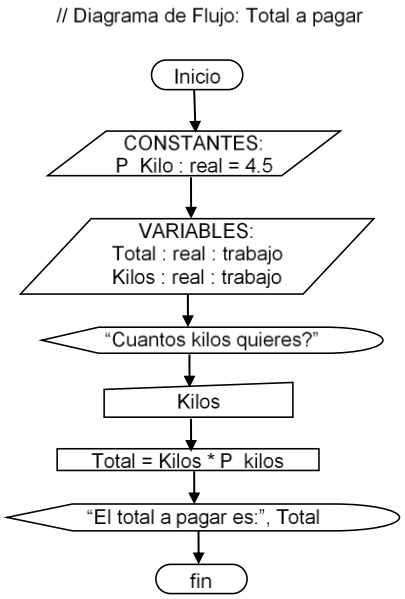
Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

Pseudocódigo: total a pagar
Constantes:
P_kilo: real = 4.5

Variables:
Total : real : trabajo
Kilos : real : trabajo
Inicio
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

Ejemplo 3

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

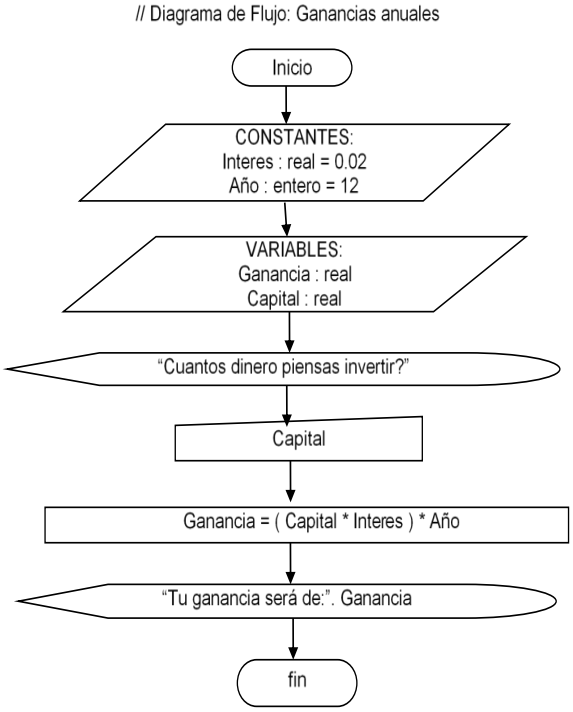
Paso I. Analizar el problema.

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

4.2.1 Ejercicios

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

4.3 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 (Joyanes, 2013).

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.

4.3.1 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.

En este momento analizaremos a las condiciones simples con las dos 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 sino; Pero si el resultado es falso, se ejecutan las instrucciones que se encuentran después de las palabras sino y las palabras finsi. Por lo cual podemos decir que los delimitadores de esta estructura son las palabra si ... entonces y finsi.

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 sino y finsi.

Al momento de diseñar un algoritmo con esta estructura se puede omitir el lado falso, es decir las instrucciones dentro del sino y el finsi; En caso de no desear hacer nada en esta parte (JOYANES, 1996).

```
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
```

Figura 13 Estructura condicional simple en pseudocódigo

En diagrama de flujo está 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.

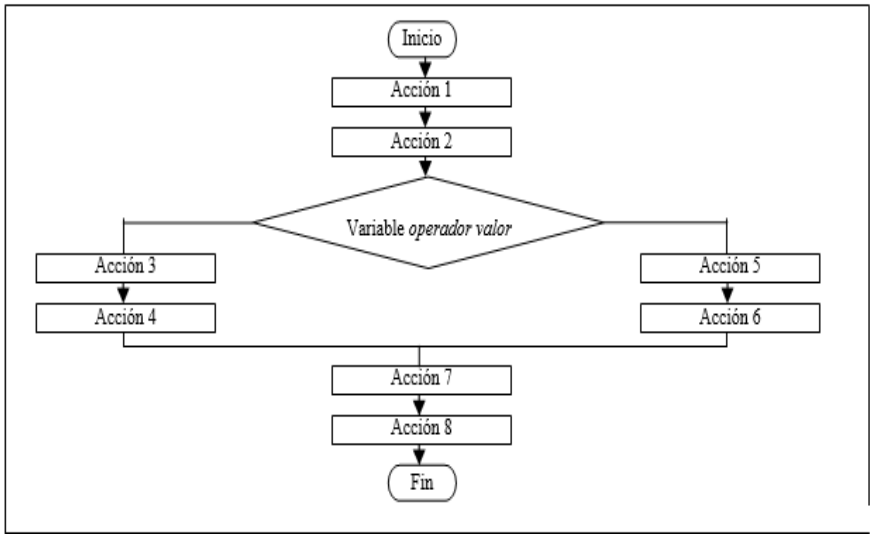


Figura 14 Estructura condicional Simple en diagrama de flujo

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 1

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

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

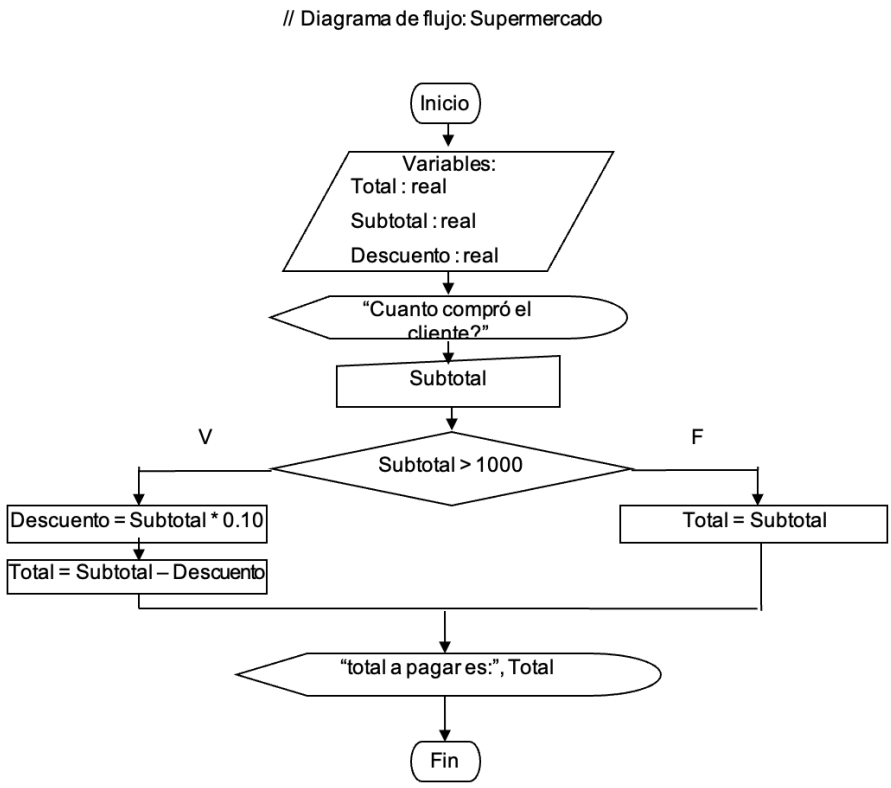
Pseudocódigo: Supermercado

Variables:
Total : real
Subtotal : real
Descuento : real
Inicio
Escribir “Cuanto compró el cliente?”
Leer Subtotal
Si Subtotal > 1000 entonces
Descuento = Subtotal * 0.10
Total = Subtotal – Descuento

sino
Total = Subtotal
finsi

Escribir “el total a pagar es:”, Total
Fin

DIAGRAMA DE FLUJO



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

Ejemplo 2

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)	▪ al1 ▪ al2 ▪ al3		Prom = (cal1 + cal2 + cal3) / 3 Cuando Prom < 6 “REPROBADO” Cuando Prom >= 6 “APROBADO”

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

Pseudocódigo: Promedio alumno

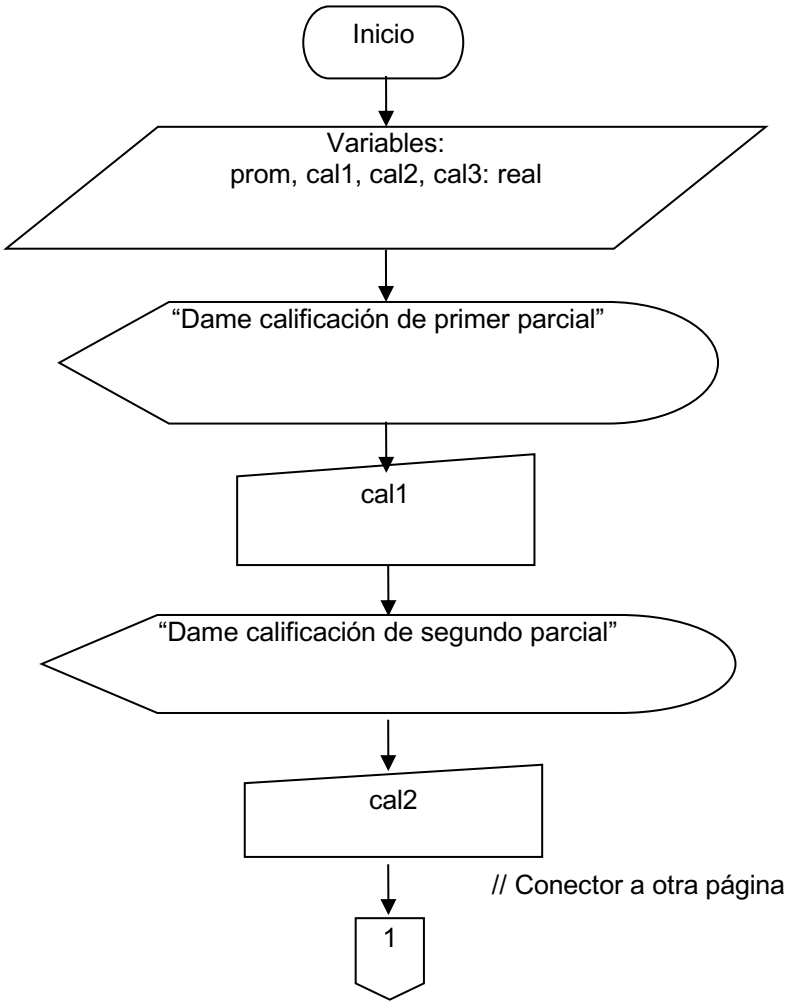
Variables: prom, cal1, cal2, cal3 : real

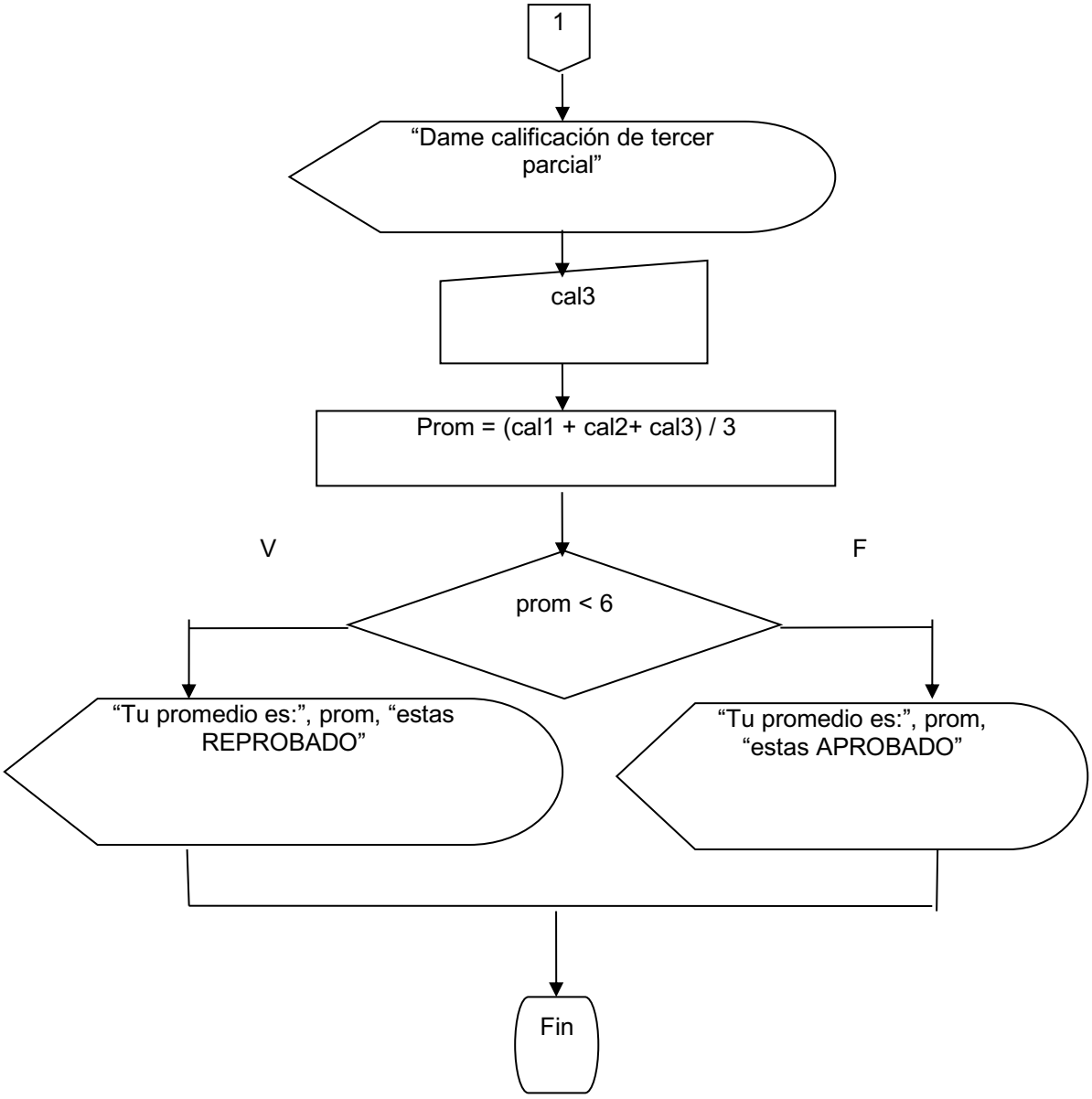
// La declaración de varias variables del mismo tipo se puede realizar siempre que el nombre de c/u este separado por una coma//

```
Inicio
 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
Si prom < 6 entonces
 Escribir “Tu promedio es:”, prom, “y estas REPROBADO”
Sino
 Escribir “Tu promedio es:”, prom, “y estas APROBADO”
Finsi

Fin
```

DIAGRAMA DE FLUJO

// Diagrama de flujo: Promedio




1
“Dame calificación de tercer parcial”
cal3
Prom = (cal1 + cal2+ cal3) / 3
V
F
prom < 6
“Tu promedio es:”, prom, “estas REPROBADO”
“Tu promedio es:”, prom, “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"

Ejemplo 3

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.

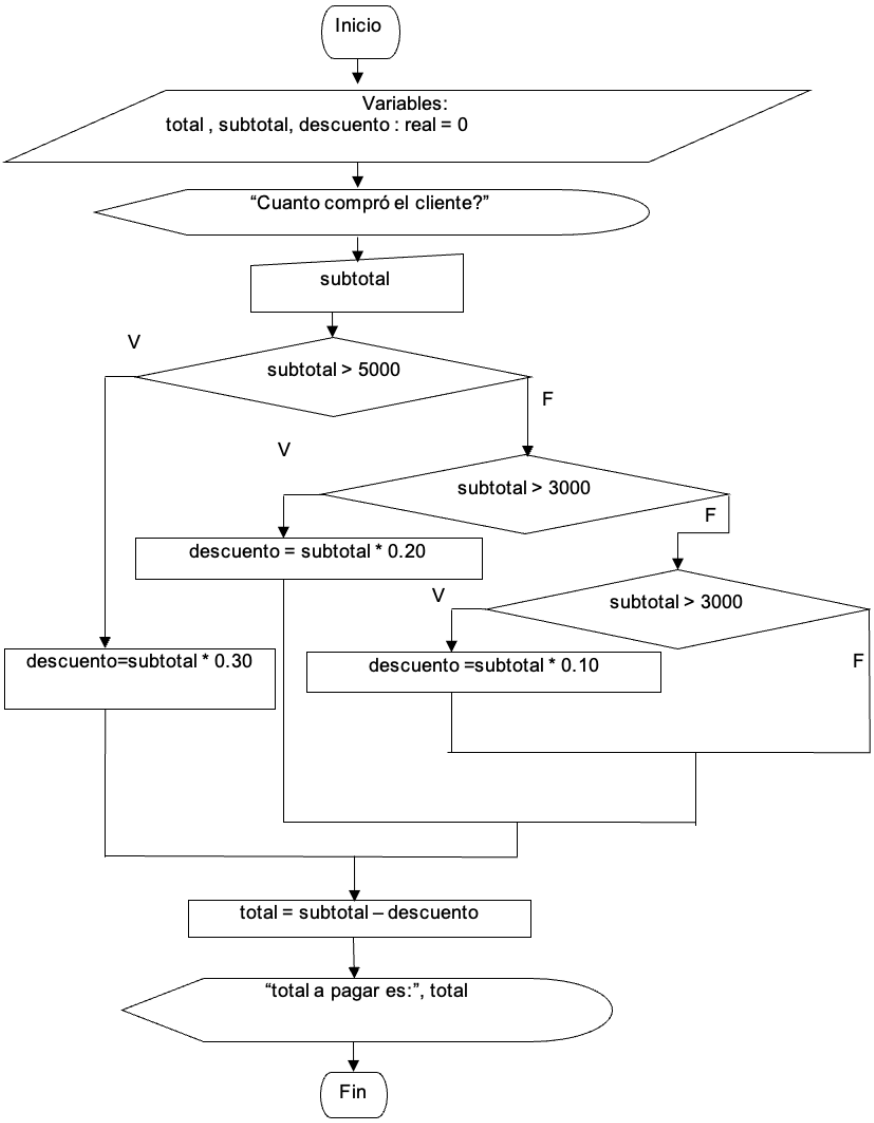
Paso I. Analizar el problema

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 = 0
// se inicializan todas las variables con cero
Inicio
   Escribir “Cuanto compró el cliente?”
   Leer subtotal
   si subtotal > 5000 entonces
descuento = subtotal * 0.30
sino
si subtotal > 3000 entonces
descuento = subtotal * 0.20
sino
si subtotal > 1000 entonces
descuento = subtotal * 0.10
            sino
       // no hace nada
            finsi
finsi
finsi
total = subtotal - descuento
Escribir “el total a pagar es:”, Total
Fin
```


Inicio
Variables:
total , subtotal, descuento : real = 0
“Cuanto compró el cliente?”
subtotal
V
subtotal > 5000
F
V
subtotal > 3000
F
descuento = subtotal * 0.20
V
subtotal > 3000
descuento=subtotal * 0.30
descuento =subtotal * 0.10
F
total = subtotal – descuento
“total a pagar es:”, total
Fin

4.3.2 Ejercicios

I. Escribe un algoritmo en pseudocódigo, diagrama de flujo 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 más 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 más y una antigüedad en su empleo de 25 años o más.
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.

4.3.3 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.

Las estructuras condicionales múltiples se analizan a continuación en las dos 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 (JOYANES, 1996).

```
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:
        Escribir " escribiste un 1"

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

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

     Fin casos
  5. Fin
```

Figura 15 Diseño que debe tener una estructura condicional múltiple

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.

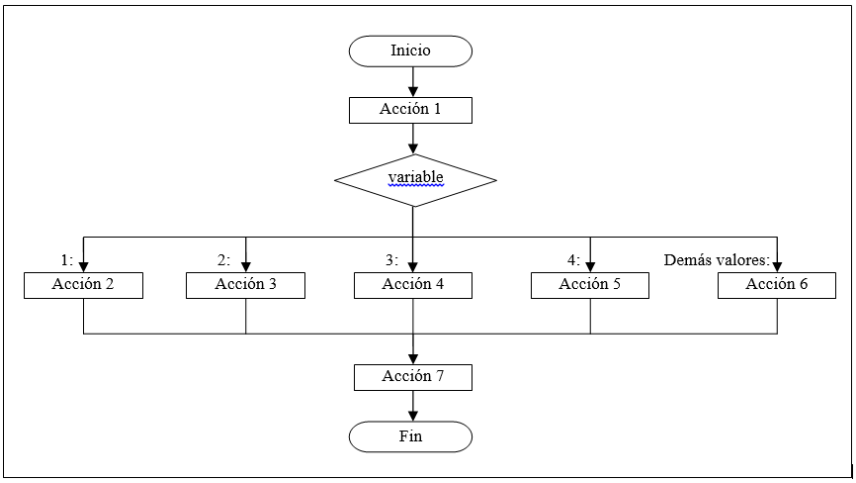


Figura 16 Diseño que debe de tener una estructura de selección múltiple.

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.

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 1

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 triángulo 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

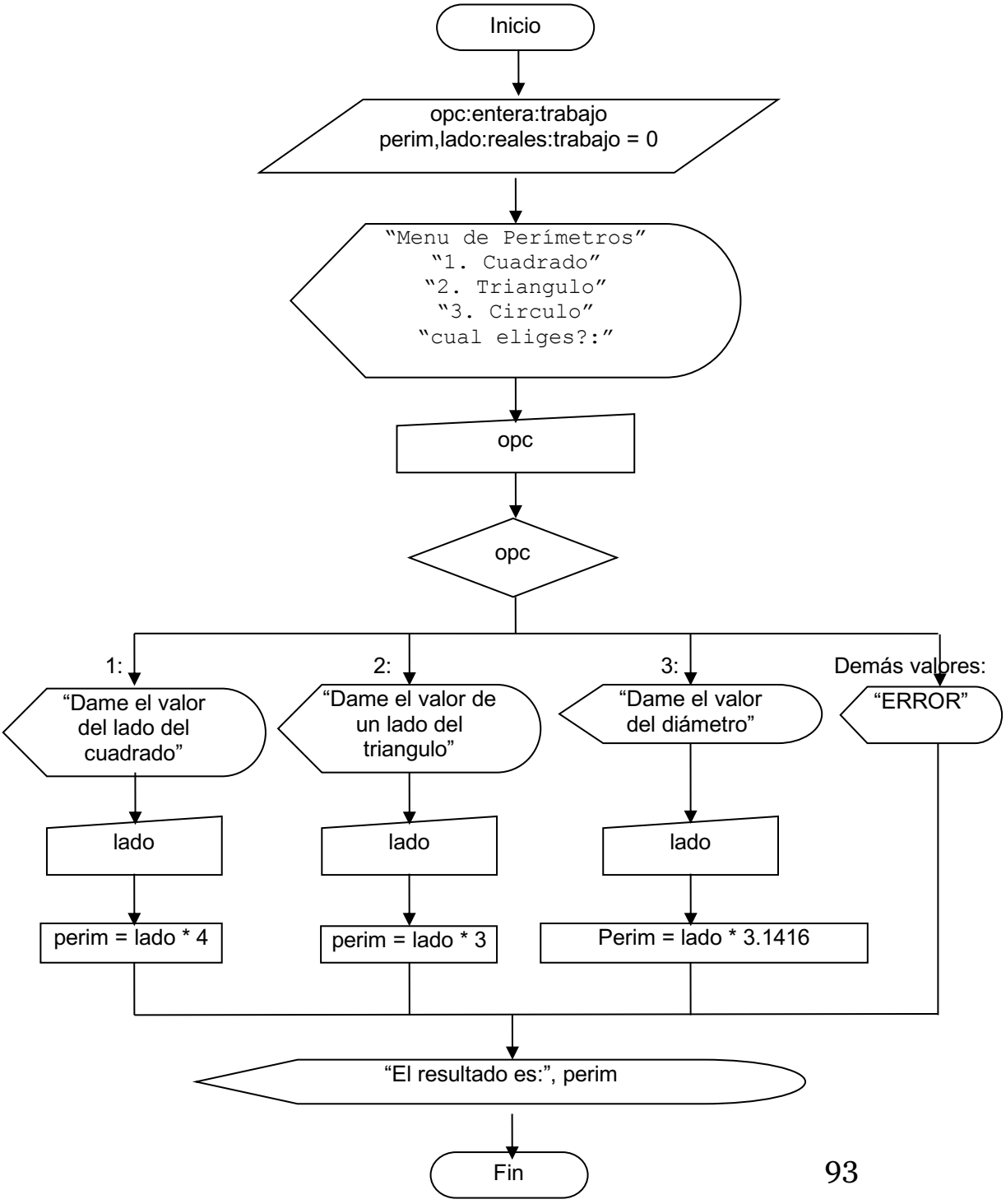
Pseudocódigo: menú perímetros

Variables:

opc : entera : trabajo
perim, lado : reales : trabajo = 0
Inicio
Escribir "Menu de Perímetros"
Escribir "1. Cuadrado"
Escribir "2. Triangulo"
Escribir "3. Circulo"
Escribir "cual eliges?:"
Leer opc
Casos para opc

cuando es igual a 1:
Escribir "dame el valor de un lado del cuadrado:"
Leer lado

```
perim = lado * 4

cuando es igual a 2:
Escribir “dame el valor de un lado del triangulo:”
Leer lado
perim = lado * 3

cuando es igual a 3:
Escribir “dame el valor del diámetro:”
Leer lado
perim = lado * 3.1416

para todos los demás valores:
Escribir “ERROR”

fincasos

Escribir “el resultado es:”, perim
Fin
```


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

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

Ejemplo 2

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

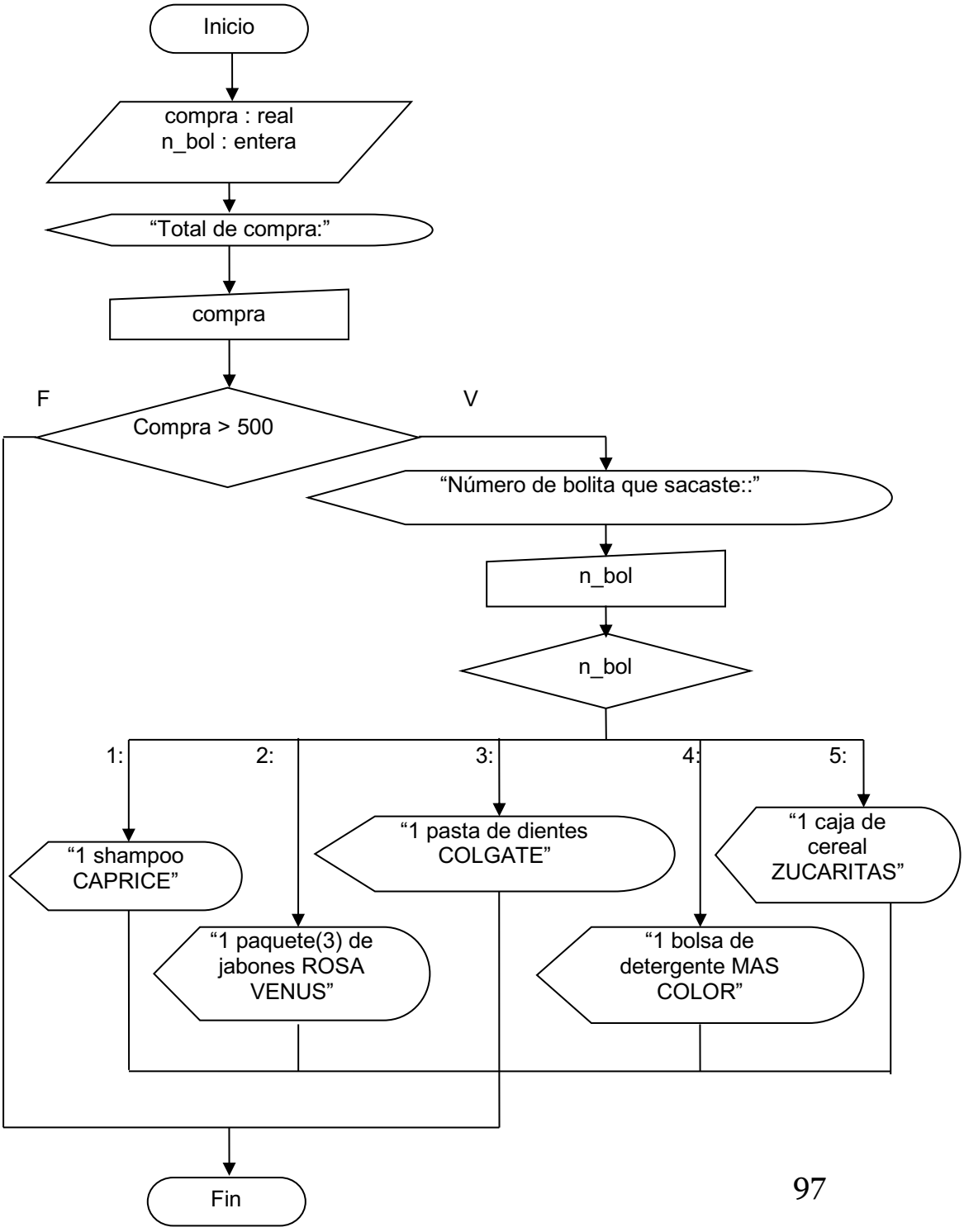
PSEUDOCÓDIGO

Pseudocódigo: Tómbola

```
Variables:
compra : real
n_bol : entera
Inicio
Escribir “Total de compra:”
Leer compra

si compra > 500 entonces
Escribir “Número de bolita que sacaste:”
Leer n_bol

Casos para n_bol
        Cuando es igual a 1:
        Escribir “1 shampoo CAPRICE”
        Cuando es igual a 2:
  Escribir “1 paquete(3) de jabones ROSA
  VENUS”
        Cuando es igual a 3:
        Escribir “1 pasta de dientes COLGATE”
        Cuando es igual a 4:
        Escribir “1 bolsa de detergente MAS COLOR”
        Cuando es igual a 5:
        Escribir “1 caja de cereal ZUCARITAS”
     Fin casos

sino
// no hace nada ya que no tiene derecho a sacar bolita
finsi
Fin
```

DIAGRAMA DE FLUJO

// Diagrama de flujo: Tómbola



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 3 == 3 → SI	"1 pasta de dientes COLGATE"
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	"1 caja de cereal ZUCARITAS"

	n_bol == 2 5 == 2 → NO n_bol == 3 5 == 3 → NO n_bol == 4 5 == 4 → NO n_bol == 5 5 == 5 → SI	
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	--

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

4.3.4 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

4.4 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 (JOYANES, 1996).

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.

4.4.1 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 (JOYANES, 1996).

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.

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.

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

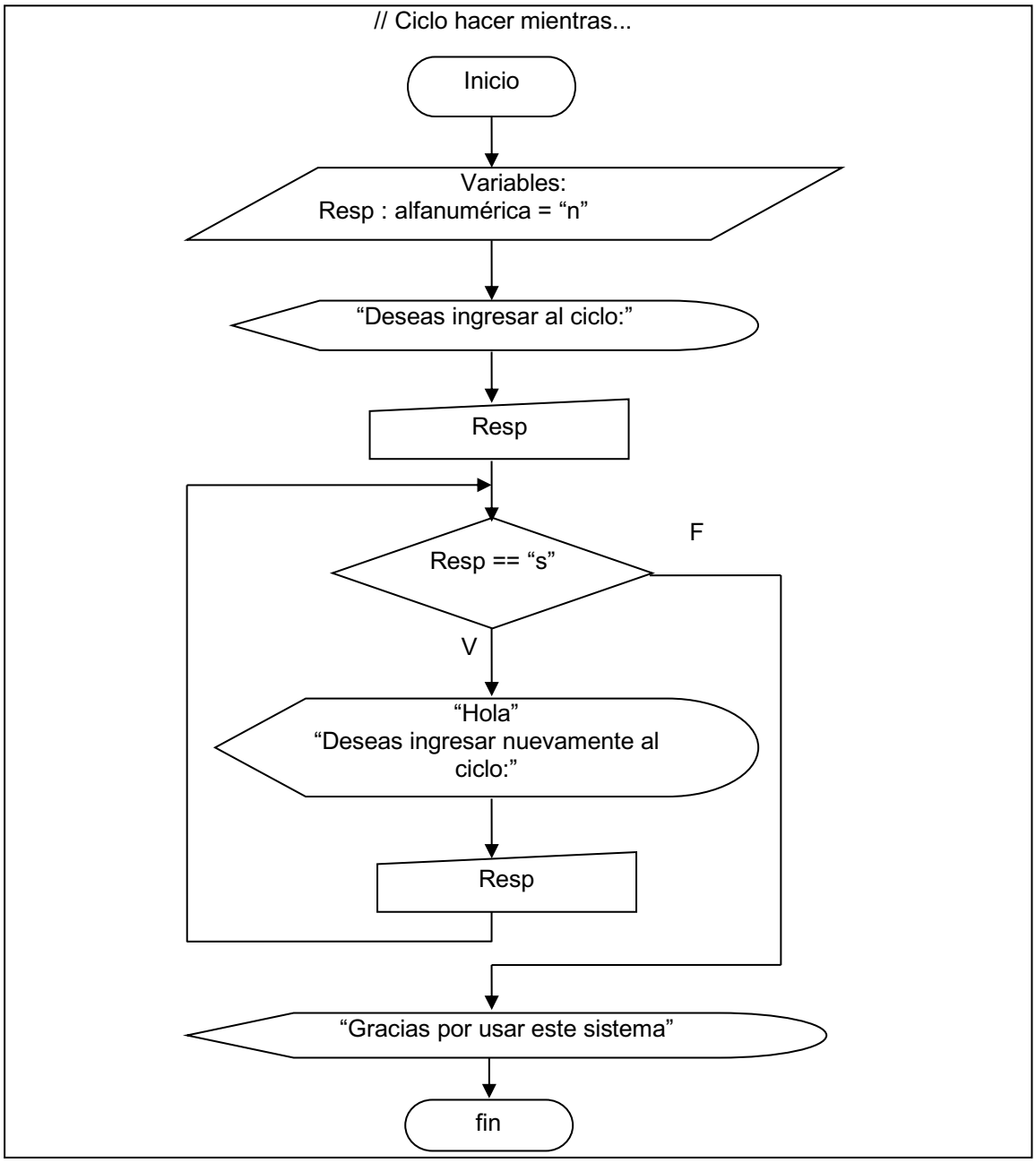
Pseudocódigo. En pseudocódigo se utilizan las instrucciones que hemos estado mencionando.

```
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"
        Escribir "hola"
        Escribir "deseas ingresar nuevamente al ciclo:"
        Leer Resp
     Fin mientras
  5. Escribir "Gracias por usar este sistema"
  6. FIN
```

Figura 17 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 (Casale J. y., 2014).


// Ciclo hacer mientras...
Inicio
Variables:
Resp : alfanumérica = “n”
“Deseas ingresar al ciclo:”
Resp
Resp == “s”
F
V
“Hola”
“Deseas ingresar nuevamente al ciclo:”
Resp
“Gracias por usar este sistema”
fin

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

Ejemplo 1

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

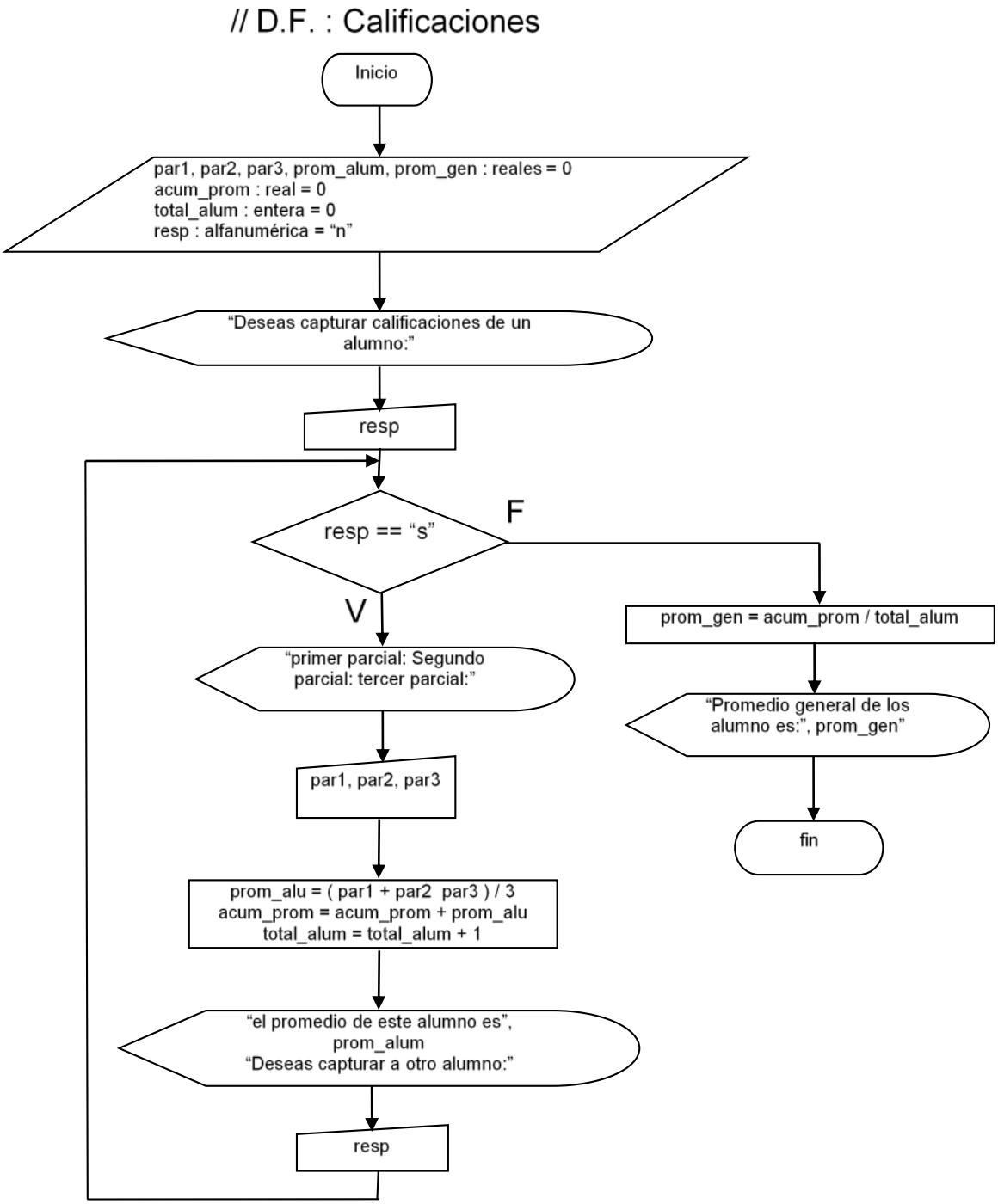
Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

Pseudocódigo: calificaciones

Variables:
par1, par2, par3, prom_alum, prom_gen : reales = 0
acum_prom : real = 0
total_alum : entera = 0
resp : alfanumérica = "n"
Inicio
Escribir "deseas capturar calificaciones de un alumno:"
Leer resp
Hacer mientras 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
fin mientras
prom_gen = acum_prom / total_alum
Escribir “Promedio general de los alumnos es:”, prom_gen
Fin
```


// D.F. : Calificaciones
Inicio
par1, par2, par3, prom_alum, prom_gen : reales = 0
acum_prom : real = 0
total_alum : entera = 0
resp : alfanumérica = "n"
"Deseas capturar calificaciones de un alumno:"
resp
resp == "s"
F
V
prom_gen = acum_prom / total_alum
"Promedio general de los alumno es:", prom_gen"
fin
"primer parcial: Segundo parcial: tercer parcial:"
par1, par2, par3
prom_alu = (par1 + par2 par3) / 3
acum_prom = acum_prom + prom_alu
total_alum = total_alum + 1
"el promedio de este alumno es", prom_alum
"Deseas capturar a otro alumno:"
resp

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 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
SEGUNDA CORRIDA (CONTINUACIÓN…)

```
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
```

Ejemplo 2

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.

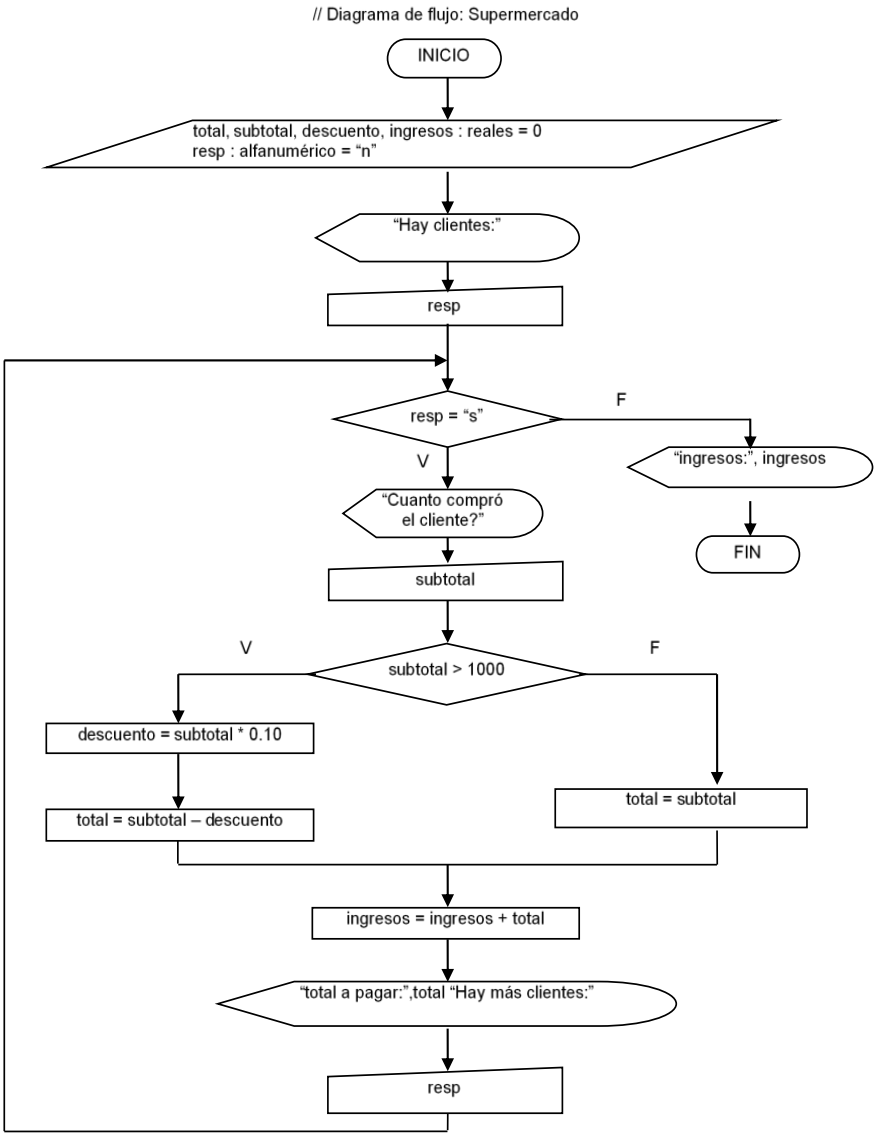
Paso I. Analizar el problema.

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”
Inicio
  Escribir “hay clientes en la tienda”
  Leer resp

Hacer mientras resp == “s”

Escribir “cuanto compró el cliente?”
Leer subtotal

Si subtotal > 1000 entonces

descuento = subtotal * 0.10
total = subtotal – descuento

sino

total = subtotal

finsi

ingresos = ingresos + total
Escribir “el total a pagar es:”, total
Escribir “Hay más clientes en la tienda:”
Leer resp

fin mientras

Escribir “ingresos:”, ingresos
Fin
```


// Diagrama de flujo: Supermercado
INICIO
total, subtotal, descuento, ingresos : reales = 0
resp : alfanumérico = “n”
“Hay clientes:”
resp
resp = “s”
F
V
“ingresos:”, ingresos
FIN
“Cuanto compró el cliente?”
subtotal
V
subtotal > 1000
F
descuento = subtotal * 0.10
total = subtotal – descuento
total = subtotal
ingresos = ingresos + total
“total a pagar:”,total “Hay más clientes:”
resp

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

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

4.4.2 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.

4.4.3 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 (Casale J. , 2012).

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.

Se recomienda que la variable a ser evaluada sea inicializada con un valor que permita romper la estructura, para evitar tener un ciclo infinito (Casale J. y., 2014).

A continuación vamos a esquematizar el diseño básico de esta estructura en las dos 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
        Escribir "hola"
        Escribir "deseas ingresar nuevamente al ciclo:"
        Leer Resp
     Hasta Resp == "n"
  3. Escribir "Gracias por usar este sistema"
  4. FIN
```

Figura 18 Pseudocódigo básico del ciclo Repetir / Hasta...
(Joyanes, 2013)

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.

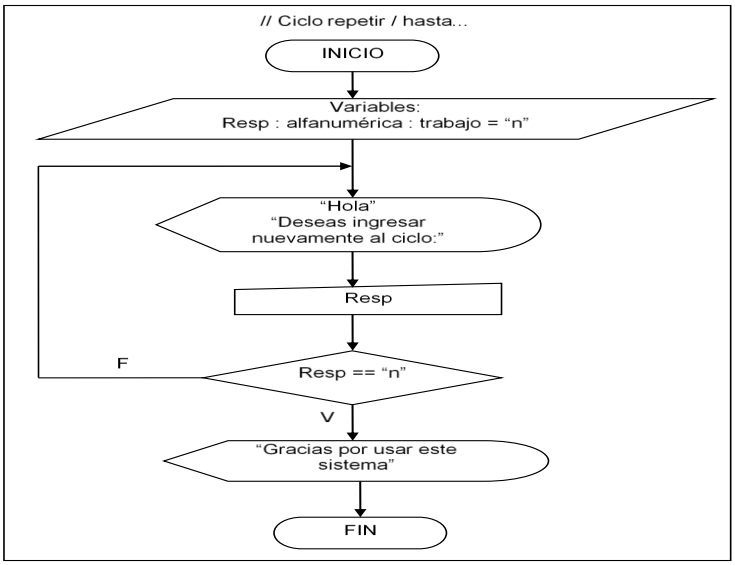


Figura 19 Diagrama de flujo del ciclo Repetir / hasta ...

(Joyanes, 2013)

Ejemplo 1

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

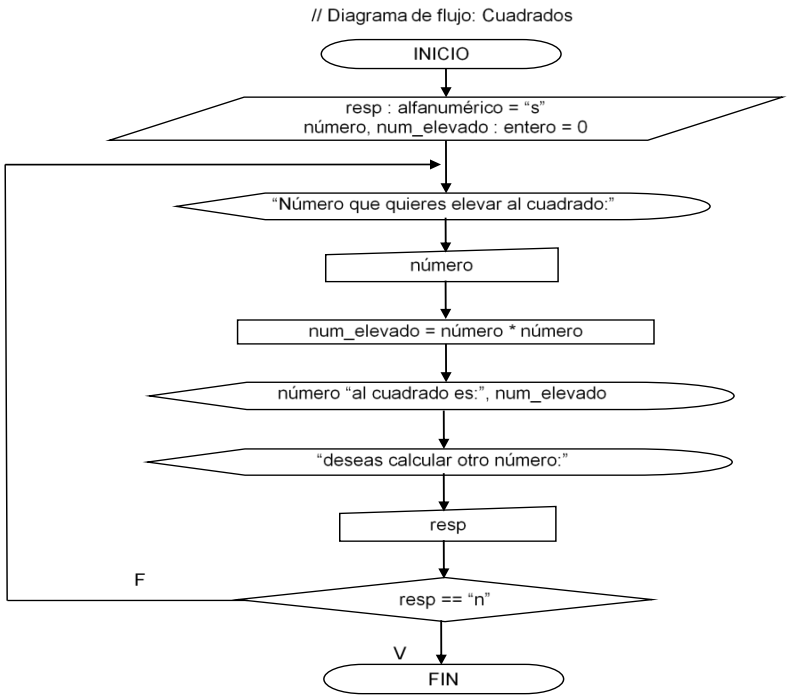
PSEUDOCÓDIGO

Pseudocódigo: Cuadrados

Variables:
resp : alfanumérico = "s"

número, num_elevado : entero = 0
Inicio
Repetir
  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
Hasta resp == “n”
Fin

DIAGRAMA DE FLUJO



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

En este ejemplo, como pudimos observar en la primera corrida de escritorio se ejecutó una vez aun cuando la respuesta a la condición fue verdadera, cosa que no ocurre con el ciclo hacer mientras…

Ejemplo 2

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.

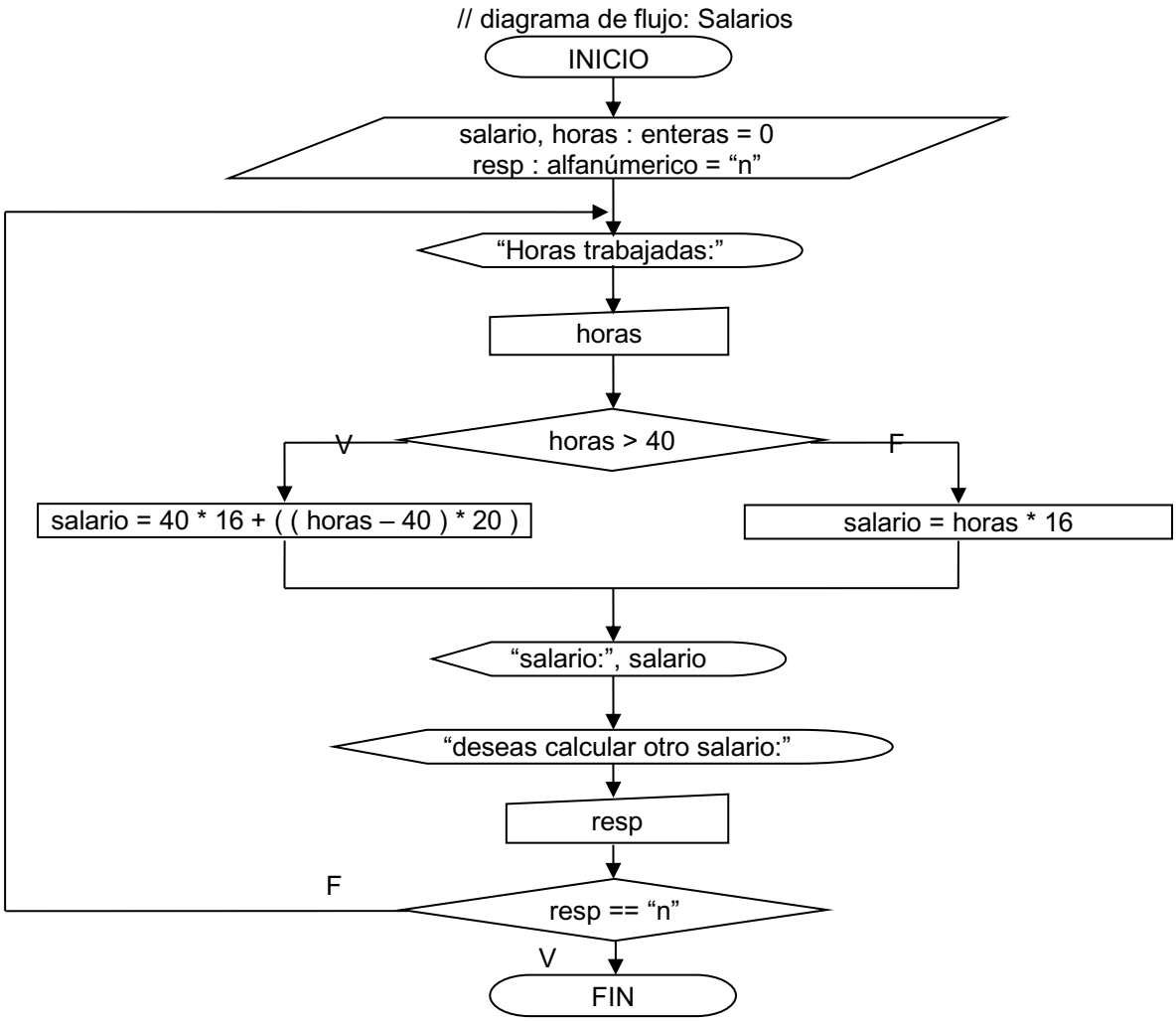
Paso I. Analizar el problema.

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”
Inicio
 repetir
  Escribir “Horas trabajadas:”
  Leer horas
  si horas > 40 entonces
   salario = 40 * 16 +( ( horas – 40 ) * 20 )
  si no
   salario = horas * 16
  fin si
  Escribir “salario:”,salario
  Escribir “deseas calcular otro salario:”
  Leer resp
 Hasta resp == “n”
Fin
```


// diagrama de flujo: Salarios
INICIO
salario, horas : enteras = 0
resp : alfanúmerico = “n”
“Horas trabajadas:”
horas
horas > 40
V
F
salario = 40 * 16 + ((horas – 40) * 20)
salario = horas * 16
“salario:”, salario
“deseas calcular otro salario:”
resp
F
resp == “n”
V
FIN

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

4.4.4 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.

4.4.5 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.

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

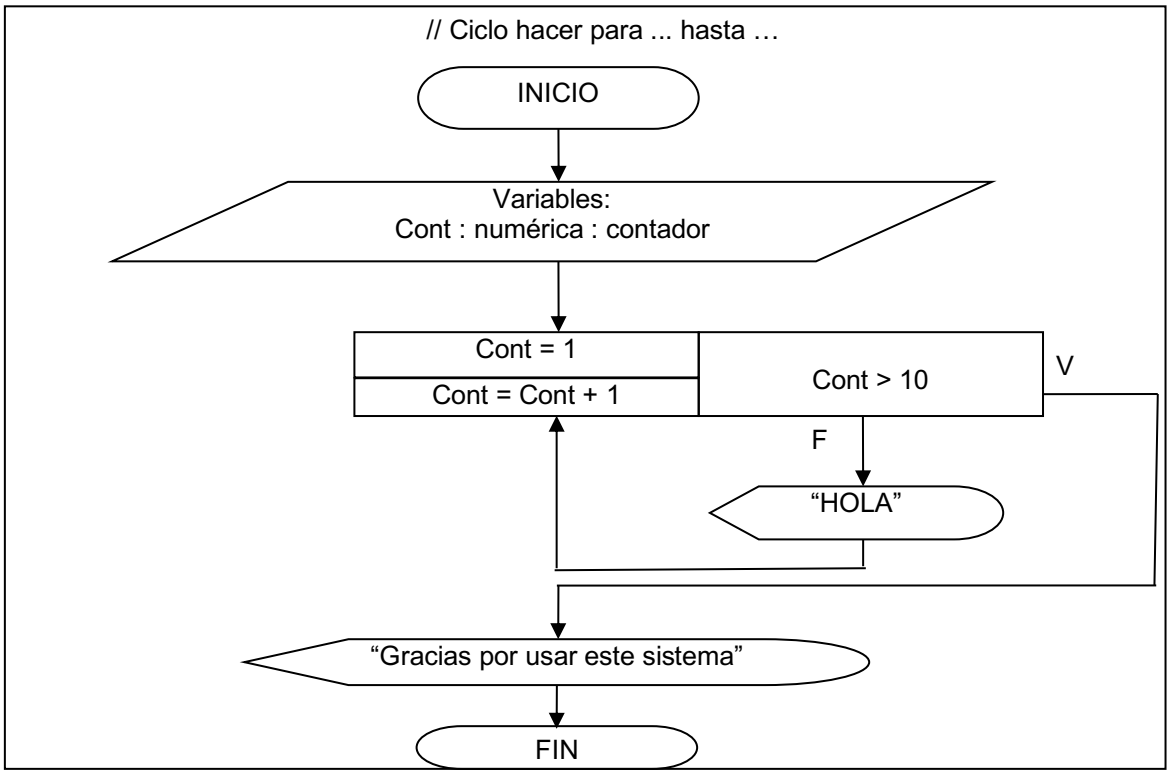
Pseudocódigo. En pseudocódigo se utilizan las instrucciones que hemos estado mencionando.

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

Figura 20 Diseño básico del ciclo hacer para … hasta …

(Casale J. , 2012)

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 (Casale J. y., 2014).



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

Ejemplo 1

Se necesita un sistema que despliega una tabla de multiplicar de un número dado por el usuario.

Paso I. Analizar el problema.

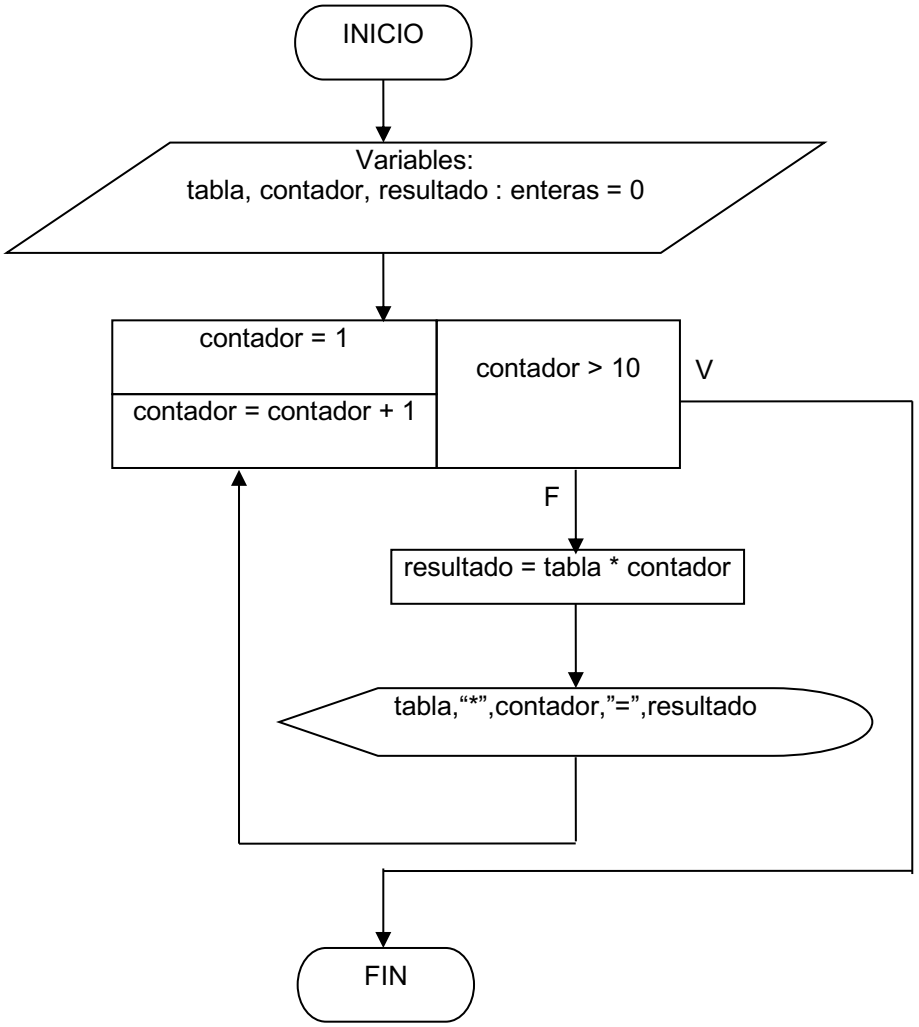
Salidas	Entrada	Procesos
▪ resultado	▪ tabla	Hacer para contador = 1 hasta contador > 10 resultado = tabla * contador contador = contador + 1

Paso II. Diseñar El algoritmo

Pseudocódigo: Tabla

```
Variables:
tabla, contador, resultado : enteras = 0
Inicio
    Escribir “tabla que deseas visualizar”
    Leer tabla
    Hacer para contador = 1 hasta contador > 10
    resultado = tabla * contador
    Escribir tabla, “*”, contador, “=”, resultado
    contador = contador + 1
Fin para
Fin
```

// Diagrama de flujo: Tabla



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

Ejemplo 2

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.

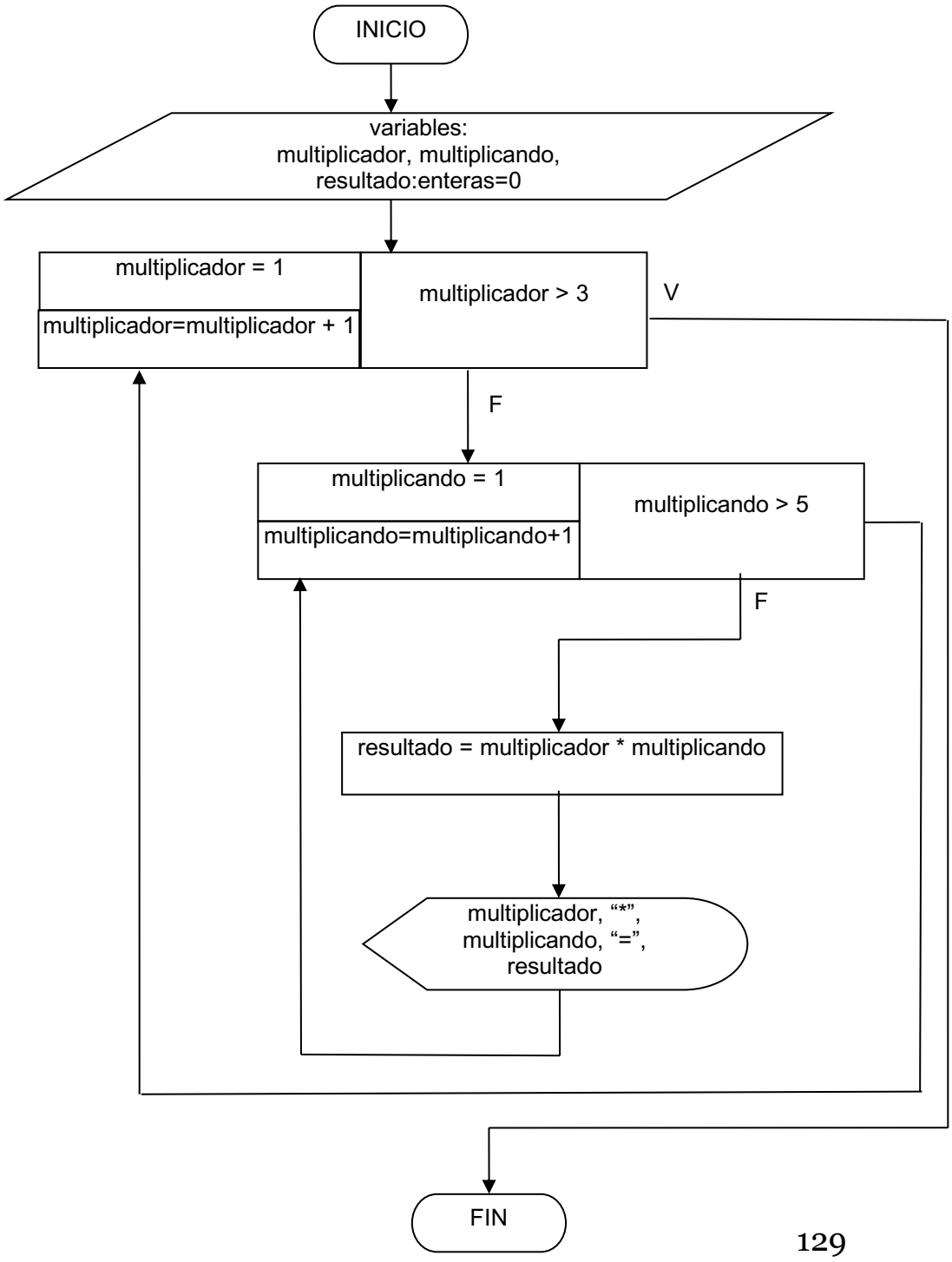
Salidas	Procesos
resultado	Hacer para multip =1 hasta multip >3 Hacer para multpdo = 1 hasta multpdo > 5 resultado = multip * multpdo escribir multip, "*", multpdo, "=", resultado multpdo = multpdo + 1

Paso II. Diseñar El algoritmo

PSEUDOCÓDIGO

```
pseudocódigo: tablas
variables:
multip, multpdo, resultado : enteras = 0
Inicio
Hacer para multip = 1 hasta multip > 3
Hacer para multpdo = 1 hasta multpdo > 5
  resultado = multip * multpdo
  escribir multip, "*", multpdo, "=", resultado
  multpdo = multipdo + 1
Fin para
  multip = multip + 1
Fin para

Fin
```

// Diagrama de flujo: Tablas



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

4.4.6 Ejercicios.

I. Diseña un algoritmo con la estructura Hacer para .. hasta .. utilizando las dos 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

4.5 Estructuras de control en Pseint

Condicional simple

Sintaxis

```
Si <condicion>
  Entonces
    ...
  Sino
    ...
FinSi
```

Evalúa la condición y ejecuta las acciones que correspondan; Entonces si es verdadera, Sino si es falsa. Siempre debe estar presente Entonces, pero puede no aparecer Sino, y así no hacer nada en caso de que la condición sea falsa. La condición puede ser una variable o una expresión lógica de tipo lógica (Novara, 2020).

Ejemplos :

```
Condicion<- A>B;
Si Condicion
  Entonces Escribir 'Si';
  Sino Escribir 'No';
FinSi

Si Cantidad > 10
  Entonces
    Escribir 'Demasiados articulos.';
    Escribir 'Solo se consideraran los 10 primeros.';
FinSi
```

Condicional múltiple

Sintaxis

```
Segun <variable> Hacer
  A: <...>;
  B,C: <...>;
...
  De Otro Modo: <...>;
FinSegun
```

Permite elegir entre dos o más posibles grupos de acciones. Evalúa el contenido de la variable y selecciona el camino a seguir. La variable debe ser de tipo numérico. Las opciones se delimitan por los dos puntos al final. Si un grupo de acciones se debe ejecutar en dos o más casos, los valores se pueden poner separados por comas en la misma línea. La opción final puede ser De Otro Modo, y se ejecuta si la variable no coincide con ninguna de las anteriores (Novara, 2020).

Ejemplo:

```
Leer a;
Segun a Hacer
1: Escribir 1;
2: Escribir 2;
3,4: Escribir '3 o 4';
De Otro Modo:
Escribir 'No esta entre 1 y 4';
FinSegun
```

Cíclicas

Ciclo Mientras Hacer

Sintaxis

```
Mientras <condicion> Hacer
  <...>
FinMientras
```

Ejecuta un grupo de acciones mientras la condición sea verdadera. Puede que no se ejecuten nunca, si al momento de entrar en la estructura la condición ya es falsa. Debe contener dentro alguna acción

que pueda modificar la condición para salir (Novara, 2020).

Ejemplo:

```
Escribir 'Ingrese un numero o 0 para salir';
Leer a;
Mientras a<>0 Hacer
Tot<-Tot+a;
Escribir 'Ingrese otro numero o 0 para salir';
FinMientras
Escribir 'Total:',Tot;
```

Ciclo Repetir Hasta Que

Sintaxis

```
Repetir
  <...>
Hasta Que <condicion>
```

Ejecuta un grupo de acciones hasta que la condición sea verdadera. Como la condición se evalúa al final, siempre las acciones serán ejecutadas al menos una vez. Debe contener dentro alguna acción que pueda modificar la condición para salir (Novara, 2020).
Ejemplo:

```
Repetir
Tot<-Tot+a;
Escribir 'Ingrese un numero o 0 para salir';
Hasta Que a=0
Escribir 'Total:',Tot;
```

Ciclo Para
Sintaxis

```
Para <I> <- <VI> Hasta <VF> ( Con Paso <P> ) Hacer
  <...>
FinPara
```

Ejecuta un bloque de instrucciones un determinado número de veces. Al ingresar al bloque, la variable <I > recibe el valor <VI > y se ejecutan las instrucciones. Luego incrementa la variable <I > en <P> se evalúa si <I > supero a <VF>. Si esto es falso se repite hasta que <I > supere a <VF>. Si se omite el paso (Con Paso <P>), la variable <I > se incrementara en 1 (Novara, 2020).

Ejemplo:

```
Escribir 'Numeros pares de 10 a 20:';
Para a<-10 Hasta 20 Con Paso 2 Hacer
Escribir a;
FinPara
```

4.5.1 Ejemplos usando estructuras de control en Pseint

A modo de práctica codifique y ejecute estos algoritmos utilizando el programa Pseint

```
// CALIFICACIONES.PSC
Proceso calificaciones
	Escribir "ingrese la calificacion";
	leer n;
	Si n<15 Entonces
		si n<11 entonces
			si n<10 entonces
				escribir "insuficiente";
			Sino
				escribir "suficiente";
			fin si
		Sino
			escribir "Bien";
		fin si
	Sino
		si n<18 Entonces
			escribir "notable";
		Sino
			escribir "sobresaliente";
```

```
                fin si
        FinSi
FinProceso

// DIAS_SEMANA.PSC
Proceso dia_de_la_semana
        Escribir "escribir el numero de dia";
        leer n;
        segun n Hacer
                1:  escribir "lunes";
                2:  escribir "martes";
                3:  escribir "miercoles";
                4:  escribir "jueves";
                5:  escribir "viernes";
                6:  escribir "sabado";
                7:  escribir "domingo";
                de Otro Modo:
                        escribir "valor incorrecto";
                        fin segun
FinProceso

// SE DESEA OBTENER 10 PRIMEROS MULTIPLOS DE 8
Proceso serie_8
        escribir "serie del 8";
        sn<-8;
        Mientras sn<=80 Hacer
                escribir sn;
                sn<-sn+8;
        FinMientras
FinProceso

// MUESTRE LOS CINCO PRIMEROS NÚMEROS Y CALCULE SU SUMA
Proceso muestre_cinco_primeros_numeros_y_su_suma
        NUM<-0;
        SUM<-0;
        Repetir
                NUM<-NUM+1;
                ESCRIBIR NUM;
                SUM<-SUM+NUM;
```

```
        Hasta Que NUM>=5
        ESCRIBIR "LA SUMA ES:",SUM;
FinProceso

// INGRESE LAS 5 NOTAS DE UN ESTUDIANTES, CALCULE SU
PROMEDIO
Proceso NOTAS_PROMEDIO_PARA
        A<-0;
        ESCRIBIR "INGRESE LAS NOTAS POR FAVOR";
        Para I<-1 Hasta 5 Hacer
                LEER N;
                A<-A+N;
        FinPara
        P<-A/5;
        ESCRIBIR "EL PROMEDIO ES",P;
FinProceso
```

5. ARREGLOS Y ESTRUCTURAS

5.1 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 (JOYANES, 1996).

5.2 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 (Joyanes, 2013).

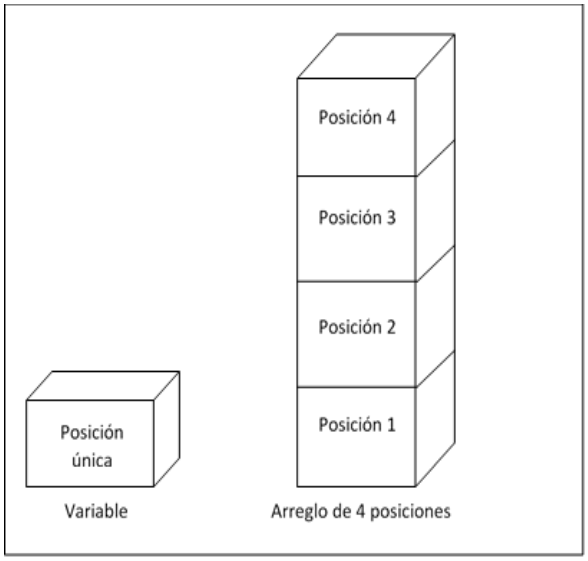


Figura 21 Representación y comparación de una variable y un arreglo

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 (Joyanes, 2013).

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

Figura 22 Forma en que se declaran los arreglos

Al declarar un arreglo, se le pueden dar valores de inicio para cada una de sus posiciones, para 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.

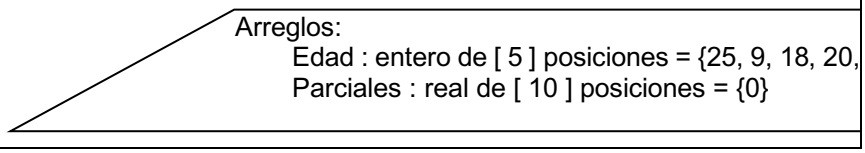
PSEUDOCÓDIGO
Arreglos: Edad : entero de [5] posiciones = {25, 9, 18, 20, 31} Parciales : real de [10] posiciones = {0} // si se coloca solo un cero todas las posiciones toman este valor
DIAGRAMA DE FLUJO


Figura 23 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.

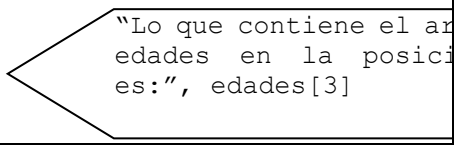
.PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Escribir "lo que contiene el arreglo edades en la posición 3 es:", edades[3]	

Figura 24 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	DIAGRAMA DE FLUJO
Leer edades[3] // almacena una dato en la posición 3 // del arreglo edades	

Figura 25 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.

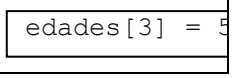
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
edades[3] = 5 * 10 // almacena un dato en la posición 3 del arreglo	

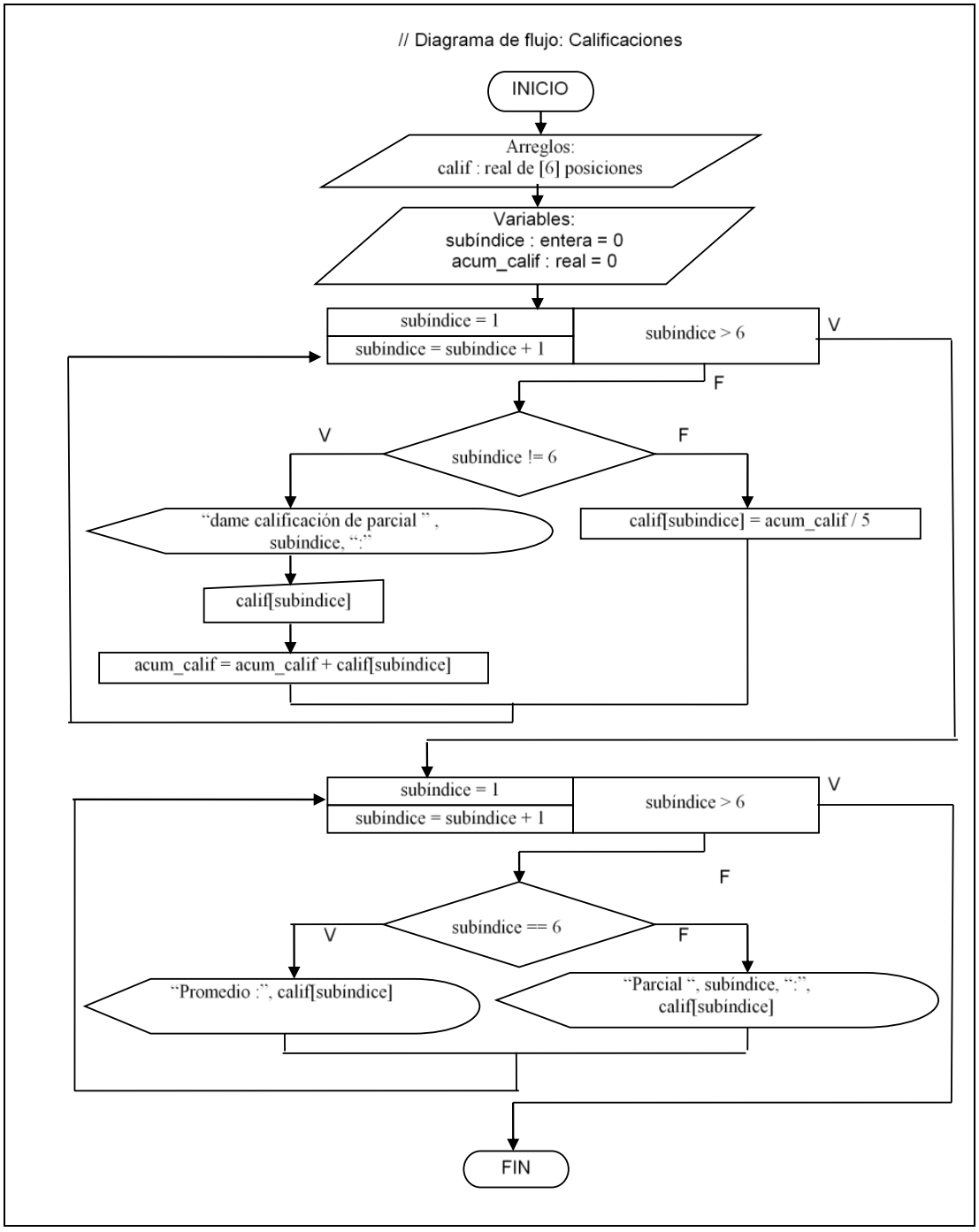
Figura 26 Forma en que se asignan datos a un arreglo

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 (Joyanes, 2013). 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 1
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 Si subíndice != 6 Escribir “dame calificación de parcial ”, subíndice, “:” Leer calif[subíndice] acum_calif = acum_calif + calif[subíndice] Si no calif[subindice] = acum_calif / 5 Fin si subíndice = subíndice + 1 Fin para 3. Hacer para subíndice = 1 hasta subíndice > 6 Si subíndice == 6 entonces Escribir “Promedio : “, calif[subíndice] Si no Escribir “Parcial “, subíndice, “:”, calif[subíndice] Fin si subíndice = subíndice + 1 Fin para 4. Fin


// Diagrama de flujo: Calificaciones
INICIO
Arreglos:
calif : real de [6] posiciones
Variables:
subíndice : entera = 0
acum_calif : real = 0
subíndice = 1
subíndice = subíndice + 1
subíndice > 6
V
F
subíndice != 6
V
F
"dame calificación de parcial " ,
subíndice, ":"
calif[subindice]
acum_calif = acum_calif + calif[subíndice]
calif[subindice] = acum_calif / 5
subíndice = 1
subíndice = subíndice + 1
subíndice > 6
V
F
subíndice == 6
V
F
"Promedio :", calif[subíndice]
"Parcial ", subindice, ":",
calif[subindice]
FIN

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

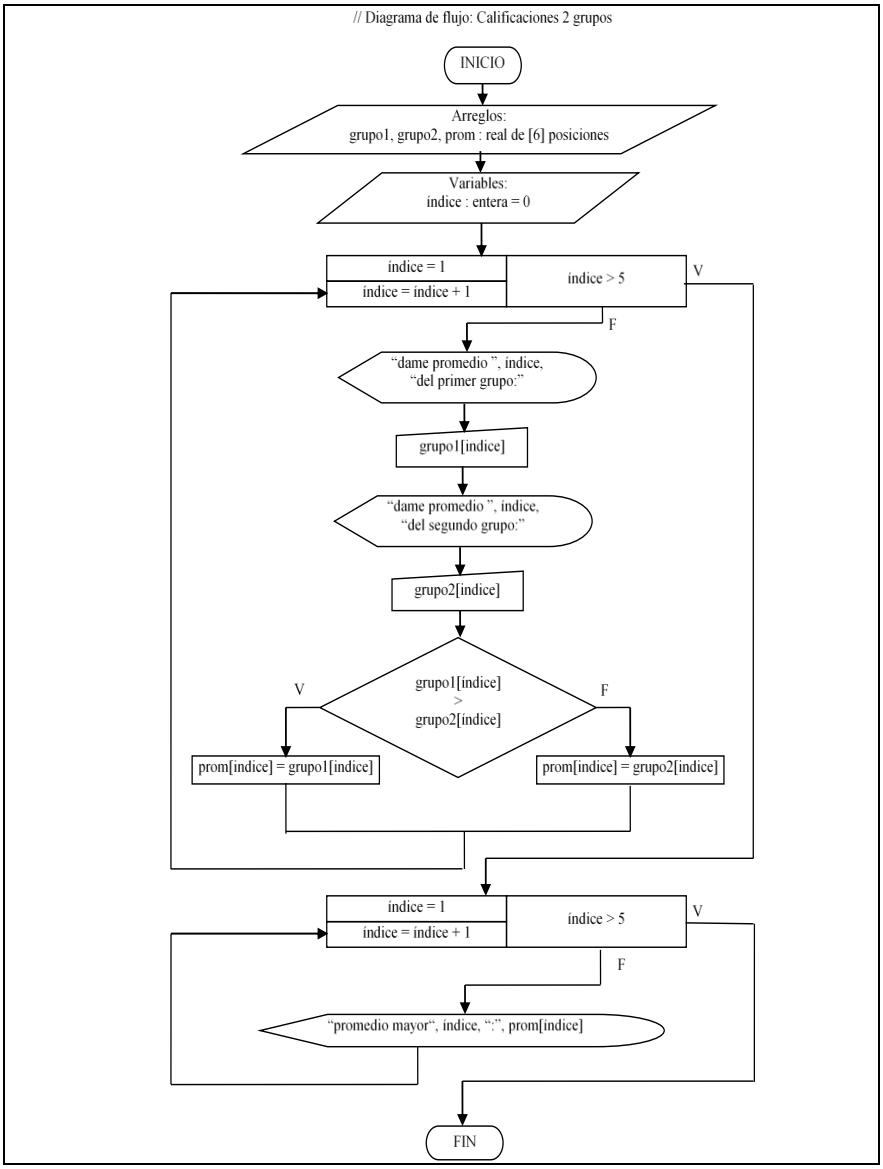
A continuación vamos a ver en el siguiente ejemplo que con un mismo subíndice se puede acceder a varios arreglos.

Ejemplo 2

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 Escribir “dame promedio ”, índice, “del primer grupo:” Leer grupo1[índice] Escribir “dame promedio ”, índice, “del segundo grupo:” Leer grupo2[índice] Si grupo1[índice] > grupo2[índice] entonces prom[indice] = grupo1[índice] Si no prom[indice] = grupo2[índice] Fin si índice = 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 de flujo: Calificaciones 2 grupos
INICIO
Arreglos:
grupo1, grupo2, prom : real de [6] posiciones
Variables:
índice : entera = 0
índice = 1
índice = índice + 1
índice > 5
V
F
"dame promedio ", índice,
"del primer grupo:"
grupo1[índice]
"dame promedio ", índice,
"del segundo grupo:"
grupo2[índice]
grupo1[índice]
>
grupo2[índice]
V
F
prom[índice] = grupo1[índice]
prom[índice] = grupo2[índice]
índice = 1
índice = índice + 1
índice > 5
V
F
"promedio mayor", índice, ":", prom[índice]
FIN

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 24 *Ejemplo 2 del manejo de arreglos*

5.2.1 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.

5.2.2 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 (Joyanes, 2013).

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:

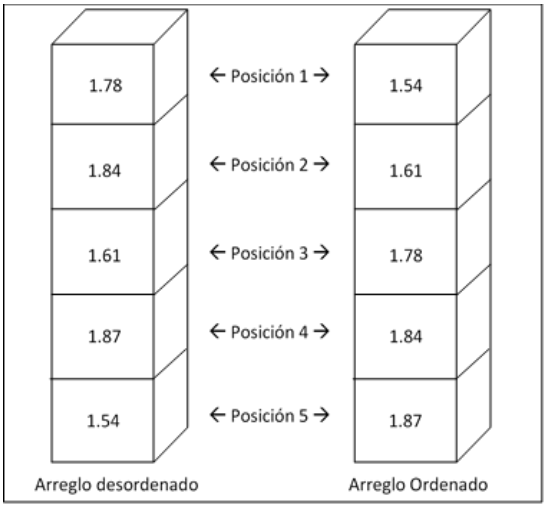


Figura 27 Vista de un arreglo llamado edades ordenado

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):

Tabla 25 *Instrucciones para ordenar el arreglo edades*

```
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
```

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 (JOYANES, 1996).

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[1]	**1.78**
edades[2]	**1.84**
edades[3]	**1.61**

➔

edades[1]	**1.78**
edades[2]	**1.61**
edades[3]	**1.84**

edades[posición] > edades[posición + 1]	**V**
Temporal=	**1.84**
Edades[posición] =	**1.61**
Edades[posición + 1] =	**1.84**

edades[4]	**1.87**
edades[5]	**1.54**

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 está en la posición 5 contra lo que está 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[1]	**1.61**
edades[2]	**1.78**
edades[3]	**1.84**

➔

edades[1]	**1.61**
edades[2]	**1.78**
edades[3]	**1.84**

edades[posición] > edades[posición + 1]	F
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[4]	1.54
edades[5]	1.87

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[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

Edades[posición] =	
Edades[posición + 1] =	

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[1]	1.61
edades[2]	1.54
edades[3]	1.78

➔

edades[1]	1.61
edades[2]	1.54
edades[3]	1.78

edades[posición] > edades[posición + 1]	**F**
Temporal=	
Edades[posición] =	
Edades[posición + 1] =	

edades[4]	**1.84**
edades[5]	**1.87**

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 está 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 (Casale J. , 2012).

5.2.3 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 (Casale J. , 2012).

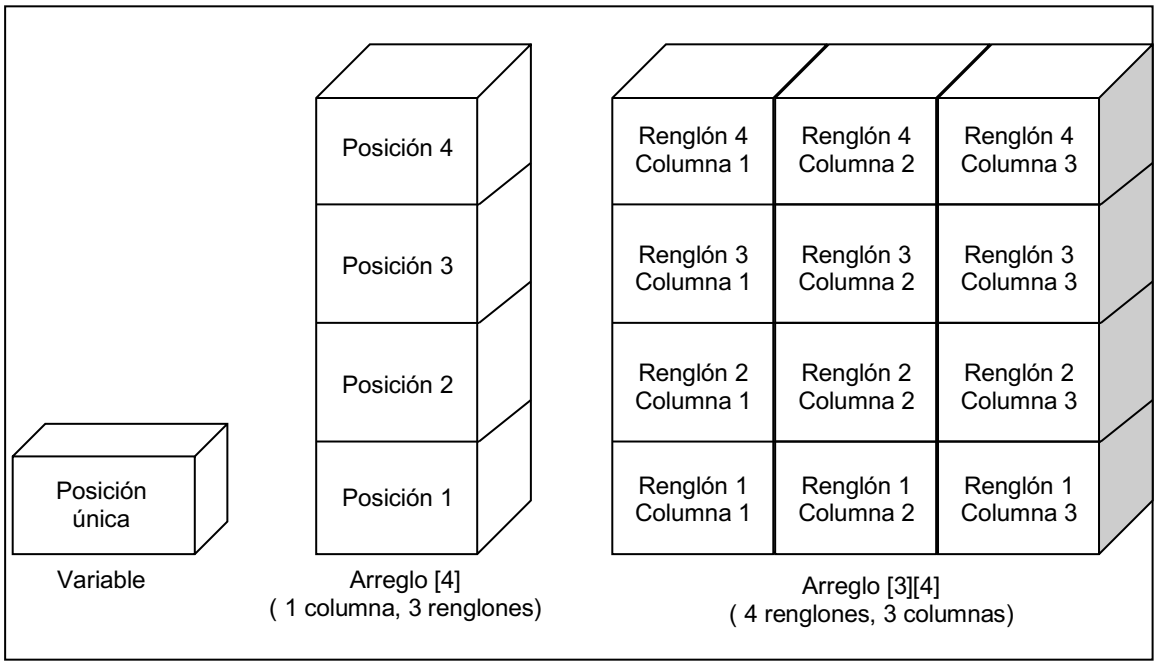


Figura 28 Comparación entre una variable, un arreglo unidimensional y un arreglo de dos dimensiones.

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

PSEUDOCÓDIGO
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]

Figura 29 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 (Casale J. y., 2014).

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.

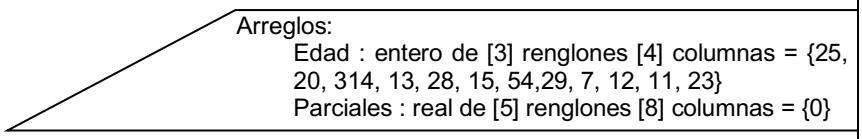
PSEUDOCÓDIGO
Arreglos: Edad : entero de [3] renglones [4] columnas = {25, 9, 18, 20, 31 4, 13, 28, 15, 54 29, 7, 12, 11, 23} Parciales : real de [5] renglones [8] columnas = {0} // si se coloca solo un cero todas las posiciones toman este valor
DIAGRAMA DE FLUJO


Figura 30 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.

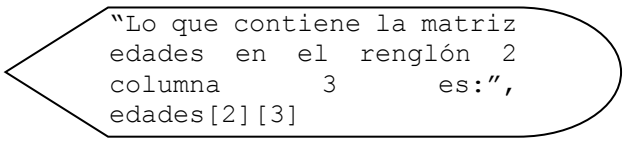
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Escribir "lo que contiene la matriz edades en el renglón 2 columna 3 es:", edades[2][3]	

Figura 31 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.

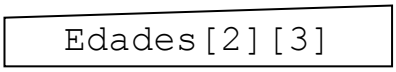
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Leer edades[2][3] // almacena una dato en el renglón 2 // columna 3 del arreglo edades	

Figura 32 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 (JOYANES, 1996).

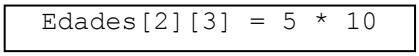
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Edades[2][3] = 5 * 10 // almacena una dato en el renglón 2 // columna 3 del arreglo edades	

Figura 33 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 1

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.

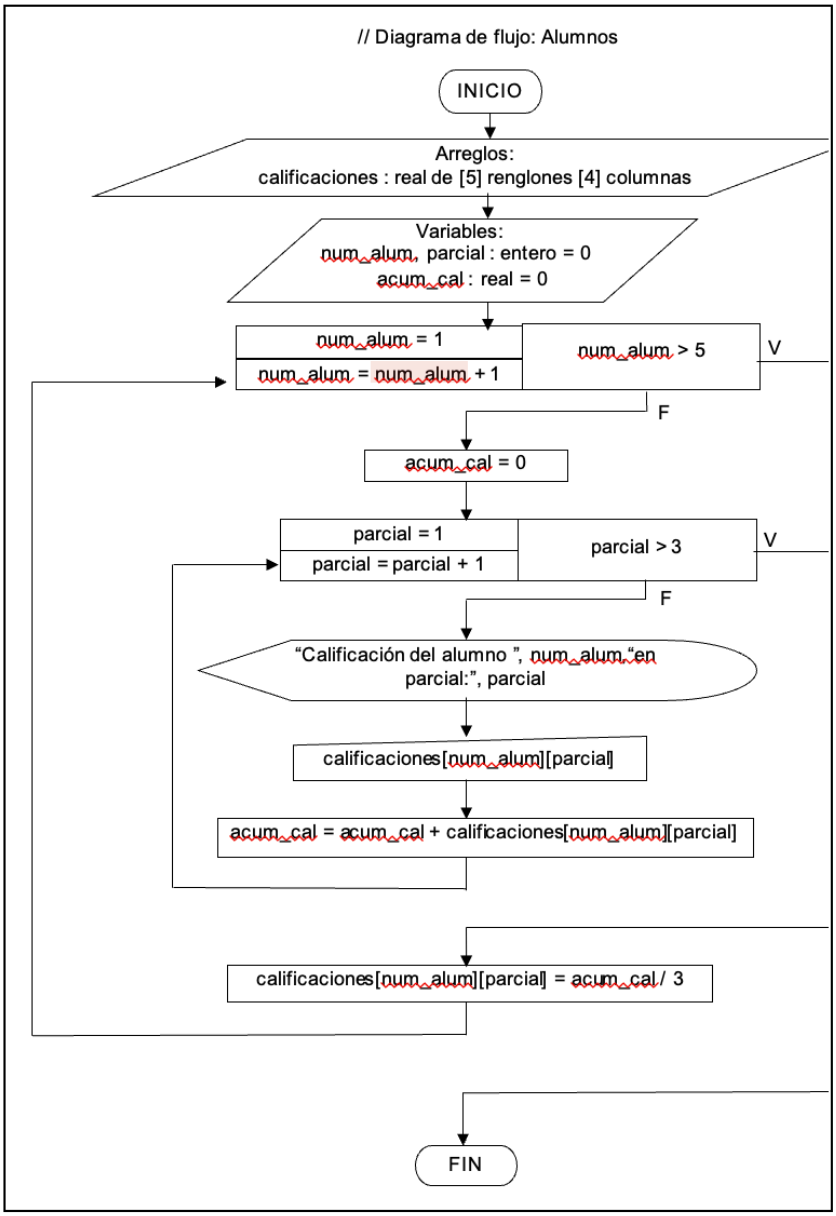
Diseñar El algoritmo
PSEUDOCÓDIGO

Pseudocódigo: alumnos
Arreglos:
  calificaciones : real de [5] renglones [4] columnas
Variables:
  num_alum, parcial : entero = 0
  acum_cal : real = 0
Inicio
 Hacer para num_alum = 1 hasta num_alum > 5
  acum_cal = 0
  Hacer para parcial = 1 hasta parcial > 3
  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
  Fin para
  calificaciones[num_alum][parcial] = acum_cal / 3
  num_alum = num_alum + 1
 Fin para
Fin

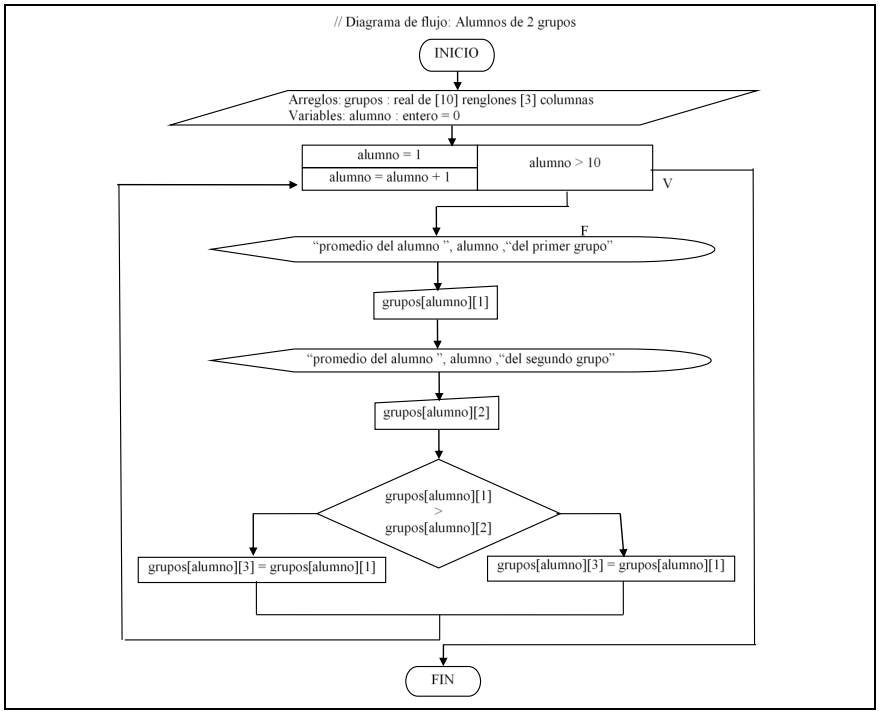
Ejemplo 2

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 Escribir “Promedio del alumno ”,alumno,“ del primer grupo:” Leer grupos[alumno][1] Escribir “Promedio del alumno ”,alumno,“ del segundo grupo:” Leer grupos[alumno][2] Si grupos[alumno][1] > grupos[alumno][2] entonces grupos[alumno][3] = grupos[alumno][1] Si no grupos[alumno][3] = grupos[alumno][2] Fin si alumno = alumno + 1 Fin para 3. Fin



.

5.2.4 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.

5.3 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.

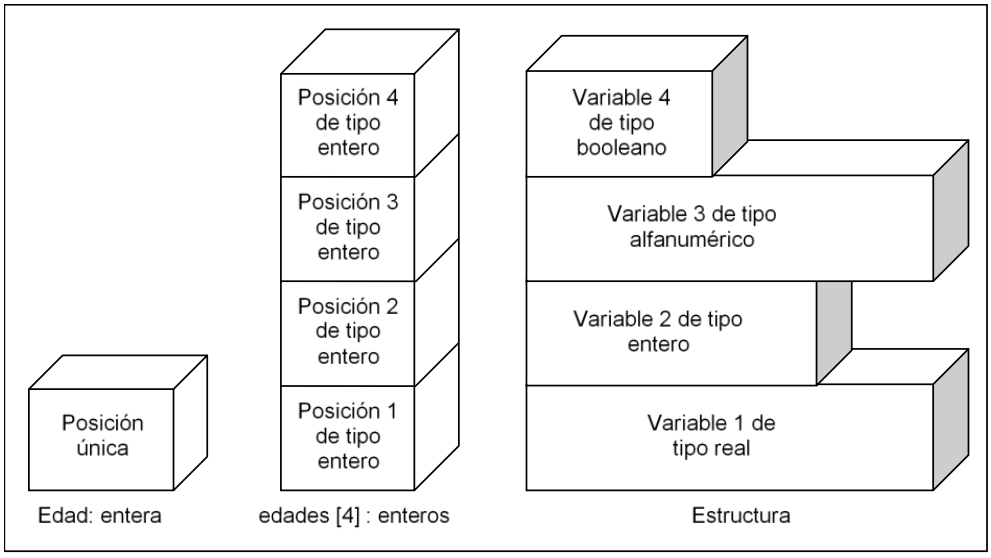


Figura 34 Diferencia entre variables, arreglos y estructuras

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 contendrán la estructura y el nombre de esta. Este proceso se conoce como definición de la estructura y se realiza en la sección estructuras.

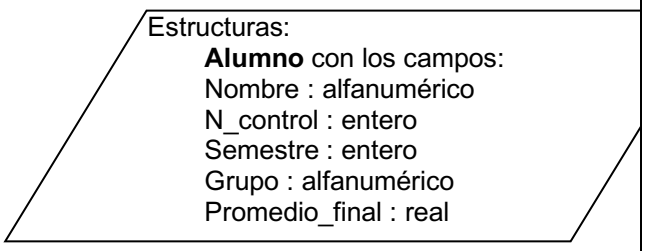
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Estructuras: Alumno con los campos: Nombre : alfanumérico N_control : entero Semestre : entero Grupo : alfanumérico Promedio_final : real	

Figura 35 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 (JOYANES, 1996).

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 (Joyanes, 2013).

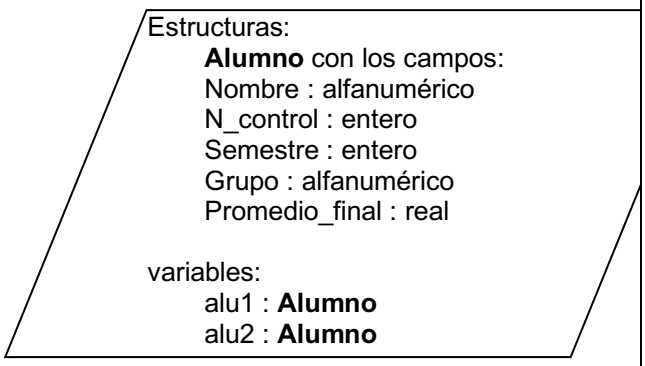
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Estructuras: Alumno con los campos: Nombre : alfanumérico N_control : entero Semestre : entero Grupo : alfanumérico Promedio_final : real variables: alu1 : Alumno alu2 : Alumno // se están declarando dos variables del // tipo Alumno	

Figura 36 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 está definida la estructura.

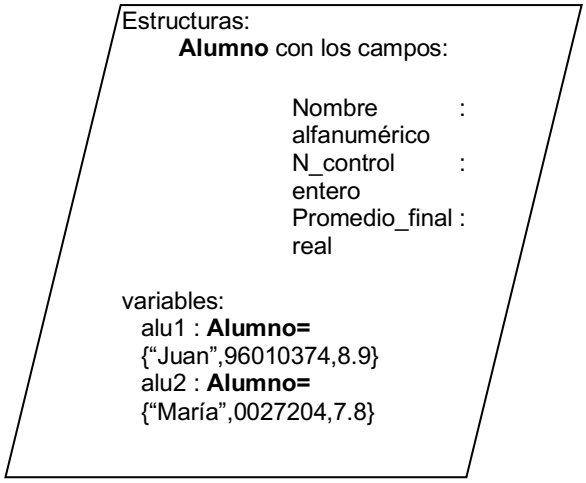
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Estructuras: Alumno con los campos: Nombre : alfanumérico N_control : entero Promedio_final : real variables: alu1 : Alumno = {“Juan”,96010374,8.9} alu2 : Alumno = {“María”,0027204,7.8}	

Figura 37 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.

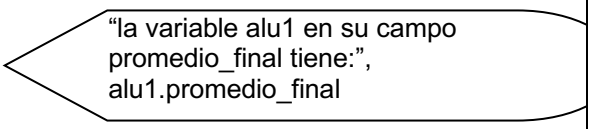
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
Escribir “la variable alu1 en su campo promedio_final tiene:”, alu1.promedio_final	

Figura 38 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	DIAGRAMA DE FLUJO
Leer alu1.promedio_final // almacena un dato en el campo promedio_final de alu1	

Figura 39 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.

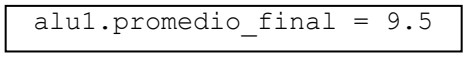
PSEUDOCÓDIGO	DIAGRAMA DE FLUJO
alu1.promedio_final = 9.5 // almacena un dato en el campo promedio_final alu1	

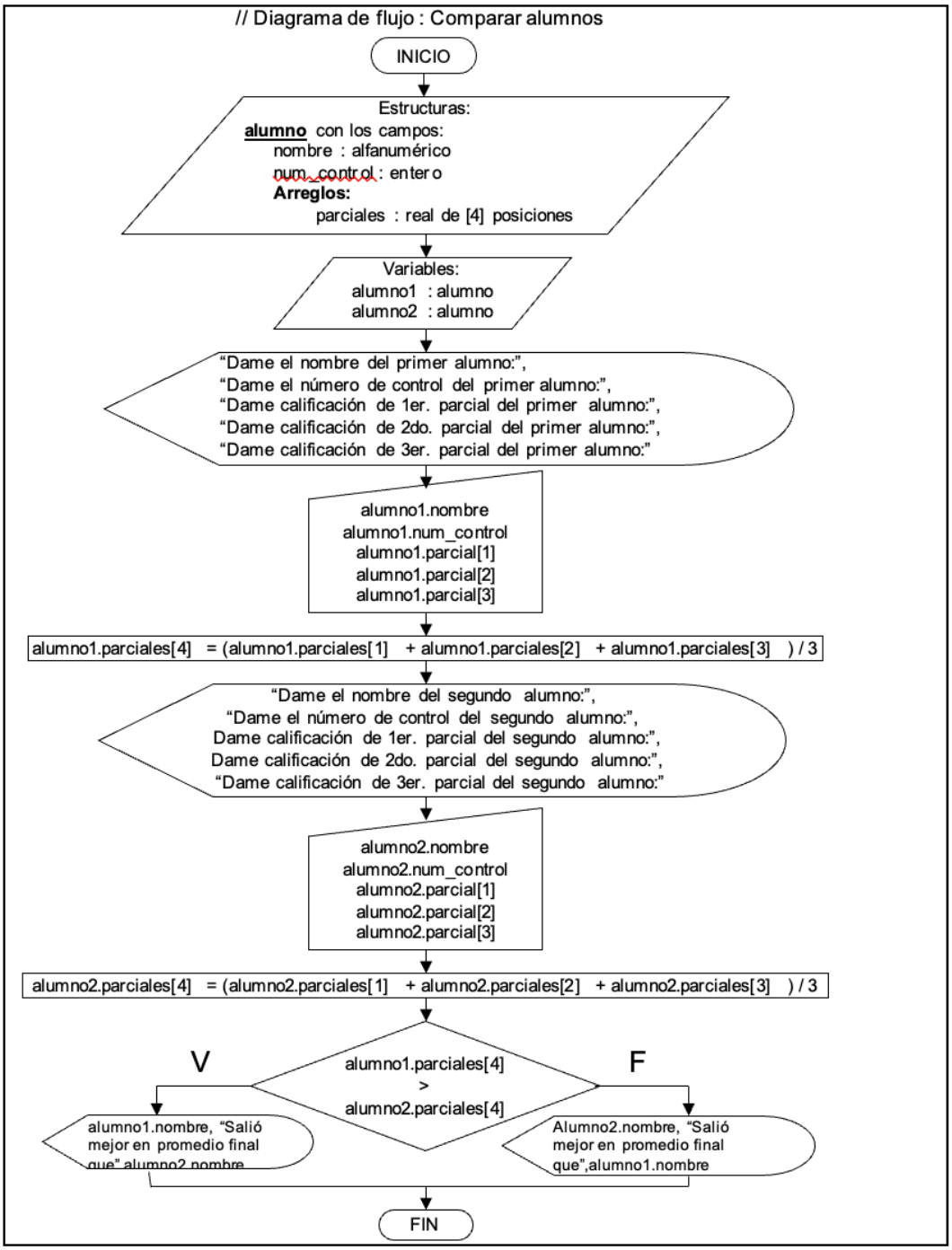
Figura 40 Asignar datos en campo de una variable del tipo estructura

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 1
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 salió 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 está 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 Escribir alumno1.nombre, “ Salio mejor en promedio final que “,alumno2.nombre Si no Escribir alumno2.nombre, “ Salio mejor en promedio final que “,alumno1.nombre Fin si 25. Fin


// Diagrama de flujo : Comparar alumnos
INICIO
Estructuras:
alumno con los campos:
nombre : alfanumérico
num_control : entero
Arreglos:
parciales : real de [4] posiciones
Variables:
alumno1 : alumno
alumno2 : alumno
"Dame el nombre del primer alumno:",
"Dame el número de control del primer alumno:",
"Dame calificación de 1er. parcial del primer alumno:",
"Dame calificación de 2do. parcial del primer alumno:",
"Dame calificación de 3er. parcial del primer alumno:"
alumno1.nombre
alumno1.num_control
alumno1.parcial[1]
alumno1.parcial[2]
alumno1.parcial[3]
alumno1.parciales[4] = (alumno1.parciales[1] + alumno1.parciales[2] + alumno1.parciales[3]) / 3
"Dame el nombre del segundo alumno:",
"Dame el número de control del segundo alumno:",
Dame calificación de 1er. parcial del segundo alumno:",
Dame calificación de 2do. parcial del segundo alumno:",
"Dame calificación de 3er. parcial del segundo alumno:"
alumno2.nombre
alumno2.num_control
alumno2.parcial[1]
alumno2.parcial[2]
alumno2.parcial[3]
alumno2.parciales[4] = (alumno2.parciales[1] + alumno2.parciales[2] + alumno2.parciales[3]) / 3
V
alumno1.parciales[4]
>
alumno2.parciales[4]
F
alumno1.nombre, "Salió mejor en promedio final que",alumno2.nombre
Alumno2.nombre, "Salió mejor en promedio final que",alumno1.nombre
FIN

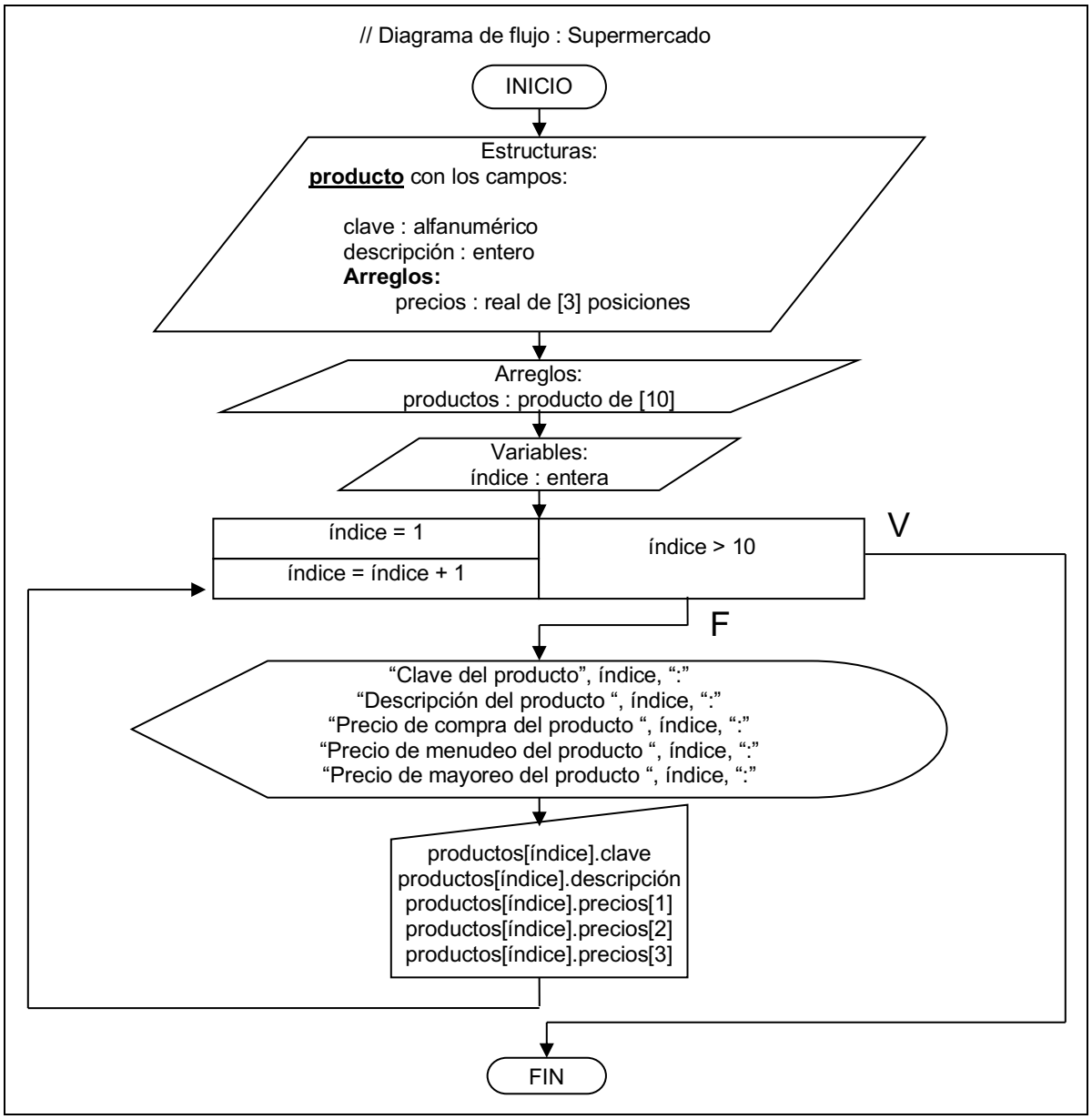
En este primer ejemplo nos damos cuenta que para introducir un dato dentro de un campo de una estructura el cual es a su vez parte de un arreglo, se tiene que indicar en el lado del campo la posición en la que se debe de guardar el dato. Pero ahora veremos como introducir datos a un arreglo del tipo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

Ejemplo 2

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 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 Para Fin


// Diagrama de flujo : Supermercado
INICIO
Estructuras:
producto con los campos:
clave : alfanumérico
descripción : entero
Arreglos:
precios : real de [3] posiciones
Arreglos:
productos : producto de [10]
Variables:
índice : entera
índice = 1
índice = índice + 1
índice > 10
V
F
“Clave del producto”, índice, “:”
“Descripción del producto “, índice, “:”
“Precio de compra del producto “, índice, “:”
“Precio de menudeo del producto “, índice, “:”
“Precio de mayoreo del producto “, índice, “:”
productos[índice].clave
productos[índice].descripción
productos[índice].precios[1]
productos[índice].precios[2]
productos[índice].precios[3]
FIN

5.3.1 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.

5.4 Arreglos en Pseint

Define un arreglo. Es decir, la cantidad de dimensiones y el valor máximo de cada una de ellas. También pueden declararse más de un arreglo en la misma línea separándolos por comas. La cantidad de dimensiones puede ser una o más, y la máxima cantidad de elementos debe ser una expresión numérica positiva. Es necesario definir un arreglo antes de utilizarlo. Se pueden definir más de un arreglo en una misma instrucción, separándolos con una coma (,) (Novara, 2020).

Sintaxis

Dimesion <identificador1> (<max1>,...,<maxN>) , ... ;

Esta instrucción define un arreglo con el nombre indicado en <identificador> y N dimensiones. Los N parámetros indican la cantidad de dimensiones y el valor máximo de cada una de ellas. La cantidad de dimensiones puede ser una o más, y la máxima cantidad de elementos debe ser una expresión numérica positiva.

Ejemplos de declaración de un arreglo:

```
Dimension Alumnos ( 10 ) ; Leer Alumno (1);

Dimension Tabla ( 10 , 5 , 3 ) , Resultados ( 5 ) ;
...
    Escribir Tabla (1,3,2);
```

Ejemplo 1

El siguiente ejemplo Mayores muestra cómo cargar datos en un arreglo y buscar luego el mayor entre dichos datos.

```
// Busca los dos mayores de una lista de N datos
Proceso Mayores
	Dimension datos[200]
	Escribir "Ingrese la cantidad de datos (de 2 a 200):"
	Leer n
	Para i<-0 Hasta n-1 Hacer
		Escribir "Ingrese el dato ",i+1,":"
		Leer datos[i]
	FinPara
	Si datos[0]>datos[1] Entonces
		may1<-datos[0]
		may2<-datos[1]
	SiNo
		may1<-datos[1]
		may2<-datos[0]
	FinSi
	Para i<-2 Hasta n-1 Hacer
		Si datos[i]>may1 Entonces
			may2<-may1
			may1<-datos[i]
		SiNo
			Si datos[i]>may2 Entonces
				may2<-datos[i]
			FinSi
		FinSi
	FinPara
	Escribir "El mayor es: ",may1
	Escribir "El segundo mayor es: ",may2
FinProceso
```

Ejemplo 2

El ejemplo Sucursales utiliza un arreglo bidimensional (matriz) para guardar las ventas de una empresa discriminadas por producto (una de las dimensiones) y por sucursal (la dimensión restante).

```
Proceso Sucursales
 Dimension Prec[5], Cant[4,5]
```

```
  Para I<-0 Hasta 4 Hacer
    Escribir 'Ingrese Precio Articulo ',I+1,':'
    Leer Prec[I]
  FinPara
  Para J<-0 Hasta 3 Hacer
    Para I<-0 Hasta 4 Hacer
      Escribir 'Ingrese Cant. de Articulo ',I+1,', en Sucursal ',J+1,':'
      Leer Cant[J,I]
    FinPara
  FinPara
  Escribir 'Cantidades por articulos:'
  Para I<-0 Hasta 4 Hacer
    Suma<-Cant[0,I]+Cant[1,I]+Cant[2,I]+Cant[3,I]
    Escribir 'Total articulo ',I+1,':',Suma
  FinPara
  Suc2<-0
  Para I<-0 Hasta 4 Hacer
    Suc2<-Suc2+Cant[1,I]
  FinPara
  Escribir 'Total Sucursal 2:',Suc2
  Escribir 'Sucursal 1, Articulo 3:',Cant[0,2]
  MayorRec<-0; NumMayor<-0; TotEmp<-0
  Para J<-0 Hasta 3 Hacer
    TotSuc<-0
    Para I<-0 Hasta 4 Hacer
      TotSuc<-TotSuc+(Cant[J,I]*Prec[i])
    FinPara
    Escribir 'Recaudaciones Sucursal ',J,':',TotSuc
    Si TotSuc>MayorRec entonces
                    MayorRec<-TotSuc; NumMayor<-J+1
    FinSi
    TotEmp<-TotEmp+TotSuc
  FinPara
  Escribir 'Recaudacion total de la empresa:',TotEmp
  Escribir 'Sucursal de Mayor Recaudacion:',NumMayor
FinProceso
```

Bibliografía

Casale, J. (2012). *Introducción a la programación*. Buenos Aires: Fox Andina; Dálaga.

Casale, J. y. (2014). *Programación desde cero*. Buenos Aires: Fox Andina:Dalaga.

Deitel, H. Y. (2004). *Como Programar en C/C++*. MEXICO: PEARSON EDUCACION.

Ferreyra, G. (2007). *Informática, Para Cursos De Bachillerato*. MEXICO: ALFAOMEGA.

Joyanes, L. (1996). *Fundamentos de Programación, Algoritmos y Estructura de Datos*. MEXICO: MCGRAW HILL.

Joyanes, L. (2013). *Fundamentos generales de programación*. Mexico, D.F.: McGraw-Hill.

Norton, P. (2006). *Introducción A La Computación*. Mexico: McGraw Hill.

Novara, P. (01 de 05 de 2020). *PSeInt es es una herramienta para asistir a un estudiante en sus primeros pasos en programacion*. Obtenido de http://pseint.sourceforge.net/

Senn, J. A. (2001). *Analisis de Sistemas de Informacion*. Mexico: McGraw Hill.

Tannenbaum, A. (2000). *Organización De Computadoras, Un Enfoque Estructurado*. Mexico: Pearson Educacion.

Descubre tu próxima lectura

Si quieres formar parte de nuestra comunidad,
regístrate en https://www.grupocompas.org/suscribirse
y recibirás recomendaciones y capacitación

compAs

Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com

Ing. Pedro Vélez Duque
Universidad Agraria del Ecuador, Profesor Titular, Ingeniero en Sistemas Computacionales (UG-GYE), Magister en Educación Informática (UG-GYE). Consultor informático.

compAs
Grupo de capacitación e investigación pedagógica

@grupocompas.ec
compasacademico@icloud.com

ISBN: 978-9942-33-463-3

@grupocompas.ec
compasacademico@icloud.com

compAs
Grupo de capacitación e investigación pedagógica