Dedicatoria A:

Ana Isabel,
Luz Piedad,
Juan Pablo y
Efraín Alberto Oviedo

TABLA DE CONTENIDO

PRESENTACIÓN

Este texto está basado en la segunda edición del libro **Algoritmos Estructurados,** pero con muchos cambios, tanto en la forma como en su contenido. Además, se le adicionaron y modificaron conceptos, tratando siempre de conservar su forma didáctica para que sea fácil de leer e interpretar por aquellos lectores que se inician en el fantástico mundo de los computadores.

La obra llegó a su fin gracias al aporte del Departamento de Recursos de Apoyos Informáticos (DRÁI) , de la Facultad de Ingeniería de la Universidad de Antioquia y de las críticas y aporte de muchos de mis compañeros y colegas, entre ellos: Fabián Ríos, Roberto Florez R, Aldrin Fredy Jaramillo, Leonel Osorno y Ovidio Montoya.

Se recomienda a los lectores iniciar la lectura del texto desde el inicio e ir analizando los ejercicios resueltos y solucionando los propuestos, ya que es fundamental dentro del proceso lógico que sigue el libro: esto es importante ya que cada capítulo nuevo utiliza los conceptos de los capítulos anteriores.

Los ejercicios resueltos están codificados en C++, y pueden ser consultados en la página web **http//ingenieria.udea.edu.co/profesores/poviedo**, donde se actualizaran y codificaran en otros lenguajes de programación.

El libro está dividido en nueve capítulos, partiendo de lo más elemental hasta llegar en el capítulo final a un tema de mayor profundidad. No implica esto que haya necesidad de comenzar por el primer tema si el lector ya conoce algunos de los tópicos tratados. Está estructurado de forma tal que cualquier persona que esté interesada en un tema específico pueda consultarlos, sin necesidad de seguir el desarrollo completo del texto.

El capítulo 1, **Conocimiento de la computadora,** presenta una reseña histórica de estas máquinas, y da a conocer diferentes definiciones y clasificaciones, hasta introducirnos en la terminología básica de las computadoras y en los sistemas numéricos con los cuales trabaja.

El segundo capítulo, **Generalidades sobre algoritmos,** trata de la lógica y de los pasos necesarios para solucionar problemas a través de la computadora; también nos introduce en los conceptos de la programación y muestra la forma de representar los algoritmos sobre el papel, lo que hace más fácil el paso a un lenguaje de programación.

En el capítulo tercero nos encontramos con la primera estructura básica de control: la **estructura secuencial.** En este capítulo se dan a conocer, paso a paso, las instrucciones de asignación, de entrada y salida de datos, y algunos otros conceptos para los cuales fueron diseñadas estas estructuras.

Estructura decisión lógica es el nombre del cuarto capítulo. En él se plantea la representación y el funcionamiento de una de las estructuras más utilizadas en el arte de programar: la estructura SI-CIERTO-FALSO.

El quinto capítulo, **Estructura repetitiva,** trata sobre el funcionamiento de la tercera estructura básica de control: el ciclo HAGA MIENTRAS. Este capítulo nos guía desde las variables tipo contador y acumulador hasta los rompimientos de control de ejecución, pasando por los esquemas cuantitativo y cualitativo y la ruptura de ciclos.

En el sexto capítulo, **Estructuras adicionales,** se encuentra la representación y el funcionamiento de las estructuras no básicas de control: CASO, PARA, HACER MIENTRAS QUE como estructuras que pueden ser usadas en casos excepcionales, ayudando a mejorar la programación.

El capítulo siete parte de la representación y documentación de los diferentes clases de **subprogramas,** para terminar con la forma en la que se activan éstos dentro de los algoritmos. Es un capítulo dinámico que plantea la verdadera utilidad de los subprogramas en la programación modularizada.

El capítulo ocho trata sobre el manejo de **arreglos,** su clasificación de una, dos y tres dimensiones que son los más utilizados en el medio, lo mismo que las operaciones de búsqueda, ordenación, inserción y borrado de elementos.

Todos los temas relacionados con este capítulo son manejados con subprogramas y vienen acompañados de una gran cantidad de ejercicios resueltos, fáciles de comprender y que explican a fondo su funcionamiento, además de muchos ejercicios propuestos.

En el capítulo nueve se introduce el concepto de **registros**, arreglo de registros, lo mismo que el concepto de **archivos de datos**. El lector puede encontrar diversos tópicos sobre archivos, tales como: componentes, clasificación, manejo de claves, procesamiento de archivos secuenciales y secuenciales indexados, lo mismo que ejercicios resueltos de gran utilidad.

A lo largo del texto se encuentran noventa ejercicios resueltos, que hacen más fácil comprender cada capítulo. También cuenta con doscientos cincuenta ejercicios propuestos, que dan al lector la oportunidad de practicar todo lo aprendido sobre los temas. Cada sección del libro tiene, al final, una parte dedicada a resumir los aspectos principales de que trata el capítulo y una serie de ejercicios que permite afianzar los conocimientos.

Por último, el texto viene acompañado de la página web http://ingenieria. udea.edu.co/profesores/poviedo, donde se encuentran noventa ejercicios que estan resueltos algorítmicamente, codificados en el lenguaje de programación C++. En cda uno de ellos se útiliza una gama de recursos con el fin de darles buena presentación a la captura y salida de información tales como: manejo de colores, tipos de letras, cuadros, menús, validación de información, manejo de teclas y librerías creadas con muchos subprogramas de gran utilidad.

El autor

Capítulo 1

CONOCIMIENTO DE LA COMPUTADORA

1.1. Historia de la computadora

No se tiene una fecha exacta determinada de cuándo se planteó el hombre la necesidad de agilizar a través de una máquina las operaciones que se le presentaban. Una de las primeras máquinas de las cuales se tiene noticia es el **Ábaco,** cuya operación data de miles de años en el cercano Oriente. Cuando esta técnica se conoció aparecen varias versiones de él, tales como el **Ábaco Chino y el Sorobán Japonés.**

Más tarde, en 1642, el francés Blaise Pascal desarrolló una máquina calculadora que era operada por una serie de discos, unidos en engranajes, que tenían los números del 0 al 9 en circunferencia, de tal forma que cuando un disco daba una vuelta, automáticamente el disco de la izquierda avanzaba un dígito. Los indicadores sobre los discos mostraban las respuestas correctas. Pero esta calculadora sólo servía para sumar.

En 1694 Gottfried Withelm Leibnitz creó una máquina que multiplicaba y dividía directamente. El dispositivo que utilizó Leibnitz, en forma de rueda escalonada, aún se usa en algunas calculadoras.

Tanto Pascal como Leibnitz se adelantaron a la tecnología de sus tiempos, pero ninguno de sus inventos era confiable; por otra parte, todos estos intentos por conseguir una máquina que pudiera procesar datos se hacían en forma aislada.

Más tarde, cuando apareció el interés por la navegación, la astronomía y el hilado textil, se crearon nuevas máquinas para la ayuda en la solución de cálculos complejos y para ingresar patrones a través de tarjetas a los telares. En 1801 Joseph Marie Jacquard diseñó tarjetas perforadas para controlar una máquina de hilado textil, en donde cada línea de tejido se presentaba en una tarjeta perforada como patrón y éste se tejía automáticamente. En 1822 Charles Babbage desarrolló un proyecto denominado máquina diferencial, la cual era capaz de producir tablas logarítmicas de seis cifras decimales de precisión. Animado por los resultados obtenidos, Babbage continuó sus experimentos para obtener una máquina analítica, pero desafortunadamente el bajo nivel de la tecnología metálica de la época impidió que llegara a un buen resultado.

El trabajo realizado por Babbage es retomado antes de finalizar el Siglo, cuando se hizo énfasis en las investigaciones para obtener máquinas programables. El norteamericano Hernán Hollerith desarrolló un mecanismo basado en tarjetas perforadas, cuya primera utilización exitosa se llevó a cabo en 1886 durante la tabulación del censo de población de 1880 en los Estados Unidos. Este equipo significó un gran adelanto en el campo de las computadoras.

El mayor impulso y fuerza del desarrollo definitivo de las computadoras fue la Segunda Guerra Mundial. Durante esta época técnicos de la IBM, empresa fundada por Hollerith, la

Universidad de Harvard y la Universidad de Pennsylvania, desarrollaron simultáneamente el **MARK 1** y el **ENIAC,** cuya utilización estaba relacionada con los misiles guiados de uso militar.

La **MARK 1** pesaba cinco toneladas y constaba de un complejo de 78 máquinas sumadoras y calculadoras, conectadas con 800 kilómetros de cables. Las instrucciones se le daban en cintas de papel y una vez que la máquina ejecutaba la primera instrucción no requería de la intervención humana. A pesar de que esta máquina electromecánica representaba un avance con respecto a las creaciones anteriores, todavía presentaba restricciones a causa de su lentitud y dificultades en su operación.

Para esta misma época, John W. MauhIy y J. P. Eckert desarrollaron una máquina que denominaron **EDVAC;** capaz de desarrollar operaciones aritméticas con números binarios y de almacenar instrucciones. El control de la computadora se realizaba mediante el alambrado de cables removibles o cinta de papel, y una vez que el procesamiento de los datos había sido iniciado, ninguna modificación podía efectuarse a pesar de que surgiera la necesidad de hacerlo.

Aun cuando las primeras computadoras se desarrollaron con propósitos militares, no tardó mucho tiempo en que se emplearan en las tareas comerciales e industriales. El gran empuje de la computadora comercial lo dio el desarrollo del transistor, por parte de los laboratorios de la Bell Telephone, que vinieron a reemplazar los tubos al vacío.

Conforme la tecnología avanzó en todas las áreas, la investigación en el campo de las computadoras se extendió y los resultados se hicieron comercialmente más prácticos en tamaño y costo. La **UNIVAC 1** (Universal Automatic Computer), instalada en el departamento de censos de los Estados Unidos en 1951, fue la primera computadora que se produjo en forma comercial y contenía varias de las características de las computadoras actuales. El uso comercial de la **UNIVAC** comenzó en 1954, y a partir de esta fecha la industria del procesamiento de datos comenzó a crecer en forma acelerada.

Las máquinas desarrolladas en la década de los años 1948 a 1958 constituyen **la primera generación de computadoras.** Éstas utilizaban bulbos como componentes básicos de sus circuitos, generando un alto consumo de energía, calor muy intenso. Además a nivel de personal se requería de treinta y cinco a cien programadores, analistas, codificadores y personal de mantenimiento, así como de un sistema de aire acondicionado. En esta generación se desarrollaron los lenguajes de programación: **FORTRAN,** orientados a la resolución de problemas numéricos y **ALGOL,** dirigido al tratamiento de problemas científicos.

En 1959 surge la **segunda generación de computadoras**, lo que significó un gran adelanto en el progreso de esta industria, debido a que los bulbos fueron sustituidos por transistores, los cuales permitieron aumentar la confiabilidad y velocidad operativa de estos equipos. Los avances en equipos periféricos también fueron notables: impresoras cada vez más rápidas, mejores lectoras de tarjetas y de cinta perforada, y sobre todo bobinas de cinta magnética capaces de memorizar y de volver a leer datos en número ilimitado. Así mismo, en esta generación proliferan diferentes modelos de computadoras, incorporando su uso en el área de los negocios. Se desarrolla el lenguaje de programación **COBOL** para estos fines y se incrementa la utilización del lenguaje **FORTRAN.** Se crean entonces los ensambladores, que utilizan un código nemotécnico para representar las instrucciones.

A partir de este momento los avances tecnológicos fueron, y continúan siendo, sorprendentes. Los transistores se sustituyeron por los circuitos integrados (**IC**). Con este cambio nace la **tercera generación,** cuya base está en los circuitos integrados monolíticos y en la miniaturización de la electrónica.

Las máquinas de esta generación (IBM 360, IBM370, POP6) son fabricadas con la característica de que las pertenecientes a una misma serie son compatibles entre sí, facilitando su uso; además, son utilizables tanto para aplicaciones de tipo científico como comercial. A

medida que estas máquinas se fueron perfeccionando surge la multiprogramación, el multiprocesamiento, las comunicaciones de datos, otros lenguajes (**BASIC** y **PL1**) y los paquetes especializados de programación. En esta época nacen las calculadoras de bolsillo y las microcomputadoras.

En la década de los años setenta se logra una gran reducción de los circuitos integrados incorporando miles de componentes con un espacio menor a una micra, lo cual coloca a las computadoras en una **cuarta generación** en la que aparece el lenguaje **PASCAL** como uno de los más poderosos, por ser de aplicación general e incluir los conceptos introductorios de lo que hoy se conoce como programación estructurada.

Actualmente existen circuitos que tienen alrededor de 260.000 elementos y se calcula que los adelantos permitirán circuitos mucho más pequeños y cien veces más veloces. Hoy se habla de supercomputadores, que caracterizarán la **quinta generación**.

1.2. Definición de una computadora

Una computadora es una máquina con gran capacidad y velocidad. Se ha convertido en un auxiliar del hombre que le presta ayuda en una enorme gama de actividades, tales como:

- Proveer a los médicos información actualizada sobre la salud del paciente.
- Preparar gráficas de patrones climatológicos y producir planes de vuelo de acuerdo con las condiciones climatológicas.
- Prestar ayuda en la oceanografía y la navegación.
- Registrar marcas y evaluar estadísticas de competencias deportivas.
- Prestar ayuda a los ingenieros en los cálculos.
- Controlar simuladores de vuelo para dar al piloto un entrenamiento inicial en tierra.
- Coordinar el funcionamiento de los semáforos para que el tránsito no sufra interrupciones.
- Verificar la cantidad de dinero depositado en una cuenta. Un empleado del banco habla con una computadora a través del teléfono.
- Proveer información sobre los productos en el mercado.

Mucho se habla de que la computadora está influyendo dentro de nuestra privacidad y sustituyendo mano de obra, creando desempleo. Estos aspectos no pueden discutirse, sin conocer a fondo lo que es una computadora, pero a medida que se avanza en su conocimiento, es posible emitir un concepto inteligente. Hay que tener en cuenta que una computadora no puede hacer algo a menos que el ser humano le diga qué hacer. Debemos controlar la computadora y no ella a nosotros.

1.3. Clasificación de las computadoras

De acuerdo con el avance tecnológico se pueden establecer categorías de computadoras, así:

- *Primera generación.* Caracterizadas por utilizar tubos al vacío en los componentes básicos de sus circuitos.
- *Segunda generación.* Los tubos al vacío son sustituidos por transistores; la memoria estaba constituida por tubos magnéticos; se usaban en aplicaciones de tipo militar, comercial, científico y utilizaban el disco magnético.
- *Tercera generación.* Caracterizadas por utilizar en sus componentes circuitos integrados monolíticos. En esta época se le adicionan equipos auxiliares, tales como terminales interactivas.
- *Cuarta generación.* Caracterizadas por la incorporación de miles de componentes, integrados en espacios sumamente pequeños. Se crean técnicas para manejar volúmenes grandes de información y se usan computadoras remotas en operación con un computador central mediante la telecomunicación, dando lugar a la telemática. Se popularizan los lenguajes de

programación de tipo interactivo, las máquinas virtuales y los manipuladores de datos. La computadora es capaz de elaborar programas.

- *Quinta generación.* Son las computadoras del futuro. Se introduce el uso de la memoria de burbuja magnética y la técnica holográfica con rayos láser; se mejoran las técnicas de comunicación con el procesador central en forma conversacional o interactiva y los lenguajes cada vez son más humanos, inclusive la voz.

Según el tipo de información que maneja una computadora, éstos se pueden clasificar en:

- *Computadoras digitales*. Son aquéllas que procesan datos discretos. Trabajan directamente contando números (dígitos) que representan cifras, letras u otros símbolos especiales. Así como los relojes digitales cuentan los segundos y minutos en una hora, los procesadores digitales cuentan valores discretos para alcanzar los resultados deseados. Las computadoras digitales son usadas en el proceso de datos, como puede ser el proceso de una contabilidad. Las computadoras digitales pueden ser:
  - *De propósitos generales.* Son utilizadas para almacenar diferentes programas y pueden ser usadas en incontables aplicaciones. Por ejemplo, una máquina que procese información y luego la imprima, que escriba nuevos programas y cambie o borre los anteriores.
  - *De propósitos especiales.* Diseñadas para realizar sólo una tarea específica, ya que los programas están permanentemente almacenados en la máquina.
- *Computadoras analógicas.* Procesan datos que están medidos en una escala continua y registrados con un determinado grado de precisión. Por ejemplo, el voltaje que puede ser medido con aproximación de un centésimo de voltio. Las computadoras analógicas son usadas para controlar procesos con alta precisión, pudiendo ser hasta en un 0.1 % el valor correcto. Se dedican al control físico de actividades, como el control de un proceso de ensamble automatizado o un sistema de control de temperatura.
- *Computadoras híbridas (analógico digitales).* Son una combinación de las dos anteriores. Por ejemplo, en la unidad de control interactivo de un hospital los dispositivos analógicos pueden medir el funcionamiento del corazón de un paciente, la temperatura y los signos vitales. Estas medidas pueden ser convertidas a números y enviadas a un componente digital del sistema. Este componente digital del sistema es usado para controlar los signos vitales del paciente y enviar una señal a la estación de las enfermeras cuando sean detectadas lecturas anormales.

Tanto las computadoras híbridas como las analógicas son importantísimas, pero las más usuales son las digitales.

Por la capacidad y estructura física. Antes de entrar en esta división, es conveniente anotar que estos tipos de computadoras están relacionadas directamente con una época específica en la historia y con el desarrollo de los computadoras.

- *Computadoras mainframe.* Son las computadoras *grandes*. Inicialmente, las computadoras eran en su mayoría mainframe y su uso estaba restringido a empresas o entidades de gran tamaño. El costo de adquisición y operación era alto y complejo. El uso de los mainframe está limitado a empresas que operan con procesos centralizados.
- *Minicomputadoras.* Son computadoras medianas: su uso ha reemplazado una buena parte a los mainframe. Su utilización es estipulada para empresas grandes.
- *Microcomputadoras de negocios.* Son computadoras pequeñas que se pueden utilizar en aplicaciones comerciales y técnicas, tanto en las pequeñas empresas como en las grandes.
- *Microcomputadoras personales y del hogar.* Son utilizadas en juegos electrónicos, juegos didácticos y diversión. Suplieron en gran parte las calculadoras programables. Se puede decir que éstas dieron origen a las anteriores por la gran demanda que tuvieron.

1.4. La computadora digital

Estructura lógica

Todas las partes de un sistema de computación operan bajo el control de una de ellas: la unidad de control.

Aunque un sistema de computación está compuesto por muchos dispositivos, cada sistema tiene cinco componentes básicos. En la figura, a continuación, se muestran estos componentes y sus relaciones entre sí.

FALTA GRAFICO PAG.7

La unidad de entrada alimenta la memoria con datos e instrucciones. La unidad de memoria almacena instrucciones y datos. Es el enlace más común entre todas las unidades o componentes.

- La memoria auxiliar proporciona capacidad masiva de almacenamiento.
- La unidad aritmética y lógica compara, suma, resta, multiplica y divide.
- La unidad de salida registra los datos que le suministra la memoria.
- La unidad de control interpreta instrucciones y emite órdenes para todos los demás componentes de la computadora.

Independientemente de cómo se introduzcan los datos, éstos se transforman en impulsos eléctricos que se transfieren a direcciones predeterminadas dentro de la memoria. La entrada se divide básicamente en dos tipos: *instrucciones para el proceso de datos y los datos a procesar*.

Las instrucciones pasan de la memoria a la unidad de control, la cual las interpreta e indica a los otros componentes qué debe hacerse con los datos o en dónde se les debe colocar dentro de la memoria. Por ejemplo, si se deben sumar los datos contenidos en dos campos para producir los datos de un tercero, la unidad de control emite la orden apropiada para la unidad aritmética y lógica. Entonces, esta unidad extrae los datos apropiados de la memoria, ejecuta la suma y coloca el resultado en la memoria.

En muchas aplicaciones se debe almacenar gran cantidad de datos y no hay espacio suficiente en la memoria. En estos casos se emplea el componente de memoria o almacenamiento auxiliar. Esta unidad se encuentra separada de la unidad central de procesamiento, pero la unidad de control siempre tiene acceso a los datos que se encuentran en ella.

La unidad de control, la memoria y la unidad aritmética y lógica, generalmente se encuentran localizadas en lo que se conoce como unidad central de procesamiento **CPU** *(Central Processing Unit)*.

Dispositivos de entrada y salida de datos

En algunos casos, los medios de entrada y salida se encuentran combinados en un solo dispositivo.

Un método común para registrar datos es emplear cintas magnéticas. En algunos aspectos no es muy diferente de una grabadora común y corriente. Así como se puede grabar música en una grabadora y luego escucharla una y otra vez, también pueden registrarse datos en una cinta magnética y leerlos una y otra vez. El dispositivo que se utiliza es una impulsadora de cinta magnética, la cual puede registrar datos en la cinta y también leerlos repetidas veces, aunque no ejecuta ambas operaciones simultáneamente. Ambas operaciones, registro y lectura, se ejecutan a altas velocidades. Cuando se leen datos que se encuentran en la cinta, la unidad impulsadora actúa como unidad de entrada; los datos son transferidos a la memoria de la computadora. Cuando los datos que están en la memoria son transferidos a la cinta, actúan como dispositivo de salida.

Los caracteres se representan en la cinta por medio de pequeños puntos magnetizados en diferentes colocaciones. En las cintas es posible almacenar grandes cantidades de datos debido a que los puntos magnetizados se pueden colocar muy próximos entre sí. Debido a esta

particularidad y a su gran velocidad, las impulsadoras de cinta magnética se utilizan con frecuencia como dispositivos de almacenamiento auxiliar.

Antes de empezar algún proceso, los datos registrados en ellos se transfieren a la memoria principal. El operador puede quitar las cintas y reemplazarlas por otras; esto significa que si una cinta se usa una vez por semana, se pueden almacenar fuera del sistema cuando no se emplee. Cuando se requieran de nuevo estos datos el operador puede colocar nuevamente la cinta en la unidad impulsadora.

Las desventajas de las cintas es que sus datos no se pueden interpretar en la cinta misma; por lo tanto, para verificar sus datos es necesario procesarla y así obtener un informe impreso; además, sus datos no se pueden reordenar fácilmente ni tampoco insertar nuevos registros entre los ya existentes.

Otro dispositivo muy útil es la terminal de máquina de escribir comúnmente denominada terminal de teletipo, que puede conectarse al computador mediante una línea telefónica normal, convirtiéndose de esta manera en un dispositivo extremadamente útil. Una sucursal que se encuentre a miles de kilómetros del sistema de computación puede enviar datos de entrada y recibir datos de salida mediante una terminal de teletipo.

El operador usa claves específicas para comunicarse con la computadora, envía datos y pide informes. Tanto el envío como la salida que produce la computadora se imprime automáticamente en un rollo de papel que pasa por la terminal.

La terminal de pantalla, que a veces recibe el nombre de télex, es muy parecida a la terminal de teletipo: la principal diferencia consiste en que los datos se exhiben en una pantalla en lugar de imprimirse. La terminal de pantalla también se puede conectar al sistema de computación mediante líneas telefónicas.

El operador se comunica con el computador por medio de un teclado. Tanto los datos de entrada como de salida aparecen en la pantalla. En la pantalla se pueden mostrar diagramas y gráficos, esto es muy útil para estudios científicos y de ingeniería. En ocasiones los sistemas para reservaciones en aerolíneas muestran un diagrama del avión señalando los sitios reservados para un vuelo en particular.

Otro dispositivo de entrada/salida, y quizá el más útil y flexible de todos, es el disco magnético. Consiste en una serie de discos metálicos que se colocan uno sobre otro, dejando un espacio de separación adecuado. Los datos se registran en las dos caras de cada disco, mediante cabezas lectoras/grabadoras, las cuales entran y salen en el espacio que hay entre los discos.

Los datos se pueden grabar y leer en el disco. En los discos cabe gran cantidad de datos. Cada dato se almacena en una dirección o localización conocida. Las cabezas lectoras/grabadoras pueden ir directamente a cualquier dirección del disco, con lo cual, la búsqueda de cualquier dato se puede hacer rápidamente. Los datos se registran en el disco mediante pequeños puntos magnéticos. Al igual que las cintas los discos se pueden retirar de la unidad de disco y guardarlos en otra parte hasta que se requieran los datos contenidos en ellos, pero también existen discos fijos que no se pueden mover.

En forma parecida operan también los dispositivos de entrada/salida: disquetes y discos compactos.

Existen también dispositivos para reconocimiento de caracteres ópticos, comúnmente conocidos como dispositivos **OCR** (*Optical Character Recognition*). La mayoría de datos que van a introducirse al computador se registran primero en un documento fuente, que puede ser manuscrito o mecanografiado. Los dispositivos **OCR** pueden leer estos datos y convertirlos directamente en impulsos eléctricos. A su vez, estos impulsos se pueden registrar directamente en cinta magnética, disco, etc. No se requiere perforar tarjetas, así que no sólo se economiza tiempo y dinero, sino que se disminuyen las posibilidades de error.

Hay dispositivos que se emplean únicamente para salida, la impresora es el dispositivo más común en los negocios. Existen muchos tamaños y versiones. Hay impresoras que pueden imprimir hasta 5.000 líneas por minuto. Sin embargo, la velocidad más usual está entre 500 y 1.000 líneas por minuto.

Se han diseñado dispositivos que registran datos de salida directamente en microfilm. A medida que se producen los datos, se proyectan sobre una pantalla y se fotografían con una cámara ultrarápida. Mediante este sistema, los datos de salida se pueden registrar en película con la misma velocidad con que opera una lectura de microfilm. Unos cuantos metros de microfilm pueden almacenar datos que requerirían miles de páginas impresas.

Un método alterno consiste en grabar los datos de salida en cinta magnética y, después, pasarlos a microfilm. Este sistema es más económico que el primero.

Otros métodos de comunicación con la computadora son el lápiz óptico, el tambor magnético y la voz.

1.5. Tipos de procesamiento

Consideremos ahora un aspecto diferente en cuanto al procesamiento de datos en un Centro de Cómputo. Sea la elaboración de una nómina que se pasa quincenalmente; las novedades respecto a tiempo extra u otros temas se reúnen cada semana y se guardan o almacenan hasta que llegue el momento de procesarlas, es decir, las novedades no se procesan en el instante en que se generan sino que se acumulan para hacerlo en el momento oportuno; a este tipo de proceso se denomina **proceso por lotes.**

Ahora supongamos que alguien va a una oficina de una aerolínea para hacer reservaciones en algún vuelo; como es obvio que el empleado no puede guardar la solicitud para procesarla luego con muchas otras, los datos deben procesarse inmediatamente. A este tipo de proceso se le denomina procesamiento **directo o en línea.**

La mayoría de los requisitos del procesamiento de datos son de uno de estos dos tipos: primero, los datos se almacenan hasta que llegue el tiempo señalado para su proceso; segundo, los datos se deben procesar tan pronto como se originan.

Cuando se tienen condiciones del primer tipo se utiliza lo que se denomina procesamiento por lotes. Se pueden recopilar grandes cantidades de datos y procesarlos en el momento oportuno, como en una nómina, por ejemplo. Cuando los datos deben procesarse inmediatamente se utiliza el procesamiento directo.

El proceso directo requiere que el sistema de cómputo tenga la posibilidad o capacidad de un acceso directo. Acceso directo se refiere a la posibilidad de ir directamente al punto donde se encuentra el registro requerido del archivo sin tener que examinar registro por registro. El almacenamiento en disco tiene la ventaja de que un registro determinado se puede leer directamente sin tener que hacerlo todos los registros anteriores.

El uso de terminales de teletipo y de pantalla que se encuentran a gran distancia del sistema central de cómputo se efectúa mediante el uso de telecomunicaciones. Telecomunicación es la comunicación de datos mediante el uso de líneas telefónicas. En la figura, a continuación, se ilustra el funcionamiento de un sistema de telecomunicaciones.

FALTA GRAFICO PAG.11 LIBRO

Al operar las terminales de pantalla o de teletipo se genera el código de computador. Este código pasa a un codificador/decodificador de datos que lo convierte en un código que puede transmitirse por cables telefónicos. En el otro extremo de la línea, donde se encuentra la computadora, hay otro codificador/decodificador de datos que convierte el código telefónico en código de computadora. Este código se usa como entrada para la unidad central de procesamiento.

La salida requerida se produce en forma de código de computadora, de tal manera que tiene que convertirse en código telefónico para ser transmitida a las terminales donde se efectúa la operación inversa, y el código de computadora produce la salida apropiada. También se pueden emplear las líneas telefónicas normales para transmitir datos. Todo lo que hay que hacer es levantar la bocina y marcar el número de la computadora; un cierto tono indica que la conexión está lista y que se puede empezar a transmitir; el teléfono se ajusta con el codificador/decodificador de datos y estamos listos para comenzar.

Como es de suponerse, las telecomunicaciones ampliaron notablemente la utilidad y conveniencia de los sistemas de computación. Todas aquellas empresas que no cuentan con su propia computadora, pueden tener ahora acceso fácil a una y, recibir un servicio inmediato.

Las telecomunicaciones, junto con los adelantos en los sistemas de comunicación, han hecho posible la práctica de lo que se denomina tiempo compartido. La computadora está programada de tal manera que admite peticiones de procesamiento hechas desde lugares lejanos y, además, reparte el tiempo disponible de procesamiento entre varios usuarios. De este modo, al mismo tiempo que el sistema está produciendo una cierta salida para algún usuario, ejecuta determinados cálculos para otro. A los clientes se les cobra una cuota que, generalmente, depende de la cantidad de tiempo de computadora empleada. Para el uso en los negocios, los sistemas de computación vienen en tres tamaños básicos: pequeño, mediano y grande. Primordialmente, el tamaño de una computadora se determina por la capacidad de almacenamiento que tenga la unidad de procesamiento, la cual, determina el tamaño de los programas y la cantidad de datos que puede manejar en cualquier momento dado.

1.6. Elaboración de programas para computadora

Las computadoras pueden ejecutar un gran número de operaciones a muy altas velocidades y con intervención mínima de seres humanos. Sin embargo, a una computadora se le debe indicar exactamente qué operaciones debe ejecutar y en qué orden. A estos conjuntos de instrucciones se les denomina programas para computadora y son elaborados por programadores. Un programa es una serie de instrucciones escritas en forma codificada que la computadora puede traducir a su propio lenguaje.

Cuando se construye una aplicación o se modifica una que ya estaba en uso, se realiza un análisis de sistemas. Este análisis determina los tipos de procesamiento de datos que se requieren, cómo y dónde deben registrarse los datos, los tipos de informe a producirse y los tipos de equipo que mejor se adaptan a esa aplicación particular.

Una vez que se ha decidido lo que se hará con los datos, se le encomienda a un programador la elaboración de un programa o conjunto de instrucciones para el sistema de computación. Al escribir el programa que se va a usar, el programador sigue una serie de etapas, que deben cumplirse.

Cuando a un programador se le pide que elabore un programa, se le suele indicar cómo debe ser la salida y qué datos de entrada están disponibles. En la etapa de análisis de sistemas se deciden estas cuestiones.

Cuando el programador analiza la entrada debe saber qué dispositivos se van a emplear, en qué orden están colocados los campos de datos en los registros y cuál es el tamaño de esos

campos, ya que a menudo los registros contienen más datos de los necesarios para cualquier trabajo.

En el cuerpo del programa se describe qué operaciones se van a ejecutar con los datos; se decide en qué orden se van a ejecutar las operaciones y en qué punto se van a tomar decisiones (ramificaciones).

Por último, el programador analiza los requisitos de la salida: si se debe grabar en cinta o imprimir un listado; si la salida va a ser impresa, cuáles deben ser los títulos y dónde se deben colocar los campos de salida.

Antes de pasar a la siguiente etapa, es necesario hacer todas esas preguntas y encontrar las respuestas. Para ello, el programador tiene que hablar con las personas que diseñaron el sistema y con las personas que utilizan los informes. Con estas últimas hablará principalmente acerca de la forma como desean tener los datos de salida, para que sean de mayor utilidad. Cuando el programador tiene toda la información necesaria acerca del programa, elabora un *algoritmo*.

Los algoritmos son muy útiles porque muestran, en orden, las operaciones que se van a ejecutar con los datos, las comparaciones, y las ramificaciones condicionales e incondicionales que formarán parte del programa.

En la figura se muestra un algoritmo sencillo y elemental, representado mediante un pseudocódigo.

```
INICIO
    LEA: CÓDIGO
    MIENTRAS CODIGO <> 0 HAGA
            LEA: NOMBRES, RENTA, PATRIMONIO
            TOTPAG = 10000 + PATRIMONIO * 0.1 + RENTA * 0.15
            ESCRIBA: CODIGO, NOMBRES, TOTPAG
            LEA: CODIGO
    FIN_ MIENTRAS
FIN_ INICIO
```

Al terminar la elaboración del algoritmo se puede verificar la exactitud general del programa propuesto. El programador puede tomar algunos datos como muestra y suponer que él es el sistema de computación. Él hará con los datos, exactamente, lo que el algoritmo indique.

La salida final que obtenga debe ser exactamente lo que espera. Si no es así, suponiendo que cometió errores por descuido, su algoritmo tiene algunas instrucciones equivocadas y debe ser corregido. Obtener la salida correcta no es garantía de que el programa final será perfecto, debido a que aún quedan muchas cosas por hacer. No obstante, esta simple verificación mostrará las imperfecciones lógicas que pueda haber en sus planteamientos. A la verificación del diagrama detallado se le conoce como prueba de escritorio.

Con base en el algoritmo terminado, se escribe una serie de instrucciones detalladas. Las que se denominan *programa fuente* y suelen estar escritas en lenguaje simbólico. Un lenguaje simbólico es aquél que la mente humana puede entender con relativa facilidad, pero que no puede entender una computadora. La selección del lenguaje simbólico para el programa fuente dependerá de muchos factores. Dos de las consideraciones principales son el tipo de tarea que se programó y la computadora que se utiliza.

Hay muchos lenguajes simbólicos, entre los cuales se encuentran **BASIC, RPG, PASCAL, FORTRAN, PL1, C, C++, JAVA**, conocidos como lenguajes de alto nivel.

Luego de escrito, el programa fuente se almacena en: disco duro, disco compacto, disquetes o en cinta magnética. El sistema de computación no puede entender directamente el lenguaje

simbólico que usó el programador. Antes debe ser traducido a lenguaje de máquina. El compilador efectúa esta traducción.

El compilador es un programa especial que única y exclusivamente traduce el lenguaje simbólico al lenguaje de máquina. Al programa traducido se le denomina *programa objeto* y puede colocarse directamente en la memoria o almacenarse en discos o cintas. La forma en que se almacena el programa depende del sistema de computación que se emplee y del número de veces que se va a correr.

Cuando el programa fuente se desea traducir a lenguaje de máquina, o sea, cuando se compila, sucede otra cosa importante. Dado que el programa fuente puede contener muchos errores, el compilador produce una lista impresa en donde aparece el programa fuente y otra lista de todos los errores que contiene el programa. El programador utiliza esta lista de mensajes de error cuando está buscando y eliminando los errores de su programa. Los errores que detecta el compilador se denominan diagnósticos del tiempo de compilación (errores de sintaxis). Indican en qué instrucción hay un error y de qué tipo es éste. Todos los errores señalados por los *diagnósticos del tiempo de compilación* deben corregirse antes de que el sistema pueda procesar los datos.

Una vez que se han corregido los errores del tiempo de compilación, deben usarse algunos datos para probar el programa objeto. Cuando el sistema de computación ejecuta el programa objeto puede encontrar algunos errores en los datos de entrada. Por ejemplo, es posible que al dispositivo de entrada se le ordene que lea cierto campo que se supone contiene datos numéricos, y en vez de esto se encuentra con datos alfabéticos. Esto hará que la computadora detenga la ejecución del programa objeto e imprima alguna advertencia de error. A estos mensajes se les denomina *diagnósticos del tiempo de ejecución*. Las instrucciones para transferencia de control defectuosas también causarán un error del tiempo de ejecución. Es posible llevar el sistema a un ciclo cerrado, es decir, el sistema ejecutará una secuencia de instrucciones y lo regresa a la primera instrucción de la secuencia. En teoría significa que la computadora ejecutará estas instrucciones para siempre; sin embargo, en la práctica, se fija un límite de tiempo para la ejecución del programa, y si éste no termina antes de que transcurra este tiempo, el sistema se detiene automáticamente e imprime un mensaje indicando porqué terminó la ejecución.

El programa fuente se almacena en memoria auxiliar, después de que se haya depurado completamente y se haya verificado que procesa los datos tal y como lo desea el programador; de esta manera se encuentra disponible para usarse en el momento que se requiera. Cuando se va a correr el programa con datos se extrae del almacenamiento, se compila y se coloca en la memoria principal en la unidad central de procesamiento como un programa objeto en lenguaje de máquina.

La documentación es la etapa final en la elaboración de un programa y tiene lugar después de que se ha almacenado y corregido. Documentar el programa se refiere a la recopilación, en un solo lugar, de toda aquella información de importancia que se usó para elaborar el programa. En un solo lugar se recopilan las descripciones de todas las tareas, los algoritmos, la codificación, y así por el estilo. Si en una fecha posterior el programa revela algún error o tiene que ser modificado, esta documentación le facilitará a cualquier otro programador saber qué se ha hecho y porqué.

También se elabora un manual de corrida (manual del operador) que contiene todas las instrucciones que necesita el operador de la computadora para correr el programa, qué archivos de entrada se van a cargar, qué tipos de salida se deben preparar, etcétera.

1.7. Terminología básica

- ***Soporte físico (Hardware).*** Se denomina así a los componentes mecánicos, eléctricos y electrónicos o *duros* de una organización computarizada. Los componentes de los circuitos pueden ser activos, pasivos o ambas cosas.

- ***Soporte lógico (Software).*** El término software fue concebido para contrastar con el hardware de un sistema de computadora. Los elementos constituyentes del software son los programas, lenguajes y procedimientos, los programas o rutinas internas preparadas en forma profesional para simplificar la programación y las operaciones de las computadora, lo mismo que las diversas ayudas que se suelen suministrar por los fabricantes para facilitar la explotación eficiente de los equipos.

- ***Programa fuente.*** Es un programa que puede traducirse en forma automática en lenguaje de máquina. Está escrito en un lenguaje concebido para facilitar la expresión de una clase de problemas o de procedimientos por parte de seres humanos; utiliza un compilador, ensamblador o traductor para realizar la mecánica de la traducción del programa fuente en un programa objeto en lenguaje de máquina.

- ***Programa objeto.*** Es un programa fuente que ha sido traducido a lenguaje de máquina. El programa fuente se convierte, entonces, en un conjunto de instrucciones de lenguaje de máquina para la solución de un problema, obtenido como el resultado final de un proceso de compilación.

- ***Ensamblador.*** El ensamblador traduce las instrucciones simbólicas de un lenguaje de programación a las instrucciones necesarias de máquina para que sean ejecutadas. En los lenguajes ensambladores se define un código especial (llamado nemónico) para cada una de las operaciones de la máquina y se introduce una notación especial para especificar el dato con el cual debe realizarse la operación. El ensamblador origina una instrucción en lenguaje de máquina por cada instrucción del lenguaje a traducir. En general es utilizado con los lenguajes de bajo nivel.

- ***Compilador.*** Son traductores más complejos. Traduce los programas que realiza el usuario al lenguaje de máquina necesario para que sean ejecutados. Además de hacer dicha traducción, el compilador chequea que el programa fuente esté correcto en cuanto a sintaxis e instrucciones propias de cada lenguaje. Los compiladores para la mayor parte de los lenguajes de programación los proporciona el fabricante.

  El compilador genera una o más instrucciones en el lenguaje de máquina por cada instrucción del lenguaje de programación. Se utiliza en los lenguajes de alto nivel. La traducción, en general, se realiza en dos pasos: primero las instrucciones en lenguaje de alto nivel se traducen a una secuencia de instrucciones en el lenguaje de bajo nivel y a continuación se efectúa la traducción al lenguaje de máquina. El compilador asigna las posiciones de memoria por cada variable que encuentra conforme lee y traduce las instrucciones.

- ***Interpretador.*** Es un programa de computador que traduce cada instrucción o sentencia del lenguaje original a una secuencia de instrucciones de máquina, antes de traducir la siguiente sentencia en lenguaje original.

- ***Lenguaje de computadora.*** Conjunto definido de caracteres que sirven para formar símbolos, palabras, instrucciones, etcétera, así como las reglas para combinar estos caracteres, dando como resultado comunicaciones significativas, por ejemplo **FORTRAN, PASCAL, PROLOG, C, C++, JAVA**.

- ***Lenguaje de máquina.*** Es el conjunto final que todas las computadoras deben utilizar (lenguaje binario). Todos los demás lenguajes de programación deben compilarse o traducirse, a la larga, a un código binario antes de introducirse en el procesador. El lenguaje binario es el lenguaje de máquina.

- ***Lenguaje de bajo nivel.*** Los lenguajes de bajo nivel son el primer paso de la sofisticación de los lenguajes de computadora. Permiten una comunicación más fácil con la computadora. Las instrucciones escritas en lenguaje de bajo nivel deben ser traducidas por medio de una traductora a lenguaje de máquina. La traducción que se hace es uno a uno, es decir, una instrucción en lenguaje de bajo nivel se transforma en una instrucción de lenguaje de máquina.

- ***Lenguaje de alto nivel.*** Es el lenguaje simbólico más fácil de entender y de programar: en él se prepara un programa para ser procesado por computadora.

  Este lenguaje es traducido a lenguaje objeto mediante un ensamblador o compilador.

  A diferencia de los anteriores éste es más flexible, fácil de aprender y más apropiado para corregir los errores de programación.

  Una instrucción de lenguaje de alto nivel es traducida a lenguaje de máquina en una o varias instrucciones. En la medida en que se simplifica el trabajo del programador, crece el trabajo del traductor.

- ***Dirección de memoria.*** Cada computador tiene una cantidad determinada de *almacenamiento interno* denominada memoria principal, además del almacenamiento externo consistente en cintas, disquetes y discos magnéticos. Las dos formas de almacenamiento difieren en sus características y propósitos. Los programas que están en ejecución, así como los datos necesarios para ello, deben residir en la memoria principal. Esta es más costosa y, en la mayor parte de los sistemas, es un recurso un tanto escaso. La información que no se requiere de inmediato se relega normalmente a los dispositivos de almacenamiento externo, donde la capacidad de almacenamiento puede ser casi ilimitada, pero donde el tiempo de recuperación de la misma suele ser considerablemente más grande. Al mismo tiempo el costo de almacenamiento externo es comparativamente más bajo, por lo que la información puede ser almacenada allí por largos períodos.

  La memoria principal de la computadora se divide en unidades pequeñas e iguales denominadas *palabras* que tienen una única dirección. Cada palabra es capaz de almacenar una unidad de información; por ejemplo, los resultados numéricos de la computadora. El tamaño de la palabra es un parámetro de diseño de la computadora y determina, digamos, el número más grande y más pequeño que puede ser almacenado.

  Las direcciones de memoria identifican cada palabra individualmente, de tal manera que la información contenida en ella pueda leerse, o almacenar nueva información.

- ***Bit, Byte, Palabra***

  Bit. Es la abreviatura de *binary digit*. Es una unidad de información que equivale a una decisión binaria, o la designación de uno de dos valores o estados posibles igualmente probables. Se suele expresar como ‘l’ o ‘0’. Representa una unidad minúscula de memoria que puede tener dos estados *encendido*, o *apagado*, donde el estado *encendido* se representa mediante uno y el *apagado* mediante cero.

  Byte. Es un conjunto de **BITS** consecutivos que se tratan como una sola entidad, por lo general son **8 BITS.** También se le conoce como el número de **BITS** necesarios para representar un carácter en la memoria de la computadora.

  Palabra. Son unidades de almacenamiento a las que se les asigna una dirección de memoria. Son tratadas por los circuitos de la computadora como una entidad, por la unidad de control como una instrucción y por la unidad aritmética como una cantidad. Este grupo de **BITS** es el más grande tratado a través del procesador central como una sola unidad. El tamaño de la palabra depende de la arquitectura de la computadora; se pueden tener palabras de uno, dos, cuatro, **ocho o más BYTES.**

- ***Campo.*** Es un grupo de caracteres que se pueden tratar como una unidad de información simple. Las instrucciones de los lenguajes están constituidas por varios campos. Dependiendo de la computadora, del lenguaje de programación utilizado y de la forma como el campo sea definido, a un campo se le asignará una o más palabras.
- ***Registro.*** Es un conjunto de campos que se refieren a una misma actividad que se desea procesar; por ejemplo, toda la información sobre un artículo de inventario, la información que contiene una lista de clase sobre un estudiante (código, nombres, programa, etc.).
- ***Archivo.*** Es un dispositivo de software utilizado para almacenar información en forma masiva, relacionada con una determinada entidad, persona o cosas. Ejemplo: suponga que la compañía A tiene 100 clientes y que se mantiene la siguiente información acerca de cada uno de ellos: nombre, dirección, balance de crédito, descuento permitido. La información individual de cada cliente se denomina *registro* y un conjunto de registros formará un archivo, es decir, un archivo de datos de los registros de los clientes. Los archivos pueden tenerse en varias formas y en diferentes medios de almacenamiento.
- ***Base de datos.*** Es un conjunto de archivos relacionados de una manera especial, que continuamente se está actualizando.

1.8. Sistemas numéricos

Un sistema numérico en base r es un sistema que utiliza distintos símbolos para representar r dígitos. Los números se representan por hileras de símbolos de dígitos. Para determinar la cantidad que el número representa es necesario multiplicar cada dígito por una potencia entera de r y luego ejecutar la suma de todos los dígitos pesados. (Dígito pesado es aquél que tiene dos valores: uno intrínseco y otro posicional). Por ejemplo, el sistema numérico decimal, que es el que utilizamos a diario, está en base 10 (r=10), lo cual significa que utiliza diez dígitos para representar todas las cantidades posibles. Los diez símbolos utilizados por dicho sistema son: 1, 2, 3, 4, 5, 6, 7, 8, 9 y 0. La hilera de dígitos 724.5 se utiliza para representar la cantidad:

$$7x10^2 + 2x10^1 + 4x10^0 + 5x10^{-1}$$

Esto es, setecientos más 2 decenas más 4 unidades + 5 décimos. El sistema numérico binario utiliza la base dos. Los dos dígitos que utiliza son el 0 y el 1. La hilera de dígitos 101101 se utiliza para representar la cantidad:

$$1x2^5 + 0x2^4 + 1x2^3 + 1x2^2 + 0x2^1 + 1x2^0 = 45$$

Para distinguir los números de diferentes bases, los dígitos serán encerrados entre paréntesis y la base del número será escrita como un subíndice. Por ejemplo, para mostrar la igualdad entre el 45 en base 10 y en base 2, escribiremos:

$$(101101)_2 = (45)_{10}$$

Además de los sistemas numéricos decimal y binario, los sistemas octal (base 8) y hexadecimal (base 16) son importantes en el trabajo de computadoras digitales. Los ocho símbolos o dígitos del sistema octal son: 0, 1, 2, 3, 4, 5, 6, 7, Los dieciséis símbolos o dígitos del sistema hexadecimal son: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Los últimos seis símbolos son desafortunadamente idénticos a las letras del alfabeto y pueden causar alguna confusión, a veces. Sin embargo, esto se utiliza por convención, y los símbolos A, B, C, D, E, F cuando se utilizan para representar dígitos hexadecimales corresponden a los números decimales 10, 11, 12,13,14,15, respectivamente.

Un número en base r se puede convertir al familiar sistema decimal ejecutando la suma de sus dígitos pesados. Por ejemplo, el número $(73,4)_8$ para convertirlo a base 10 se hace como sigue:

$$(736.4)_8 = 7x8^2 + 3x8^1 + 6x8^0 + 4x8^{-1}$$

Y se ejecutan las operaciones en sistema decimal, esto es:

$$7x64 + 3x8 + 6x1 + 4/8 = (478,5)_{10}$$

El equivalente número decimal del hexadecimal F3 se obtiene efectuando los siguientes cálculos:

$$(F3)_{16} = Fx16^1 + 3x16^0 = 15x16 + 3 = (243)_{10}$$

La conversión desde decimal a su equivalente representación en base r se ejecuta separando el número en su parte entera y su parte decimal y convirtiendo cada parte por separado. La conversión de un entero decimal a su equivalente en base r se ejecuta mediante divisiones sucesivas por la base r y se acumulan los residuos en orden inverso a como fueron apareciendo. La conversión de la parte decimal a su equivalente en base r se obtiene mediante multiplicaciones sucesivas por la base r y se acumulan los dígitos enteros obtenidos en el orden de aparición.

El siguiente ejemplo ilustra estos procedimientos:

La conversión del número decimal 41,6875 a binario se ejecuta separando primero la parte entera 41 y la parte decimal 0,6875 y dividiéndolos sucesivamente por la base (2).

ENTERO 41		FRACCION 0.6875				
41 |						
20 | 1		0.6875	*	2	=	1.3750
10 | 0		0.3750	*	2	=	0.7500
5 | 0		0.7500	*	2	=	1.5000
2 | 1		0.5000	*	2	=	1.0000
1 | 0						(fracción cero)
0 | 1	(cociente cero)	$(0.6875)_{10}$ = $(0.1011)_2$				

$$(41)_{10} = (101001)_2$$

Por consiguiente $(41.6875)_{10} = (101001.1011)_2$

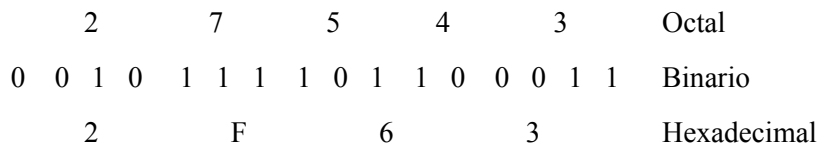
La parte entera se convierte dividiendo 41 por la base a la cual se va a trasladar **(r = 2)** y se obtiene un cociente de 20 y un residuo de 1. El cociente se divide de nuevo por la nueva base **(r = 2)** y se obtiene un nuevo cociente y un nuevo residuo. Este proceso se repite hasta que el cociente sea cero. El número equivalente en binario será entonces el residuo obtenido en el orden inverso de aparición.

La parte fraccionaria se convierte multiplicándola por la nueva base **(r=2)** y se obtiene una parte entera y una fracción. La nueva fracción (sin la parte entera) se multiplica de nuevo por la nueva base **(r=2)** para obtener una nueva parte entera y una nueva fracción. Este proceso se repite hasta que la parte fraccionaria llegue a ser cero o hasta que el número de dígitos obtenidos dé la precisión deseada. La fracción binaria se obtiene de los dígitos enteros obtenidos al realizar cada multiplicación, colocándolos en el orden de aparición. Finalmente, se unen la parte entera y fraccionaria y se obtiene el resultado requerido.

Representación binaria, octal y hexadecimal

La conversión desde y hacia, de representación numérica binaria, octal y hexadecimal, juega un papel muy importante en computadores digitales. Debido a que dos al cubo es igual a ocho y dos a la cuarta es igual a dieciséis, cada dígito octal corresponde a tres dígitos binarios y cada dígito hexadecimal corresponde a cuatro dígitos binarios. La conversión de binaria a octal se ejecuta fácilmente dividiendo el número binario en grupos de tres dígitos de derecha a

izquierda. A cada grupo de tres dígitos se le asigna el correspondiente dígito octal, y la hilera de dígitos así obtenida da la representación octal del número binario. Consideremos un registro de 16 bits. Físicamente uno puede pensar que el registro está compuesto por 16 celdas de almacenamiento binario, cada una de las cuales es capaz de almacenar un valor, cero o uno. Supongamos que la configuración de bits almacenada en un registro es como la mostrada en la figura:



Debido a que un número binario consiste en una hilera de unos y ceros, el registro de 16 bits se puede utilizar para almacenar un número binario desde uno hasta 2 a la 16 menos 1. Para el particular ejemplo mostrado en la figura, el número almacenado en binario es equivalente al decimal **12131.**

Comenzando desde la derecha, partimos el registro en grupos de tres bits. A cada grupo de tres bits se le asigna su equivalente octal y se coloca en la parte superior del registro. La hilera de dígitos octales, así obtenida, representa el equivalente octal del número binario. La conversión de binario a hexadecimal es similar, excepto que los dígitos se agrupan en conjuntos de cuatro dígitos. El correspondiente dígito hexadecimal para cada grupo de cuatro dígitos se anota en la parte inferior del registro. La hilera de dígitos, así obtenida, representa el número hexadecimal equivalente al número binario.

El correspondiente dígito octal para cada grupo de tres bits se encuentra en la siguiente tabla:

Número octal	Octal codificado en binario	Decimal equivalente
0	000	0
1	001	1
2	010	2
3	011	3
4	100	4
5	101	5
6	110	6
7	111	7

La correspondencia entre dígitos hexadecimales y su equivalente en grupos de cuatro bits se encuentran en la siguiente tabla:

Número hexadecimal	Hexadecimal codificado en binario	Decimal equivalente
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7
8	1000	8
9	1001	9
A	1010	10
B	1011	11

C	1100	12
D	1101	13
E	1110	14
F	1111	15

1.9. Representación de datos

La información binaria en computadoras digitales está almacenada en memoria, en registros procesadores. Los registros pueden contener datos o información de control. La información es un bit o grupo de bits, utilizado para especificar la secuencia de comandos de señales necesarias para manipular los datos de otros registros. Los datos son números u otra información codificada en binario operados para lograr los resultados computacionales requeridos. Aquí se presentarán los más comunes tipos de datos encontrados en computadores digitales y se mostrará cómo los diferentes tipos de datos son representados en forma codificada en binario en los registros de computadores.

Los tipos de datos encontrados en los registros de computadoras digitales se pueden clasificar en alguna de las siguientes categorías: a) números, utilizados en cálculos aritméticos; b) letras del alfabeto, utilizadas en procesamiento de datos, y c) otros símbolos, utilizados para propósitos específicos. Todos los tipos de datos, excepto números binarios, se representan en registros de computadores en forma codificada en binario. Esto se debe a que los registros son construidos con elementos biestables, los cuales son dispositivos que sólo pueden almacenar unos o ceros. El sistema numérico binario es el sistema natural en computadoras digitales. Pero, algunas veces es conveniente emplear diferentes sistemas numéricos, especialmente el sistema numérico decimal, el cual es el ejecutado por los humanos para efectuar cálculos aritméticos.

El caso es que una hilera de unos y ceros almacenados en un registro puede representar un número binario, un número octal codificado en binario o un número hexadecimal codificado en binario. Los registros en computadoras digitales contienen muchos bits. Si especificamos su contenido por su representación en binario, requeriremos una hilera muy larga de dígitos binarios. Es más conveniente especificar el contenido de los registros por su equivalente octal o hexadecimal. El número de dígitos se reduce en una tercera parte en representación octal, y en una cuarta parte en representación hexadecimal. Por ejemplo, el número binario 111111111111 tiene doce dígitos. Si lo expresamos en forma octal utiliza cuatro dígitos (7777), en hexadecimal utiliza tres dígitos (FFF). Los computadores utilizan la representación octal o hexadecimal para especificar el contenido de los registros.

Representación alfanumérica

Muchas aplicaciones de computadores digitales requieren el manejo de datos que no son únicamente números, sino también letras del alfabeto y ciertos caracteres especiales. Un conjunto de caracteres alfanuméricos es un conjunto de elementos que incluye los diez dígitos decimales, las 26 letras del alfabeto y caracteres especiales tales como $, +, =, *, etc. Tal conjunto contiene entre 32 y 64 (si se incluyen únicamente letras mayúsculas), entre 64 y 128 (si se incluyen letras mayúsculas y minúsculas), o entre 128 y 256 (si se incluyen caracteres especiales). En el primer caso se requerirán códigos binarios de seis bits, en el segundo caso, códigos binarios de siete bits y en el tercer caso códigos binarios de 8 bits.

Es bueno aclarar que el computador siempre utilizará un byte (un conjunto de 8 bits) para almacenar datos alfanuméricos. En algunos sistemas de codificación utiliza los 8 bits para representar los caracteres; en otros, como el ASCII, utiliza 7 bits para representar los caracteres y el octavo como bit de paridad para controlar la transmisión de datos alfanuméricos.

El código binario estándar alfanumérico es el ASCII *(American National Standard Code For Information Interchange)*, el cual utiliza ocho bits para codificar 256 caracteres. Los códigos

binarios para letras mayúsculas, dígitos decimales y unos cuantos caracteres especiales se muestran en la siguiente tabla:

Carácter binario	Código	Carácter binario	Código
A	1000001	0	0110000
B	1000010	1	0110001
C	1000011	2	0110010
D	1000100	3	0110011
E	1000101	4	0110100
F	1000110	5	0110101
G	1000111	6	0110110
H	1001000	7	0110111
I	1001001	8	0111000
J	1001010	9	0111001
K	1001011		
L	1001100	Blanco	0100000
M	1001101		0101110
N	1001110	(	0101000
O	1001111	+	0101011
P	1010000	$	0100100
Q	1010001	*	0101010
R	1010010	)	0101001
S	1010011	-	0101101
T	1010100	/	0101111
U	1010101	,	0101100
V	1010110	=	0111101
W	1010111		
X	1011000		
Y	1011001		
Z	1011010		

Los códigos binarios juegan un papel muy importante en las operaciones en computadoras digitales. Los códigos deben estar en binario debido a que los registros sólo pueden guardar información binaria. Las operaciones especificadas para computadoras digitales deben tener en cuenta el significado de los bits almacenados en los registros, de tal manera que las operaciones se ejecuten sobre operandos del mismo tipo. Los códigos binarios se pueden asignar para cualquier conjunto de elementos: colores, notas musicales, piezas de ajedrez, etc. Los códigos binarios se utilizan también para formular instrucciones que especifiquen información de control a la computadora. Aquí nos ocuparemos únicamente de la representación de datos.

Representación numérica

Los números utilizados en cálculos se designan por un signo, la magnitud del número y algunas veces un punto decimal. El signo se utiliza para determinar si el número es positivo o negativo. La posición del punto decimal se utiliza para representar fracciones o números con parte entera y fraccionaría.

El signo del número se puede considerar como un conjunto de dos elementos, el más y el menos. Estos dos elementos se pueden asignar con un código binario de un bit. Por convención, el cero representará el signo más y el uno el signo menos. El bit del signo, por lo general, se coloca en la posición más a la izquierda del registro.

La representación del punto decimal o binario en un registro es complicado, debido a que el punto debe estar entre dos elementos biestables del registro. Hay dos formas de especificar la posición del punto decimal en un registro: dándole una posición fija o empleando la

representación de punto flotante. El método de punto fijo asume que el punto decimal está siempre en una posición fija. Las dos posiciones más comúnmente usadas son: a) el punto decimal en el extremo izquierdo del registro, para tener el número almacenado como una fracción, y b) el punto decimal en el extremo derecho de un registro, que hace que el número almacenado sea un entero. En cualquiera de los dos casos el punto decimal no está realmente presente, pero su presencia es asumida para que el número sea tratado como fracción o como entero. La representación de **punto** flotante utiliza un segundo registro, o parte del registro, para almacenar un número que designa la posición real del punto decimal.

Antes de explicar cómo los números de punto fijo se representan en los registros es necesario definir el complemento de un número. Los complementos se utilizan en computadoras digitales para representar números negativos debido a que su representación facilita las operaciones aritméticas. Hay dos tipos de suplementos para cada sistema numérico en base **r**: a) el complemento a **r**, y b) el complemento a **r-1**. Cuando la base es dos (sistema binario) los complementos reciben los nombres de complemento a dos y complemento a uno, respectivamente; si la base es diez (sistema decimal) los nombres serían complemento a diez y complemento a nueve, respectivamente.

El complemento **r-1** de un número en base de **r** se obtiene restando cada dígito del número **r-1**. Para números decimales **r-1** es nueve, y para números binarios **r-1** es 1. Por consiguiente, el complemento a nueve del decimal 835 es 164 y se obtuvo restando cada dígito de nueve; el complemento a uno del número binario 1010 es 0101 y se obtuvo restando cada dígito de uno. Sin embargo, considerando que 1-0=1 y 1-1=0 el complemento a uno de números binarios se obtiene cambiando los unos por ceros y los ceros por unos.

El complemento **r-1** de números octales y hexadecimales se obtiene restando cada dígito de 7 o F, respectivamente. Cuando estos números están codificados en binario, el complemento se obtiene cambiando unos por ceros y ceros por unos.

El complemento a **r** de un número en base **r** se obtiene sumando 1 al complemento a **(r-1)**. Por consiguiente, el complemento a 10 del número decimal 835 es 164+1=165, el cual se obtuvo sumando uno al complemento a 9. El complemento a 2 del número binario 1010 es 0101+1=0110, y se obtuvo sumando uno al complemento a uno.

Representación numérica entera

Cuando un número binario de punto fijo es positivo, el signo se representa por un cero y la magnitud por un número positivo binario. Cuando el número es negativo el signo se representa por un uno, pero el resto del número se puede representar de tres maneras diferentes que son: representación signo–magnitud; representación complemento a uno con signo y representación complemento a dos con signo.

La representación signo-magnitud de un número negativo es el bit del signo seguido por la magnitud del número. En las otras dos representaciones, el número negativo se representa o en complemento a uno o en complemento a dos. Como ejemplo consideremos el número 9 almacenado en un registro de 7 bits. Si el número 9 es positivo su representación será 0001001, la cual es única. El bit que está a la izquierda es cero indicando que el número es positivo, los otros seis bits representan el número 9 en binario. Sólo existe una manera de representar el +9; sin embargo, existen tres maneras diferentes para representar el -9, las cuales son:

- Representación signo-magnitud: 1001001
- Representación signo-complemento a uno: 1110110
- Representación signo-complemento a dos: 1110111

La representación signo-magnitud se obtiene de la representación de +9 (0001001) cambiando solamente el bit del signo.

La representación signo-complemento a uno de (-9) se obtiene cambiando todos los bits de la representación de +9 (1001001); los unos por ceros y los ceros por unos, excluyendo el signo del bit.

La representación signo-complemento a dos se obtiene hallando el complemento a dos de la representación de +9 (1001001), excluyendo el bit del signo.

La razón por la que se utilizan las representaciones signo-complemento a uno y signo-complemento a dos, es que facilitan la ejecución de las operaciones. Veamos el siguiente ejemplo: +23 y -35 se representan con un signo seguido por la magnitud del número. Para sumar estos dos números es necesario sustraer la magnitud más pequeña de la mayor y utilizar el signo de la cantidad mayor como signo del resultado, esto es:

$$(+23) + (-35) = - (35-23) = - 12$$

El proceso de sumar dos números signados, cuando los números negativos se representan en la forma signo-magnitud, requiere que se comparen sus signos. Si los dos signos son iguales se suman las magnitudes y el resultado tiene el mismo signo. Si los signos son diferentes se comparan las magnitudes (en valor absoluto) de ambos números y del mayor se resta el menor, y el signo del resultado será el del número mayor; se necesita, por consiguiente, determinar el signo del resultado. Todo lo anterior es un proceso que cuando se implementa por *hardware* requiere una larga secuencia de decisiones y de controles, así como circuitos que comparen, sumen y resten números.

Comparemos ahora el procedimiento anterior con el procedimiento que se sigue cuando se suman dos números binarios signados o cuando los números negativos se representan en la forma signo-complemento a dos. Este procedimiento es muy simple y se puede establecer como sigue: se suman ambos números (incluido el signo del bit) y se descarta cualquier acarreo al sumar los dígitos más a la izquierda de los números.

A continuación se muestran algunos ejemplos de cómo se suman dos números negativos, cuando se representan en la forma signo-complemento a dos. (Utilizando estructuras de 7 bits).

```
 + 6    0000110          - 6    1111010
          +                       +
 + 9    0001001          + 9    0001001
 ----   -------          ----   -------
 +15    0001111            + 3  0000011


 + 6    0000110          - 9    1110111
          +                       +
 - 9    1110111            - 9  1110111
 ----   -------          ----   -------
 - 3    1111101            -18  1101110
```

Note que los números negativos deben estar inicialmente en representación signo-complemento a dos, y el resultado obtenido después de la suma, si es negativo, está también en la representación signo-complemento a dos. Los dos números en los cuatro ejemplos se suman, incluyendo el bit del signo, y cualquier acarreo en el bit del signo se descarta.

Este procedimiento es mucho más simple que el utilizado para sumar números en representación signo-magnitud. Requiere solamente una decisión y un circuito para sumar dos números. El procedimiento requiere que los números negativos sean almacenados en la forma signo-complemento a dos.

El procedimiento que se sigue para obtener la suma de dos números binarios, cuando los números negativos están en la representación signo- complemento a uno, es similar y se puede enunciar como sigue: sume los dos números, incluyendo el bit del signo. Si resulta algún acarreo luego de sumar los bits del signo se le suma uno (1) al resultado y el acarreo se descarta. Los siguientes ejemplos ilustran este procedimiento:

```
   +6     0000110          -6     1111001
                                              +
   -9     1110110               +9     0001001
  ----   ---------          ----   ---------
   -3     1111100                     0000010

+

   1

                                    ---------
                                     0000011
```

Note que todos los números negativos, incluyendo los resultados, están en la representación signo-complemento a uno.

La ventaja de la representación signo-complemento a dos sobre las otras dos representaciones (signo-complemento a dos y signo-magnitud) es que solamente tiene una representación para el cero. Las otras dos representaciones tienen cero positivo y cero negativo. Por ejemplo, para sumar (+9) y (-9) en representación signo-complemento a uno se tiene:

```
  +9     0001001
                +
  -9     1110110
 ----   ---------
  -0     1111111
```

Y el resultado es un cero negativo, es decir, el complemento del cero positivo (0000000).

El número cero con el bit de signo asociado aparecerá en un registro en algunas de las siguientes formas, dependiendo de la representación utilizada para números negativos:

	+0	-0
Sígno-magnitud	0000000	1000000
Signo-complemento a uno	0000000	1111111
Signo-complemento a dos	0000000	Ninguna

En las primeras dos representaciones existe cero positivo y cero negativo. En la representación signo-complemento a dos sólo existe el cero positivo; esto debido a que el complemento a dos de 0000000 (cero positivo) es 0000000.

Resta aritmética. La resta de dos números binarios signados, cuando los números negativos están representados en la forma signo-complemento a dos, es muy simple y se puede enunciar así:

Tome el complemento a dos del sustraendo (incluyendo el bit del signo) y súmela al minuendo (incluyendo el bit del signo).

Este procedimiento se basa en el hecho de que la operación de resta se puede convertir en una operación de suma, cambiando el signo sustraendo.

Cambiar un número positivo por uno negativo es muy fácil: basta con hallar su complemento a dos (incluyendo el bit del signo). La operación contraria también es cierta, debido a que el complemento a dos del complemento a dos restaura el número en su valor original.

Debido a la disponibilidad de procedimientos simples para sumar y restar números, cuando los números negativos se representan en la forma signo-complemento a dos, la mayoría de los computadores adoptan esta representación, en vez de la familiar signo-magnitud. Otra de las razones por la cual se escoge la representación signo-complemento a dos es porque evita la presencia del cero negativo.

En los ejemplos que se han visto anteriormente se han utilizado estructuras de 7 bits para representar los números enteros. Cada computador utiliza estructuras diferentes y de diferente cantidad de bits para representar los datos en memoria.

Representación numérica de punto flotante

La representación de un número con decimales (real) necesita dos partes: la primera parte representa un número de punto fijo, con signo llamado **Mantisa.** La segunda parte la conforma la representación del exponente, que indica la posición real del punto. Cuando se usa una estructura (palabra) de 32 bits, el exponente está representado por un valor denominado **característica** y la mantisa debe estar normalizada, lo cual indica que la posición más significante de ésta es un número diferente de cero. Esto nos permite decir que en una estructura como ésta los números reales se almacenan en la forma **mantisa, característica,** donde el punto no está representado sino asumido, y puede asumirse a la izquierda o a la derecha de la mantisa; generalmente se asume a la izquierda, lo que indica que la mantisa es tomada como una fracción.

El signo del valor representado está en el bit más a la izquierda de la palabra, si el valor almacenado en ese bit es uno, el valor representado es negativo y si es cero es positivo.

La distribución de los bits de la palabra, para representar el signo, la mantisa y la característica se puede hacer de dos formas, dependiendo de la manera como esté representado el exponente; representación con exponente en base 2; representación con exponente en base 16.

a. Representación con exponente en base 2

En este caso la distribución de los 32 bits se hace de la siguiente manera:

- **Signo:** bit 1.
- **Mantisa:** del bit 2 al 24 y es representada como un número binario.
- **Característica:** del bit 25 al 32 y es representada como un número decimal codificado en binario.

El valor almacenado en la estructura es: ± (. mantisa * 2^e).

Donde *e* es el exponente y se asume el punto a la izquierda de la mantisa.
Por ejemplo, si se tiene la siguiente representación, veamos cual es su valor almacenado:

```
                1                   2                   3
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0 1 1 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 1
s                   mantisa                   Característica
```

Primero se convierte la característica a base 10, $(10000011)_2 = (131)_{10}$

Segundo se determina el valor del exponente, e = (característica)$_{10}$ – 128 (el sesgo de 128 indica que se pueden representar exponentes positivos y negativos). $e = 131 - 128 = 3$

Tercero el valor almacenado es: $(.11011 \times 2^3)_2 = (110.11)_2 = (6{,}75)_{10}$

Ahora hagamos el paso contrario, almacenar en una estructura de 32 bits que utiliza el exponente en base 2 el número –34,25

Primero se pasa el número a binario. El bit de signo será 1.

$(34{,}25)_{10} = (100010.01)_2$

Segundo se denota el valor del exponente (*e*). Como la mantisa debe tomarse como una fracción, entonces (100010.01) = $(.10001001 \times 2^6)_2$, por lo tanto $e = 6$ y la mantisa 10001001.

Tercero se determina el valor de la característica.

Característica = $e + 128 = 6 + 128 = 134$

Cuarto se convierte la característica a binario.

$(134)_{10} = (10000110)_2$.

La estructura quedaría así:

```
                1                   2                   3
1 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2     1         1
  0 0 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 1 1 0
```

s mantisa Característica

b. Representación con exponente en base 16

En este caso la distribución de los 32 bits se hace de la siguiente manera:

- **Signo:** bit 1.
- **Característica:** del bit 2 al 8 y es representada como un número decimal codificado en binario.
- **Mantisa:** del bit 9 al 32 y es un número hexadecimal codificado en binario.

Si se tiene la siguiente estructura, calculemos su valor almacenado:

```
                1                   2                   3
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
0 1 0 0 0 0 1 1 1 1 0 1 1 1 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0
```

s característica mantisa (hexadecimal codificado en binario)

El valor almacenado será = $(.\text{mantisa} * 16^e)$ y e=característica-64

Primero se convierte la característica a base 10.

$(1000011)_2 = (67)_{10}$

Segundo se determina el valor del exponente.

$e = (\text{característica})_{10} - 64$, o sea $67-64 = 3$

Tercero se convierte la mantisa a base 16.

$(.11011110001)_2 = (.DE2)_{16}$

Cuarto el valor almacenado será

$(.DE2 * 16^3)_{16} = (DE2)_{16} = (3554)_{10}$

Ejemplo

Almacenar en una estructura de 32 bits que utiliza exponente en base 16 el decimal -1017.5

- Se convierte el número a base 16
  $(-1017.5)_{10} = (-3F9.8)_{16}$
- Se calcula el exponente
  $(-3F9.8)_{16} = (-.3F98*16^3)$

  Por lo tanto e = 3 y la mantisa será $(-.3F98)_{16}$
- Se convierte la mantisa a binario
  $(-.3F98)_{16} = (-.0011111110011)_2$
- Se calcula el valor de la característica
  característica = e + 64 = 3 + 64 = 67
- Se convierte la característica a binario
  $(67)_{10} = (1000011)_2$

  La estructura quedaría así:

```
                1               2               3
1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2
1 1 0 0 0 0 1 1 0 0 1 1 1 1 1 1 1 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0
```

s característica mantisa

1.10. Tipos de campos

Para que una computadora pueda hacer cálculos debe tener almacenados en la memoria principal los valores con los cuales ha de trabajar; por lo tanto, se necesita un espacio interno para guardar esos valores en forma temporal.

Esos espacios internos de la memoria son grupos de bits biestables denominados **campos;** a éstos se les asigna una única dirección de memoria y su capacidad o valor máximo que se puede almacenar en ellos dependerá del tamaño del grupo. Los campos pueden ser de dos clases:

***Campos variables*:** son todos aquellos campos que permiten que el contenido almacenado en el espacio de memoria asignado, pueda ser alterado en el transcurso de la ejecución del programa; o sea, en un momento tiene un valor y más adelante puede tener otro distinto, pero nunca más de un valor al mismo tiempo. Esta clase de campos, que en lo sucesivo se denominarán simplemente **variables,** se identifican por un nombre con el cual actúan durante todo el transcurso del proceso, por lo tanto, deben ser únicos. El nombre dado a la variable debe iniciar con una letra seguida, si se quiere, de letras o dígitos. Es el programador quien da nombre a sus variables teniendo en cuenta la regla anterior.

Ejemplos de nombres de variables:

A
B1
C1AZ
NOMBRE
SALARIOHORA

Es buena práctica de programación utilizar nombres de variables significativas (nemotécnicas), es decir, que sugieran lo que ellas representan, ya que esto hace que el algoritmo, y por ende el programa, sea más legible y fácil de comprender.

Si se quiere dar nombre a una variable que va a almacenar un pago, por ejemplo, perfectamente lo podemos llamar K, pero sería más diciente si la denominamos **PAG** o más

directamente **PAGO.** Tampoco estos nombres deben ser tan e*xcesivamente largos debido a que dificultan la escritura del algoritmo.*

Campos constantes: es otra forma de manejar el grupo de elementos asignados en memoria, pero que a diferencia de las variables su contenido no puede cambiar durante el proceso. Esta clase de campos se identifica directamente por la información que tiene almacenada y es asignada desde el momento en que se hace la compilación. Pueden existir campos constantes, o simplemente constantes, de tipo numérico, carácter o lógico. Por ejemplo:

40,6 'ANAISABEL' -10 .V.

1.11. Clases de información

Los principales tipos de información, entre otros, que se manejan en una computadora son:

Información numérica: es toda aquella información con o sin decimales (entera o real) con la cual se pueden hacer cálculos aritméticos o matemáticos. Esta información puede estar representada como una constante o una variable. Por ejemplo:

- Como constantes:
  20
  30,6
  2000

- Como variables cuyo contenido siempre será numérico:

NUMERO	PAGO	CANTIDAD
20.666	110.5	4556

Información como caracteres: muchos procesos o programas no sólo requieren de la manipulación de valores numéricos, sino también de letras y símbolos especiales con los cuales, lógicamente, no se pueden efectuar operaciones de cálculo matemático. Por ejemplo: los nombres de personas.

Esta información también puede representarse como constante o como variable. Por ejemplo:

- Como constante:
  "A" Las comillas indican que se trata del carácter A y no de la variable A.
  "JUAN PABLO"
  "ALBERTO"
  "1" Muy diferente al número 1, su forma de almacenamiento es distinta.
  "*B4"
  "+* _"

- Como variable:

LETRA	NOMBRE 1	NOMBRE 2	CARÁCTER
A	JUAN PABLO	ALBERTO	*B4

Información lógica: este tipo de información sólo tiene dos posibles valores: **falso o verdadero.** Al igual que el tipo de información anterior, éste tampoco puede ser utilizado para efectuar cálculos matemáticos. Se utilizarán las notaciones:

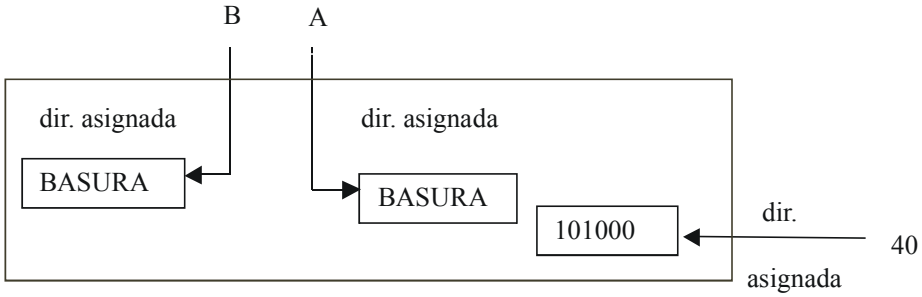
.**V**. para representar el valor verdadero, y

.**F**. para representar el valor falso, con el fin de diferenciar estos valores con las variables V y F. Esta información también puede ser el contenido de una variable, por ejemplo:

ENCONTRADO	VERDAD
.F.	.V.

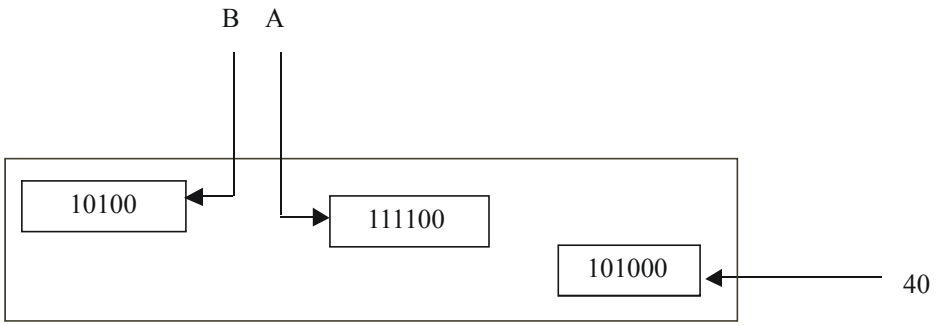
Aspectos importantes a tener en cuenta

a Los nombres de variables deben ser únicos: no pueden representar dos cosas al mismo tiempo.
b Una variable sólo puede almacenar un único valor.
c Una variable no puede almacenar tipos de información distintos. Sólo un tipo: numérica, carácter o lógica.
d Los números son diferentes como tales y como caracteres (1diferente de "1").
e Es aconsejable que los nombres dados a las variables se asemejen a lo que representan.
f Es necesario diferenciar el nombre de la variable y el contenido que ésta tiene en un momento determinado.
g A todos los campos se les asigna una única dirección de memoria, que le permite a la computadora llevar información al campo desde un medio externo, sacar información desde memoria y representarla en un medio de salida.
h Las direcciones de memoria son asignadas después de la compilación; en esta fase se asignan los valores de los campos constantes. A las variables, aunque tienen asignadas sus direcciones de memoria, se les da valor en la etapa de ejecución; antes de ésta los campos variables tienen un valor indeterminado (basura). Por ejemplo:
Si en un programa dos de la instrucciones son:
Entre información a B.
A la variable A llévele B más 40.
i Inmediatamente antes de la ejecución cada campo de la instrucción o mandato (y en general todos los campos que intervienen en el programa) tiene asignada en forma aleatoria una dirección de memoria, y los campos constantes su valor.



En la ejecución los campos constantes no cambian y son asignados los valores a los campos variables. Estos valores son llevados a la memoria a través de una instrucción de asignación o una entrada de datos.

Si el valor que se entra a B es 20, internamente los contenidos o valores almacenados son:



Capítulo 2

GENERALIDADES SOBRE ALGORITMOS

2.1. La lógica

Cuando se desarrolla un algoritmo que da solución a un problema determinado, previamente se han debido cumplir los pasos anteriores a él. Como éstos son previos a la construcción del programa que ejecutará la computadora, debe haber coherencia y concatenación en cada uno de los pasos seleccionados para la solución del problema.

El orden en que se disponen los pasos del algoritmo debe ser riguroso; esto implica que deben existir unos pasos antes que otros u otros antes de unos. No se podrá multiplicar A por B si, previamente, no se conocen sus respectivos valores.

El algoritmo es la antesala del programa que ejecutará la computadora, y cuando éste se traslada al lenguaje escogido para representarlo se debe conservar el orden preestablecido en él, independientemente del lenguaje seleccionado. Un algoritmo, una vez construido, puede expresarse en lenguajes diferentes de programación y ejecutarse en computadoras distintas; sin embargo, el algoritmo será siempre el mismo. De ahí que los errores lógicos que se cometan en la elaboración de éste pasarán al lenguaje y, por ende, a la computadora, el cual reproducirá exactamente lo que se le ha mandado; éste no tiene el poder para detectar errores humanos.

FALTA GRAFICO PAG. 37

2.2. Procedimiento

Un procedimiento es un conjunto de instrucciones o pasos descritos mediante palabras, para llegar a la solución o resultado(s) de un problema que no involucra cálculos matemáticos; pero aunque la descripción de cada paso rigurosamente debe conservar un orden, la entendibilidad o generalidad de éste depende en forma exclusiva de la persona que lo construye. Posiblemente, una persona distinta divida un paso en varios o condense dos o más pasos en uno solo, según él lo entienda. Miremos el siguiente ejemplo:

Ejercicio resuelto No. 1

Hacer un procedimiento que muestre los pasos necesarios para cambiar una llanta *pinchada* de un carro.

Una posible solución sería:

1 *Iniciar*
2 Sacar la llanta y herramientas de la maleta del carro
3 Sacar la llanta pinchada
4 Colocar la llanta buena
5 Guardar la llanta pinchada y la herramienta
6 Subirse al carro
7 Reanudar el viaje

8 *Terminar*

Posiblemente alguien dirá: antes de hacer una de las tareas propuestas es necesario hacer otras, tales como:

Para llegar al paso uno:
- Primero debe orillar el carro
- Detener el carro
- Bajarse del carro
- abrir la maleta del carro
- Sacar las señales de peligro y colocarlas a cierta distancia

Para llegar al paso dos:
- Introducir la llave en la tuerca
- Aflojar levemente las tuercas
- Levantar mediante un gato hidráulico el carro
- Sacar las tuercas

Todo esto puede ocurrir antes o después de cada uno de los pasos descritos inicialmente; de tal manera, que un procedimiento puede ser tan refinado o tener tantas instrucciones como cree quien lo elabora; en cambio en los algoritmos los pasos no dependen de quien los realiza, sino de la lógica del problema.

Otra solución sería:
1 *Iniciar*
2 Orillar el carro
3 Detener el carro
4 Bajarse del carro
5 Abrir la maleta del carro
6 Sacar las señales de peligro y colocarlas a cierta distancia
7 Sacar la llanta y herramientas de la maleta del carro
8 Introducir la llave en la tuerca
9 Aflojar levemente las tuercas
10 Levantar mediante un gato hidráulico el carro
11 Sacar las tuercas
12 Sacar la llanta pinchada
13 Colocar la llanta buena
14 Apretar las tuercas
15 Guardar la llanta pinchada, la herramienta y las señales
16 Subirse al carro
17 Reanudar el viaje
18 *Terminar*

2.3. La expresión

Una expresión es un grupo de operadores que actúan sobre operandos, para proporcionar un único resultado. La expresión puede ser:

Aritmética: en este caso los operandos son constantes o variables numéricas unidas a través de operadores aritméticos, donde el resultado obtenido de la expresión es un número. Las expresiones aritméticas que involucran más de un operador son evaluadas dependiendo de la prioridad que tenga el operador, de acuerdo con la siguiente tabla:

Operador	**Prioridad**	**Significado**
**, ó, ^	Mayor	Potenciación

* , /, %	↓	Multiplicación, división, módulo
+ , –	Menor	Suma, resta

Si dos o más operadores consecutivos tienen la misma prioridad, las operaciones se ejecutarán en la instrucción de izquierda a derecha. Ejemplo:

Si se tiene la expresión: A * * 2 / 5 * B - 5 y los valores almacenados en A y B son 5 y 20, respectivamente, la evaluación de acuerdo al orden de prioridad será

5 * * 2 = 25
25 / 5*20 = 100
100 - 5 = 95

Si se requiere que una o más operaciones se realicen primero que otras, entonces éstas se encierran entre paréntesis y dentro de éstos se conserva la jerarquía de los operadores. Ejemplo:

La operación $\frac{A + B}{C - A} + 20$ debe representarse como: (A + B)/(C - A) + 20

Lógica: en este tipo de expresiones existe por lo menos un operador lógico actuando sobre variables numéricas, lógicas o caracteres. El resultado siempre será uno de los valores falso o verdadero.

Los operadores lógicos son de dos clases: *relacionales y booleanos*.

Los operadores *relacionales* permiten realizar comparaciones de tipo numérico, carácter o lógico, y tienen la misma prioridad en su evaluación. Estos se muestran en la siguiente tabla.

Operador	**Significado**
=	Igual
< >	Diferente de
< =	Menor igual que
> =	Mayor igual que
>	Mayor que
<	Menor que

Ejemplos

Si las variables X y Z tienen almacenadas internamente los valores 30 y 40, respectivamente, el resultado de las siguientes expresiones será:

Expresión	**Significado**
X = Z	.F.
X > Z	.F.
(X + 20) <> (Z+1)	.V.
Z > = X	.V.
'A' > 'B'	.F.

Los operadores *booleanos* operan sobre información lógica, uniendo condiciones simples para formar condiciones compuestas. Éstos operadores son:

Operador	**Prioridad**	**Significado**
~ (NOT)	Mayor	Negación
∧ (AND)	↓	'Y', Lógica
∨ (OR)	Menor	'O', Lógica

Donde el primero es llamado *operador unario*, porque actúa sobre un solo valor de verdad; y los dos siguientes, *operadores binarios*, puesto que operan sobre dos valores de verdad.

Si a y b son condiciones de tipo lógico, los resultados al aplicarles los operadores booleanos se muestran en la siguiente tabla.

a	**b**	**~a**	**a ∧ b**	**a ∨ b**
.V.	.V.	.F.	.V.	.V.
.V.	.F.	.F.	.F.	.V.
.F.	.V.	.V.	.F.	.V.
.F.	.F.	.V.	.F.	.F.

O sea que si se unen dos condiciones a través del operador ∧, el resultado solamente será verdadero si las dos condiciones son verdaderas; y si se unen a través del operador ∨, el resultado será falso si las dos condiciones son falsas.

Ejemplo

Si las variables X, Z y L tienen almacenados los valores 23, 47 y .V., respectivamente, las expresiones que se muestran tendrán los siguientes resultados:

Expresión	**Resultado**
(X > Z) ∨ (Z > 20)	.V.
(X < Z) ∧ (~(Z > 20))	.F.
(Z > 50) ∧ L	.F.
~L	.F.
(Z > 100) ∧ (X < 3) ∧ (~L)	.F.

Constante		Variable	
Ejemplos		***Ejemplos***	
"ALGORITMO"	"A"	A	PESO
"L"	" * "	SALARIO	LÓGICA
0	60.456		
.V.			

Estas dos últimas son denominadas expresiones simples, pero tienen gran utilidad en el desarrollo de un algoritmo.

2.4. Pasos para la solución de un problema a través de la computadora

Cuando se pretende obtener resultados de un problema por computadora es necesario darle el modelo de solución, o sea, toda la serie de pasos que ella debe seguir para llegar a obtener resultados. Si el modelo que se le entrega es incorrecto ésta no lo corrige: arrojará resultados equívocos; de tal manera, que es necesario tener mucho cuidado, mucha disciplina en el trabajo para que esto no suceda.

Existe una serie de pasos y etapas que deben cumplirse con el fin de minimizar los errores humanos que puedan cometerse.

Definición del problema

Es absolutamente necesario tener un enunciado entendible donde se especifique, qué es lo que se requiere resolver y qué resultados se deben obtener a través de la computadora.

Análisis del problema

Es aquí donde se hace el planteamiento matemático y lógico de la solución del problema; por lo tanto, se hace necesario identificar qué es lo que tengo en el momento (datos de entrada), qué es lo que deseo que la computadora produzca (datos de salida o resultados), y cuál es el proceso que se debe hacer, de tal manera que a partir de los datos de entrada se pueda llegar a los resultados.

Es necesario, también, tener en cuenta los recursos que se tienen y si éstos son aptos para el proceso, lo mismo que el medio y la forma como se va a almacenar la información a procesar y la información a salir.

Crear el algoritmo

Si se cumplió a cabalidad el paso anterior, ya se tiene la concepción de la solución del problema; por lo tanto, esta etapa consiste en hacer una descripción de los pasos lógicos que dan solución al problema, hasta obtener los resultados requeridos.

Prueba de escritorio

Si al crear un algoritmo que da solución a determinado problema se cometen errores de lógica, éstos llegarán también a la computadora; y como él no puede detectarlos, arrojará errores en los resultados. La prueba de escritorio permite detectar los posibles errores que cometa el programador en el diseño del algoritmo, para corregirlos antes de continuar con el siguiente paso. Esta prueba consiste en la selección de diferentes datos de entrada al algoritmo y en el seguimiento de la secuencia de cada una de las etapas, hasta obtener los resultados. Al hacer un análisis comparativo de la salida obtenida con los resultados reales, se podrá establecer si el algoritmo funciona o si, por el contrario, es necesario hacerle algunos ajustes.

Codificación

Como la computadora no admite los gráficos o palabras con que se diseña el algoritmo, es necesario pasar éstos a un lenguaje de programación reconocido por la computadora (codificar).

Primero se selecciona el lenguaje de programación con el cual se va a trabajar y luego se pasan, uno a uno, los pasos del algoritmo a instrucciones del lenguaje, de acuerdo con las normas de sintaxis que éste tenga.

Transcripción

El programa escrito en papel es necesario llevarlo a un medio de entrada que sea admitido por la computadora, bien sea que se transcriba a través de una terminal para que sea grabado en disco o disquete o se grabe directamente en disquete a través de una máquina especial para este propósito. El programa transcrito se conoce como *programa fuente*.

Compilación

Es posible que al hacer la codificación se cometan errores de sintaxis gramaticales que riñan con las normas establecidas en el lenguaje de programación elegido. Como se dijo antes, el compilador analiza, una a una, las instrucciones del programa codificado para detectar si están bien escritas o no. Si existen errores se producirá un listado de éstos indicando su localización y causa del error, para corregirlos e iniciar de nuevo la fase de compilación.

Cuando no existan errores el programa se traduce al lenguaje de máquina, único lenguaje que la computadora entiende, lo cual da origen a lo que se conoce como *programa objeto.* En esta fase se les asigna dirección de memoria a todos los campos y además se le da valor a los campos constantes.

Ejecución

Esta fase consiste en dar una orden para que sean ejecutadas, una a una, las instrucciones del programa objeto; por lo tanto, es necesario suministrar datos de entrada cada vez que la

computadora encuentre una orden de este tipo, de igual manera como se hizo en la prueba de escritorio. En esta fase se le da valor a los campos variables.

Es necesario analizar los resultados obtenidos para darse cuenta si están de acuerdo con los resultados reales que se desean conseguir; de lo contrario, hay que hacer modificaciones en las instrucciones que causan el error o errores, en caso de que éstos sean leves. Si son errores graves de lógica es necesario reanalizar el problema.

Documentación externa

Cuando a través de distintas pruebas se establece que el programa está a punto, es decir, funciona produciendo resultados correctos, es necesario documentarlo, para que éste pueda ser utilizado por distintos programadores o usuarios sin necesidad de recurrir directamente a su autor. La documentación debe incluir aspectos relevantes del proceso como: enunciado del problema, narración de la solución, método empleado, definición de campos variables utilizados, listado del programa fuente, dispositivos de computación utilizados, listado de la salida producida. Si el programa utiliza subprogramas, éstos también deben estar documentados.

2.5. El algoritmo

Si se analizan los pasos para la solución de un problema a través de la computadora se nota que el algoritmo es bastante importante, y a él se debe llegar cuando se ha entendido el problema y se ha hecho un análisis correcto de las características de éste.

Es común ver estudiantes que por ligereza omiten uno o varios pasos y luego se dan cuenta de que han invertido su tiempo en lo equivocado, porque no han verificado si lo que están haciendo es correcto o no.

Si no se está seguro de la implementación de uno o algunos de los pasos, es preferible pedir ayuda especializada para aclarar las dudas que surjan.

La palabra algoritmo es muy antigua; toma su nombre del famoso matemático y astrónomo árabe Al-khôwarizmi (siglo IX), quien escribió un tratado sobre manipulación de números y ecuaciones titulado *al-jabrw'almugabala*.

Un *algoritmo* es una secuencia de pasos o instrucciones que representan la solución de un determinado tipo de problema.

Cuando se quiere solucionar un problema a través de la computadora, se exige un algoritmo que muestre la secuencia de solución del mismo.

Características de los algoritmos

Las características fundamentales que debe cumplir todo algoritmo son:

- ***Entrada***

La entrada hace referencia a la información proporcionada al algoritmo, la cual debe sufrir un proceso para obtener los resultados.

Un algoritmo tiene cero o más datos de entrada. Estos valores le son dados por medio de una instrucción o mandato que se debe cumplir al ejecutarse el algoritmo. Si no existen datos de entrada es porque una o más instrucciones generan los valores de partida, de los que hará uso el algoritmo para producir los datos o valores de salida.

- ***Salida***

Todo algoritmo debe proporcionar uno o más valores como resultado, una vez se ha ejecutado la secuencia de pasos que lo conforman.

La salida es la respuesta dada por el algoritmo o el conjunto de valores que el programador espera se le proporcionen.

Estos resultados pueden ser de cualquier tipo: uno o más valores numéricos, valores lógicos o caracteres. La facilidad o complejidad de un algoritmo no la determinan la cantidad de datos que se desean obtener. Un algoritmo puede tener un alto grado de complejidad y, sin embargo, producir un solo valor como resultado.

- ***Limitado o finito***

Todo algoritmo debe tener un número de instrucciones que limitan el proceso en algún momento, es decir, la ejecución debe detenerse. No puede existir un algoritmo, por muy grande que sea o por muchos resultados que produzca, que se quede en forma indefinida ejecutando sus instrucciones o repitiendo la ejecución de un subconjunto de ellas.

- ***Finalización***

Un algoritmo debe indicar el orden de realización de cada uno de sus pasos. Debe mostrar la primera, la intermedia y la última instrucción que debe realizarse. Esto permite mostrar que en algún momento debe culminar la acción o tarea que hace el algoritmo.

- ***Claridad***

Todo el conjunto de pasos debe ser entendible y factible de realizar, de tal manera, que al hacer un seguimiento del algoritmo éste produzca siempre los resultados requeridos. No puede entonces existir incertidumbre en las acciones a tomar cuando se sigue la lógica (flujo del programa) del algoritmo.

Todo algoritmo debe tener tres partes:

Entrada. Información dada al algoritmo, o conjunto de instrucciones que generen los valores con que ha de trabajar, en caso de que no tenga datos de entrada.

Proceso. Cálculos necesarios para que a partir de un dato de entrada se llegue a los resultados.

Salida. Resultados finales o transformaciones que ha sufrido la información de entrada a través del proceso.

Entrada Proceso Salida **(OJO GRAFICAR)**

Ejercicio resuelto Nº 2

Se desea conocer cuántos meses han transcurrido entre enero de 1951 y enero de 2002.

Si se mira con detenimiento el problema se nota que su solución es simple, posiblemente con una calculadora se demoraría menos en obtener los resultados que elaborando un algoritmo.

La construcción de algoritmos exige conocimiento de técnicas y algo de arte. Por ello es necesario que el estudiante afronte la solución de los problemas en forma independiente; subir en escala desde los casos más triviales hasta llegar a diseñar algoritmos más complejos, de tal manera, que el éxito está en resolver la mayor cantidad de ejercicios que se pueda.

Análisis del algoritmo

En este paso debemos conocer: punto de partida (datos de entrada), punto de llegada (datos de salida), y proceso (cálculos necesarios para obtener los resultados).

Datos de entrada

- Observe que estos valores se están proporcionando en el enunciado. Se está dando el año inicial y el año final como valores fijos, o sea, que actuarán como campos constantes en el algoritmo. Esto permite determinar que hay cero datos de entrada a través de variables.

Datos de salida

- Número total de meses transcurridos entre los años 1951 y 2002. ¡Esto es lo que se pide! Se necesitará de una variable que almacene internamente esta información.

Proceso

Para conocer el número de meses es necesario, primero, conocer el número de años que hay entre las dos fechas. Se necesitará de otra variable que almacene este valor que se obtiene restando el año menor del mayor (2002 - 1951); este resultado se multiplica por doce (número de meses del año) y así se obtendrá el número de meses que hay en dicho período, el cual debe ser guardado en otra variable.

Definición de variables a utilizar

En cada algoritmo es necesario escoger los nombres con los cuales se van a representar aquellos valores que no conocemos en el momento de construir el algoritmo, o sea, los nombres que se le asignan a los datos de entrada (si los hay), a los datos de salida y a las variables que no son de entrada ni salida pero que son necesarios para almacenar valores temporales intermedios, útiles para obtener los resultados y que se denominan variables auxiliares.

Se puede decir que en la elaboración de un algoritmo intervienen tres clases de variables: variables que representan los datos de entrada (si los hay), variables auxiliares y variables que representan los datos de salida.

Como se dijo, estos nombres deben iniciar con una letra seguida de letras o dígitos y se aconseja que sean nemotécnicas.

NATRAN: Número de años transcurridos entre 1951 y 2002.
NMESES: Número de meses que hay en el período 1951-2002.

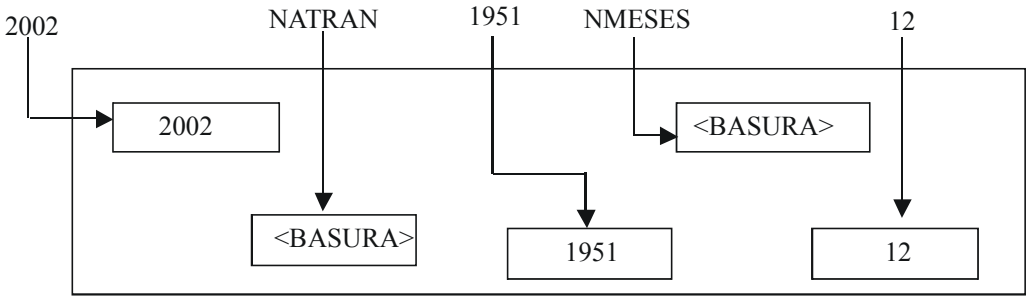
Con las bases expuestas podemos ya construir el algoritmo.

1 Inicia el algoritmo.
2 A NATRAN llévele el resultado de la expresión 2002 - 1951.
3 A NMESES llévele el valor que hay en NATRAN * 12.
4 Muestre el valor que hay almacenado en la variable NMESES.
5 Termina el algoritmo.

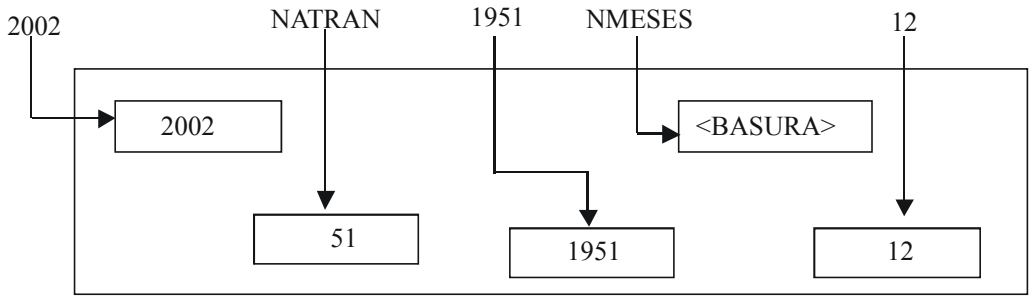
Vamos a analizar qué sucede cuando llevamos el anterior algoritmo a un lenguaje de programación y lo ejecutamos a través de una computadora.

a. Después de compilarlo e inmediatamente antes de ejecutarse.

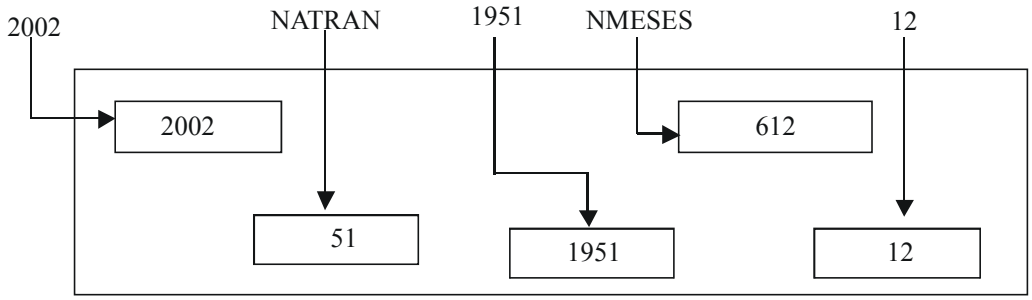
A todos los campos se les asigna una dirección de memoria y a los campos constantes se les asigna valor.



b. Después de ejecutarse la instrucción 2.

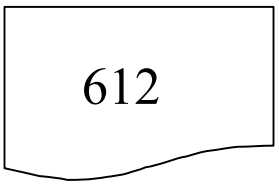


c. Después de ejecutarse la instrucción 3.



d. Después de ejecutarse la instrucción 4.

Se extraerá de memoria el valor almacenado en la variable NMESES y se llevará al medio externo elegido para la salida de información. Una vez ejecutado el mandato el valor de NMESES estará en el medio externo y en memoria. La salida del valor de una variable no implica que su valor desaparezca de la memoria de la computadora.



Resultado que se muestra por el medio externo de salida. El símbolo representa salida por impresora de papel.

Generalización del algoritmo

Los algoritmos deben ser construidos para ser utilizados en situaciones diferentes, donde se requieran los mismos resultados. Para el caso anterior, el algoritmo debe servir no sólo para calcular los meses comprendidos entre 1951 y 2002 sino para cualquier subrrango de años. Si el enunciado se cambia por:

Se desea conocer cuántos meses han transcurrido entre los inicios de dos años cualesquiera dados...

Análisis

Datos de entrada

Como en el momento de construir el algoritmo no se conocen cuáles son los valores del año inicial y final, éstos deben representarse mediante variables para que sus valores sean dados en el momento de la ejecución.

• Valor del año inicial o año menor.
• Valor del año final o año mayor.

Datos de salida

• Número de meses transcurridos entre el año inicial y el año final.

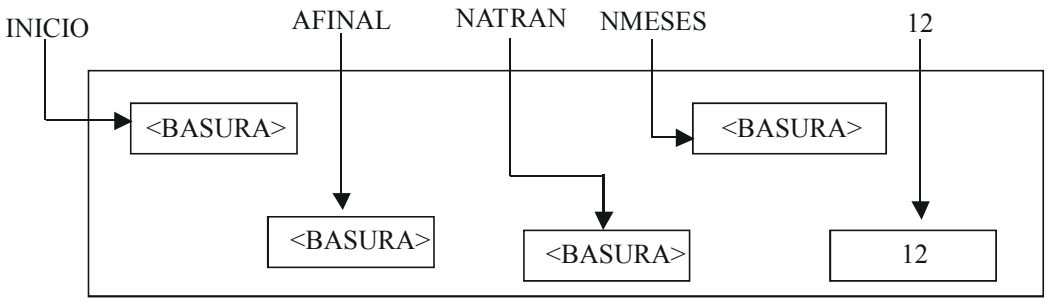
Definición de variables a utilizar

AINICIO: Año inicial.
AFINAL: Año final.
NATRAN: Número de años transcurridos entre AlNICIO y AFINAL.
NMESES: Número de meses que hay en el período AFINAL - AINICIO.

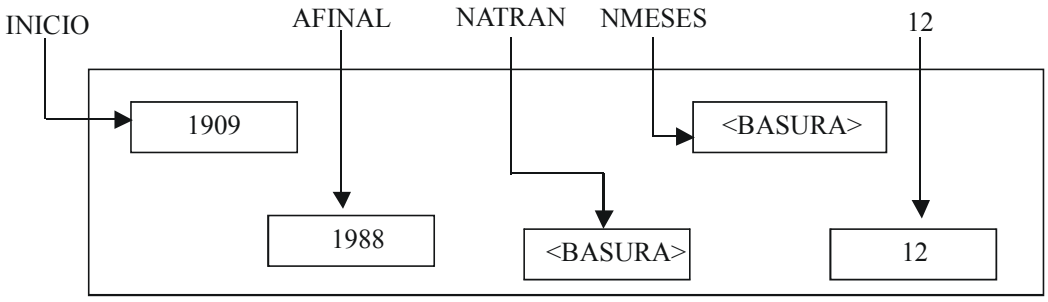
1 Inicia el algoritmo.
2 Acepte valores para AINICIO y AFINAL.
3 A NATRAN llévele AFINAL - AINICIO.
4 A NMESES llévele NATRAN * 12.
5 Muestre el valor que hay almacenado en la variable NMESES.
6 Termina el algoritmo.

Veamos qué sucede al ejecutarse el algoritmo...

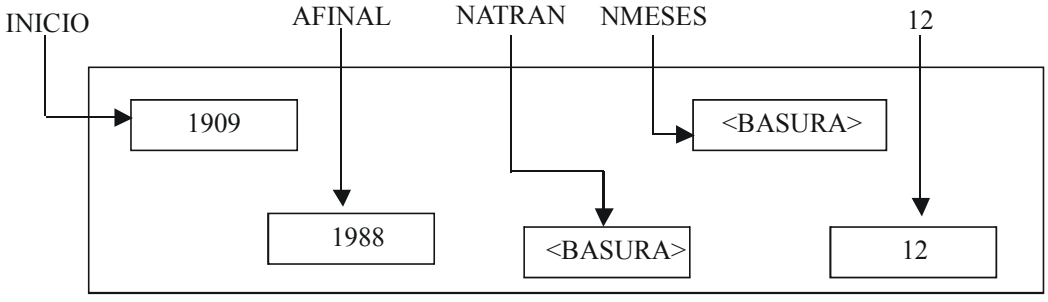
a. Después de compilado e inmediatamente antes de la ejecución.



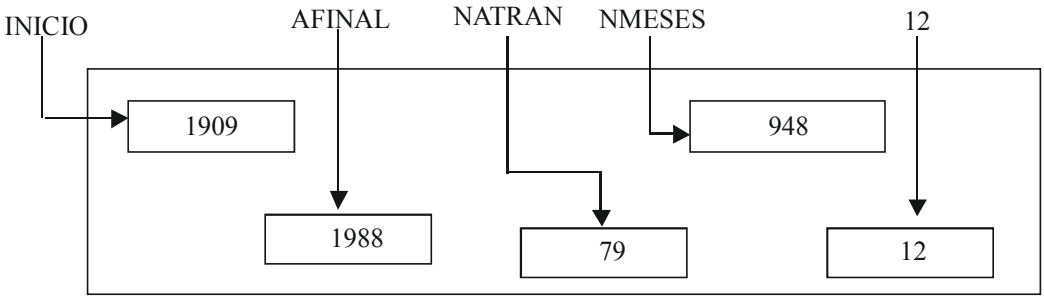
b. Si los valores a entrar en la variables AINICIO y AFINAL son en su orden 1909 y 1988, después de ejecutada la instrucción 2.



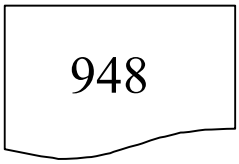
c. Después de ejecutarse la instrucción 3.



d. Después de ejecutarse la instrucción 4.



e. Después de ejecutarse la instrucción 5.



Resultado que se lleva al medio de salida

Si no se requiere volver a ejecutar el programa y nos *despedimos* de la computadora, son desasignadas las direcciones de memoria, por lo tanto, ya no se puede referenciar ninguna variable estructurada

2.6. Concepto de programación

La programación es un conjunto de técnicas utilizadas para desarrollar programas que sean fáciles de leer, depurar (poner a punto) y modificar o mantener. Está basada en el teorema de Bhöm y Jacopini (1966), que establece que un programa propio puede ser escrito utilizando sólo tres tipos de estructuras de control: *Secuencial, Condición lógica o Selectiva y Repetitiva o tipo Bucle.*

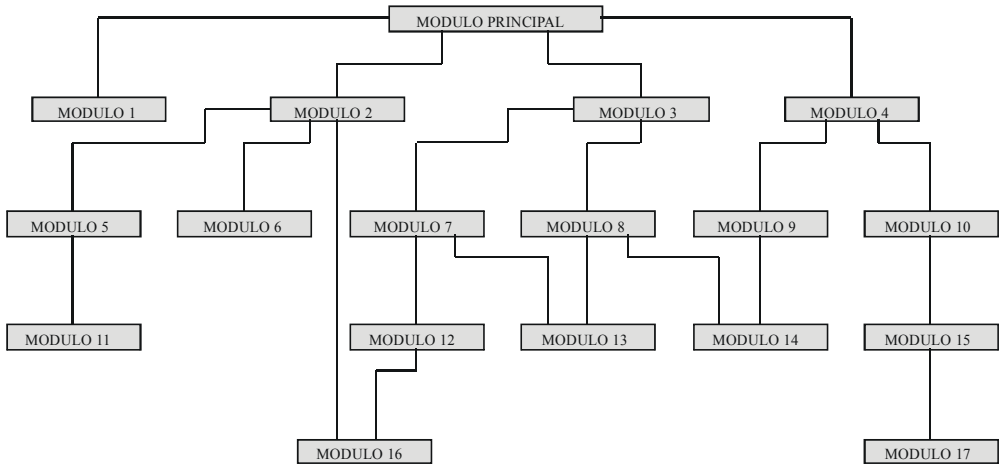
Un programa se define como propio si cumple las siguientes características:
- Para control del programa sólo existe un punto de entrada y otro de salida.
- Existen caminos desde la entrada hasta la salida que se pueden seguir y que pasan por todas las partes del programa.
- Todas las instrucciones son ejecutables y no existen bucles infinitos.

Un programa, no importa lo que haga, se puede escribir usando sólo las tres estructuras antes citadas, donde cada una de ellas tiene un solo punto de entrada y un solo punto de salida.

Según Dijkastra, la programación estructurada se auxilia en los recursos abstractos en lugar de los recursos concretos de que dispone un determinado lenguaje de programación, esto consiste en descomponer una determinada acción compleja en términos de un número de acciones simples, que constituyen instrucciones de computadora disponibles.

La programación estructurada es un conjunto de técnicas de programación que incluye:
- Un número limitado de estructuras de control.
- Diseño descendente (Top-Down).
- Descomposición modular con independencia de los módulos.

Esta programación utiliza el principio *dividir para reinar*, esto es, para escribir un programa complejo se parte de una descripción lo más detallada posible del problema o módulos que sean independientes entre sí y que tengan, cada uno, una entrada y una sola salida (nivel l). Cada módulo resultante se descompone en subconjuntos que tratan de ser independientes (nivel 2). Se vuelve a aplicar este enfoque simple a cada uno de los submódulos resultantes; para todos o para algunos submódulos la solución puede ser trivial, pero en caso de que no lo sea se vuelve a subdividir lográndose el nivel 3, y así sucesivamente.



Los algoritmos estructurados hacen uso de los postulados de la programación estructurada. O sea que todo problema, no importa lo que haga, puede ser representado a través de un algoritmo que use las estructuras: Secuencial, Decisión Lógica y Repetitiva.

2.7. Representación de algoritmos

Los algoritmos deben ser representados usando algún método que les permita ser independizados del lenguaje de programación que se requiera utilizar. Los métodos más usuales son: *diagramas y pseudocódigos.*

Diagrama

Un diagrama es la representación, mediante gráficos, de cada uno de los pasos que dan solución a un problema determinado. Cada gráfico utilizado representa la acción o mandato que se debe ejecutar dentro del algoritmo.

Diagrama de flujo

Un diagrama de flujo es un conjunto de figuras geométricas unidas mediante una flecha, donde cada figura indica la acción a ejecutar y la flecha el orden en que han de ejecutarse las acciones. Existe una figura para representar cada instrucción o acción a ejecutar.

Ejemplo

En los tres ejemplos siguientes se representará el ejercicio No 2, de acuerdo al método de representación expuesto.

Figura que representa principio y fin del programa

Entrada de datos a través del teclado de una terminal

Asignación de una expresión a una variable

Salida de información por impresora de papel

FALTA GRÁFICO PAG. 53 LIBRO

Diagrama rectangular

Este no utiliza flechas de conexión, sino que se construye, fundamentalmente, con tres figuras que representan las estructuras de control de la programación estructurada.

Un diagrama rectangular empieza con un rectángulo vacío que se va llenando de arriba hacia abajo, en la medida en que es necesario representar un determinado paso. A diferencia del anterior, no tiene figuras especiales para demarcar los medios de entrada y de salida, pero dentro de la figura escogida para representar la instrucción se escribe la acción a ejecutar.

Los bloques de inicio y fin del algoritmo pueden omitirse ya que se asume que el orden de ejecución de los pasos es siempre de arriba a abajo, pero si se desea, pueden adicionarse.

Ejemplo

```
INICIO
LEA: AINICIO, AFINAL
NATRAN = AFINAL - AINICIO
NMESES = NATRAN * 12
ESCRIBA: NMESES
TERMINE
```

Pseudocódigo

El pseudocódigo es tan claro como el diagrama y es una herramienta útil para el seguimiento de la lógica de un algoritmo y, sobretodo, facilita la transcripción a un lenguaje de programación. Tiene la desventaja de que el programador trata de escribir los pasos del algoritmo utilizando palabras reservadas, propias del lenguaje en el cual está acostumbrado a trabajar.

Es necesario, entonces, al igual que en el diagrama, establecer unos parámetros o formas de expresar cada instrucción, independientemente de la terminología de los lenguajes de programación.

El pseudocódigo es la representación de los pasos del algoritmo a través de palabras, utilizando una nomenclatura estandarizada para denotar el significado de cada paso. Dentro de éste es permitido la sangría con el fin de que se visualice, en forma sencilla, el grupo de instrucciones pertenecientes a una determinada acción.

Ejemplo

```
INICIO
    LEA: AINICIO, AFINAL
    NATRAN = AFINAL - AINICIO
    NMESES = NATRAN * 12
    ESCRIBA: NMESES
TERMINE
```

Observe cómo el grupo de instrucciones entre las palabras *inicio y termine* se han rodado un poco hacia la derecha (*sangría*), lo cual permite visualizar en una forma sencilla el grupo de instrucciones pertenecientes a ese bloque. Las soluciones a problemas propuestos, de ahora en adelante, se harán utilizando pseudocódigo, pero se pueden utilizar cualesquiera de los tres métodos propuestos. La forma general o formato de cada paso, será expuesta en detalle en los capítulos posteriores.

Aspectos a tener en cuenta

a. Trate de entender y hacer el mejor análisis posible del problema a solucionar.
b. A todos los campos utilizados en un programa se les asigna una única dirección de memoria.
c. Los campos variables (*variables*) se les asignan valores en el momento de ejecución, bien sea a través de una instrucción de entrada de datos o una instrucción de asignación.
d. Cuando se le asigna valora una variable; lo que había antes en esa determinada dirección es destruido, no se puede recuperar.
e. No todos los valores de partida al hacer el algoritmo son datos de entrada, sólo lo son aquellos valores que desconocemos en el momento, pero que serán suministrados en la fase de ejecución.
f. Al extraer de memoria uno o más valores hacia un medio externo, éstos permanecen; no se borran de la memoria de la computadora.
g. Usted aprende a construir algoritmos haciendo algoritmos. Trate de hacer el máximo número de algoritmos propuestos. Busque, si es necesario, enunciados en otros textos.

2.8. Ejercicios propuestos

1. Haga un procedimiento que describa los pasos para cruzar una calle por un puente peatonal.
2. Describa los pasos necesarios para llevar una amiga a cine.
3. Describa los pasos para cambiar una bombilla quemada.
4. Elabore un procedimiento que muestre los pasos necesarios para hacer una llamada telefónica, donde no se sabe el número del teléfono, pero sí el nombre de la persona.
5. Haga un procedimiento que muestre los pasos para cambiar la llanta pinchada de una bicicleta.
6. Muestre los pasos necesarios para hacer un desayuno de huevos duros, hasta servirlos.
7. Escriba las siguientes expresiones algebraicas como expresiones algorítmicas:

a. $X^2 + 4YZ$ b. $\frac{X + Y}{Z} + \frac{3X}{5} + 4Y$ c. $\frac{4X^2 - 2X + 8}{c - d}$

d. $\frac{-b + \sqrt{(b^2 - 4ac)}}{2a}$ e. $\frac{4}{3}\pi$ f. $A \div BC \div (\sqrt{D})^3$

g. $(A^3)^2 - \frac{BC}{DE}$

8. Escriba las siguientes expresiones algorítmicas como expresiones algebráicas:

a. B * *2 – X * Y b. 5 * X * *2 – X * *3 * 5

c. (A + B) / (C – A) d. X * Y + Y * *(1 / 2)

9. Si el valor interno de la variable A = 4, el de B = 5, de C = 1, L =.V. (Verdadero). Muestre cuáles son los valores impresos en el siguiente algoritmo.

```
INICIO
        X = B * A _ B **2/4 * C
        Y = A * B/3 ** 2
        Z = (((B + C) /2 * A + 10) * 3 * B) _ 6
ESCRIBA: X, Y, Z
TERMINE
```

10. Haga un seguimiento del siguiente algoritmo y diga qué valores se muestran en las variables P, R y S. (use los valores definidos en 9)

```
INICIO
    P = A ** (1/2) ** B
```

```
    R = A * B + A ** (1/2)
    S = B * A - B ** 2 / 4 * C
    ESCRIBA: P, R, S
TERMINE
```

11. Usando los valores de A, B, C y L, arriba mencionados, calcule el valor almacenado en las siguientes variables:
    a. $X = (A > B) \wedge (\sim L) \vee (X \ ^{3} \ 30)$
    b. $Y = (B <= 100) \wedge \sim (A > C) \wedge (C = 1)$
    c. $Z = (B = 5) \vee (C = 30) \wedge \sim L$
    d. $W = ((B + 20) > (C - 1)) \vee ((A + 5) \leq 50)$

Capítulo 3

ESTRUCTURA SECUENCIAL

La estructura secuencial permite representar aquellas instrucciones que se ejecutan una tras otra, en secuencia; o sea, instrucciones en que la salida de una es la entrada de la próxima instrucción. También se puede decir que son una o más instrucciones seguidas, donde ninguna de ellas hace que se bifurque el control de ejecución del algoritmo, o que se repita la ejecución de una o más instrucciones. Mediante esta estructura se pueden representar instrucciones de asignación, entrada y salida de datos e invocación a subprogramas. Para diferenciar una de otra, se añaden características intrínsecas de cada instrucción.

Ésta, como todas las estructuras de la programación estructurada, tiene una sola entrada y una sola salida.

3.1. Representación

```
INICIO
    <instrucción 1>
    <instrucción 2>
    <instrucción 3>
          .
          .
          .
    <instrucción n>
FIN_INICIO
```

Ejemplo

```
INICIO
LEA: A, B
C= A + B
ESCRIBA: C

FIN_INICIO
```

3.2. Instrucción de asignación

Por medio de esta instrucción se asigna a una variable el resultado de evaluar una expresión. Al lado izquierdo de la instrucción está la variable, al derecho la expresión y uniéndolas el signo "=", denominado operador de asignación.

Al ejecutarse la instrucción se evalúa la expresión, la cual da como resultado un único valor; éste es llevado a la dirección de memoria asignada a la variable que recibe la información. Al hacerse la asignación es destruido cualquier otro valor que tenga la variable. El dato o valor que arroje la expresión debe ser del mismo tipo que la variable: si el resultado es numérico la variable debe ser numérica, si es lógico o carácter la variable también lo debe ser.

Formato
<variable> = <expresión>

Ejemplos

SALDO = VFINAL - VINICIAL
AREA = Pl * R ** 2
A = (B * C) / (K - 20) – (M + L)
RESUL = (X > Y) ∧ (B > C)
SUMA = 0
VARIABLE1 = VARIABLE2
NOMBRE = “CARMEN”

3.3. Instrucción de entrada de datos

Cuando en el análisis del problema a solucionar se detectan cuáles son los valores necesarios para que la computadora haga los cálculos pertinentes, se dice que éstos son los datos de entrada y que serán proporcionados en el momento de hacer la ejecución.

Una instrucción de entrada de datos es una orden, para que desde un medio externo se entren valores, y dichos datos sean llevados a las direcciones de memoria, asignadas a las variables utilizadas para almacenar los datos de entrada en el proceso.

Formato
LEA: <lista de variables separadas por coma>

Ejemplo. LEA: CODIGO, NOMBRE

Los valores que se entran deben estar en el mismo orden de las variables en la lista.

Si el registro que se conforma para dar los datos es:

6646095 MANUEL

Al ejecutarse la orden dada en el ejemplo anterior, los valores almacenados en las direcciones de memoria asignadas serían:

FALTA GRÁFICO PAG.59 LIBRO

3.4. Instrucción de salida de datos

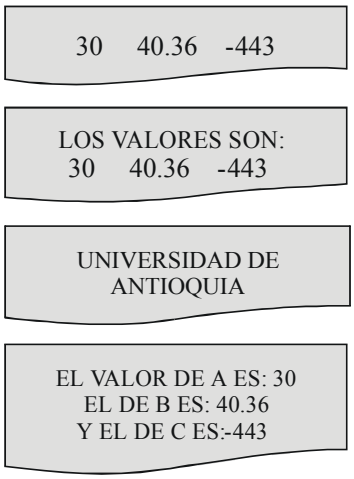
Esta instrucción hace lo contrario que la anterior; mediante la entrada se llevan valores desde un medio externo a la memoria y mediante la instrucción de salida se extraen valores de la memoria hacia un medio externo de salida permitido.

En ésta se permite acompañar los resultados con comentarios explicativos de los valores que se muestran en el medio de salida.

Formato
ESCRIBA: <lista de variables separadas por comas o comentarios>

Ejemplos
ESCRIBA: A, B, C
ESCRIBA: “LOS VALORES SON:”, A, B, C
ESCRIBA: “UNIVERSIDAD DE ANTIOQUIA”
ESCRIBA: “EL VALOR DE A ES:”, A,”EL DE B ES:”, B, “Y EL DE C ES:”, C

Al ejecutarse cada una de las instrucciones anteriores, y si las variables A, B y C contienen los valores: A = 30; B = 40.36; C = -443, las salidas serán en su orden:



Cuando en una instrucción de salida se tiene un grupo de caracteres entre comillas éste será impreso sin ninguna modificación en el medio de salida, y lo que aparezca sin comillas se asume que es una variable; por lo tanto, se extraerá de la memoria el contenido de ésta.

Ejercicio resuelto Nº 3

Hacer un algoritmo que, dados los dos lados diferentes de un rectángulo, encuentre el perímetro y el área del mismo.

Análisis

Como no nos dicen en este momento cuáles son los valores de los dos lados del rectángulo, debemos separar espacio en memoria a través de variables para que estos valores sean dados posteriormente, de tal manera que el algoritmo debe proveer el perímetro y el área de cualquier rectángulo. Al ser un rectángulo (lo asegura el enunciado), conociendo los valores de los dos lados diferentes podemos obtener los resultados pedidos.

Datos de entrada

- Valor de un lado.
- Valor del otro lado.

Datos de salida

Nos piden como resultado:

- El valor de perímetro.
- El valor del área del rectángulo.

Proceso

Los cálculos necesarios para obtener los resultados partiendo de los datos de entrada, son:

• Perímetro = suma de los cuatro lados del rectángulo.
• Area = lado que representa la base *lado que representa la altura.

Definición de variables

L1: Valor del lado que representa la base.
L2: Valor del lado que representa la altura.
P: Perímetro.
ÁREA: Área del rectángulo.

Algoritmo

```
INICIO
    LEA: L1, L2
    P = 2 * L1 + 2 * L2
    AREA = L1 * L2
    ESCRIBA: “EL PERIMETRO ES:”, P
    ESCRIBA: “EL AREA ES:”, AREA
FIN_INICIO
```

Prueba de escritorio

Si el registro de entrada es:

24 15

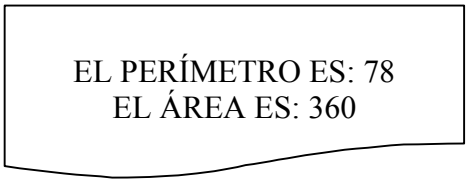
Valor de L1 Valor de L2

↓ ↓ ↑ ↑

L1	L2	P	AREA
24	15	78	360

FALTA TEXTO Y GRAFICO PAG.61 LIBRO

Salida



Ejercicio resuelto Nº 4

A la mamá de Juan le preguntan su edad, y contesta: tengo 3 hijos, pregúntele a Juan su edad. Alberto tiene 2/3 de la edad de Juan, Ana tiene 4/3 de la edad de Juan y mi edad es la suma de las tres. Hacer un algoritmo que muestre la edad de los cuatro.

Análisis

Es necesario que al algoritmo se le proporcione la edad de Juan para calcular la edad de Alberto, de Ana y de la mamá.

Datos de entrada

- Edad de Juan.

Datos de salida

- Edad de Alberto.
- Edad de Juan. (En este caso es un dato de entrada y salida al mismo tiempo).
- Edad de Ana.
- Edad de la mamá.

Proceso

- Edad de Alberto = 2/3 * edad de Juan.
- Edad de Ana = 4/3 * edad de Juan.

- Edad de la mamá = Edad de Alberto + Edad de Juan + Edad de Ana.

Definición de variables

EDJUAN: **Edad de Juan.**
EDALBER: **Edad de Alberto.**
EDANA: **Edad de Ana.**
EDMAMA: **Edad de la mamá.**

Algoritmo

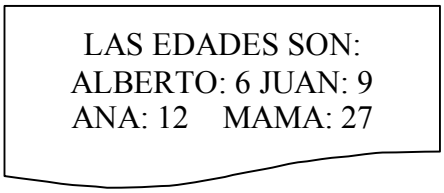
```
INICIO
    LEA: EDJUAN
    EDALBER = 2 * EDJUAN / 3
    EDANA = 4 * EDJUAN / 3
    EDMAMA = EDJUAN + EDALBER + EDANA
    ESCRIBA: “LAS EDADES SON: ALBERTO =”, EDALBER,
    “JUAN =”, EDJUAN, “ANA =”, EDANA, “MAMA =”, EDMAMA
FIN_INICIO
```

Prueba de escritorio

Si el valor de la edad de Juan es 9, las otras edades serán:

↑	↑	↑	↓
EDJUAN	EDALBER	EDANA	EDMAMA
9	6	12	27

Salida



Ejercicio resuelto Nº 5

Hacer un seguimiento (prueba de escritorio) del siguiente grupo de instrucciones.

```
INICIO
    SUMA = 0
    X = 20
    SUMA = SUMA + X
    Y = 40
    X = X + Y ** 2
    SUMA = SUMA + X / Y
    ESCRIBA: “EL VALOR DE LA SUMA ES:”, SUMA
FIN_INICIO
```

Prueba de escritorio

SUMA	X	Y
~~0~~	~~20~~	~~40~~
~~20~~	1620	
60.5		

La raya horizontal indica que el valor subrayado desaparece de la dirección de memoria y es reemplazado por un nuevo valor.

Salida

FALTA GRÁFICO PAG.64 LIBRO

Aspectos a tener en cuenta

a. En un algoritmo a las variables se les da valor, bien sea por una asignación o por una instrucción de entrada.
b. El valor que se le asigna a una variable en cualquiera de los casos debe ser del mismo tipo de dato que la variable.
c. En la prueba de escritorio se deben mostrar los cambios que sufren todas las variables del algoritmo.
d. La instrucción de asignación no puede ser tomada como una igualdad, ya que la variable que recibe el valor puede aparecer también al lado derecho de la expresión.
e. Si una variable aparece en más de una instrucción de entrada o asignación, su valor es destruido cada que se ejecuta una nueva instrucción.

3.5. Ejercicios propuestos

12. Un empleado trabaja 48 horas en la semana a razón de $5.000 hora. El porcentaje de retención en la fuente es del 12,5% del salario bruto. Se desea saber cuál es el salario bruto, la retención en la fuente y el salario neto del trabajador.

13. Escriba un algoritmo que, dados dos valores A y B, encuentre:
    A - B
    A + B
    A * B
    A / B

14. Elabore un algoritmo que lea un número y obtenga su cuadrado y su cubo.

15. Elabore un algoritmo que lea un número negativo e imprima el número y el positivo del mismo.

16. Elabore un algoritmo que lea las variables Ay B y pase el valor de A a B y de B a A. Sin usar más variables

17. Dado el radio de un círculo. Haga un algoritmo que obtenga el área del círculo y la longitud de la circunferencia.

18. Se tiene la siguiente información de un empleado:
    - código del empleado,
    - nombres,
    - número de horas trabajadas al mes,
    - valor hora trabajada,
    - porcentaje de retención en la fuente.

Haga un algoritmo que muestre: código, nombres, salario bruto y salario neto.

19. Dado el valor del lado en un triángulo equilátero, haga un algoritmo que obtenga el perímetro, el valor de la altura y el área del triángulo.

20. Haga un algoritmo que determine los parámetros A, B y C de una recta que pasa por los puntos (X1, Y1) y (X2, Y2). La ecuación de la recta es:
    $AX + BY + C = 0$.

21. Dados los tres lados de un triángulo, haga un algoritmo que encuentre: perímetro, semiperímetro y el área del triángulo.

Capítulo 4

ESTRUCTURA DECISIÓN LÓGICA

La escritura de algoritmos implica, en muchas ocasiones, llegar a algo más que una simple secuencia. Es el caso, cuando existe una serie de caminos o alternativas a escoger dependiendo del resultado de una determinada situación. La estructura *decisión lógica* es utilizada para seleccionar la ruta que debe tomar la ejecución de instrucciones de un algoritmo, o también el flujo que debe llevar el control de ejecución cuando se presentan tomas de decisiones.

Por ejemplo, si usted desea viajar a Bogotá desde Medellín debe escoger una de estas alternativas: vía aérea o vía terrestre. ¿cuál escoge? Sin duda el resultado va a depender de la respuesta que dé a algunas preguntas que debe hacerse, tales como: si el pasaje es más barato vía terrestre, entonces tomo esta alternativa. Como tengo prisa en llegar y tengo dinero, entonces viajo en avión. Esto sucede en los algoritmos cuando es necesario tomar una decisión o camino a seguir.

La *estructura decisión lógica* o selectiva está formada por una condición de tipo lógico que puede ser simple o compuesta, de la que salen dos posibles caminos: un conjunto de acciones o secuencias a ejecutar, si el resultado de la condición es verdadera; u otro conjunto de acciones o secuencias a realizar, si el resultado de la condición es falsa. Se puede dar el caso de que falte uno de los grupos de instrucciones, pero nunca los dos; esto sería un camino nulo que implica no tomar ninguna acción.

4.1. Representación

```
Si <expresión lógica> entonces
    secuencia 1 l camino 1
  SINO
    secuencia 2 l camino 2
FIN_SI
```

Ejemplo

```
SI A > B ENTONCES
    Escriba: «EL VALOR DE A ES MAYOR QUE B»
  SINO
    Escriba: «EL VALOR DE B ES MAYOR O IGUAL QUE A»
FIN_SI
```

4.2. Funcionamiento

Al igual que las otras estructuras, la estructura de decisión lógica tiene una única entrada y una única salida. Al llegar a la estructura se evalúa la condición; si ésta es:

Verdadera, se ejecuta el grupo de instrucciones que conforman la secuencia 1 y continúa hacia abajo con las demás instrucciones del algoritmo que están después del FIN_SI, o sea que toma el camino 1 y no se ejecutan las instrucciones que conforman la secuencia 2.

Si es *Falsa,* se ejecuta el grupo de instrucciones que conforman la secuencia 2 y omite la ejecución de las instrucciones de la secuencia 1, o sea que toma el camino 2 y continúa hacia abajo.

Las instrucciones que conforman las dos secuencias pueden pertenecer a cualquier estructura, incluida la decisión lógica. Es común, dentro de la programación, que falte una de las dos secuencias, en cuyo caso significa que al estar ausente y tomar ese camino *no haga nada* y continúe con las estructuras siguientes.

El grupo de instrucciones que conforman la secuencia 1 está entre el ENTONCES y el SINO y la secuencia 2 entre el SINO y el FIN_SI.

Ejercicio resuelto Nº 6

Escribir las estructuras que calculen y muestren el valor de X de acuerdo a lo siguiente:

X = 1 si Y > Z
X = 2 si Y <= Z

Solución

```
SI Y > Z ENTONCES
      X=1
  SINO
     X=2
FIN_SI
ESCRIBA: X
```

Prueba de escritorio

Si los valores de Y y Z son en su orden: 10 y -6

Y	Z	X
10	-6	1

Salida

FALTA GRÁFICO PAG. 69 LIBRO

Ejercicio resuelto Nº 7

Hacer un algoritmo que, dados dos valores numéricos A y B, escriba un mensaje diciendo si A es mayor, menor o igual a B.

Análisis

Datos de entrada

- Los valores de A y de B.

Datos de salida

- Un mensaje de acuerdo al valor de A y de B.

Proceso

Se compara A con B, sí:

A > B. Entonces se escribe el mensaje «A es mayor que B», si la condición es falsa se pregunta sí:

A = B. Si ésta es verdadera se escribe el mensaje «A es igual que B», y si es falsa, por defecto A es menor que B.

Algoritmo

```
INICIO
    LEA: A, B
    SI A > B ENTONCES
            ESCRIBA: «A ES MAYOR QUE B»
      SINO
            SI A = B  ENTONCES
                    ESCRIBA: «A ES IGUAL A B»
              SINO
                    ESCRIBA: «A ES MENOR QUE B»
            FIN_SI
    FIN_SI
FIN_INICIO
```

Prueba de escritorio

Si los valores de entrada son:

A = 20
B = 25

los valores almacenados internamente serían:

A	B
20	25

Salida

FALTA GRÁFICO PAG. 70 LIBRO

Ejercicio resuelto Nº 8

Si A, B, C, D y E son condiciones de tipo lógico, construir las estructuras correspondientes que muestren:

- A y B (Parte verdadera de A y parte verdadera de B).
- A y no B y C (Parte verdadera de A, parte falsa de B y parte verdadera de C).
- A y no B y no C.
- No A (y luego seguirá explorando D y, en su caso, E).
- No A y D y E.
- No A y D y no E.
- No A y no D.

Solución

Para mostrar A y B, primero hay que evaluar la condición A y luego por la salida verdadera de A evaluar la condición B, entonces la solución del numeral 1 sería el grupo de instrucciones que pertenezcan a la salida verdadera de la condición B.

```
SI A ENTONCES
    SI B ENTONCES
            SECUENCIA 1
      SINO
            SI C ENTONCES
```

```
                    SECUENCIA 2
               SINO
                    SECUENCIA 3
            FIN_SI
     FIN_SI
  SINO
     SECUENCIA 4
     SI D ENTONCES
            SI E ENTONCES
                    SECUENCIA 5
               SINO
                    SECUENCIA 6
            FIN_SI
       SINO
            SECUENCIA 7
     FIN_SI
FIN_SI
```

Ejercicio resuelto Nº 9

Escribir las estructuras que calculen y muestren el valor de X, de acuerdo con lo siguiente:

X = 0 si Y < A y (A< B < C)
X = 1 si A <= Y < B
X = 2 si B <= Y < C
X = 3 si C <= Y
X = 4 si no se cumple ninguna de las condiciones anteriores.

Solución

Como existen condiciones compuestas, éstas deben unirse a través de operadores lógicos.

```
SI (Y < A) ∧ ((A < B) ∧ (B < C)) ENTONCES
     X = 0
  SINO
     SI (A < = Y) ∧ (Y < B) ENTONCES
            X = 1
       SINO
            SI (B < = Y) ∧ (Y < C) ENTONCES
                    X = 2
               SINO
                    SI (C < = Y) ENTONCES
                            X = 3
                      SINO
                            X = 4
                    FIN_SI
            FIN_SI
     FIN_SI
FIN_SI
```

Prueba de escritorio

Si los valores que contienen las variables son:

Y = 20
A = 10
B = 5
C = 2

Salida

FALTA GRÁFICO PAG. 72 LIBRO

Ejercicio resuelto Nº 10

Cierta universidad para liquidar el pago de matrícula de un estudiante le exige los siguientes datos:

- Número de inscripción
- Nombres
- Patrimonio.
- Estrato social.

La universidad cobra un valor constante para cada estudiante de $50.000. Si el patrimonio es mayor que $2´000.000 y el estrato superior a 3, se le incrementa un porcentaje del 3% sobre el patrimonio. Hacer un algoritmo que muestre:

- Número de inscripción.
- Nombres.
- Pago de matrícula.

Análisis

Datos de entrada

- Número de inscripción.
- Nombres.
- Patrimonio.
- Estrato social.

Datos de salida

- Número de inscripción.
- Nombres.
- Pago por matrícula.

Proceso

Inicialmente se asume que el valor a pagar es el valor constante $50.000; se hace la comparación del patrimonio con 2´000.000 y del estrato con tres; en caso de ser cierta la condición al valor constante se le incrementa el 3% sobre el patrimonio. Este valor adicional se obtiene multiplicando el patrimonio por 3 y dividiéndolo sobre 100 ó, lo que sería lo mismo, multiplicar 0,03 por el patrimonio.

Definición de variables

NI: Número de inscripción.
NOM: Nombres.
PAT: Patrimonio.
EST: Estrato social.
PAGMAT: Pago por matrícula.

Algoritmo

```
INICIO
    LEA: NI, NOM, PAT, ES
```

```
    PAGMAT = 50000
    SI (PAT > 2000000) ∧ (ES > 3) ENTONCES
    PAGMAT = PAGMAT + 0.03 * PAT
    FIN_SI
    ESCRIBA: "EL ESTUDIANTE CON NUMERO DE INSCRIPCION", NI,
          "Y NOMBRE", NOM, "DEBE PAGAR: $", PAGMAT
FIN_INICIO
```

Prueba de escritorio

Si el registro de entrada de datos está conformado de la siguiente forma:

0001 JUAN PABLO 1´500.000 4

Los valores almacenados en la variable serían:

NI	NOM	PAT	ES	PAGMAT
0001	JUAN PABLO	1´500.000	4	50.000

Salida

EL ESTUDIANTE CON NUMERO DE INSCRIPCION 0001 Y NOMBRE JUAN PABLO DEBE PAGAR $50.000

Ejercicio resuelto Nº 11

Escribir un algoritmo que acepte tres números enteros diferentes y muestre el mayor de ellos.

Análisis

El enunciado aclara que no existen números iguales, por lo tanto existe un valor menor, uno medio y uno mayor.

Datos de entrada

• Los tres números. Estos deben almacenarse en variables distintas.

Datos de salida

• El valor del número mayor.

Proceso

Existen varias formas de encontrar el valor mayor entre un grupo de datos. Usted puede pensar y desarrollar una diferente a la aquí expuesta. El proceso a utilizar sería:

NUMERO1 será el mayor si:
(NUMERO1 > NUMERO2) ∧ (NUMERO1 > NUMERO3).

NUMERO2 será el mayor si:
(NUMERO2 > NUMERO1) ∧ (NUMERO2 > NUMERO3).

NUMERO3 será el mayor si:

(NUMERO3 > NUMERO1) ∧ (NUMERO3 > NUMERO2).

Definición de variables

N1: Valor del primer número a entrar.
N2: Valor del segundo número.
N3: Valor del tercer número.
MAYOR: Variable que almacenará el valor mayor entre N1, N2 y N3.

Algoritmo

```
INICIO
    LEA: Nl, N2, N3
    SI (N1 > N2) ∧ (N1 > N3) ENTONCES
            MAYOR = N1
      SINO
            SI N2 > N3 ENTONCES
                    MAYOR = N2
              SINO
                    MAYOR = N3
            FIN_SI
    FIN_SI
    ESCRIBA: "EL VALOR MAYOR ENTRE: ", N1, ",", N2, "Y", N3, "ES:", MAYOR
FIN_INICIO
```

Prueba de escritorio

Registro de datos

25 26 56

N1	N2	N3	MAYOR	
25	26	56	~~25~~	
			~~26~~	
			56	Último número almacenado en mayor

Salida

FALTA GRÁFICO PAG. 76 LIBRO

Ejercicio resuelto Nº 12

Determinar la cantidad de dinero recibida por un trabajador por concepto de las horas semanales trabajadas en una empresa, sabiendo que cuando las horas de trabajo exceden de 40, el resto se considera horas extras y se pagan al doble de una hora normal, cuando no exceden de 8; si las horas extras exceden de 8, se pagan las primeras 8 al doble de lo que se paga una hora normal y el resto al triple. Del trabajador se conocen los siguientes datos: nombres, número de horas trabajadas en la semana y valor recibido por una hora normal de trabajo.

Análisis

Datos de entrada

• Nombres del trabajador.
• Número de horas trabajadas.

• Valor hora de trabajo.

Datos de salida
• Nombre del trabajador.
• Salario devengado.

Proceso

Se compara el valor de las horas trabajadas con 40 que es la jornada normal.

SI horas trabajadas > 40

Se calculan las horas extras trabajadas como horas trabajadas - 40. Como los costos de horas extras son distintos para la cantidad de horas superiores a 8, se debe comparar el valor obtenido en horas extras con 8.

SI horas extras > 8 ENTONCES

Las horas extras excedentes de 8 serían: horas extras - 8. En este caso lo percibido por el trabajador por concepto de horas extras sería:

Pago por horas extras = Pago por hora normal * 2 * 8 + pago por hora normal * 3 * horas extras excedentes de 8.

SINO

Pago por horas extras = Pago por horas normal *2*horas extras. El salario devengado en este caso sería:

Salario = Pago por hora normal * 40 + Pago por horas extras.

SINO

En este caso las horas trabajadas durante la semana son menores o iguales a 40 (no existen horas extras). Por lo tanto, lo devengado sería:

Salario = Horas trabajadas * valor hora normal.

Definición de variables

NOM: Nombre del trabajador.
NHT: Número de horas trabajadas.
VHN: Valor hora normal trabajada.
HET: Horas extras trabajadas.
HEE8: Horas extras que exceden de 8.
SALARIO: Pago total que recibe el trabajador.

Algoritmo

```
INICIO
    LEA: NOM, NHT, VHN
    SI NHT > 40 ENTONCES
            HET = NHT - 40
            SI HET > 8 ENTONCES
                    HEE8 = HET - 8
                    SALARIO = 40 * VHN + 16 * VHN + HEE8 * 3 * VHN
              SINO
                    SALARIO = 40 * VHN + HET * 2 * VHN
            FIN_SI
       SINO
            SALARIO = NHT * VHN
    FIN_SI
    ESCRIBA: “EL TRABAJADOR”, NOM, “DEVENGO: $”, SALARIO
FIN_INICIO
```

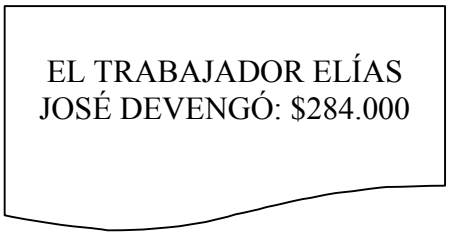
Prueba de escritorio

Registro de entrada

ELIAS JOSE	53	4000

NOM	NHT	VHN	HET	HEE8	SALARIO
ELIAS JOSE	53	4000	13	5	284000

Salida



Ejercicio resuelto Nº 13

Un almacén efectúa una promoción en la cual se hace un descuento sobre el valor de la compra total, según el color de la bolita que el cliente saque al pagar en caja. Si la bolita es blanca no se le hará descuento alguno, si es verde se le hará un 10% de descuento, si es amarilla un 25%, si es azul un
50% y si es roja un 100%. Hacer un algoritmo para determinar la cantidad final que un cliente deberá pagar por su compra. Se sabe que sólo hay bolitas de los colores mencionados.

Análisis

Datos de entrada
- Valor de la compra.
- Color de la bolita; ésta almacenará un valor alfabético.

Datos de salida
- Valor a pagar teniendo en cuenta los posibles descuentos.

Proceso

Conocido el color de la bolita que le ha tocado al cliente se puede establecer si tiene o no descuento por el valor de la compra y el porcentaje que le corresponde.

Si la bolita que saca es:

Blanca, el descuento es del 0%;

si es verde, el descuento es del 10%;

si es amarilla, el descuento es del 25%;

si es azul, el descuento es del 50%;

si no es de ninguno de los colores anteriores, por defecto será roja al no haber más colores, por lo que no es necesario hacer la pregunta si el color de la bolita es roja, en cuyo caso el descuento es del 100%.

Definición de variables

VALCOMP: Valor de la compra.
COLOR: Color de la bolita.
VALPAG: Valor a pagar.
PDES: Porcentaje de descuento.

Algoritmo

```
INICIO
     LEA: VALCOMP, COLOR
     SI COLOR = “BLANCO”  ENTONCES
              PDES=0
       SINO
              SI COLOR = “VERDE” ENTONCES
                       PDES = 10
                 SINO
                       SI COLOR=”AMARILLO” ENTONCES
                                PDES = 25
                          SINO
                                SI COLOR = “AZUL” ENTONCES
                                         PDES = 50
                                   SINO
                                         PDES = 100
                                 FIN_SI
                       FIN_SI
              FIN_SI
     FIN_SI
     VALPAG = VALCOMP - PDES * VALCOMP / 100
     ESCRIBA : “ EL CLIENTE DEBE PAGAR:$”, VALPAG
FIN_INICIO
```

Prueba de escritorio

Registro de Entrada

543450 AZUL

VALCOMP	COLOR	PDES	VALPAG
543450	AZUL	50	271275

Salida

FALTA GRÁFICO PAG. 80 LIBRO

Ejercicio resuelto Nº 14

Una empresa con tres departamentos tiene establecido un plan de incentivos para sus vendedores. Al final del período, a cada departamento se le pide el importe global de las ventas. A los departamentos que excedan el 33% de las ventas totales se les adiciona al salario de los vendedores un porcentaje equivalente al 20% del salario mensual. Las nóminas de los tres departamentos son iguales. Si se tienen los siguientes datos:

- Ventas del departamento 1
- Ventas del departamento 2
- Ventas del departamento 3
- Salario de los vendedores de cada departamento

Hacer un algoritmo que determine cuánto recibirán los vendedores de cada departamento al finalizar el período.

Análisis

Datos de entrada

- Los tres importes de ventas de los departamentos.
- Salario de los vendedores.

Datos de salida

- Valor recibido por salario en cada uno de los departamentos.

Proceso

Para poder obtener el 33% de las ventas totales, primero es necesario conocer el total de ventas de la empresa. Total de ventas = ventas depto 1 + ventas depto 2 + ventas depto 3.

El 33% de las ventas totales o porcentaje sobre las ventas será: ventas totales * 0.33 o ventas totales * 33/100. Este porcentaje sirve para determinar si las ventas de un departamento superan este valor e indicará si los vendedores reciben o no un 20% más sobre su salario.

```
SI ventas de un departamento > porcentaje de ventas ENTONCES
    salario recibido = salario + salario * 0.2;
  SINO
    salario recibido = salario
FIN_SI
```

Continuación

Definición de variables

VD1:	Ventas del departamento 1
VD2:	Ventas del departamento 2
VD3:	Ventas del departamento 3
SALAR:	Salario que reciben vendedores en cada departamento.
TOTVEN:	Total ventas en la empresa.
PORVEN:	Porcentaje equivalente al 33% de ventas totales.
SALAR1:	Salario de los vendedores en el depto. 1
SALAR2:	Salario de los vendedores en el depto. 2
SALAR3:	Salario de los vendedores en el depto. 3

Algoritmo

```
INICIO
    LEA: VD1, VD2, VD3, SALAR
    TOTVEN = VD1 + VD2 + VD3
    PORVEN = 0.33 * TOTVEN
    SI VD1 > PORVEN ENTONCES
            SALAR1 = SALAR + 0.2 * SALAR
      SINO
            SALAR1 = SALAR
    FIN_SI
    SI VD2 > PORVEN ENTONCES
            SALAR2 = SALAR + 0.2 * SALAR
      SINO
            SALAR2 = SALAR
    FIN_SI
    SI VD3 > PORVEN ENTONCES
```

```
            SALAR3 = SALAR + 0.2 * SALAR
        SINO
            SALAR3 = SALAR
    FIN_SI
    ESCRIBA: “SALARIO VENDEDORES DEPTO. 1”, SALAR1,
    “SALARIO VENDEDORES DEPTO. 2”, SALAR2, “SALARIO
    VENDEDORES DEPTO. 3”, SALAR3
FIN_INICIO
```

Prueba de escritorio

Registro de Entrada

4200000 250000 3300000 380320

VD1	VD2	VD3	SALAR	TOTVEN	PORVEN	SALAR1
4200000	250000	3300000	380320	9700000	3201000	456384

SALAR 2 SALAR 3

VD2 VD3

380320 456384

Salida

FALTA GRÁFICO PAG. 83 LIBRO

Ejercicio resuelto Nº 15

Se tienen cuatro esferas (A, B, C, D) de las cuales se sabe que tres son de igual peso y una diferente. Elaborar un algoritmo que determine cuál es la esfera diferente y si es de mayor o menor peso.

Análisis

Datos de entrada

- Peso de la esfera A
- Peso de la esfera B
- Peso de la esfera C
- Peso de la esfera D

Datos de salida

- Mensaje que diga cuál es la esfera de diferente peso y si es de mayor o menor peso que las otras tres.

Proceso

El enunciado del problema afirma que existen tres esferas que tienen igual peso pero no se conocen, por lo tanto, es necesario averiguarlo. Si ya se

Continuación

tienen las tres esferas de igual peso, la restante es la diferente; para saber si es de mayor o menor peso que las otras, basta compararla con el peso de cualquiera de las otras tres.

Las comparaciones posibles serán:

SI (peso de A = Peso de B) ∧ (peso de A = Peso de C), entonces
D es la diferente.
SI peso de D es mayor que peso de A, entonces
D es de mayor peso;
SINO, D es de menor peso.
SINO
SI (peso de A = peso de B) ∧ (peso de A = peso de D), entonces
C es la diferente.
SI peso de C > peso de A, entonces
C es de mayor peso;
SINO, C es de menor peso.
SINO
SI (peso de A = peso de C) ∧ (peso de A = peso de D),
B es la diferente.
SI peso de B > peso de D, entonces
B es de mayor peso;
SINO, B es de menor peso.
SINO, A es la diferente.
SI peso de A > peso de B, entonces
A es de mayor peso;
SINO, A es de menor peso.

Definición de variables

PESOA: Peso de la esfera A
PESOB: Peso de la esfera B
PESOC: Peso de la esfera C
PESOD: Peso de la esfera D

Algoritmo

```
INICIO
    LEA: PESOA, PESOB, PESOC, PESOD
    SI (PESOA = PESOB) ∧ (PESOA = PESOC) ENTONCES
        SI PESOD > PESOA ENTONCES
            ESCRIBA: "LA ESFERA D ES LA DIFERENTE Y
        SINO
            ESCRIBA: "LA ESFERA D ES LA DIFERENTE Y
        FIN_SI
    SINO
        SI (PESOA = PESOB) ∧ (PESOA = PESOD) ENTONCES
            ESCRIBA: "LA ESFERA C ES LA DIFERENTE"
            SI PESOC > PESOA ENTONCES
                ESCRIBA: "Y ES DE MAYOR PESO"
            SINO
                ESCRIBA:"Y ES DE MENOR PESO"
            FIN_SI
        SINO
            SI (PESOA = PESOC) ∧ (PESOA = PESOD) ENTONCES
                ESCRIBA: "LA ESFERA B ES LA DIFERENTE"
                SI PESOB > PESOD ENTONCES
```

```
                        ESCRIBA: “Y ES DE MAYOR PESO”
                    SINO
                        ESCRIBA: “Y ES DE MENOR PESO”
                    FIN_SI
                SINO
                    ESCRIBA: “LA ESFERA A ES LA DIFERENTE”
                    SI PESOA > PESOB ENTONCES
                        ESCRIBA: “Y ES DE MAYOR PESO”
                    SINO
                        ESCRIBA: “Y ES DE MENOR PESO”
                    FIN_SI
            FIN_SI
        FIN_S
    FIN_SI
FIN_INICIO
```

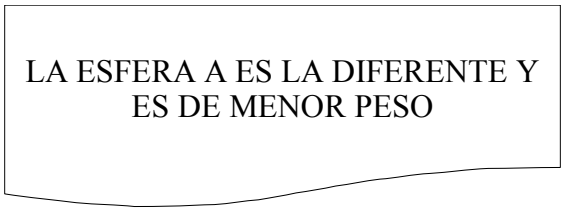
Prueba de escritorio

Registro de entrada

30	54	54	5

PESOA	PESOB	PESOC	PESOD
30	54	54	54

Salida



Aspectos a tener en cuenta:

a. Las variables que hacen parte de una expresión de tipo lógico deben tener un valor previo asignado.

b. Cuando se conforma un registro de entrada de datos, los valores de los campos deben ir separados al menos por un espacio en blanco. Si el campo es de tipo real, el punto debe estar presente y todo carácter que esté entre comillas hará parte de una constante alfanumérica. Si el campo indica miles, millones, etc., no debe estar presente el punto para indicar estas cantidades. El punto sólo se debe utilizar para indicar la parte decimal. Por ej. 1.000.000 (un millón), debe indicarse como 1000000.

? Si un campo indica porcentaje, los caracteres que lo determinan(%,$) no deben estar presentes en el registro de entrada. Serán erróneas las expresiones:
a > $100, P = 30%, k > 33%, etc.

c. Si la comparación en una condición involucra constantes alfanuméricas, éstas deben estar entre comillas; por ejemplo NOMBRE> “ISABEL”.

d. Una variable que tenga un contenido lógico, puede hacer parte de una expresión lógica compuesta. Si A = B > 20, o sea, se le asigna. F. o. V.

La siguiente expresión será correcta: X > Y ^ A

4.3. Ejercicios propuestos

22. Elaborar un algoritmo que entre el nombre de un empleado, su salario básico por hora y el número de horas trabajadas en el mes; escriba su nombre y salario mensual si éste es mayor de $450.000, de lo contrario escriba sólo el nombre.

23. Dados los valores A, B y C que son los parámetros de una ecuación de segundo grado, elaborar un algoritmo para hallar las posibles soluciones de dicha ecuación.

24. Se tienen tres esferas (A,B,C) de diferente peso, elaborar un algoritmo que determine cuál es la esfera de mayor peso.

25. Hacer un algoritmo que determine cuál es el mayor en un grupo de cuatro datos diferentes.

26. Hacer un algoritmo que determine la suma del valor menor y mayor en un grupo de 4 datos.

27. Elaborar un algoritmo que determine el valor equivalente en el sistema numérico decimal, de un número de cinco dígitos octales.

28. Dados tres valores positivos determinar si éstos no forman triángulo o si forman triángulo, decir si éste es: equilátero, isósceles o escaleno y obtener el área del triángulo.

29. Hacer un algoritmo que entre la ordenada (Y) y la abscisa (X) de un punto de un plano cartesiano y, determine si pertenece o no a la recta Y = 3X + 5.

30. Elaborar un algoritmo que entre un punto (x, y); diga si está, o no, en el área determinada por la parábola Y = 4 - X^2 y la recta Y = X - 3.

31. Elaborar un algoritmo que entre el par ordenado A,B; determine si está, o no, en el área comprendida entre las rectas Y = 2X - 2, Y = X + 1, X = 20.

32. Un almacén de escritorios hace los siguientes descuentos: si el cliente compra menos de 5 unidades se le da un descuento del 10% sobre la compra; si el número de unidades es mayor o igual a cinco pero menos de 10 se le otorga un 20% y, si son 10 o más se le da un 40%. Hacer un algoritmo que determine cuánto debe pagar un cliente si el valor de cada escritorio es de $800.000.

33. En un juego de preguntas que se responde “SI” o “NO”, gana quien responda correctamente las tres preguntas. Si se responde mal cualquiera de ellas, ya no se pregunta la siguiente y termina el juego. Las preguntas son:

1. ¿Simón Bolívar libertó a Colombia?
2. ¿Camilo Torres fue un guerrillero?
3. ¿El Binomio de Oro es un grupo de música vallenata?

Diseñe el registro de entrada.

34. Una frutería ofrece las manzanas con descuento según la siguiente tabla:

No. de manzanas compradas	*% descuento*
0 – 2	0%
3 – 5	10%
6 – 10	15%
11 en adelante	20%

Determinar cuánto pagará una persona que compre manzanas en esa frutería.

35. Cierta universidad tiene un programa para estimular a los estudiantes con buen rendimiento académico. Si el promedio es de 4,5 o más y el alumno es de pregrado entonces cursará 28 créditos y se le hará un 25% de descuento. Si el promedio es mayor o igual a 4,0 pero menor que 4,5 y el alumno es de pregrado, entonces cursará 25 créditos y se le hará un 10% de

descuento. Si el promedio es mayor que 3,5 y menor que 4,0 y es de pregrado, cursará 20 créditos y no tendrá ningún descuento. Si el promedio es mayor o igual a 2,5 y menor que 3,5 y es de pregrado, cursará 15 créditos y no tendrá descuento. Si el promedio es menor de 2,5 y es de pregrado, no podrá matricularse. Si el promedio es mayor o igual a 4,5 y es de posgrado, cursará 20 créditos y se le hará un 20% de descuento. Si el promedio es menor de 4,5 y es de posgrado cursará 10 créditos y no tendrá descuento.

Hacer un algoritmo que determine cuánto debe pagar un estudiante y cuántos créditos registra si el valor de cada crédito es de $50000 para pregrado y $300000 para posgrado.

36. Un proveedor de computadores ofrece descuento del 10%, si cuesta $1.000.000 o más. Además, independientemente, ofrece el 5% de descuento si la marca es ABACO. Determinar cuánto pagará, con lVA incluido, un cliente cualquiera por la compra de una computadora.

37. Determinar el precio de un pasaje de ida y vuelta por avión, conociendo la distancia a recorrer, el número de días de estancia y sabiendo que si la distancia es superior a 1.000 km, y el número de días de estancia es superior a 7, la línea aérea le hace un descuento del 30%. El precio por kilómetro es de $89,50.

Capítulo 5

ESTRUCTURA REPETITIVA

Hasta el momento se ha encontrado que cada una de las instrucciones que conforman el algoritmo se ejecutan una, y sólo una vez, en el mismo orden en que aparecían. Los algoritmos de este tipo son realmente simples, ya que no incluyen una estructura que permita que un grupo de instrucciones se ejecute varias veces, como resultado de una determinada condición.

La mayoría de los problemas dentro de la programación exigen que un grupo de instrucciones que hacen un cálculo determinado no se hagan para un ente específico, sino que sea aplicado a muchos para realizar el mismo cálculo. Por ejemplo: el cálculo del promedio de créditos de los estudiantes de una universidad. En este caso es necesario el establecimiento de un modelo que obtenga el promedio para uno y que a la vez sirva para todos los estudiantes, es decir, que el grupo de pasos que conforman el modelo de solución de un determinado proceso dentro del algoritmo, se repita tantas veces como sea necesario.

La estructura repetitiva, también conocida como estructura MIENTRAS o MIENTRAS-QUE-, permite ordenar la realización de una o más instrucciones (secuencia), cero o más veces, con base en el valor de verdad que arroje la evaluación de una expresión de tipo lógico. Esta expresión le permite al algoritmo tomar la decisión de repetir o dejar de ejecutar el grupo de instrucciones.

La estructura está formada por dos partes: la expresión de tipo lógico que es evaluada cada vez que se intenta repetir el proceso y, el grupo de instrucciones donde debe haber, por lo menos, una que permita modificar el resultado de la expresión lógica. De lo contrario, nunca se terminaría la repetición de la ejecución de las instrucciones y sería un proceso infinito.

5.1. Representación

```
MIENTRAS <expresión lógica> HAGA
    <secuencia>
FIN_MIENTRAS
```

Ejemplo

```
MIENTRAS A > B HAGA
    ESCRIBA: A
    A = A - 1
FIN_MIENTRAS
```

5.2. Funcionamiento

Cuando se llega a la estructura se evalúa la expresión lógica; sí ésta es:

Falsa, no se ejecuta la secuencia de instrucciones y continuará hacia abajo si hay más instrucciones o, terminará la ejecución si no hay más instrucciones; es decir, la secuencia se repetirá cero veces.

Si es *Verdadera*, se ejecuta la secuencia una vez y automáticamente (no hay que decírselo) regresa a evaluar la expresión; si nuevamente es verdadera se repite la ejecución de la secuencia y vuelve a evaluar la expresión. Este proceso es cíclico porque se repite siempre que la condición sea verdadera; también se puede decir que la secuencia no se repite o deja de repetirse cuando la condición es falsa.

5.3. Variables tipo contador

Muchas veces en los procesos repetitivos es necesario hacer el conteo de sucesos o acciones internas del ciclo; este proceso de conteo se hace con una variable que se va incrementando cada vez que el ciclo se repite.

El contador es una variable que se incrementa o disminuye en un valor constante cada que ocurre una acción o suceso. La forma general de los contadores es la siguiente:

CONTADOR = CONTADOR + <valor constante>

Como el contador aparece al lado derecho en la expresión, antes de usarse se le debe dar un valor que borre lo que había en la dirección de memoria asignada a la variable utilizada. Esa orden de borrado debe indicarse una vez; por lo tanto, debe estar antes de activar el ciclo donde se necesite el contador, de lo contrario se repetirá la orden de borrado cada vez que se repite el ciclo. Ejemplo:

Una orden de borrado puede ser: CONTADOR = 0.

5.4. Variables tipo acumulador

Un acumulador o totalizador es una variable cuya misión es almacenar cantidades variables resultantes de procesos sucesivos. La diferencia con el contador radica en que el incremento o disminución de cada suma es variable en lugar de constante, como en el caso del contador.

La forma general del acumulador es:

ACUMULADOR = ACUMULADOR + <expresión>

Como también aparece al lado derecho, antes de utilizarlo se hace necesario asignarle un valor inicial, por ejemplo: ACUMULADOR = 0.

Ejercicio resuelto Nº 16

Hacer un algoritmo que encuentre la suma de los primeros cinco números naturales.

Análisis

Datos de entrada

No hay. Como se dijo antes, los datos de entrada son aquellos valores que se desconocen en el momento de hacer el algoritmo, en este caso los valores a sumar son conocidos, se sabe que se sumarán los números 1, 2, 3, 4 y 5, por lo tanto se pueden generar en el proceso. Observe que no tiene lógica la instrucción LEA: 1 o LEA: 2, además, en la instrucción de lectura debe haber por lo menos una variable y no un valor constante.

Datos de salida

• La suma de los cinco números.

Continuación

Proceso

Se hará uso de un contador que sirva para generar los números entre 1 y 5 y al mismo tiempo controle la condición del ciclo; también se usará un acumulador para que, en la medida que se genere un valor, lo totalice. En este caso pueden presentarse varias alternativas, dependiendo del valor inicial que se le dé al contador, el cual influirá de modo directo en la forma de establecer la expresión lógica que controlará el ciclo.

Definición de variables

NUM: Variable tipo contador que generará los números a sumar y servirá para controlar el número de veces que debe repetirse el ciclo; debe partir de un valor inicial asignado fuera del ciclo.

SUMA: Variable tipo acumulador que totalizará el valor generado en NUM, siempre que su contenido esté entre 1 y 5.

Algoritmo

Solución 1:

```
INICIO
    NUM=0                          (1) Inicializa el contador
    SUMA=0                         (2) Inicializa el acumulador
    MIENTRAS NUM < 5 HAGA
        NUM=NUM+1                  (3) Modifica la expresión
        SUMA=SUMA+NUM              (4) Actualiza el acumulador
    FIN_MIENTRAS
    ESCRIBA: "LA SUMA ES:", SUMA   (5) Termina el ciclo cuando
FIN_INICIO                                 NUM vale 5
```

Prueba de escritorio

Miremos el funcionamiento del algoritmo paso por paso.

Se ejecuta la instrucción (1)	NUM	
	0	contenido de NUM
Se ejecuta la instrucción (2)	SUMA	
	0	contenido de SUMA

Al llegar a la estructura MIENTRAS se evalúa la expresión: ES NUM < 5

(ES 0 < 5): Sí, se ejecutan 3 y 4 una vez

3) se le adiciona 1 al contenido de NUM, su nuevo valor será 0 + 1 = 1

NUM	
1	se destruye el valor 0

4) se le agrega NUM a SUMA o sea 0 + 1, su nuevo valor es

SUMA	
1	se destruye el valor anterior

Al llegar al final de las instrucciones de la secuencia del ciclo, regresa a evaluar la condición: es NUM < 5 (ES 1 < 5): Sí, se ejecutan 3 y 4

(3) NUM = NUM + 1 (1 + 1) NUM
2

(4) SUMA = SUMA + NUM (1 + 2) SUMA
3

Regresa a evaluar la condición : ES NUM < 5 (ES 2 < 5): Sí, se ejecutan 3 y 4

(3) NUM = NUM + 1 (2 + 1) NUM
3

(4) SUMA = SUMA + NUM (3 + 3) SUMA
6

Regresa a evaluar la condición : ES NUM < 5 (ES 3 < 5): Sí, se ejecutan 3 y 4

(3) NUM = NUM + 1 (3 + 1) NUM
4

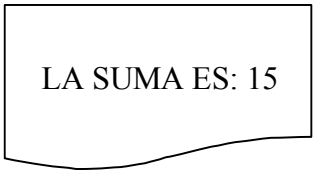
(4) SUMA = SUMA + NUM (6 + 4) SUMA
10

Regresa a evaluar la condición .- ES NUM < 5 (ES 4 < 5): Sí, se ejecutan 3 y 4

(3) NUM = NUM + 1 (4 + 1) NUM
5

(4) SUMA = SUMA + NUM (10 + 5) SUMA
15

Regresa a evaluar la condición: ES NUM < 5 (ES 5 < 5): No, como la condición es falsa salta a ejecutar la instrucción (5) que está después de la estructura MIENTRAS, o sea que se sale del ciclo. Al ejecutarse la instrucción (5) se mostrará el contenido actual de la variable SUMA, por tanto la salida será:



Solución 2:

```
INICIO
    NUM=0
    SUMA=0
    MIENTRAS NUM <=5 HAGA
        SUMA=SUMA + NUM
        NUM=NUM + 1
    FIN_MIENTRAS
    ESCRIBA= "LA SUMA ES:", SUMA
FIN INICIO
```

Solución 3:

```
INICIO
    NUM = 1
    SUMA = 1
    MIENTRAS NUM <  5 HAGA
            NUM = NUM + 1
            SUMA = SUMA + NUM
    FIN_MIENTRAS
    ESCRIBA: "LA SUMA ES:", SUMA
FIN_INICIO
```

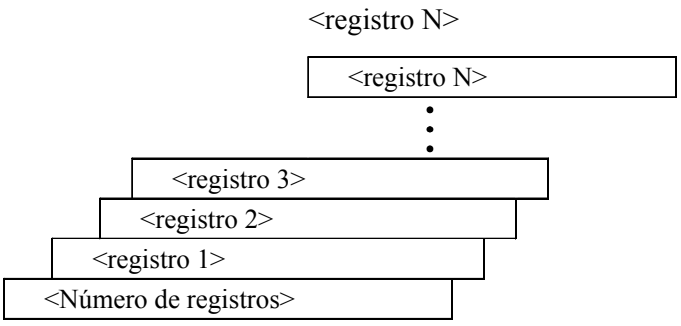
Cuando se implementa un ciclo dentro de un algoritmo puede ocurrir que previamente a él se conozca el número de iteraciones que debe hacer; otras veces, aunque se sabe que el ciclo tiene un número finito de iteraciones, la definición del problema no permite conocer el número de ellas; esto implica establecer dos esquemas de solución, uno para cada caso, dependiendo de si dentro de la definición del problema se da o no el número de veces que se debe repetir el ciclo.

En muchos casos el esquema a utilizar depende de la forma como se maneje la cantidad de información o registros a tener en cuenta dentro del proceso. Por ejemplo: si se desea calcular el promedio crédito a un conjunto de estudiantes, es posible que se conozca o no la cantidad de ellos.

5.5. Esquema cuantitativo

El esquema cuantitativo es utilizado cuando se conoce el número de veces que debe repetirse un ciclo determinado, antes de activarse la estructura repetitiva. Para el ejemplo resuelto No. 18 antes de llegar al ciclo se proporciona la cantidad de estudiantes, que en este caso sería igual a la cantidad de registros a procesar.

El número de iteraciones o cantidad de registros puede ser un valor constante o, generalmente, una variable de entrada cuyo valor es proporcionado al algoritmo antes de activarse el ciclo. La organización de los registros a procesar (archivo) en el caso de un modelo cuantitativo es:



Donde N es el número de registros y su valor es proporcionado al principio del proceso.

Ejercicio resuelto Nº 17

(Generalización del ejercicio Nº 16)

Hacer un algoritmo que encuentre la suma de los primeros N números naturales.

Continuación

Análisis

Datos de entrada

- La cantidad de números a tener en cuenta en la suma.

Datos de salida

- La suma de los primeros N números naturales.

Proceso

Primero se debe conocer la cantidad de números naturales a sumar y luego generar y sumar los números comprendidos entre 1 y esa cantidad. El proceso sería el mismo utilizado en el ejercicio 16.

Definición de variables

N: Cantidad de números naturales a sumar.
NUM: Contador que genera los números entre 1 y N y que a su vez controla el ciclo.
SUMA: Suma de los números entre 1 y N.

Algoritmo

```
INICIO
    SUMA = 0
    NUM = 1
    LEA: N
    MIENTRAS NUM ≤ N HAGA
        SUMA = SUMA + NUM
        NUM = NUM + 1
    FIN_MIENTRAS
    ESCRIBA: "LA SUMA ES:", SUMA
FIN_INICIO
```

Prueba de escritorio

Si el valor de N es 7:

N	NUM	SUMA
7	~~1~~	~~0~~
	~~2~~	~~1~~
	~~3~~	~~3~~
	~~4~~	~~6~~
	~~5~~	~~10~~
	~~6~~	~~15~~
	~~7~~	~~21~~
	8	28

Salida

FALTA GRAFICO PAG. 97 LIBRO

Ejercicio resuelto Nº 18

Por cada uno de los N estudiantes de una universidad, donde cada uno cursa 4 materias se tienen los siguientes datos:
- Código del estudiante.
- Nota materia 1.
- Número de créditos de la materia 1.
- Nota materia 2.
- Número de créditos de la materia 2.
- Nota materia 3.
- Número de créditos de la materia 3.
- Nota materia 4.
- El Número de créditos de la materia 4.

Hacer un algoritmo que encuentre para cada estudiante el número de créditos cursados y el promedio crédito.

Análisis

Datos de entrada
- Número de estudiantes.
- Código estudiante.
- Nota materia 1.
- Número de créditos materia 1.
- Nota materia 2.
- Número de créditos materia 2.
- Nota materia 3.

Continuaciòn

- Número de créditos materia 3.
- Nota materia 4.
- Número de créditos materia 4.

Datos de salida
- Código del estudiante.
- Número de créditos cursados.
- Promedio crédito.

Proceso

Primero se entrará al algoritmo el número de estudiantes y luego con este valor se construirá un ciclo donde, en cada ejecución, encuentre el promedio crédito para cada estudiante. El promedio se obtiene mediante la siguiente fórmula:

Número de créditos cursados = **FALTA FÓRMULA PAG. 98 LIBRO**

Definición de variables

NE: Número de estudiantes.
COD: Código.
N1: nota materia 1.
NC1: Número de créditos materia 1.
N2: nota materia 2.

NC2: Número de créditos materia 2.
N3: nota materia 3.
NC3: Número de créditos materia 3.
N4: nota materia 4.
NC4: Número de créditos materia 4.
NUME: Contador de estudiantes y controlador de ciclo.
NCC: Número de créditos cursados.
PC: Promedio crédito.

Algoritmo

```
INICIO
    NUME = 1
    LEA: NE
    MIENTRAS NUME < = NE HAGA
        LEA: COD, N1, NC1, N2, NC2, N3, NC3, N4, NC4
        NCC = NC1 + NC2 + NC3 + NC4
        PC = (NI * NC1 + N2 * NC2 + N3 * NC3 + N4 * NC4) / NCC
        ESCRIBA: "EL ESTUDIANTE CON CODIGO", COD,
        NUME = NUME + 1
    FIN_MIENTRAS
FIN_INICIO
```

Prueba de escritorio

Si el número de estudiantes es 5 y los datos de cada curso los siguientes:

928 2.5 4.0 3.5 4.0 2.8 3.0 4.4 4.0

764 3.5 4.0 4.6 3.0 3.8 3.0 4.2 5.0

282 3.0 3.0 3.5 3.0 3.6 4.0 5.0 4.0

432 4.2 4.0 3.0 4.0 2.5 4.0 4.8 3.0

251 4.3 4.0 4.0 4.0 4.3 2.0 5.0 5.0

5

↑ ↑↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↑ ↑

NE	COD	N1	NC1	N2	NC2	N3	NC3	N4	NC4	NUME	NCC	PC
~~5~~	~~261~~	~~3.0~~	~~4~~	~~4.0~~	~~4~~	~~4.3~~	~~2~~	~~5.0~~	~~5~~	~~1~~	~~15~~	~~4.45~~
	~~432~~	~~4.2~~	~~4~~	~~3.0~~	~~4~~	~~2.5~~	~~4~~	~~4.8~~	~~3~~	~~2~~	~~15~~	~~3.55~~
	~~282~~	~~3.0~~	~~3~~	~~3.5~~	~~3~~	~~3.6~~	~~4~~	~~5.0~~	~~4~~	~~3~~	~~14~~	~~3.85~~
	~~764~~	~~3.5~~	~~4~~	~~4.6~~	~~3~~	~~3.8~~	~~3~~	~~4.2~~	~~5~~	~~4~~	~~15~~	~~4.01~~
	~~928~~	~~2.5~~	4	3.5	4	2.8	3	4.4	~~4~~	~~5~~	~~15~~	3.33

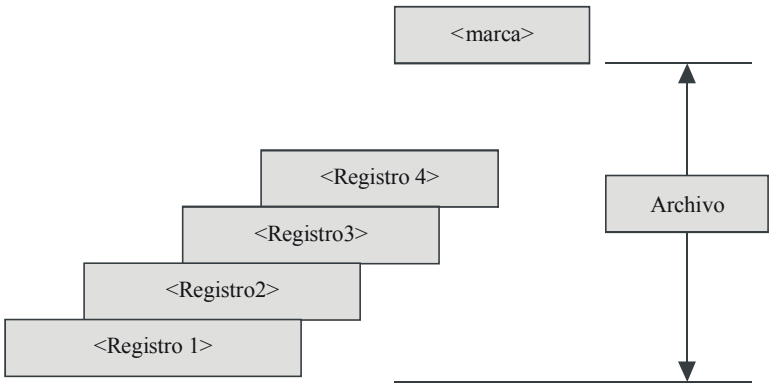
Salida

EL ESTUDIANTE CON CODIGO 251 CURSO 15 CREDITOS Y SU PROMEDIO FUE: 4,45
EL ESTUDIANTE CON CODIGO 432 CURSO 15 CREDITOS Y SU PROMEDIO FUE: 3,55
EL ESTUDIANTE CON CODIGO 282 CURSO 14 CREDITOS Y SU PROMEDIO FUE: 3,85
EL ESTUDIANTE CON CODIGO 764 CURSO 15 CREDITOS Y SU PROMEDIO FUE: 4,01
EL ESTUDIANTE CON CODIGO 928 CURSO 15 CREDITOS Y SU PROMEDIO FUE: 3,33

5.6. Esquema cualitativo

Este esquema es utilizado cuando no se conoce el número de veces que debe repetirse la secuencia de un determinado ciclo. En esta clase de ciclo el usuario decide cuándo terminar con el proceso repetitivo sin importar cuantas iteraciones se hayan efectuado; a diferencia de los ciclos anteriores, en donde el número de iteraciones está determinado desde el principio.

La implementación del modelo se hace demarcando el grupo de registros o datos a procesar mediante información inválida para el proceso; por lo tanto, se deben conocer las características de los datos y con base en esto usar una marca como registro final que delimite la información que interviene dentro del proceso. La marca o registro centinela, como también se le conoce, tiene el significado de no hay más datos, o simplemente no hay más registros para procesar; en otros términos se llegó al final del archivo. El registro que contiene la marca (centinela) se adiciona al final del archivo como último registro. La organización de la información sería:



El valor de la marca es producto de la escogencia de un campo dentro del registro que tenga demarcado su rango de valores posibles. La marca escogida será un valor del mismo tipo pero que no esté en el rango posible de valores permitidos para ese determinado campo. Si la marca es del mismo tipo del campo escogido, quiere decir que si el campo seleccionado es numérico el valor de la marca también lo debe ser, lo mismo sucede si el campo es alfanumérico o lógico.

Ejercicio resuelto Nº 19

Hacer un algoritmo que encuentre la suma y el promedio de un grupo de datos numéricos positivos que entran de a uno por registro.

Análisis

Datos de entrada

• El valor de cada número

Datos de salida

• La suma de los números
• El promedio del grupo de números.

Proceso

Como los datos entran de a uno por registro, se lee un número, se suma y se vuelve a leer el próximo para sumarlo también. ¿Cuándo terminar este proceso? Como en este caso solamente hay un campo en el registro se deben analizar las características de éste para escoger el valor que no esté permitido para él, de acuerdo a lo que nos dice el enunciado. Ahora bien, el campo es numérico positivo, por lo que ni el valor cero ni los valores negativos serían tenidos en cuenta en la suma de los números. Esto permite escoger un valor para la marca que esté en el rango menor o igual a cero. Si se escoge el valor cero, significa que se adicionará un registro con este valor y que cuando se encuentre es porque no hay más valores a sumar. En otras palabras, se lee un número, se compara con cero, si es mayor que cero se tiene en cuenta para la suma, se lee el próximo y se le hace la misma comparación y el proceso será cíclico, hasta que se encuentre un valor igual a cero; esto hará que se termine el ciclo de lectura.

Para encontrar el promedio es necesario contar los números que se van a sumar y su cálculo debe hacerse fuera del ciclo cuando se obtenga la suma total.

Continuación

Promedio = ; pero este N no está dado por el enunciado, por lo tanto hay que calcularlo.

Definición de variables

NUM: Variable que almacenará uno a uno cada número o valores a sumar, una vez se ejecuta una instrucción de entrada.
SUMA: Acumulador que totalizará los valores a entrar
N: Contador que contabilizará el total de números.
PROM: Promedio de los números.

Algoritmo

```
INICIO
    SUMA, N = 0                                                    (1)
    LEA: NUM                                                       (2)
    MIENTRAS NUM > 0 HAGA
        SUMA = SUMA + NUM                                          (3)
        N = N + 1                                                  (4)
        LEA: NUM                                                   (5)
    FIN_MIENTRAS
    PROM = SUMA / N                                                (6)
    ESCRIBA: "LA SUMA ES:", SUMA "Y EL PROMEDIO", PROM             (7)
FIN_INICIO
```

Prueba de escritorio

Si el grupo de números es 20, 50, 75, 90, 1 el grupo de registros (archivo) estaría conformado de la siguiente manera:

FALTA GRAFICO PAG. 102 LIBRO

Registro adicional con la marca 0

90

75

50

20

Hagamos un seguimiento, paso a paso, de lo que sucede al ejecutarse cada instrucción.

Se ejecuta la instrucción (1) SUMA, N

0 0

Se ejecuta la instrucción (2) NUM

20 (Valor del primer registro)

Al llegar a la estructura MIENTRAS se evalúa la expresión: ES NUM > 0 (ES 20 > 0): sí, se ejecutan (3), (4) y (5)

(3) Se le agrega NUM a SUMA, o sea, 0 + 20;
su nuevo contenido será: SUMA

20

(4) Se le adiciona 1 a N, o sea, 0+1;
nuevo contenido de N N

1

(5) Se lee un nuevo valor de NUM; NUM

50

Llega al final del ciclo y regresa a evaluar la condición: ES NUM > 0 (ES 50>0): sí, se ejecutan (3), (4) y (5)

(3) Se le agrega NUM a SUMA, o sea, 20+50;
su nuevo contenido será: SUMA

70

(4) Se le adiciona 1 a N, o sea, 1+1;
nuevo contenido de N N

2

(5) Se lee un nuevo valor de NUM; NUM

75

Llega al final del ciclo y regresa a evaluar la condición: ES NUM > 0 (ES 75>0): sí, se ejecutan (3), (4) y (5)

(3) Se le agrega NUM a SUMA, o sea, 70 + 75;
su nuevo contenido será: SUMA

145

(4) Se le adiciona 1 a N, o sea, 2+1;
nuevo contenido de N N

3

(5) Se lee un nuevo valor de NUM; NUM
90

Llega al final del ciclo y regresa a evaluar la condición: ES NUM > 0 (ES 90>0): sí, se ejecutan (3), (4) y (5)

(3) se le agrega NUM a SUMA, o sea, 145 + 90;
su nuevo contenido será: SUMA

235

(4) se le adiciona 1 a N, o sea, 3 + 1;

nuevo contenido de N N

4

(5) Se lee un nuevo valor de NUM NUM
1

Llega al final del ciclo y regresa a evaluar la condición: ES NUM > 0 (ES 1>0): sí, se ejecutan (3), (4) y (5) (3) Se le agrega NUM a SUMA, o sea, 235 + 1;

su nuevo contenido será: SUMA

236

(4) Se le adiciona 1 a N, o sea, 4+1;
nuevo contenido de N N

5

(5) Se lee un nuevo valor de NUM; NUM
0

Se evalúa la condición ES NUM > 0 (ES 0 > 0): NO; como la condición es falsa, se ejecutará la instrucción siguiente al ciclo MIENTRAS.

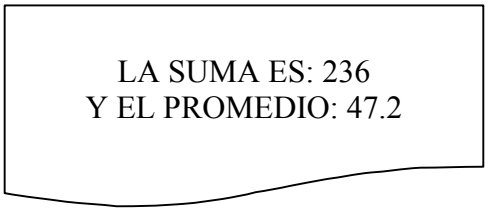
(6) Se calcula el valor del PROM como SUMA / N

(236 / 5) PROM

47.2

(7) Se muestra el último valor de SUMA y el valor de PROM

Salida



Ejercicio resuelto Nº 20

En un supermercado un ama de casa pone en su carrito los artículos que va tomando de los estantes. La señora quiere asegurarse de que el cajero le cobre bien lo que ella ha comprado; por lo cual, cada vez que toma un artículo distinto le coloca un código numérico mayor que cero, anota la cantidad de artículos iguales y su precio y, determina cuánto dinero gastará en este artículo; a esto le suma lo que iría gastando en los demás artículos, hasta que decide que ya tomó todo lo que necesitaba. Hacer un algoritmo que le muestre a la señora el número del artículo, la cantidad de artículos de cada especie, el importe total por artículo y el importe total de la compra.

Análisis

Datos de entrada

- Código asignado al artículo.
- Cantidad de artículos iguales de cada precio.
- Precio de cada artículo.

Continuación

Datos de salida

- Código asignado al artículo.
- Precio de cada artículo.
- Valor total a pagar por cada clase de artículo.
- Valor total de la compra.

Proceso

Como no se conoce la cantidad de artículos que comprará la señora, se debe usar un esquema cualitativo. Se sabe que los códigos que la señora coloca son numéricos mayores que cero; por lo tanto, se puede escoger éste como marca que indique que no hay más artículos por registrar.

Los cálculos dentro del proceso serán:

Valor a pagar por artículo = cantidad * precio del artículo. Este total, como debe mostrarse por artículo, debe imprimirse dentro del ciclo.

Valor total de la compra = Suma de valores totales a pagar por artículo; éste debe mostrarse después del ciclo, que es cuando se tiene su último valor.

Definición de variables

CODA: Código asignado por artículo.
CAN: Cantidad de artículos iguales.
PRECIO: Precio del artículo.
VALTPA: Valor total por artículo.
VTC: Valor total de la compra.

Algoritmo

```
INICIO
    VTC = 0
    ESCRIBA: "CODIGO ARTICULO CANTIDAD PRECIO UNITARIO
    LEA: CODA
    MIENTRAS CODA > 0 HAGA
        LEA:CAN, PRECIO
        VALTPA = CAN * PRECIO
        VTC = VTC + VALTPA
        ESCRIBA: CODA, CAN, PRECIO, VALTPA
        LEA: CODA
    FIN_MIENTRAS
    ESCRIBA: "TOTAL DE LA COMPRA:", VTC
FIN_INICIO
```

FALTA GRAFICO PAG. 107 LIBRO – ARREGLAR -

Prueba de escritorio

Registro con la marca 0

6	1	5000.00
5	2	10300.20
4	6	3800.00
3	4	2252.50
2	12	345.00
1	5	520.50

CODA		CAN	PRECIO	VALTP	VTC
1	5	520.00	2602.50	0.00	
2	12	345.00	4140.00	2602.50	
3	4	2252.50	9010.00	6742.50	
4	6	3800.00	22800.00	15752.50	
5	2	10320.00	20640.40	38552.50	

Salida

Código artículo	Cantidad	Precio unitario	Subtotal
1	5	520.50	2602.50
2	12	345.50	4140.00
3	4	2252.50	9010.00
4	6	3800.00	22800.00
5	2	10300.20	20640.40
6	1	5000.00	5000.00
Total de la compra:		**$ 64.192.90**	

5.7. Formato general de los dos esquemas

```
INICIO
    <inicializar contador>
    LEA: <cantidad de iteraciones>
    MIENTRAS <expresión lógica> HAGA
        LEA: <registro de datos>
        <PROCESO>
    FIN_MIENTRAS
```

FIN_INICIO

Esquema cualitativo

```
INICIO
    LEA: <campo escogido>
    MIENTRAS <expresión lógica> HAGA
    LEA: <resto del registro>
            <PROCESO>
            LEA: <campo escogido>
    FIN_MIENTRAS
FIN_INICIO
```

5.8. Variables tipo bandera o switche

La bandera es una variable que generalmente usa dos valores excluyentes o diferentes, su contenido es uno cualquiera de dos valores definidos por el programador, el cual ordena cuando cambia su contenido.

La bandera es utilizada dentro de la programación como un seleccionador de una de dos alternativas a seguir dentro del algoritmo. Antes de llegar a la expresión que se utilice para bifurcar la ejecución, la bandera debe tener asignado uno de los dos valores.

Los valores escogidos para la bandera pueden ser de cualquier tipo de dato, por ejemplo:

Tipo de dato	Valores de la bandera
numérico	0 y 1, 1 y 2
lógico	.V. y .F.
carácter	'S' y 'N', "SI" y "NO", '0' y '1', "ENCONTRADO" y "NO ENCONTRADO"

La bandera también puede utilizarse para la implementación de un modelo cualitativo, cuando no es posible seleccionar un valor que no esté dentro del rango de valores permitido para el campo seleccionado dentro del registro.

Ejercicio resuelto Nº 21

Elaborar un algoritmo que genere los primeros N términos de la sucesión:

17 15 18 16 19 17 20 18 21...

Análisis

Datos de entrada

- La cantidad de términos a generar.

Datos de salida

- Cada uno de los términos de la sucesión a generar.

Proceso

El algoritmo debe ser útil para generar cualquier cantidad de términos que se le solicite y ésta se le debe dar al algoritmo antes de empezar a generar los términos de la sucesión.

Para generar los términos se parte del número 17, sumando alternadamente -2 y 3 al último valor generado. Esto implica que debe conocerse cuándo se suma -2 y cuándo se suma 3; una solución es utilizar una bandera que tome los valores 1 y 2, la cual se inicializa en 1; luego, en una estructura condicional se pregunta si su valor es 1; si es cierto se resta 2 al término, si no, se le suma 3 y se cambia el valor de la bandera de 1 a 2.

Definición de variables

NT: Número de términos a generar.
TER: Término generado, parte del valor 17.
CONT: Contador y generador de términos entre 1 y NT.
BAN: Bandera, es utilizada para seleccionar si se suma -2 o 3.

Algoritmo

```
INICIO
    LEA: NT
    TER = 17
    CONT, BAN = 1
    MIENTRAS CONT ≤ NT HAGA
            ESCRIBA: TER
            SI BAN = 1 ENTONCES
                    TER = TER - 2
                    BAN = 2
              SINO
                    TER = TER + 3
                    BAN = 1
            FIN_SI
            CONT=CONT + 1
    FIN_MIENTRAS
FIN_INICIO
```

Prueba de escritorio

Hagamos una prueba que genere 10 términos:

NT	TER	CONT	BAN
10	~~17~~	~~1~~	~~1~~
	~~15~~	~~2~~	~~2~~
	~~18~~	~~3~~	~~1~~
	~~16~~	~~4~~	~~2~~
	~~19~~	~~5~~	~~1~~
	~~17~~	~~6~~	~~2~~
	~~20~~	~~7~~	~~1~~
	~~18~~	~~8~~	~~2~~
	~~21~~	~~9~~	~~1~~
	~~19~~	~~10~~	~~2~~
	22	11	1

Salida

FALTA GRAFICO PAG. 110 LIBRO

Ejercicio resuelto Nº 22

Hacer un algoritmo que encuentre la suma de los valores de un conjunto de números enteros, la cantidad de valores negativos, positivos, iguales a cero y el total de números en el conjunto.

Análisis

Datos de entrada
- El valor entero a sumar.

Datos de salida
- La suma de todos los números enteros leidos.
- La cantidad de valores negativos.
- La cantidad de valores positivos.
- La cantidad de valores iguales a cero.
- El total de números en el conjunto.

Proceso
Como no se conoce la cantidad de valores a tener en cuenta en la suma, es necesario utilizar un modelo cualitativo. Al tener el registro de entrada un sólo campo y el rango de valores de éste son los números enteros (negativos, cero y positivos), no existe un valor del mismo tipo, con quién compararlo para terminar el ciclo. Una forma de implementar el modelo cualitativo en casos como éste es mediante el uso de una bandera.

Si se escogen los valores "SI" y "NO" para la bandera, ésta empezará en "SI", y cuando ya no existan más valores a sumar se le dará el valor de "NO". Esto implica entrar un valor, sumarlo, y de acuerdo a si hay más valores o no, darle el valor de "SI' o "NO" a la bandera. Para realizar las sumas basta comparar el valor con cero y determinar si éste es cero, negativo o positivo.

Definición de variables
NUM: Número o valor a sumar.
SUMA: Sumatoria de los números.
VALN: Cantidad de valores negativos.
VAL0: Cantidad de valores iguales a cero.
VALP: Cantidad de valores positivos.
TOTALN: Total de números sumados. Este valor se calculará después del ciclo como VALP + VALN + VAL0.
SIGA: Variable tipo bandera que se inicializará en "SI"; cuando no existan más valores tomará el valor de "NO".

Algoritmo

INICIO

```
    VALN, VAL0, VALP, SUMA = 0
    SIGA = "SI"
    MIENTRAS SIGA = "SI" HAGA
            LEA: NUM
            SUMA = SUMA + NUM
            SI NUM > 0 ENTONCES
                    VALP= VALP + 1
              SINO
                    SI NUM = 0  ENTONCES
                            VAL0 = VAL0 + 1
                      SINO
                            VALN = VALN + 1
                    FIN_SI
            FIN_SI
            LEA: SIGA
    FIN_MIENTRAS
    TOTALN = VALP + VALN + VAL0
    ESCRIBA:    "SUMA DE LOS  NÚMEROS:", SUMA,                                    "NU
                                    TOTALN
FIN_INICIO
```

Si los valores a sumar son 20, -15, 30, 0, -4, 0, 50, la conformación de la entrada de datos del algoritmo sería:

```
NO
50
SI
0
SI
-4
SI
0
SI
30
SI
-15
SI
20
```

VALN	VALO	VALP	SUMA	SIGA	TOTALN	NUM
~~0~~	~~0~~	~~0~~	~~0~~	~~SI~~	7	~~20~~
~~1~~	~~1~~	~~1~~	~~20~~	~~SI~~		~~-15~~
~~2~~	~~2~~	~~2~~	~~5~~	~~SI~~		~~30~~
		~~3~~	~~25~~	~~SI~~		~~0~~
			~~25~~	~~SI~~		~~-4~~
			~~21~~	~~SI~~		~~0~~
			~~21~~	~~SI~~		~~50~~
			71	NO		

Salida

SUMA DE LOS NÚMEROS: 81
NÚMEROS NEGATIVOS: 2
NÚMEROS IGUALES A CERO: 2
NÚMEROS POSITIVOS: 3
TOTAL DE NÚMEROS: 7

5.9. Ruptura de ciclos

Dentro de la programación, algunas veces, es necesario hacer que un ciclo se detenga abruptamente (aborte), porque ya se cumplió algo que se necesitaba o se estaba buscando, por lo que, posiblemente, no se alcance a satisfacer completamente en una forma normal la culminación de la cantidad de veces que debe ejecutarse o repetirse un ciclo. La ruptura se hace cambiando el sentido de la expresión lógica que controla el ciclo, para que ésta sea falsa y no se continúe ejecutando la secuencia de instrucciones.

Ejercicio resuelto Nº 23

Se tiene una serie de N números enteros. Se quiere saber si dentro de ellos existe, por lo menos, uno cuyo valor sea cero.

Análisis

- La cantidad de números.
- El valor de cada número.

Datos de salida

- La salida será un mensaje cuyo contenido dependerá de si se encuentra o no el valor cero dentro de los N números.

Continuación

Si hay un valor cero, la salida será: "SI HAY UN CERO ENTRE LOS NUMEROS"; y si no la salida será: "NO HAY UN VALOR IGUAL A CERO".

Proceso

Como el enunciado dice que son N números este valor será dado al algoritmo; por lo tanto, se puede implementar un esquema cuantitativo a través de un contador que toma valores entre 1 y la cantidad de números.

Como el objetivo del algoritmo es decir si se encuentra, o no, un valor cero dentro del grupo de números, cada vez que se entre un valor se compara con cero, si éste es:

- Diferente de cero se entra el próximo valor y se le hace la misma operación.
- Si es igual a cero se debe detener la búsqueda, por lo tanto se le cambia de valor a la bandera utilizar.

Cuando se termine el ciclo es necesario saber si se encontró, o no, un valor cero; esto lo hacemos mediante una variable tipo bandera cuyo valor inicial será F. y se le cambiará de contenido cuando se encuentre un valor igual a cero.

Definición de variables

NN: Cantidad de números en el grupo.
NUM: Cada valor de un número del grupo.
CONTA: Contador de valores entre 1 y NN.

ENCONTRADO: Variable tipo bandera cuyo valor inicial es F.

Algoritmo

```
INICIO
    LEA: NN
    CONTA = 1
    ENCONTRADO = .F.
    MIENTRAS (CONTA < = NN) ∧ (ENCONTRADO =.F.) HAGA
            LEA: NUM
            SI NUM = 0 ENTONCES
                    ENCONTRADO =.V.
              SINO
                    CONTA = CONTA + 1
            FIN _SI
    FIN_MIENTRAS
    SI ENCONTRADO = .V.  ENTONCES
            ESCRIBA: "SI HAY UN CERO ENTRE LOS NUMEROS"
      SINO
            ESCRIBA: "NO HAY UN CERO ENTRE LOS NUMEROS"
    FIN_SI
FIN_INICIO
```

Prueba de escritorio

Si los valores son 6.

PRUEBA 1

Valores: 30 -15 10 -4 1 122

NN	CONTA	ENCONTRADO	NUM
6	~~1~~	.F.	~~30~~
	~~2~~		~~-15~~
	~~3~~		~~10~~
	~~4~~		~~-4~~
	~~5~~		~~1~~
	~~6~~		~~122~~
	7		

Salida

NO HAY UN CERO ENTRE LOS NÚMEROS

PRUEBA 2

Valores: 140 89 0 -14 16 0

NN	CONTA	ENCONTRADO	NUM
6	~~1~~	.F.	~~140~~
	~~2~~		~~89~~
	~~3~~	.V.	0

Salida

SÍ HAY UN CERO
ENTRE LOS NÚMEROS

5.10. Rompimientos de control de ejecución

Los rompimientos de control en la ejecución de un algoritmo se presentan cuando es necesario detener el proceso porque se ha producido un cambio y se deben dar reportes de lo que ha acontecido antes de la ruptura. En estos casos los registros a procesar están agrupados por lotes; cada uno de ellos pertenece a una determinada entidad, persona o cosa y tiene un campo cuyo contenido cambia cuando se pasa de un lote a otro. Para detectar el cambio de un lote de registros a otro se hace uso de una variable auxiliar, que almacene el valor que tenía el campo antes del cambio de contenido. Al leer un registro y el valor del campo no cambia, se sigue el proceso normal; si su valor es diferente al de la variable auxiliar, se detiene el control de la ejecución para efectuar procesos distintos al de la rutina o secuencia.

Ejercicio resuelto Nº 24

Una empresa con varias sucursales tiene la siguiente información por cada uno de sus empleados:

- El código del empleado.
- El código de la sucursal.
- Nombres.
- El salario mensual.

La información o registros de los empleados de cada sucursal se encuentran juntos dentro del archivo.

Hacer un algoritmo que produzca un reporte de los empleados, dando totales de salarios por cada sucursal.

Análisis

Datos de entrada
- Código del empleado.
- Código de la sucursal.
- Nombres.
- Salario mensual.

Datos de salida
- Código de la sucursal.
- Código del empleado.
- Nombres.

Continuación

- Salario.
- Total de salarios por sucursal.

Proceso

Se debe implementar un esquema cualitativo ya que no se conoce la cantidad de empleados que tiene la empresa, el ciclo de lectura terminará cuando se encuentre un código de empleado cuyo valor sea cero. Para dar los totales por sucursal se hace necesario detectar cuándo se pasa de un registro perteneciente a una sucursal a otro de una sucursal distinta; para reconocer el cambio se hará uso de una variable auxiliar que

almacene el código de la sucursal anterior, de tal manera, que cuando el código de la sucursal leída sea diferente al código de la sucursal anterior se produzca el reporte de los totales de salarios por sucursal. La implementación de la ruptura se hará mediante un ciclo MIENTRAS, cuya expresión lógica será verdadera si el código de la sucursal anterior igual al código de la sucursal leída y falsa cuando los códigos sean distintos.

El formato de los reportes o salida por sucursal será el siguiente:

REPORTE DE SALARIOS SUCURSAL: XX

CÓDIGO	NOMBRES	SALARIO
XXX	XXX	XXX.xx

TOTAL SALARIOS DE LA SUCURSAL: $XX.XXX.xx

FALTA GRAFICO PAG. 117 LIBRO

Definición de variables

COD: Código del empleado.
CODS: Código de la sucursal.
NOM: Nombres del empleado.
SAL: Salario.
TOTSUC: Total de salarios por cada sucursal.
CODSA: Código de la sucursal anterior.

Algoritmo

```
INICIO
    LEA: COD, CODS, NOM, SAL
    MIENTRAS COD <> 0 HAGA
            ESCRIBA: "REPORTE DE SALARIOS SUCURSAL:",
            CODS
            ESCRIBA: "CODIGO NOMBRES SALARIO"
            CODSA = CODS
            TOTSUC = 0
            MIENTRAS (CODS = CODSA) ∧ (COD <> 0) HAGA
                    ESCRIBA: COD, NOM, SAL
                    TOTSUC = TOTSUC + SAL
                    LEA: COD
                    SI COD <> 0 ENTONCES
                            LEA: CODS, NOM, SAL
                    FIN_SI
            FIN_MIENTRAS
            ESCRIBA: "TOTAL SALARIOS DE LA SUCURSAL:$",
    FIN_MIENTRAS
FIN _INICIO
```

Prueba de escritorio

0

311 04 JAIRO ANTONIO 388734

118 03 FLORENCIA MARIA 425000

119 03 RUTH DE JESÚS 119325

212 02 CORINA FRANCISCA 823514

117 02 ALBERTINA ISABEL 191221

099 02 GUILLERMO SEGUNDO 222604

211 01 LUISA BERTA 312516

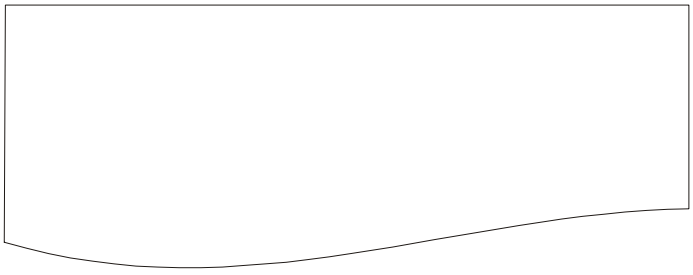
121 01 GRACIELA DEL CARMEN 253423

114 01 ELIAS JOSE 180000

COD	CODS	NOM	SAL	CODSA	TOTSUC
~~114~~	~~01~~	~~Elias Jose~~	~~180000~~	~~01~~	~~0~~
~~121~~	~~01~~	~~Graciela del Carmen~~	~~253423~~		~~180000~~
~~211~~	~~01~~	~~Luisa Berta~~	~~312516~~		~~433423~~
~~099~~	~~02~~	~~Guillermo Segundo~~	~~222604~~	~~02~~	~~745939~~
~~117~~	~~02~~	~~Albertina Isabel~~	~~191321~~		~~0~~
~~212~~	~~02~~	~~Corina Francisca~~	~~823514~~		~~222604~~
~~119~~	~~03~~	~~Ruth De Jesús~~	~~119325~~	~~03~~	~~413925~~
~~118~~	~~03~~	~~Florencia Maria~~	~~425000~~		~~1239439~~
~~311~~	04	Jairo Antonio	388934	04	~~0~~
0					~~119325~~
					~~544325~~
					~~0~~
					388934

- **Salida**

-



Ejercicio resuelto Nº 25

Una empresa de ventas tiene la información de sus vendedores clasificada por sucursal y dentro de cada sucursal, por vendedor. Cada registro de un vendedor posee los siguientes datos: código de la sucursal, código del vendedor, número de la factura y valor de la venta.

Hacer un algoritmo que muestre la relación de ventas hechas por cada vendedor, dando subtotales por vendedor, por sucursal y el gran total de ventas de la empresa.

Análisis

Datos de entrada

- Código de la sucursal.
- Código del vendedor.
- Número de la factura.

- Valor de la venta.

Datos de salida
- Código de la sucursal.
- Código del vendedor.
- Número de la factura.
- Valor de la venta.
- Total de ventas por cada vendedor.
- Total de ventas por sucursal.
- Total de ventas de la empresa.

Proceso
Como no se conoce la cantidad de sucursales que tiene la empresa es necesario implementar un esquema cualitativo. Para el control del ciclo de entrada de datos se escoge el campo código de la sucursal y, el ciclo se repetirá hasta que se encuentre un valor de código de sucursal igual a cero.

Para dar totales por vendedor es necesario hacer un rompimiento de control de ejecución cada que, dentro de una sucursal, se cambie de un vendedor a otro. Para ello se escoge una variable auxiliar que almacene el código del vendedor anterior y se actualiza cuando entre información de un vendedor distinto. Como hay que dar totales por sucursal hay que hacer otro rompimiento de control cuando se cambie de una sucursal a otra. Esta clase de problema se denomina rompimiento de control de segundo nivel; pueden existir rompimientos de más niveles. La salida de información estará de acuerdo con el siguiente formato:

Continuación

VENTAS DEL MES SUCURSAL XX

VENDEDOR	NRO. FACTURA	VALOR VENTA
XXX XXX	XXX.xx	
XXX XXX	XXX.xx	
TOTAL VENDEDOR		
XXX XXX	XXX.xx	
TOTAL VENDEDOR		XXX
TOTAL SUCURSAL		XXX
TOTAL EMPRESA	XXX	

Definición de variables

COSUC:	Código de la sucursal
COSUCA:	Código de la sucursal anterior
COVEN:	Código del vendedor
COVENA:	Código del vendedor anterior
NF:	Número de la factura
VV:	Valor de la venta
TOMEN:	Total de ventas de un vendedor
TOTSUC:	Total de ventas de una sucursal
TOTE:	Total de ventas de la empresa

Algoritmo

```
INICIO
    TOTE = 0
    LEA: COSUC, COVEN, NF, VV
    MIENTRAS COSUC <> 0 HAGA
        COSUCA = COSUC
        TOTSUC = 0
        ESCRIBA: "VENTAS DEL MES SUCURSAL:", COSUC
        ESCRIBA: "VENDEDOR   NRO. FACTURA   VALOR VENTA"
        MIENTRAS (COSUC = COSUCA) ∧ (COSUC <> 0) HAGA
            COVENA = COVEN
            TOTVEN = 0
            MIENTRAS (COVEN = COVENA) ∧ (COSUC <> 0)                    ∧ (
                ESCRIBA: COVEN, NF,VV
                TOTVEN = TOTVEN + VV
                LEA: COSUC
                SI COSUC <> 0 ENTONCES
                    LEA: COVEN, NF, VV
                FIN_SI
            FIN_MIENTRAS
            ESCRIBA: "TOTAL VENDEDOR:$", TOTVEN
            TOTSUC = TOTSUC + TOTVEN
        FIN_MIENTRAS
        TOTE = TOTE + TOTSUC
        ESCRIBA: "TOTAL. SUCURSAL:$", TOTSUC
    FIN_MIENTRAS
    ESCRIBA: "TOTAL EMPRESA:$", TOTE
FIN_INICIO
```

Prueba de escritorio

0

02	17	19	300000.0
02	16	15	395.80
02	16	11	800.00
01	92	31	489.50
01	92	30	3000.00
01	89	25	586.00
01	89	23	2243.50

COSUC	COSUCA	COVEN	COVENA	NF	VV	TOTVEN	TOTSUC	TOTE
01	01	89	89	23	2243,50	0	0	0
01		89		25	586,00	2243,50		
01		92	92	30	3000,00	2829,50	2829,50	
01		92		31	489,50	0		

02 02 16 16 11 800,00 3000,00
02 16 15 395,80 3489,50 6319,00 6319,00
02 17 17 19 300000.00 0 0
0 800,00
1195,80 1195.80
0
300000.00 301195.80 307514.80

Salida

Ventas del mes sucursal 01

Vendedor	Nro. factura	Valor venta
89	23	2243.50
89	25	586.00
Total vendedor		2829.50
92	30	3000.00
92	31	489.50
Total vendedor		3489.50
Total sucursal		6319.00

Ventas del mes sucursal 02

Vendedor	Nro. factura	Valor venta
16	11	800.00
16	15	395.80
Total vendedor		1195.80
17	19	300000.00
Total vendedor		300000.00
Total sucursal		301195.80
Total empresa		307514.80

FALTA GRAFICO PAG. 123 LIBRO

Ejercicio resuelto Nº 26

Cada uno de cuatro atletas desea saber si ha bajado o subido de peso desde la última vez que se pesó. Para esto cada uno se pesa cinco veces, calcula el promedio de su peso actual y lo compara con el peso anterior. Hacer un algoritmo que determine para cada atleta si bajó, subió de peso o está estable y cuánto es el incremento en caso de que exista diferencia respecto al peso anterior.

Análisis

Datos de entrada

- Promedio de peso de la última vez que se pesó cada atleta.
- Cada uno de los valores de las cinco pesadas.
- Diferencia de pesos.

Datos de salida

- Un mensaje que diga si bajó, subió o está estable.
- El incremento de peso en caso de que exista diferencia entre los dos pesos.

Proceso

Se usará un esquema cuantitativo ya que se conoce el número de atletas. Por cada uno de ellos se entrarán cinco valores de pesos obtenidos, para calcular el promedio de peso actual.

PROMEDIO DE PESO = (PESO1 + PESO2 +...+PESO5) / 5

La diferencia de pesos se obtiene restando el peso anterior del promedio de peso actual. Al comparar esta diferencia con cero se puede establecer si bajó, subió de peso o está estable.

Definición de variables

PROAN: Promedio de peso anterior.
PESO: Cada uno de los 5 pesos.
PROAC: Promedio de peso actual.
DP: Diferencia de pesos.
CONA: Contador de atletas (entre 1 y 5).
CONP: Contador de pesos (entre 1 y 5).
SUMAP: Suma de los 5 pesos de cada atleta.

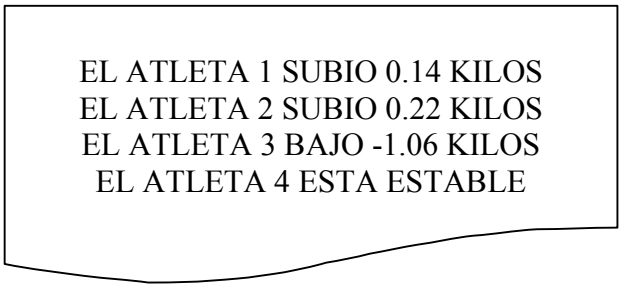
Algoritmo

```
INICIO
    CONA = 1
    MIENTRAS CONA ≤ 4 HAGA
            LEA: PROAN
            CONP, SUMAP = 0
            MIENTRAS CONP < 5 HAGA
                        LEA: PESO
                        SUMAP = SUMAP + PESO
                        CONP = CONP + 1
            FIN_MIENTRAS
            PROAC = SUMAP / 5
            DP = PROAC - PROAN
            SI DP > 0 ENTONCES
                    ESCRIBA: "EL ATLETA", CONA, "SUBIO", DP,
              SINO
                    SI DP = 0 ENTONCES
                            ESCRIBA: "EL ATLETA", CONA, "ESTA
                      SINO
                            DP =  DP * -1
                            ESCRIBA: "EL ATLETA", CONA, "BAJO", DP,
                    FIN_SI
            FIN_SI
            CONA = CONA + 1
    FIN_MIENTRAS
FIN_INICIO
```

Prueba de escritorio

75,0	74,9	75,1	74,8	75,2	75,0
80,5	79,0	79,5	80,,0	79,8	78,9
70,0	71,0	70,5	70,6	70,0	69,0
67,34	67,0	67,3	68,0	68,1	67,0

Salida



Ejercicio resuelto Nº 27

Elaborar un algoritmo que encuentre el factorial de un número positivo.

Análisis

Datos de entrada

- El número a calcularle al factorial.

Datos de salida

- El factorial del número.

Proceso

El factorial de un número N se obtiene como:

N! = 1 * 2 * 3 * 4 *... * (N - 1) * N

Por definición: 1! = 1 y 0! = 1

Para calcular la productoria es necesario generar, a través de un contador, los números entre 1 y N e ir acumulando cada producto en una variable

Continuación

inicializada en uno (1), por ser éste el módulo de la multiplicación. El proceso de acumulación se hace cada vez que se genera un nuevo número.

Definición de variables

N:	Número a calcularle el factorial
CON:	Contador y generador de números entre 1 y N
FAC:	Factorial del número N

Algoritmo

```
INICIO
    LEA: N
    CON, FAC = 1
    MIENTRAS CON ≤ N HAGA
            FAC = FAC * CON
            CON = CON + 1
    FIN_MIENTRAS
    ESCRIBA: "EL FACTORIAL DE", N, " ES: ", FAC
FIN_INICIO
```

Prueba de escritorio

Valor de N: 5

N	CON	FAC

5	1	1
	2	1
	3	2
	4	6
	5	24
	6	120

Salida

Ejercicio resuelto Nº 28

Elaborar un algoritmo que, dado un valor de X, determine el valor de F(X) definida por la siguiente serie:

Teniendo en cuenta los términos cuyo valor absoluto sean mayores o iguales a 10-3.

Análisis

Datos de entrada

- El valor de X.

Datos de salida

- El valor de F(X).

Proceso

La serie puede ser expresada como:

En este caso el valor del exponente coincide con el número factorial del denominador; para generar este valor se hará uso de un contador que parta de cero y se incremente de uno en uno, hasta llegar a N.

El término a generar será:

Término = (X * * contador) / (contador!)

El signo de cada término se obtiene como:

Signo = ((-1) * * contador)

El ciclo será controlado por el valor del término, ya que aquellos valores de los términos menores que 10-3 no se tendrán en cuenta.

El factorial será calculado multiplicando el factorial anterior por el contador cada que éste se incrementa en uno.

Definición de variables

I:	Contador; generador de exponente y número factorial.
FX:	Acumulador de la suma de los términos de la serie.
FAC:	Factorial del contador.
TER:	Término de la serie generado.

Algoritmo

```
INICIO
    LEA: X
    FX, I = 0
    TER, FAC = 1
    MIENTRAS [[TER]] >= 10 ** (-3) HAGA
        FX = FX + TER
        I = I + 1
        FAC = FAC * I
        TER = X ** I / FAC*(-1) ** I
    FIN_MIENTRAS
    ESCRIBA: "EL VALOR DE F(X) ES: ", FX
FIN_INICIO
```

Prueba de escritorio

X	I	FX	TER	FAC
2	0	0	1	1
	1	1	- 2	1
	2	- 1	2	2
	3	1	- 1.3733	6
	4	- 0.3333	0.5667	24
	5	0.3333	- 0.2467	120
	6	0.0667	0.0989	720
	7	0.1556	- 0.0254	5040
	8	0.1302	0.0063	40320
	9	0.1365	-0.0014	362880
	10	0.1351	0.0003	3628800

Salida

Ejercicio resuelto Nº 29

En una elección realizada en Piedra Dura se presentaron tres candidatos: Pedro, Pablo y Vilma. Por cada votante se tiene la siguiente información:

Piedracédula: Número de la cédula del votante.

Opción: Campo que indica el candidato escogido, si el valor del campo es:

1: votó por Pedro
2: votó por Pablo
3: votó por Vilma
4: votó en blanco

Elaborar un algoritmo que muestre el número de votos depositados por cada candidato y el total de votantes.

Análisis

Datos de entrada

- Piedracédula de cada votante
- Opción que indica el candidato elegido

Datos de salida
- Número de votos por Pedro
- Número de votos por Pablo
- Número de votos por Vilma
- Número de votos en blanco
- Número total de votos

Proceso
Como no se conoce el total de votantes se debe implementar un esquema cualitativo; el campo escogido dentro del registro será Piedracédula, el cual dentro de sus posibles valores no tiene uno que sea cero o negativo. El ciclo de entrada de datos terminará cuando encuentre un valor de cero para el campo Piedracédula.

El campo opción puede tomar uno cualquiera de los valores entre 1 y 4, pero sólo uno de estos valores en cualquier momento. Si Fred votó por Vilma y su cédula es 588 su registro sería:

588 3

Esta clase de campos es muy utilizada dentro de la programación, fundamentalmente cuando se trata de problemas de tipo estadístico.

Continuación

Definición de variables

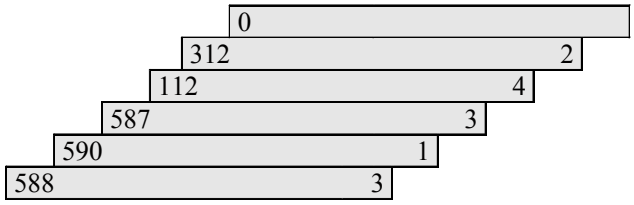
PC:	Número de piedracédula.
OP:	Opción elegida.
NVPE:	Contador del número de votos por Pedro.
NVPA:	Contador del número de votos por Pablo.
NVVI:	Contador del número de votos por Vilma.
NVBL:	Contador del número de votos en blanco.
NTV:	Número total de votos.

Algoritmo

```
INICIO
    NVPE, NVPA, NVVI, NVBL = 0
    LEA: PC
    MIENTRAS PC > 0 HAGA
            LEA: OP
            SI OP = 1 ENTONCES
                    NVPE = NVPE + 1
              SINO
                    SI OP = 2 ENTONCES
                            NVPA = NVPA + 1
                      SINO
                            SI OP = 3 ENTONCES
                                    NVVI = NVVI + 1
                              SINO
                                    NVBL = NVBL + 1
                            FIN_SI
                    FIN_SI
            FIN_SI
            LEA:PC
    FIN_MIENTRAS
```

```
    NTV = NVPE + NVPA + NVVI + NVBL
    ESCRIBA: "VOTOS TOTALES", NTV, "REPARTIDOS ASI:
FIN_INICIO
```

Prueba de escritorio



PC	OP	NVPA	NVPE	NVVI	NVBL	NTV
~~588~~	~~3~~	~~0~~	~~0~~	~~0~~	~~0~~	
~~590~~	~~1~~	1		~~1~~		
~~587~~	~~3~~		1	2	1	5
~~112~~	~~4~~					
312	2					

Salida

```
VOTOS TOTALES: 5
REPARTIDOS ASI:
PABLO 1   PEDRO 1
VILMA 2   EN BLANCO 1
```

Ejercicio resuelto No. 30

Una industria de sobres de tarjetas de navidad dispone en la actualidad de 500 sobres, distribuidos así: 300 sobres de 15 cm de largo por 10 cm de ancho y 200 sobres de 20 cm de largo por 15 cm de ancho

Para que sea posible introducir una tarjeta en un sobre se requiere que el largo y ancho de la tarjeta sean menores o iguales al largo y ancho del sobre. La industria requiere un algoritmo que imprima el número de sobres que necesitará y el número de sobres que tendrá que producir para cubrir la demanda de tarjetas. Por cada tarjeta pedida se suministra el largo y el ancho de la misma. La industria sólo produce sobres cuando no tiene existencias de las dimensiones pedidas.

Análisis

Datos de entrada

- Largo del sobre pedido.
- Ancho del sobre pedido.

Continuación

Datos de salida

- Cantidad de sobres pedidos.
- Cantidad de sobres a fabricar.

Proceso

No se conoce el número de sobres que se pedirán; por lo tanto, debe implementarse un esquema cualitativo.

Por cada sobre pedido se debe examinar si se tiene o no en las existencias.

Si largo $\leq$ 15 y ancho $\leq$ 10 disminuye en 1 la cantidad de sobres de 15 * 10 en caso de tener existencia, es decir, si es > 0. Si no hay existencias de esta clase de sobres se debe preguntar si existen sobres de 20 * 15. Si los hay se disminuye en uno esta clase de existencias, y si no, se incrementa en uno el número de sobres a fabricar.

No si el largo $\leq$ 20 y ancho $\leq$ 15, se disminuye en 1 la cantidad de existencias de esta clase de sobres si los hay y, si no, se debe incrementar en 1 la cantidad de sobres a fabricar.

Definición de variables

LA:	Largo del sobre pedido.
AN:	Ancho del sobre pedido.
NSP:	Cantidad de sobres pedidos.
NSAF:	Número de sobres a fabricar.
SD1510:	Existencias de sobres de 15 * 10.
SD2015:	Existencias de sobres de 20 * 15.

Algoritmo

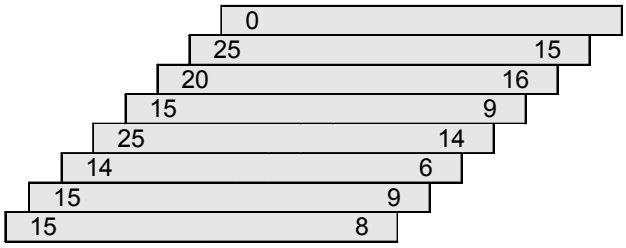
```
INICIO
    NSP, NSAF = 0
    SD1510 = 300
    SD2015 = 200
    LEA: LA
    MIENTRAS LA > 0 HAGA
        LEA: AN
        NSP = NSP + 1
        SI (LA <= 15) ∧ (AN <= 10) ENTONCES
            SI SD1510 > 0 ENTONCES
                SD1510 = SD1510 -1
              SINO
                SI SD2015 > 0 ENTONCES
                    SD2015 = SD2015 -1
                  SINO
                    NSAF = NSAF + 1
                FIN_SI
            FIN_SI
          SINO
            SI (LA <= 20 ) ∧ (AN <= 15) ENTONCES
                SI SD2015 > 0 ENTONCES
                    SD2015 = SAD2015 - 1
                  SINO
                    NSAF = NSAF - 1
                FIN_SI
              SINO
                NSAF = NSAF + 1
            FIN_SI
        FIN_SI
        LEA: LA
    FIN_MIENTRAS
    ESCRIBA: "SOBRES PEDIDOS:", NSP, "SOBRES A FABRICAR", NSF
```

FIN_INICIO

Prueba de escritorio

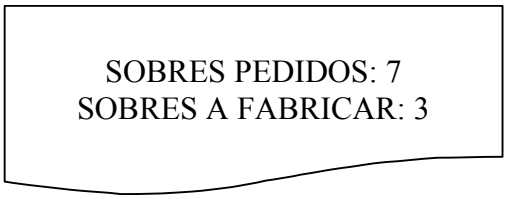
Vamos a asumir que las existencias de sobres son de 3 y 2, respectivamente.

Archivo de datos



LA	AN	SD1510	SD2015	NSAF	NSP
~~15~~	~~8~~	~~3~~	~~2~~	~~0~~	~~0~~
~~13~~	~~9~~	~~2~~	1	~~1~~	~~1~~
~~14~~	~~6~~	~~1~~		~~2~~	~~2~~
~~25~~	~~14~~	0		3	~~3~~
~~15~~	~~9~~				~~4~~
~~30~~	~~16~~				~~5~~
25	15				~~6~~
					7

Salida



Aspectos a tener en cuenta:

a. El esquema cuantitativo se implementa a través de un contador, la expresión lógica que gobierna el ciclo se establece de acuerdo al valor inicial que se le asigne a éste.
b. Los contadores y acumuladores deben partir de un valor inicial.
c. Una vez que se entra a un ciclo, éste debe ejecutar todo el conjunto de instrucciones que lo componen.
d. En la implementación de cualquiera de los dos esquemas, las variables que están involucradas en la expresión lógica deben tener almacenado un valor.
e. Cuando existen cálculos totales, algunos se hacen dentro del ciclo y otros pueden hacerse fuera del ciclo.
f. El uso de una bandera exige que ésta tenga asignado un valor en el momento de evaluarse.
g. Si un ciclo ya cumplió su misión, para que este termine antes de que la condición en forma normal sea falsa, es necesario alterar la expresión lógica.

5.11. Ejercicios propuestos

38. Hacer un algoritmo que encuentre la suma de los números impares comprendidos entre 1 y N.

39. Elaborar un algoritmo que encuentre la desviación típica de un grupo de datos positivos.
40. Encontrar la raíz cuadrada, el cuadrado y el cubo de un grupo de números enteros positivos.
41. Elaborar un algoritmo que encuentre el mayor valor entre un grupo de datos positivos.
42. Hacer un algoritmo que encuentre la suma de los valores mayor y menor entre un grupo de datos positivos.
43. Hacer un algoritmo que determine la cantidad de tríos de valores, entre un conjunto de ternas que representen triángulos rectángulos.
44. En cada uno de una serie de registros se encuentran tres valores que posiblemente representan los tres lados de un triángulo. Hacer un algoritmo que determine cuántos triángulos equiláteros, isósceles y escalenos hay.
45. Hacer un algoritmo que encuentre la cantidad de valores enteros que hay entre un par de números positivos.
46. Realizar un algoritmo que escriba la posición de la cantidad de puntos que estén ubicados en el primer cuadrante de una circunferencia con centro (0,0).
47. Varias ambulancias recorren la ciudad y cuando se recibe en la CENTRAL una llamada se informa la ubicación de la emergencia mediante coordenadas, lo mismo que la ubicación de todas las ambulancias.

    La central es el punto (0,0) u origen de las coordenadas.

    Se sabe que existen N ambulancias en servicio.

    Realice un algoritmo que, dada la información necesaria, informe las coordenadas de la ambulancia más cercana al punto de emergencia.

48. Dados N valores, diseñe un algoritmo que haga el siguiente proceso:
    • Si el valor es menor que cero calcular su cubo.
    • Si el valor está entre 0 y 100 calcular su cuadrado.
    • Si el valor está entre 101 y 1000 calcular su raíz cuadrada.
49. Elaborar un algoritmo que determine cuántos son los intereses generados en cada uno de los N períodos, por un capital de X pesos, que se invierte a una cantidad de P por ciento, sin retirar intereses.
50. Elaborar un algoritmo que encuentre el factorial de los números comprendidos entre 1 y N.
51. Hacer un algoritmo que entre dos valores A y B y encuentre A^B mediante sumas únicamente.
52. Elaborar un algoritmo que muestre los enteros desde 1 hasta N y sus cuadrados, calculados solamente con sumas y utilizando el método propuesto.

Número	Cuadrado	Método
1	1	1
2	4	1+3
3	9	1+3+5
4	16	1+3+5+7
5	25	1+3+5+7+9
.	.	.
.	.	.
.	.	.
N		

53. Partiendo de la función Y = 3X, elaborar un algoritmo que calcule todos los posibles valores de Y, sabiendo que X puede tomar el valor de cualquier múltiplo entero de 7 en el rango [0,1000].
54. Elaborar un algoritmo que, dado un valor de X, obtenga el valor de E mediante la suma de la serie:

$$E = 1 + X + \frac{X^2}{2!} + \frac{X^3}{3!} + \frac{X^4}{4!}$$

    Si X es mayor o igual que 0, incluyendo los términos que sean mayores que 10^{-6}.

Si X es menor que cero $E = \frac{1}{E^{|x|}}$

55. Elaborar un algoritmo que calcule para X = 0. 1, 0.2,… , 0.9, el valor de la serie
$\frac{X}{2} + \frac{2X^2}{3} + \frac{3X^3}{4} + \frac{4X^4}{5} + \cdots$

    Para cada valor de X la suma se calcula hasta que el término generado sea menor o igual a un valor T dado.

56. Elaborar un algoritmo que, dado un valor de X, determine la suma de los N primeros términos de la siguiente serie:
$\frac{X}{2!} + \frac{2X^2}{4!} + \frac{3X^3}{6!} + \frac{4X^4}{8!} + \frac{5X^5}{16!} + \cdots$

57. Hacer un algoritmo que lea un valor de X y calcule el valor de los primeros N términos de la serie:
$1 - \frac{X^3}{2*1} + \frac{X^5}{3*4} + \frac{X^7}{5*6} + \frac{X^9}{7*8} - \cdots$

58. Elaborar un algoritmo que encuentre la suma de los primeros N términos de la sucesión de FIBONACCI.
    Ésta se genera empezando por los números 0 y 1 y calculando cada nuevo miembro como la suma de los dos miembros precedentes de la sucesión, así:
    0 1 1 2 3 5 8 13 ...

59. Se tiene una serie de registro donde cada uno de ellos contiene 4 valores, que corresponden a las longitudes de diferentes patas de una mesa.

    Elaborar un algoritmo que chequee si con los cuatro valores leídos en cada registro se puede fabricar una mesa no coja, de cuatro patas o, una mesa no coja de tres patas en caso de que no se pueda fabricar una de cuatro. El algoritmo debe mostrar el número de mesas de cuatro y tres patas que se pueden formar, lo mismo que la cantidad de patas sobrantes.

60. Una empresa tiene un número variable de empleados y de cada uno de ellos posee la siguiente información:

    Código.
    Nombres.
    Número de hijos.
    Salario por hora.
    Número de horas trabajadas al mes.

    La retención por cada empleado se determina así:

    Para salarios menores de $300.000: si el número de hijos es mayor de 6 no hay retención; si el número de hijos es menor o igual a 6, se le retiene un porcentaje igual a 6 menos el número de hijos dividido por 2.

    Para salarios iguales o mayores a $300.000: si el número de hijos es menor de 3, se le retiene un 3%; si el número de hijos es mayor o igual a 3 se le retiene un porcentaje igual a 10 dividido por el número de hijos.
    El subsidio por cada hijo es de $1.200.

    Elaborar un algoritmo que muestre: código, nombres, devengado, retención, subsidio y total a pagar.

61. Un vendedor desea calcular la comisión total sobre la venta de varios artículos. Al vendedor le corresponde el 10% de comisión sobre artículos, cuyo precio es menor o igual a $1.000 y el 7% de comisión sobre aquellos artículos cuyo precio sea mayor de $1.000.

Elabore un algoritmo para obtener la comisión total, si se conoce que el vendedor hizo N ventas.

62. Se desea obtener el promedio de N grupos que están en un mismo año escolar, sabiendo que cada grupo puede tener M alumnos, cada alumno puede cursar K materias y en todas las asignaturas se promedian tres calificaciones. Hacer un algoritmo que calcule el promedio de cada alumno, el promedio de cada grupo y el promedio de los grupos.
63. Una oficina de seguros ha reunido datos concernientes a todos los accidentes de tránsito ocurridos en el área metropolitana de Medellín en el último año. Por cada conductor involucrado en un accidente se toman los siguientes datos: año de nacimiento, sexo (1: Femenino, 2: Masculino), registro del carro (1: Medellín, 2: Otras ciudades). Hacer un algoritmo que muestre:
    - El porcentaje de conductores menores de 25 años.
    - Porcentaje de conductores del sexo femenino.
    - Porcentaje de conductores masculinos con edades entre 12 y 30 años.
    - Porcentaje de conductores cuyos carros están registrados fuera de Medellín.
64. Una empresa extranjera de aviación fumiga cosechas contra una gran variedad de plagas. Los valores cobrados a los granjeros dependen de lo que éste desea fumigar y de cuántas hectáreas se fumigan, de acuerdo a la siguiente distribución:
    Tipo 1: fumigación contra malas hierbas, 10 dólares por hectárea.
    Tipo 2.- fumigación contra langostas, 15 dólares por hectárea.
    Tipo 3: fumigación contra gusanos, 20 dólares por hectárea.
    Tipo 4: fumigación contra todo lo anterior, 30 dólares por hectárea.

    Si el área a fumigar es mayor de 1.000 hectáreas, el granjero goza de un 5% de descuento. Además, cualquier granjero cuya cuenta sobrepase los 3.000 dólares se le descuenta un 10% sobre la cantidad que exceda dicho precio. Si se aplican ambos conceptos, el correspondiente a la superficie se considera primero. Por cada pedido se tiene la siguiente información: nombre del granjero, tipo de fumigación solicitada (1, 2, 3,4) y el número de hectáreas a fumigar. Por cada solicitud se debe suministrar: nombre del granjero y valor a pagar.
65. Se desea invertir una cantidad de dinero de manera que llegue a ser X pesos en Y años. Si la tasa actual de interés es de R%, entonces la cantidad a invertir (el valor presente de X) está dado por la siguiente fórmula:

$$\frac{X}{(1 + 0.01 * R)}$$

66. Hacer un algoritmo que obtenga el promedio de tiempo por semana, por mes y por año de un atleta que recorre la misma ruta durante 5 días a la semana.
67. A lo largo de un día un cajero procesa a las personas que llegan a efectuar movimientos bancarios. Estos movimientos son esencialmente consignaciones y retiros. Determinar la cantidad total de dinero obtenido por concepto de consignaciones y retiros en todo el día, lo mismo que un balance que indique si hubo más retiros que consignaciones y la diferencia absoluta.
68. Hacer el control de un tablero que registra los puntos de set de un partido de tenis de 5 sets sin muerte súbita, es decir, que gana el set aquél que llegue a 6 puntos o más, con dos puntos de ventaja sobre el adversario. Debe mostrarse un mensaje con el nombre del jugador que gana cada set y el nombre del ganador del partido.
69. En un curso se practican 4 evaluaciones con los siguientes porcentajes: 25%, 20%, 25% y 30%, respectivamente. Por cada estudiante se informa código y las cuatro notas. Hacer un algoritmo que calcule la nota definitiva de cada estudiante, el promedio de notas definitivas del curso y el porcentaje de perdedores.
70. Hacer un algoritmo que calcule el valor presente de P pesos al 1 % de interés, para períodos de uno a veinte años en pasos de dos años (ver fórmula ejercicio de renglón 63).

71. La criba de Eratóstenes es una técnica para generar números primos. Se comienza escribiendo todos los enteros impares desde 3 hasta N; luego se elimina cada tercer elemento después de 3, cada quinto elemento después de 5, etc., hasta que los múltiplos de todos los enteros impares menores que $\sqrt{N}$ hayan sido eliminados. Los enteros que quedan constituyen la lista exacta de los números primos entre 3 y N. Diseñar un algoritmo que genere los números primos entre 1 y 1.000, utilizando la técnica de la criba.
72. Suponga que una tienda en particular vende todas sus mercancías a un precio de $1000 o menos. Suponga, además, que todos los compradores pagan con una moneda de $1000 todas sus compras. Diseñe un algoritmo para leer los precios de los artículos vendidos y para calcular el número de monedas que debe dar a cambio, de tal manera, que devuelva un mínimo de moneda fraccionaria. Por ejemplo, si el precio de la venta es de $450, el cambio debe ser una moneda de $500 y una moneda de $50.

73. Escriba un algoritmo para leer e imprimir un texto de entrada y determinar el número de oraciones y palabras involucradas. Asuma que:
    a. Cada oración termina con un punto, y que el carácter punto no es utilizado para ningún otro propósito.
    b. Las palabras están conformadas por letras y separadas por blanco.

74. Cierta sucesión parte de los números 0, 1, 1; de ahí en adelante los nuevos términos se forman mediante la suma de los tres términos inmediatamente anteriores, así:

    0,1,1,2,4,7,13,24 ....

    Hacer un algoritmo que genere y muestre los primeros N términos de la sucesión.
75. Elaborar un algoritmo que entre la hora de un día en horas, minutos y segundos; se desea obtener la hora a los 30 segundos siguientes.
76. Dado un mes del año y si el año es o no bisiesto, hacer un algoritmo que calcule los días del mes.
77. Un distribuidor de juguetes ha hecho una excelente compra de 10.000 juguetes en cajas rectangulares de diversos tamaños. El distribuidor desea poner las cajas en esferas plásticas de brillantes colores y sellarlas como paquetes de sorpresa. Las esferas son de cuatro diámetros diferentes: 4, 6, 8 y 10 pulgadas; por lo que para realizar todo en orden desea saber cuántas esferas de cada diámetro debe comprar; la diagonal de una caja rectangular, cuyas dimensiones son A, B y C, está dada por y es, además, la dimensión mayor. El distribuidor debe calcular las longitudes de las diagonales de las cajas y determinar el número de las que son de 4 pulgadas o menos, las comprendidas entre 4 y 6 pulgadas, etc. Las dimensiones de cada caja están en un registro. Diseñar un algoritmo que determine el número de esferas de cada tamaño que se necesitan para empacar los juguetes.

78. Un restaurante paga a sus meseros dos clases de comisiones:
    - Una comisión del 7% sobre toda venta.
    - Otra comisión que depende del tipo de venta: 15% si la venta es de contado, 10% si la venta se hizo en cheque y, 5% si se hizo con tarjeta de crédito.

    El restaurante tiene por cada venta:

    Identificación del vendedor (1, 2, 3)

    Tipos de ventas (1: contado; 2: cheque; 3: tarjeta)

    Cuantía de la venta.

    Elaborar un algoritmo que obtenga el total a pagar a cada uno de los empleados.

79. Se tiene la siguiente información por cada uno de los N estudiantes de la universidad:

Edad
Sexo (1: masculino; 2: femenino)
Carrera (1: ingeniería; 2: otra carrera)

Hacer un algoritmo que obtenga:
- Promedio de edad de los estudiantes de Ingeniería.
- Porcentaje de hombres en la universidad.
- Porcentaje de mujeres que estudian Ingeniería.

80. Un almacén tiene 4 departamentos numerados consecutivamente del 1 al 4, los cuales venden artículos de diferente naturaleza. Por cada artículo se tiene la siguiente información:
- Código del artículo.
- Código departamento.
- Cantidad vendida.
- Precio de venta unitario.
- Precio de costo unitario.
- Indicativo que dice si el producto es importado o colombiano (1: Colombia- no;2: Importado).

Hacer un algoritmo que determine, por cada departamento, por cuántos y por cuáles productos importados se obtuvo una utilidad superior a $1000.000,oo y la utilidad total por departamento.

81. Se desea organizar una competencia de motociclismo, para la que se han establecido 5 categorías:
Primera: cilindrajes de motos hasta 100cc.
Segunda: cilindrajes de motos de 101cc hasta 250cc.
Tercera: cilindrajes de motos de 251cc hasta 350cc.
Cuarta: cilindrajes de motos de 351cc hasta 500cc.
Quinta: cilindrajes de motos superiores a 500cc.

Por cada participante se tiene un registro con los siguientes datos:
- Identificación.
- Nombres.
- Categoría en la que se inscribió.
- Cilindraje de la moto.

Elaborar un algoritmo que:
a. Compruebe la validez de la categoría dada, respecto al cilindraje por cada participante.
b. Si el participante cumple el requisito anterior debe mostrar su identificación y su nombre.
c. Muestre la cantidad de participantes aceptados en cada categoría.
d. Muestre el total de participantes aceptados.

82. Elaborar una algoritmo que haga el siguiente censo para una empresa de transporte.
a. Número de vehículos cuyo modelo sea anterior a 1995.
b. Número de vehículos cuyo modelo sea de 1995 o posterior y cuya capacidad sea menor de 35 pasajeros.
c. Número de buses cuyo modelo sea posterior a 1995 con capacidad mayor de 35 pasajeros.
d. Número de busetas con capacidad menor de 35 pasajeros.
e. El total de vehículos de la empresa.

Por cada vehículo la empresa tiene la siguiente información:
- Tipo de vehículo (1: buseta; 2: bus)
- Modelo del vehículo.
- Capacidad del vehículo.

83. En una encuesta realizada entre los posibles electores se obtienen los siguientes tipos de respuestas:
    1. Abstencionista.
    2. Los de la pomada.
    3. Los belicosos.
    4. Los capos.
    5. Los fascistas.
    6. La izquierda goda.
    7. No sabe por quien votará.

    Cada respuesta se encuentra en un registro de los cuales no se sabe cuántos hay. Hacer un algoritmo que calcule el porcentaje que representa cada respuesta, respecto al número total de encuestados.

84. Un banco tiene el problema de calcular los intereses que a fin de año habrá de acreditar a cada uno de los clientes. Por cada cliente tiene la siguiente información:
    - Cédula
    - Capital
    - El día del calendario (año de 360 días) en que el capital fue ingresado al banco.
    - Tasa de interés a aplicar:

    Interés = Ki / 100 + (360 - M)/360

    Donde:
    K es el capital
    M es el día
    i es la tasa de interés.

    También desea calcular el número de clientes, el total del capital depositado y el total de intereses.

85. Una gasolinera presta 4 clases de servicios. Por cada servicio que preste se tienen los siguientes datos: clase de servicio prestado (valores del 1 al 4), jornada en la que se prestó el servicio (M: mañana, T: tarde) y valor del servicio. Al final del día se requiere determinar el valor producido por cada clase de servicio, el número de veces que se prestó cada servicio, el servicio que más veces se prestó y si éste se prestó más en la mañana o en la tarde.

86. Se tiene un conjunto de registros donde cada uno de ellos contiene los siguientes campos:
    - Sexo (1: masculino; 2: femenino).
    - Año de nacimiento.
    - Estado civil (S: soltero; C: casado).

    Se desea saber cuántas personas pueden votar en cada una de las dos próximas elecciones que se realizan cada dos años, siempre en los años pares y en el mes de marzo. Sólo pueden votar quienes tengan 18 años cumplidos el 31 de diciembre del año anterior. La información pedida se debe discriminar así:
    - El número de hombres solteros.
    - EI número de hombres casados.
    - El número de mujeres solteras.
    - El número de mujeres casadas.
    - El total de personas solteras y casadas.
    - El total de personas que pueden votar.

87. En el supermercado BARATA, cada semana el gerente solicita al jefe de cada una de las secciones la siguiente información: código del artículo, existencia, demanda y costo de fábrica del artículo. La demanda se informa tabulada (1:buena demanda, 2:poca demanda). El gerente desea un informe por artículo de cada sección, como sigue:

    Sección, código del artículo, número de artículos a pedir, costo total por artículo y por sección y el gran total de los pedidos por las secciones.

    Para determinar el número de artículos a pedir se tiene el siguiente criterio: si el artículo tiene poca demanda o una existencia menor de 5 unidades, se piden 20 unidades, de lo contrario no se pide. Si el artículo tiene buena demanda y existencia mayor de 50 unidades, se piden 100, de lo contrario 200.

88. Hacer un algoritmo que determine si lo números N1, N2, N3, N4, N5 están ordenados ascendentemente.
89. Calcular y escribir las tablas de multiplicar del 13 al 26 en los siguientes rangos: 25 al 35, 45 al 50 y 60 al 85 lo mismo que el promedio de cada rango.
90. Hacer un algoritmo que obtenga la suma de los múltiplos de M entre varios M y N leídos (M < N, y ambos números enteros).
91. Se tienen N registros donde cada uno contiene:
    - Edad.
    - Tipo (1: alto, 2: bajo, 3: medio).
    - Nacionalidad (1: latino, 2: germano, 3: indú).

    Hacer un algoritmo que obtenga:
    - Número de indúes altos.
    - Total de latinos.
    - Germanos bajos menores de 30 años.
    - Total de indúes medianos.
    - Total de germanos.
    - Latinos bajos menores de 40 años.

92. Por cada uno de N estudiantes se tiene la nota obtenida en una materia. Hacer un algoritmo que obtenga: la nota máxima, la mínima, cuántos perdieron y cuántos ganaron la asignatura.
93. Se tiene un grupo de registros con las siguientes características: K registros tienen un valor R. M registros tienen un valor P. Los valores de M y K se indican por medio del primer registro (registro identificador).

    Elaborar un algoritmo para encontrar el valor del siguiente cálculo:
    CALCULO = (S(K, R) - S(M,P))/(K + M)

    Donde:
    S (K, R) es la suma de los K valores R.
    S (M, P) es la suma de los M valores P.

94. Hacer un algoritmo que lea el número total de experimentos de física que entrarán seguidamente.

    Por cada experimento se tiene:
    - Velocidad inicial VI
    - Velocidad final VIF
    - Distancia recorrida por un móvil D.

    $$\text{Aceleración} = \frac{VF**2 - VI**2}{2 * D}$$

    Determinar:

- Número de móviles acelerados (aceleración positiva).
- Número de móviles desacelerados (aceleración negativa).
- Número de móviles sin aceleración.
- Valor promedio de la aceleración.

95. Encontrar la cantidad de números primos que hay entre N y M (N < M)
    Si el kilovatio hora de energía es a $1000 y el metro cúbico de agua a $2000, hacer un algoritmo que determine:
    - Total a pagar por agua y energía por cada usuario.
    - Total recaudado por agua.
    - Total recaudado por energía y número total de usuarios en cada estrato.

96. Las Empresas Públicas desean crear un algoritmo que calcule los pagos que deben efectuarse por concepto de servicios; para ello cuenta con la siguiente información por usuario:
    - El código del usuario.
    - Avalúo de la propiedad.
    - Lectura anterior de agua.
    - Lectura actual de agua.
    - Lectura anterior de energía.
    - Lectura actual de energía.
    - Lectura anterior de teléfono.
    - Lectura actual de teléfono.
    - Otras entidades.

    Otras entidades tiene un valor de 1 si hay cobro para éstas y 2 si no hay cobro. En caso de que sea 1 vendrá a continuación otro registro que contiene:
    - El código del usuario.
    - Clave teléfonica.
    - Pago teléfonica.
    - Clave reparaciones.
    - Pago reparaciones.

    Clave teléfonica y reparaciones tiene el valor de 1 si existe el concepto; si no, tiene el valor de 2.

    Además posee los siguientes datos adicionales.
    - Tasa municipal de aseo: 1% del avalúo.
    - Agua: consumo libre primeros 20 metros cúbicos y a $2000 metro cúbico excedente.
    - El energía: $1000 el kilovatio/hora
    - Teléfono: $6 el impulso.

    Los consumos se obtienen de la diferencia de lecturas.

    El algoritmo debe producir por usuario: código; pagos por agua, luz, teléfono; tasa municipal de aseo; otras entidades; y el total a pagar.

97. Se tiene la siguiente información por cada una de las materias que cursa cada estudiante:
    - Código del estudiante.
    - Código de la materia.
    - Nota definitiva.

    No se conoce el número de estudiantes ni el número de materias cursadas por cada estudiante.

    Elaborar una algoritmo que muestre:

- Código del estudiante.
- Número de materias cursadas.
- Nota mayor y en qué materia la obtuvo.

98. La empresa de aviación PAJAROS VOLADORES posee 3 aviones con capacidad de 100 pasajeros cada uno.
    La empresa tiene asignadas 3 rutas, las cuales vuelan una vez al día entre:
    - Medellín - Apartadó, ruta 1
    - Medellín - Caucasia, ruta 2
    - Medellín - Quibdó, ruta 3

    La empresa tiene como política cancelar el vuelo cuando:

a. El número de pasajeros reservados sea inferior al 20% del cupo del avión.

b. Cuando el número de reservaciones para éste es menor que el excedente del cupo normal para otra ruta, en este caso se harán los dos vuelos para una misma ruta.

    Elaborar un algoritmo que calcule:
    - Número de pasajeros por vuelo.
    - Rutas a las cuales se cumplió el vuelo.
    - Número de vuelos cancelados.
    - En caso de cancelar vuelos qué rutas fueron.

    Por cada reservación se tiene un registro con el número de la ruta (1, 2 ó 3).

99. En el proceso de registros de materias a cursar, una universidad tiene la siguiente información por materia: código del estudiante, código de la materia, número de créditos. Las materias que cursará cada estudiante están reunidas por grupo. Hacer un algoritmo que muestre por estudiante: código del estudiante, código de la materia, número de créditos de la materia y total de créditos a cursar por el estudiante.
100. Una empresa tiene la siguiente información por cada uno de sus empleados: código empleado, código sección, código de departamento, código de la sucursal y salario del empleado.
    Hacer un algoritmo que muestre el código y el salario de cada empleado dando totales por sección, departamento y sucursal, y el total del salario a pagar por parte de la empresa.
101. Los registros que contienen la información de los empleados de una empresa se encuentran clasificados por sección, y a su vez, éstas por sucursal; el gerente de la empresa precisa de un listado de sueldos de los empleados, donde se especifique la información de cada uno por sección y los totales de sueldo por sección y por sucursal que están ubicados en los rangos: $450.000 a $600.000, $601.000 a $900.000 y mayores que $900.000. Diseñe el archivo de entrada.
102. Una compañía vende 5 productos diferentes. Para la venta de éstos emplea cierto número de vendedores, donde cada vendedor está encargado de la venta de cierto tipo de producto (pero un producto puede ser vendido por varios vendedores). La compañía necesita un listado de comisión por vendedor; para hacer esto tiene un registro por cada vendedor con los siguientes datos: código del vendedor, código del producto, cantidad de unidades vendidas, precio mínimo de venta por unidad y precio de venta por unidad (el precio mínimo de venta por unidad es el precio que se le fija al vendedor para vender el producto).

    Por cada vendedor se debe mostrar: código del vendedor, precio mínimo de venta total, precio de venta total, comisión (ésta se calcula de acuerdo al código del artículo de la siguiente manera):

Código del artículo	*Comisión*

0..999	15 % del precio de venta total. 40% de la diferencia del precio de venta total y el precio de venta mínimo total.
2000..2999	10 % del precio mínimo de venta total más el 50 % de la diferencia del precio de venta total, con respecto al precio mínimo de venta total.
3000..3999	10 % por unidad más el 5 % del precio mínimo de venta total.
4000..4999	$ 200 por unidad

Capítulo 6

ESTRUCTURAS ADICIONALES

Aunque cualquier programa apropiado se puede elaborar utilizando solamente las tres estructuras básicas de control descritas antes, el uso de éstas se convierte en soluciones particulares de la implementación de la estructura decisión lógica o mientras. Es necesario analizar en cuáles de estos casos particulares se puede utilizar la estructura adicional, teniendo en cuenta que lo que ellas hagan también se puede solucionar con una de las estructuras básicas.

6.1. Estructura caso o selección múltiple

Esta estructura permite seleccionar una, dentro de un conjunto de alternativas, con base en el valor almacenado en un campo variable denominado selector o campo controlador de la estructura. Es una estructura selectiva múltiple donde, de acuerdo con el valor que tenga el controlador, se realiza una determinada tarea una sola vez, es decir, no repite la ejecución de la tarea o secuencia. De acuerdo al valor que tenga el controlador el control de ejecución del programa pasa a uno de varios puntos de éste, evitando así una serie de preguntas (estructura decisión lógica o un anidamiento de las mismas).

Aunque el valor que puede almacenar el controlador puede, en algunos casos, pertenecer a distintos tipos de datos, lo más general es que los posibles valores sean números naturales del número uno en adelante.

Representación

```
CASOS DE <variable>
    CASO 1:
        <SECUENCIA 1>
    CASO 2:
        <SECUENCIA 2>
    .
    .
    CASO N:
        <SECUENCIA N>
    OTROS
    CASOS:      <SECUENCIA N+1>
FIN_CASOS
```

Ejemplo

```
CASOS DE K

    CASO 1:     K = K + 1
```

```
                ESCRIBA: K

    CASO 2:     K = K + 2
                ESCRIBA: K

    CASO 3:     K = K + 3
                ESCRIBA: K

    OTROS
    CASOS:      ESCRIBA K

FIN_CASOS
```

Funcionamiento

Al llegar a la estructura se evalúa el valor almacenado en la variable utilizada como controlador, para determinar cuál de las secuencias se efectúa. Una secuencia se ejecutará sí, y sólo sí, uno de los rótulos (casos) correspondientes coincide con el valor actual del controlador. Por tanto, el valor actual del controlador determina cuál de las secuencias se va a ejecutar. Si el valor del controlador no coincide con ninguno de los rótulos descritos y la secuencia por defecto (N+1) está presente, entonces ésta será ejecutada. Si esta secuencia no está presente (es opcional usarla) y el valor del controlador no coincide con ninguno de los rótulos, no será ejecutada ninguna de las secuencias; por lo tanto, continuará con la estructura siguiente.

En forma general se puede decir, que si el valor del controlador es i, se ejecutará una sola vez la secuencia correspondiente al caso i y que la secuencia (N+1) se ejecutará si está presente, para cualquier valor del controlador que no coincida con ningún caso.

Ejercicio resuelto No. 31

Una empresa tiene cuatro escalas de salario numeradas consecutivamente del 1 al 4, además, tiene un programa de incentivos de acuerdo a la categoría y si el número de unidades producidas es mayor de 50. Si está en la categoría 1 se le da un incremento de sueldo equivalente al 5% de su salario, si está en la 2 del 7%, en la 3 del 10% y en la 4 del 15%; esto es por cada empleado.

Se tiene la siguiente información: nombres, salario mensual, categoría y número de unidades producidas.

Hacer un algoritmo que determine el total devengado por cada empleado.

Análisis
Datos de entrada
- Nombres.
- Salario mensual.
- Categoría.
- Número de unidades producidas.

Datos de salida
- Nombres
- Total devengado.

Proceso
Se usará un esquema cualitativo, ya que no se conoce el número de empleados. Como las categorías son numéricas consecutivas, se puede utilizar la estructura caso y, dentro de cada una de estas preguntas si tiene incentivos o no.

Definición de variables

NOM: Nombres del trabajador.
SALM: Salario mensual.
CAT: Categoría.
NUP: Número de unidades producidas.
TDEV: Total devengado.

Algoritmo

```
INICIO
    LEA: NOM
    MIENTRAS NOM <> '*' HAGA
            LEA: SALM, CAT, NUP
            TDEV = SALM
            CASOS DE CAT
                    CASO 1: SI NUP > 50 ENTONCES
                                    TDEV = TDEV + 0.05 * SALM
                            FIN_SI
                    CASO 2: SI NUP > 50 ENTONCES
                                    TDEV = TDEV + 0.07 * SALM
                            FIN_SI
                    CASO 3: SI NUP > 50 ENTONCES
                                    TDEV = TDEV + 0.1 * SALM
                            FIN_SI
                    CASO 4: SI NUP > 50 ENTONCES
                                    TDEV = TDEV + 0.15 * SALM
                            FIN_SI
            FIN_CASOS
            ESCRIBA: NOM, "DEVENGO $", TDEV
            LEA: NOM
    FIN_MIENTRAS
FIN_INICIO
```

Prueba de escritorio

Archivo de datos

```
                ALBERTO     250000      3           45
          JUAN          300000      2           52
    LUZ           150000      4           30
ANA         200000      1           80
```

NOM	SALM	CAT	NUP	TDEV
~~ANA~~	~~200.000~~	~~1~~	~~80~~	~~210000~~
~~LUZ~~	~~150.000~~	~~4~~	~~30~~	~~150000~~
~~JUAN~~	~~300.000~~	~~2~~	~~52~~	~~321000~~
ALBERTO	250.000	3	45	250000

Salida

ANA DEVENGO:	$ 210.000
LUZ DEVENGO	$ 150.000
JUAN DEVENGO:	$ 321.000
ALBERTO DEVENGO:	$ 250.000

Ejercicio resuelto No. 32

En un programa deportivo se ha concluido que la duración de la sesión de práctica depende del peso de la persona y de las condiciones que presenta en la revisión médica. La revisión médica califica a las personas como de condición 3, 4, 5 ó 6. El tiempo de cada persona es igual al peso por una rata que se basa en su condición; ésta es respectivamente: 0.15, 0.21, 0.22 y 0.26. Elaborar un algoritmo que calcule las duraciones para las sesiones individuales.

Análisis
Datos de entrada
- Identificación de la persona.
- Peso de la persona.
- Condición.

Datos de salida
- Identificación de la persona.
- Tiempo de duración de la sesión de práctica.

Proceso
Se debe utilizar un esquema cualitativo. El proceso a utilizar se hace por persona y en forma repetida hasta que se encuentre una identificación igual a cero. Se utilizará una estructura caso, cuyo selector será la condición de la persona. Como el rango de la condición es de tres a seis, dentro del algoritmo será modificado de uno a cuatro. (Algunos lenguajes de programación aceptan rangos no consecutivos, inclusive con casos especificados como un conjunto de valores).

Definición de variables
lDEN: Identificación de la persona.
PESO: Peso de la persona.
CON: Condición.
T: Tiempo de duración de la sesión.
R: Rata de condición.

Algoritmo

```
INICIO
    LEA: IDEN
    MIENTRAS IDEN > 0 HAGA
        LEA: PESO, CON
        CON = CON - 2
        CASOS DE CON
            CASO 1:  R = 0.15
            CASO 2:  R = 0.21
```

```
                    CASO 3:  R = 0.22
                    CASO 4:  R = 0.26
                FIN_ CASOS
                T = PESO * R
                ESCRIBA: IDEN, T
                LEA: IDEN
        FIN_MIENTRAS
FIN_INICIO
```

Prueba de escritorio

Archivo de datos

```
                                  0
                         005                 69                 4
                    004                 68                 4
               003                 65                 3
          002                 80                 6
     001                 78                 5
```

IDEN	PESO	CON	R	T
~~001~~	~~78~~	~~5~~	~~0.22~~	~~17.16~~
~~002~~	~~80~~	~~6~~	~~0.26~~	~~20.80~~
~~003~~	~~65~~	~~3~~	~~0.15~~	~~9.75~~
~~004~~	~~68~~	~~4~~	~~0.21~~	~~14.28~~
005	69	4	0.21	14.49

Salida

```
001     17.16
002     20.80
003     9.75
004     14.28
005     14.49
```

6.2. Estructura PARA

La estructura *para* permite que una o más instrucciones *secuencia* se repitan cero o más veces, mientras los valores de una progresión aritmética de razón creciente o decreciente se vayan asignando a una variable denominada variable de control del ciclo para. El control del ciclo se hace en forma automática con base en parámetros que establece el programador.

Esta estructura es usada cuando se tiene un esquema cuantitativo y el contador que controla el ciclo se incrementa o disminuye en un valor constante. La diferencia como se implementa esta clase de ciclos con la estructura *mientras*, es que el *para* maneja la inicialización del contador y su incremento en forma automática, es decir, lo hace la estructura.

Los valores de los parámetros que maneja este ciclo son controlados por la estructura y, una vez establecidos sus valores o activado el ciclo, no es factible cambiarlos.

Representación

```
PARA VC = LI, LF, INC HAGA
    <secuencia>
FIN_PARA
```

Donde:

VC: Variable numérica entera que sirve de control del ciclo (contador en la estructura mientras).

Ll: Expresión entera utilizada como límite inicial del rango de valores que puede tomar VC (valor inicial del contador).

LF: Expresión entera usada como límite final del rango de valores que puede tomar VC (valor final del contador o número de repeticiones del ciclo en el esquema cuantitativo).

INC: Expresión entera que indica el valor del incremento o disminución de la variable de control; si el valor es uno se puede omitir.

Ejemplo

```
PARA I = 1, 5, 1 HAGA
    ESCRIBA: I
FIN_ PARA
```

Este ejemplo usando la estructura mientras sería:

```
I = 1
MIENTRAS 1 ≤ 5
    ESCRIBA: I
    I = I + 1
FIN_MIENTRAS
```

Funcionamiento

Cuando se activa la estructura, es asignado el valor de Ll a VC (VC = LI) en forma automática; se compara VC con LF: Si VC > LF, no se ejecuta la secuencia y salta a ejecutar la instrucción siguiente al ciclo (se ejecuta cero veces): Si VC ≤ LF, se ejecuta la secuencia de instrucciones una vez y automáticamente regresa al principio del ciclo para actualizar la variable de control, incrementándola con el valor de INC (VC = VC + INC); se compara nuevamente VC con LF: Si VC > LF, se deja de ejecutar la secuencia pasando a la instrucción siguiente: Si VC ≤ LF, se ejecuta otra vez la secuencia y regresa a actualizar VC. Este proceso continúa hasta que VC toma un valor superior a LF.

En caso de que INC sea negativo, la secuencia de instrucciones se ejecuta siempre que la variable de control tome un valor mayor o igual que el límite final (VC ≥ LF).

También se puede decir que si el incremento (INC) es positivo, la secuencia se ejecuta siempre que la variable de control sea menor o igual al límite final (VC ≤ LF).

Ejercicios resueltos No. 33

Encontrar el factorial de un número N positivo (problema resuelto No. 27 mediante la estructura MIENTRAS).

Análisis

El mismo ejercicio realizado anteriormente pero utilizando la estructura PARA, ya que es un esquema cuantitativo y el incremento del contador es constante.

Algoritmo

```
INICIO
```

```
    LEA: N                                              (1)
    FAC = 1                                             (2)
    PARA CON = 1, N, 1 HAGA
        FAC = FAC * CON                                 (3)
    FIN_PARA
    ESCRIBA: "EL FACTORIAL DE", N, "ES:", FAC           (4)
FIN_INICIO
```

Prueba de escritorio

Miremos el algoritmo, paso a paso, suponiendo un N igual a 4.

Se ejecuta la instrucción 1 N 4

Se ejecuta la instrucción 2 FAC 1

Se activa el ciclo PARA y ocurre lo siguiente:

- Se inicializa CON en 1 (CON = 1) CON 1

- Se compara CON con N
  Es CON $\leq$ N (Es 1 $\leq$ 4): sí, se ejecuta la instrucción 3 una vez

  (3) Se lleva 1 * 1 a FAC FAC 1

  Regresa automáticamente al principio del ciclo y:

- Incrementa CON en 1 (CON = 1 + 1) CON 2

- Compara CON con N
  Es CON $\leq$ N (Es 2 $\leq$ 4). sí, se ejecuta la instrucción 3.

  (3) Se lleva 1 * 2 a FAC FAC 2

  Regresa al principio del ciclo y:

- Incrementa CON en 1 CON 3

- Compara CON con N
  Es CON $\leq$ N (Es 3 $\leq$ 4): sí, se ejecuta la instrucción 3.
  (3) Se lleva 3 * 2 a FAC FAC 6

  Regresa al principio del ciclo y:

- Incrementa CON en 1 CON 4

- Compara CON con N
  Es CON $\leq$ N (Es 4 $\leq$ 4): sí, se ejecuta la instrucción 3.
  (3) Se lleva 4 * 6 a FAC FAC 24

  Regresa al principio del ciclo y:

- Incrementa CON en 1 CON
  5

- Compara CON con N
  Es CON $\leq$ N (Es 5 $\leq$ 4): no, se sale de la estructura ciclo y ejecuta la instrucción siguiente (4).

(4) Al ejecutarse esta instrucción se mostrará el siguiente resultado:

Ejercicio resuelto No. 34

Por cada uno de los estudiantes de una universidad se tiene un registro identificador que contiene: código del estudiante, nombre y número de materias cursadas. A continuación de este registro vienen los registros identificadores de materias cursadas, tantos como materias vio durante el semestre. Por cada materia cursada se tiene la siguiente información: código de la materia, nombre y número de créditos. Hacer un algoritmo que muestre por estudiante: la información de cada materia, el número de créditos cursados, el promedio crédito y el promedio crédito ponderado de todos los estudiantes.

Análisis

Datos de entrada
En este caso se tienen dos clases de registros de entrada. Si es un registro identificador de estudiante, éste contiene:

- Código del estudiante.
- Nombres del estudiante.
- Número de materias cursadas.

Si es un registro identificador de materia, se tienen los siguientes datos de entrada:

- Código de la materia.
- Nombre de la materia.
- Número de créditos de la materia.

Datos de salida
- Código del estudiante.
- Nombres.
- Número de materias cursadas.
- Código de cada materia.
- Nombre de cada materia.
- Número de créditos de cada materia.
- Nota de cada materia.
- Total de créditos cursados.
- Promedio crédito del estudiante.
- Promedio crédito ponderado de los estudiantes.

Proceso

Como no se conoce la cantidad de estudiantes se hace uso de un esquema cualitativo, que termine cuando encuentre un valor de código de estudiante igual a cero. Por cada registro identificador de estudiante se elabora un ciclo cuantitativo utilizando la estructura PARA, cuya variable de control fluctúe entre 1 y el número de materias cursadas por el estudiante.

Información a calcular por estudiante:

Total de créditos = Suma de los créditos de las materias cursadas. Este cálculo se hace a través de un acumulador que totalice el total de créditos.

Promedio crédito = Sumatoria de créditos¡ * Nota¡ dividido por el total de créditos cursados.

Promedio crédito ponderado = Sumatoria de los promedio créditos dividido por el total de estudiantes.

Formato de la salida de información:

CODIGO: XXXXXX	NOMBRES:XXXXXX	MATCURSADAS:XX
CODIGO MATERIA	NOMBRES	NOTA
XXXXX	XXXXXXXXXXX	X.XX
:	:	:
:	:	:
CREDITOS CURSADO:	XX	
PROMEDIO CREDITO:	X.XX	
CODIGO:XXXXXX	NOMBRES:XXXXXXX	MATCURSADAS:XX
:	:	:
PROMEDIO DEL GRUPO		

Definición de variables

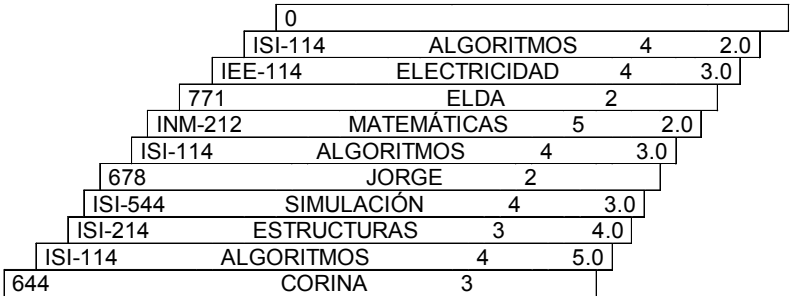
CODE: Código del estudiante.
NOME: Nombre del estudiante.
NMC: Número de materias cursadas.
CODM: Código de la materia.
NOMM: Nombre de la materia.
NC: Número de créditos de la materia.
NOTA: Calificación obtenida en cada materia.
NCC: Número de créditos cursados.
PC: Promedio crédito.
PCP: Promedio crédito ponderado.
NT: Número total de estudiantes.
SUMA1: Sumatoria de los créditos¡ *Nota¡.
SUMA2: Sumatoria de los promedio créditos.
I: Variable de control del ciclo PARA.

Algoritmo

```
INICIO
    NT, SUMA2 = 0
    LEA: CODE
    MIENTRAS CODE > 0 HAGA
        LEA: NOME, NMC
        NCC, SUMA1 = 0
        ESCRIBA: "CODIGO", CODE, "NOMBRES", NOME, "MATCURSADAS", NMC
        ESCRIBA:"CODIGOMATERIA NOMBRES NOTA"
        PARA I = 1, NMC, 1 HAGA
```

```
                    LEA: CODM, NOMM, NC, NOTA
                    NCC = NCC + NC
                    SUMA1= SUMA1 + NC * NOTA
                    ESCRIBA: CODM, NOMM, NOTA
                FIN_PARA
                PC = SUMA1 / NCC
                NT = NT + 1
                SUMA2 = SUMA2 + PC
                ESCRIBA: "CREDITOS CURSADOS:", NCC
                ESCRIBA: "PROMEDIO CREDITO:", PC
                LEA: CODE
        FIN_MIENTRAS
        PCP = SUMA2 / NT
        ESCRIBA: "PROMEDIO DEL GRUPO:", PCP
FIN_INICIO
```

Prueba de escritorio



~~ME~~	NMC	CODM	NOMM	NC	NOTA	NCC	PC	PCP	NT	SUMA1	SUMA2	I
~~INA~~	~~3~~	~~ISI-114~~	~~ALGORITMOS~~	~~4~~	~~5.0~~	~~0~~			~~0~~	~~0~~	~~0~~	~~1~~
		~~ISI-214~~	~~ESTRUCTURAS~~	~~3~~	~~4.0~~	~~4~~				~~20.0~~		~~2~~
		~~ISI-544~~	~~SIMULACIÓN~~	~~4~~	~~3.0~~	~~7~~				~~32.0~~		~~3~~
						~~11~~	~~4.0~~		~~1~~	~~44.0~~	4	~~4~~
~~GE~~	~~2~~	~~ISI-114~~	~~ALGORITMOS~~	~~4~~	~~3.0~~	~~0~~				~~0~~		~~1~~
		~~INM-214~~	~~MATEMÁTICAS~~	~~5~~	~~2.0~~	~~4~~				~~12~~		~~2~~
~~A~~	~~2~~					~~9~~	~~2.44~~		~~2~~	~~22~~	~~6.44~~	~~3~~
		~~IEE-114~~	~~ELECTRICIDAD~~	~~4~~	~~3.0~~	~~0~~				~~0~~		~~1~~
		ISI-114	ALGORITMOS	4	2.0	~~4~~				~~12~~		~~2~~
						8	2.5	2.98	3	20	8.94	3

Salida

CODIGO: 664	NOMBRES: CORINA	MATCURSADAS: 3
CODIGO MATERIA:	NOMBRES	NOTA:
ISI-114	ALGORITMOS	5.0
ISI-214	ESTRUCTURAS	4.0
ISI-544	SIMULACIÓN	3.0
CREDITOS CURSADOS:	11	
PROMEDIO CREDITO:	4.0	
CODIGO: 678	NOMBRES: JORGE	MATCURSADAS:2
CÓDIGO MATERIA:	NOMBRES:	NOTA:
ISI-114	ALGORITMOS	3.0
INM-212	MATEMÁTICAS	2.0
CREDITOS CURSADOS:	9	
PROMEDIO CREDITO:	2.44	
CÓDIGO: 771	NOMBRES: ELDA	MATCURSADAS: 2
CÓDIGO MATERIA	NOMBRES:	NOTA:
IEE-114	ELECTRICIDAD	3.0
ISI-114	ALGORITMOS	2.0
CRÉDITOS CURSADOS:	8	
PROMEDIO CRÉDITO:	2.5	
PROMEDIO GRUPO	2.98	

6.3. Estructura REPETIR (HACER MIENTRAS QUE)

Esta es otra estructura repetitiva de control que es implementada para efectuar un ciclo. Es parecida a la estructura *mientras* y en algunos aspectos se complementan. Se utiliza en situaciones en las que se desea que una secuencia se repita, al menos una vez, antes de comprobar la condición de repetición y está formada por dos partes: una expresión de tipo lógico y la secuencia de instrucciones, donde una de ellas debe estar modificando la expresión.

La diferencia especial entre la estructura *repetir* y la estructura *mientras* es que en la primera la secuencia de instrucciones se ejecuta por lo menos una vez, antes de evaluar la condición de repetición y, en la estructura *mientras* la condición de terminación es evaluada primero y, por lo tanto, el grupo de instrucciones o campo de la sentencia puede ser o no ejecutado.

Representación

```
REPETIR
    <secuencia>
MIENTRAS <expresión lógica>
```

Ejemplo

```
REPETIR
    A= A + 1
    ESCRIBA: A, B
MIENTRAS A <= B
```

Funcionamiento

Al entrar a la estructura se ejecuta la secuencia de instrucciones una vez y se evalúa la condición; si ésta es falsa, se sale de la estructura y continúa con la instrucción siguiente; y si es verdadera, se repite la ejecución de la secuencia y vuelve a evaluar la condición; el proceso se repite mientras que la condición sea verdadera.

La secuencia, entonces, se ejecutará repetidamente mientras que la expresión lógica sea cierta, asegurándose que la secuencia se ejecuta por lo menos una vez. La estructura *repetir* puede implementarse en ciclos cuyos esquemas sean cualitativos o cuantitativos. *Las instrucciones no se repiten cuando la condición es falsa*.

Ejercicio resuelto No. 35

Hacer un algoritmo que encuentre la suma de los primeros N números naturales, (Ejercicio resuelto No. 17, mediante la estructura MIENTRAS).

Análisis
El mismo realizado para el ejercicio propuesto No. 17, pero utilizando la estructura *repetir*.

Algoritmo

```
INICIO
    SUMA = 0                          (1)
    NUM = 1                           (2)
    LEA: N                            (3)
    REPETIR
        SUMA = SUMA + NUM             (4)
        NUM = NUM +1                  (5)
    MIENTRAS NUM <= N
    ESCRIBA: "LA SUMA ES: ", SUMA     (6)
FIN_INICIO
```

Prueba de escritorio

Para un N igual a 4 miremos la ejecución del algoritmo, paso a paso.

Se ejecuta la instrucción 1	SUMA 0
Se ejecuta la instrucción 2	NUM 1
Se ejecuta la instrucción 3	N 4
Se ejecuta la instrucción 4	SUMA 1
Se ejecuta la instrucción 5	NUM 2

Se evalúa la condición es NUM <= N (Es 2 <= 4): si, se ejecutan 4 y 5.

Ejecución de 4	SUMA 3
Ejecución de 5	NUM 3

Se evalúa la condición es NUM <= N (Es 3 <= 4): si, se ejecutan 4 y 5.

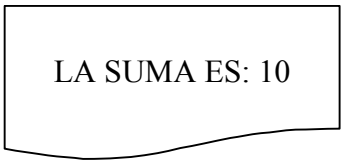
Ejecución de 4	SUMA
	6
Ejecución de 5	NUM
	4

Se evalúa la condición es NUM <= N (Es 4 <= 4): si, se ejecutan 4 y 5.

Ejecución de 4	SUMA
	10
Ejecución de 5	NUM
	5

Se evalúa la condición es NUM <= N (Es 5 <= 4): no, se sale y ejecuta la instrucción 6.

Salida



Ejercicio resuelto No. 36

Se recomiendan becas para estudiantes según su promedio crédito. El tipo de recomendación se basa en la siguiente tabla:

Promedio	**Recomendación**
≥4.8	Muy fuerte recomendación
≥4.5 y < 4.8	Fuerte recomendación
≥4.0 y < 4.5	Recomendado
< 4.0	No recomendado

Por cada estudiante se tiene un registro de datos con la siguiente información: código y promedio crédito. Diseñar un algoritmo que informe por estudiante: código, promedio y recomendación, promedio crédito general de los solicitantes y el número de recomendaciones de cada tipo.

Análisis

Datos de entrada
- Código del estudiante.
- Promedio crédito.

Datos de salida
- Código.
- Promedio.
- Tipo de recomendación (se hará mediante un mensaje).
- Número de estudiantes muy fuerte recomendados.
- Número de estudiantes de fuerte recomendación.
- Número de estudiantes recomendados.
- Número de estudiantes no recomendados.
- Promedio crédito general de los estudiantes.

Proceso

Se utilizará un esquema cualitativo implementado mediante la estructura *repetir*, ya que se supone que al menos un estudiante haga solicitud. El ciclo se ejecutará iterativamente hasta que encuentre un código de estudiante igual a cero. Por cada registro se hará un análisis de acuerdo al valor que tenga su promedio crédito, para tomar la decisión sobre qué tipo de recomendación se le da. También, de acuerdo con el valor del promedio crédito se debe ir contando el número de estudiantes de cada tipo de recomendación.

Para el cálculo del promedio crédito general se debe usar un acumulador que totalice los promedios créditos.

Definición de variables

COD: Código del estudiante.
PC: Promedio crédito.
EMFR: Número de estudiantes de muy fuerte recomendación.
EFR: Número de estudiantes de fuerte recomendación.
ER: Número de estudiantes recomendados.
ENR: Número de estudiantes no recomendados.
SPC: Sumatoria de los promedios crédito.
NTE: Número total de estudiantes.
PCG: Promedio crédito general.

Algoritmo

```
INICIO
    EMFR, EFR, ER, ENR, SPC = 0
    LEA: COD
    HAGA
        LEA: PC
        SPC = SPC + PC
        SI  PC >= 4.8  ENTONCES
            EMFR = EMFR + 1
            ESCRIBA: COD, PC, "MUY FUERTE RECOMENDACION"
          SINO
            SI  PC >= 4.5  ENTONCES
                EFR = EFR+ 1
                ESCRIBA: COD, PC, "FUERTE RECOMENDACION"
              SINO
                SI  PC >= 4.0 ENTONCES
                    ER = ER +1
                    ESCRIBA: COD, PC, "RECOMENDADO"
                  SINO
                    ENR = ENR +1
                    ESCRIBA: COD, PC, "NO RECOMENDADO"
                FIN_SI
            FIN_SI
        FIN_SI
        LEA: COD
    MIENTRAS COD <> 0
    NTE = EMFR + EFR + ER +ENR
    PCG = SPC / NTE
    ESCRIBA: EMFR, EFR, ER, ENR, PCG
FIN_INICIO
```

Prueba de escritorio

Archivo de datos

0
326 5.0
251 4.2
356 4.7
342 4.9
331 3.5

COD	PC	EMFR	EFR	ER	ENR	SPC	NTE	PCG
331	3.5	0	0	0	0	0	5	4.46
342	4.9	1	1	1	1	3.5		
356	4.7	2				8.4		
251	4.2					13.1		
326	5.0					17.3		
0						22.3		

Salida

331	3.5	NO RECOMENDADO		
342	4.9	MUY FUERTE RECOMENDACION		
356	4.7	FUERTE RECOMENDACIÓN		
251	4.2	RECOMENDADO		
326	5.0	MUY FUERTE RECOMENDACIÓN		
2	1	1	1	4.46

Ejercicio resuelto No. 37

Elabore un algoritmo que forme un cuadro con línea doble en el centro de la pantalla de un computador.

Análisis
Datos de entrada
- Fila donde inicia el cuadro.
- Fila donde termina el cuadro.
- Columna donde inicia el cuadro.
- Columna donde termina el cuadro.

Datos de salida
- El cuadro en pantalla

Proceso
Se hará uso de:
- La función gotoxy (c,f): ubica el cursor en la columna c y en la fila f de la pantalla.
- El ordinal de los caracteres ASCCII: 205,188,200 y 186 (buscarlo en la tabla de caracteres ASCCII).

Ejemplo: Escriba: carácter (205); imprime =

Definición de variables

FI: Fila donde inicia el cuadro.
FF: Fila donde termina el cuadro.
CI: Columna donde inicia el cuadro.
CF: Columna donde termina el cuadro.

Algoritmo

```
INICIO
    LEA: FI, FF, CI, CF //  Se coloca la línea superior e inferior
    PARA I = CI + 1, CF – 1, 1 HAGA
            gotoxy (I, FI)
                    escriba: caracter (205)
            gotoxy (I, FF)
                    escriba: caracter (205)
    FIN PARA

    // Se colocan las líneas verticales
    PARA I = FI + 1, FF, – 1, 1 HAGA
    gotoxy (CI, I)
            escriba: carácter (186)
    gotoxy (CF, I)
            escriba: carácter (186)
    FIN PARA


    // Se colocan las esquinas
    gotoxy (CI, FI)
            escriba: caracter (201)
    gotoxy (CF, FI)
            escriba caracter (187)
    gotoxy (CI, FF)
            escriba: caracter (200)
    gotoxy (CF, FF)
            escriba: carácter (188)
FIN_INICIO
```

Haga la prueba de escritorio con el siguiente registro:

7 19 19 62

Ejercicio resuelto No. 38

En las empresas públicas se ingresa la siguiente información por usuario:

- Código de instalación.
- Nombres y apellidos.
- Lectura actual de agua (m^3).
- Lectura anterior de agua (m^3).
- Lectura actual de energía (kw/h).
- Lectura actual de energía (kw/h).
- Lectura anterior de energía (kw/h).
- Estrato socioeconómico.

Para el estrato uno se tiene estipulado un descuento del 50% sobre el valor de la factura.

Para los estratos dos y tres se tiene estipulado un descuento del 30% sobre el valor de la factura.

Para el estrato cuatro no existe recargo ni descuento.

Para los estratos cinco y seis se tiene estipulado un recargo del 20% sobre el valor de la factura.

Elabore un algoritmo para conocer:

1. Total a pagar por consumo de agua y energía por usuario (1 kw/h vale $1000 y un m^3 de agua $1200).
2. Total recaudado ($) por agua y energía por las empresas públicas.
3. Total recaudado por estrato.

Análisis

Datos de entrada

- Código de instalación.
- Nombre del usuario.
- Lectura actual del agua.
- Lectura anterior del agua.
- Lectura actual de energía.
- Lectura anterior de energía.
- Estrato socioeconómico.

Datos de salida

- Código de instalación.
- Nombre del usuario.
- Estrato socio económico.

- Total a pagar por agua y energía cada usuario.
- Total recaudado por agua y energía por las empresas públicas.
- Total recaudado por cada estrato.

Proceso

La entrada de información se hará a través del procedimiento captura, excepto el estrato socioeconómico que se capturará a través de la tecla enter en un menú donde el desplazamiento se hace a través de flechas. El procedimiento captura tendrá los siguientes parámetros de envío:

Código de instalación	CI
Nombre del usuario	NOM
Lectura actual de agua	LEACA
Lectura anterior de agua	LEANA
Lectura actual de energía	LEACE
Lectura anterior de energía	LEANE

Debido a que este procedimiento valida la información que captura, dentro de él las variables numéricas deben definirse como caracteres y, luego de la validación, convertirlos a números.

* Ver procedimientos en el capítulo siguiente

Subalgoritmo

```
PROCEDIMIENTO CAPTURA (CI, NOM,LEACA,LEANA,LEACE,LEANE)
LEA:  CI
SI CI <> 0 ENTONCES
     LEA: NOM, LEACA, LEANE,LEACE, LEANE
FIN_CAPTURA
```

El procedimiento cálculo obtiene lo que el usuario debe pagar por concepto de agua y energía y acumula lo recaudado por estrato.

Parámetro de recibo

- Lectura actual de agua LEACA
- Lectura anterior de agua LEANA
- Lectura actual de energía LEACE
- Lectura anterior de energía LEANE
- Porcentaje aplicado al estrato POR

Parámetros de envío

- Pago por agua PCA
- Pago por energía PCE
- Total a pagar FACTURA
- Total recaudado por estrato RES

Subalgoritmo

```
PROCEDIMIENTO CALCULO (LEACA, LEANA, LEACE, LEANE, POR,
    PCA, PCE, FACTURA, RES)
    PCA = (LEACA - LEANA) * MCA * POR
    PCE = (LEACE - LEANE) * KWH * POR
    FACTURA = PCA + PCE
    RES = RES + FACTURA
FIN_CALCULO
```

Definición de variable

CI:	Código de instalación.
NOM:	Nombre del usuario.
ES:	Estrato socioeconómico.
LEACA:	Lectura actual del agua.
LEANA:	Lectura anterior de agua.
LEACE:	Lectura actual de energía.
LEANE	Lectura anterior de energía.
PCA:	Pago de agua por usuario.
PCE:	Pago de energía por usuario
FACTURA:	Pago total del usuario
MCA:	Valor metro cúbico de agua
KWH:	Valor Kilovatio hora
RE1,RE2,RE3,RE4,RE5, RE6:	Totales recaudados por cada estrato.
TRAE:	Total recaudado por agua.
TREE:	Total recaudado por energía.

Algoritmo

```
INICIO
    RE1, RE2, RE3, RE4, RE5, RE6, TRAE, TREE = 0
    CAPTURA (CI, NOM, LEACA, LEANA, LEACE, LEANE)
    MIENTRAS CI <> 0 HAGA
            ESCRIBA: "ESTRATO 1"
            ESCRIBA: "ESTRATO 2"
            ESCRIBA: "ESTRATO 3"
            ESCRIBA: "ESTRATO 4"
            ESCRIBA: "ESTRATO 5"
            ESCRIBA: "ESTRATO 6"
```

```
            ESCRIBA: "SELECCIONE  OPCION"
            LEA: ES
            CASOS DE ES
                  CASO 1:
                        CALCULO (LEACA, LEANA, LEACE,
                  CASO 2:
                        CALCULO (LEACA, LEANA, LEACE,
                  CASO 3:
                        CALCULO (LEACA, LEANA, LEACE,
                  CASO 4:
                        CALCULO (LEACA, LEANA, LEACE,
                  CASO 5:
                        CALCULO (LEACA, LEANA, LEACE,
                  CASO 6:
                        CALCULO (LEACA, LEANA, LEACE,
            FIN_CASOS
            TRAE = TRAE + PCA
            TREE = TREE + PCE
            ESCRIBA: NOM, ES, PCA, PCE, FACTURA
            CAPTURA (CI, NOM, LEACA, LEANA, LEACE, LEANE)
      FIN_MIENTRAS
      ESCRIBA: TRAE, TREE, RE1, RE2, RE3, RE4, RE5, RE6
FIN_INICIO
```

Aspectos a tener en cuenta

a. Al llegar a la estructura CASO, la variable que hace las veces de controlador debe tener asignado un valor.
b. Al entrar a la estructura CASO sólo se ejecuta una de las secuencias: aquélla que coincida con el valor del selector.
c. En la estructura PARA las variables de control límite inicial y límite final, pueden ser expresiones cuyos valores son manejados en forma automática, por lo que éstas no deben ser manipuladas por el usuario dentro de la secuencia.
d. El uso de la variable de control después de un ciclo PARA no es aconsejable, porque en algunos casos se desconoce su valor; por lo tanto, se deberá destruir su valor mediante una asignación para su posterior utilización.
e. La estructura REPETIR puede usarse en esquemas cuantitativos o cualitativos, siempre que se tenga la certeza de que la secuencia se debe ejecutar al menos una vez.
f. A las variables de la condición del REPETIR se les debe asignar un valor antes de evaluar la condición.
g. Aunque con MIENTRAS se puede implementar cualquier ciclo, REPETIR y PARA algunas veces facilitan la programación.

6.4. Ejercicios propuestos

103. Se tienen 300 registros donde cada uno de ellos contiene:
- Estatura (1: Alto, 2: Bajo, 3: Mediano).
- Nacionalidad (1: Latino, 2: Germano, 3: Hindú).
- Edad.

Hacer un algoritmo que determine: número de indúes altos, latinos medianos, total de latinos, germanos bajos, total de germanos menores de 30 años y latinos bajos menores de 40 años.

104. En la elección del candidato estudiantil se presentaron 5 candidatos numerados consecutivamente del 1 al 5. Cada elector depositó su voto con el número de su candidato

favorito. Al final del escrutinio se desea saber cuántos votos obtuvo cada candidato y el porcentaje de votos por cada participante.

*Utilizando la estructura **PARA**, solucione los siguientes 11 ejercicios:*

105. Hacer un algoritmo que calcule e imprima los números primos comprendidos entre 1 y 100.
106. Desarrolle un algoritmo que lea tres registros, donde cada uno de ellos contenga las ordenadas y abscisas de tres puntos sobre el plano cartesiano e investigue si los tres puntos forman una línea recta.
107. Cierta universidad tiene N estudiantes. Elabore un algoritmo que encuentre el promedio de edad de los estudiantes mayores de 21 años y el promedio de edad del resto de estudiantes. Por cada estudiante se tiene un registro que contiene su código y edad.
108. En un grupo de N registros se tienen las notas de: ALGORITMOS, MATEMATICAS y ESPAÑOL de cada uno de los estudiantes. Diseñe un algoritmo que obtenga la nota promedio de cada estudiante.
109. Elabore un algoritmo que lea 100 números y determine la media de los números positivos y negativos.
110. El seno de X puede calcularse, de forma aproximada, sumando los N primeros términos de la serie infinita:

$$\text{Sean } X = X - \frac{X^3}{3!} + \frac{X^5}{5!} - \frac{X^7}{7!} + \cdots\cdots$$

Elabore un algoritmo para calcular el seno de X, mediante la suma de los primeros N términos.

111. Diseñe un algoritmo que convierta los N primeros números positivos a números romanos.
112. Escribir un algoritmo que calcule X^n donde:
X, puede ser cualquier número real.
n, un número positivo, nulo o negativo.
113. Elabore un algoritmo que obtenga el cuadrado de los primeros N números, utilizando sumas únicamente.
114. Elabore un algoritmo que obtenga el cubo de los primeros N números. Hágalo utilizando multiplicaciones y sumas únicamente.
115. Elabore un algoritmo que permita calcular el factorial de los primeros N números usando sumas únicamente.

Utilizando la estructura **REPETIR**, solucione los siguientes 10 ejercicios:

116. Una empresa posee los siguientes datos en el registro de cada uno de sus clientes:
Nombre, sexo, ('M' o "F), edad, altura (en metros), peso (en libras), color de los ojos (1 para azules, 2 para castaños y 3 para los demás), color del cabello (1 para castaño, 2 para rubio y 3 para los demás). Diseñe un algoritmo para que lea el archivo e imprima los nombres de:

a) Todas las mujeres de cabello rubio y ojos azules, que miden entre 1.65 metros y 1.75 metros y que pesen menos de 120 libras.
b) Todos los hombres de ojos castaños de más de 1.70 metros de altura y que pesen entre 180 y 220 libras.

117. Se tiene un conjunto de registros con la siguiente información: cédula y número de hijos. Elaborar un algoritmo que determine cuántas personas tienen hijos y cuántas no tienen; además, se desea conocer el promedio de número de hijos.
118. En un *peaje* se desea saber cuántos carros y cuántos buses pasaron en un día, el promedio de personas que viajan en carro y el promedio de los que viajan en bus. Elabore un algoritmo para encontrar lo anterior, teniendo en cuenta que por cada vehículo que pase se

elabora un registro donde se especifica el tipo de vehículo (1 si es bus y 2 si es un carro diferente) y el número de personas que lo ocupan.

119. Un administrador desea obtener los datos necesarios para elaborar una curva de salarios en su empresa, en función del puntaje asignado por valorización de méritos. Las fórmulas empleadas para este propósito son las siguientes:

$$S = \begin{cases} 20 + 0.485P^2 & \text{para } P < 120 \\ (18000) / (1 + P / 18000) & \text{para } P \geq 120 \end{cases}$$

Elabore un algoritmo que encuentre S para valores de P desde 20 a 200 con incrementos de 5, donde:

S = Salario en pesos.

P = Puntaje.

120. Para un período de 10 años se invierten 250 mil pesos en los años 1, 3, 5, 7 y 9. Determinar el valor del capital al final de los años 1, 2, 3, 4, 5, 6, 7, 8, 9 y 10, con intereses del 20% anual.
121. Una estación climática proporciona un par de temperaturas diarias (máxima, mínima - no es posible que alguna o ambas temperaturas sea 0 grados). La pareja fin de temperaturas es 0, 0. Elabore un algoritmo para determinar el número de días, cuyas temperaturas se han proporcionado, las medidas máxima y mínima, el número de errores (temperaturas de 0 grados) y el porcentaje que representan.
122. Haga un algoritmo que calcule y tabule los valores de la función:
$f(x, y) = (x^2 - y^2)/(x^2 + y^2)$.
123. Elabore un algoritmo que lea dos números M y N (M < N) y calcule los múltiplos de M comprendidos entre M y N.
124. Por cada estudiante de una universidad se tienen los siguientes datos: código del estudiante, nombre, valor crédito del estudiante, número de créditos tomados y, valor del recargo en la matrícula.
Hacer un algoritmo que entre la anterior información y muestre por cada estudiante: código, nombre, valor crédito, número de créditos, valor del recargo y total pagado, lo mismo que el monto pagado por todos los estudiantes.
125. Diseñar un algoritmo para calcular el número de puntos con coordenadas de valores enteros que están contenidos en la siguiente elipse:

$$\frac{X^2}{16} + \frac{Y^2}{25} = 1$$

Notas: Los puntos sobre la elipse se consideran dentro de ella

El intervalo de coordenadas está limitado por los ejes mayor y menor de la elipse, es decir:

$-4 \leq X \leq 4$ y $-5 \leq Y \leq 5$

126. Elabore un algoritmo que calcule, usando un menú, si cada uno de un conjunto de números positivos es: par, primo y perfecto (un número es perfecto cuando la suma de todos sus divisores, excepto por él mismo da el mismo número).

Capítulo 7

SUBPROGRAMAS

A lo largo del texto se han tratado los problemas como un todo, como si fuesen un solo módulo. Una de las grandes ventajas que tiene la programación estructurada es la división de problemas grandes en subproblemas y, a su vez, éstos también pueden dividirse en problemas más pequeños. Esta técnica le permite dar más entendibilidad y facilidad en la construcción y corrección de los errores que se presentan en la solución de un problema determinado. Cada una de las divisiones que se hacen para obtener la solución de un problema se denomina módulo y éstos son implementados por medio de los subprogramas.

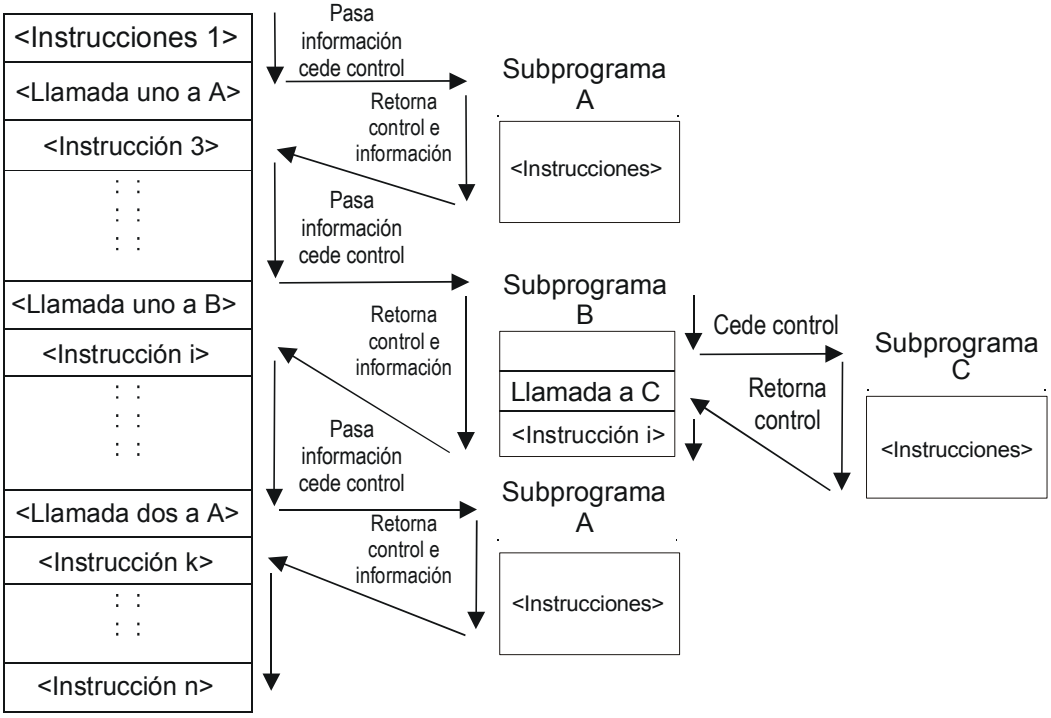
Un subprograma es un algoritmo diseñado para efectuar una tarea particular, bajo la dependencia de un algoritmo u otro subprograma que lo utiliza. De esta manera, el subprograma es diseñado para ser utilizado por otros procesos más amplios, dentro de los cuales existe un flujo de recibo y entrega de información.

La utilización de subprogramas en la solución de problemas grandes tiene muchas ventajas. Por ejemplo, para tareas que deban efectuarse más de una vez, la modularidad evita la necesidad de programación redundante, esencialmente el mismo conjunto de instrucciones; en vez de ello, un módulo puede construirse e invocarse cada vez que se necesite hacer la misma labor, por lo tanto, el uso de módulos permite o hace que los programas sean más cortos. De igual manera, la fragmentación en módulos individuales proporciona claridad, facilidad de distribución de trabajo en equipo y, por ser algoritmos cortos, son más fáciles de escribir, corregir y su estructura lógica es más clara que la de los programas que no están divididos en módulos.

El subprograma, por ser un algoritmo, debe cumplir con las mismas características de éste y hacer tareas similares como aceptar datos, escribir datos y hacer cálculos; sin embargo, es utilizado para un propósito específico. El subprograma recibe datos del algoritmo o subalgoritmo que lo invoca y éste le devuelve resultados. Su labor puede compararse con la de un jefe que le da instrucciones a un empleado (Subprograma), para que realice una determinada labor y así poder continuar con su trabajo; se detiene mientras su subalterno realiza la tarea y cuando éste le devuelve los resultados reanuda su labor. Asimismo, cuando un algoritmo invoca un subprograma le cede el control de ejecución a éste, por lo tanto, detiene la ejecución de la próxima instrucción hasta que el subprograma se ejecute, le entregue resultados y le devuelva el control de ejecución.

La comunicación que se establece entre el algoritmo llamante y el subprograma se hace a través de variables denominadas parámetros, que hacen las veces de recipiente, unas de recibo y otras de envío de información.

Un subprograma puede ser invocado tantas veces como se necesite y, a su vez, los subprogramas pueden invocar a otros subprogramas, como puede verse en la figura siguiente.



En el gráfico las flechas indican el flujo de ejecución del algoritmo; observe que cuando se ejecuta una instrucción de llamada a un subprograma, el algoritmo llamante se detiene, ya que no solamente le transfiere información al subprograma sino también el control de ejecución; esto hace que se active el subprograma, se les dé valor a las variables que lo conforman y se ejecuten una a una las instrucciones. Una vez que finaliza su ejecución, tiene la información requerida por el algoritmo llamante; por lo tanto, se la entrega y devuelve el control de ejecución a la instrucción siguiente o a la misma instrucción desde donde fue llamado, dependiendo de la clase de subprograma.

En la figura también se muestra que un algoritmo puede invocar las veces que sea necesario al mismo subprograma y que un subprograma puede, a su vez, invocar a otros subalgoritmos.

Cuando un algoritmo invoca a un subprograma es como si las instrucciones de éste se copiaran dentro del algoritmo y las ejecutara; en el caso del subprograma A, no tiene sentido tener el mismo grupo de instrucciones repetidas dentro del algoritmo; como los subprogramas pueden compilarse y ponerse a punto independientemente del programa que los invoque, éstos se diseñan para que sean usados en diferentes programas. Por ejemplo, si en un sistema de cómputo con frecuencia se necesitan ordenamientos de los elementos de un vector, una buena solución es diseñar un subprograma que haga el ordenamiento y cada vez que sea necesario implementarlo, simplemente se le encomienda la tarea al subprograma (No hay que hacerlo).

7.1. Clasificacion de los subprogramas

La división de los subprogramas se hace con base en el número o cantidad de valores que el subprograma le envía al programa o subprograma llamante.

7.1.1. Procedimiento o subrutinas

Son subprogramas que devuelven cero o más valores al programa o subprograma que lo invoca (utiliza) y retornan el control de ejecución a la instrucción siguiente desde donde se llaman.

a. Representación

PROCEDIMIENTO Nombre (PARÁMETROS)

```
    Instrucciones
FIN_NOMBRE
```

El Nombre dado al subprograma debe cumplir con las normas para conformar nombres de variables y debe ser único (no pueden existir dos o más subprogramas con el mismo nombre).

Los Parámetros son una lista de campos variables separados por comas y son utilizados para establecer la comunicación entre el módulo que lo activa y el subprograma. Unos parámetros serán de recibo (almacenan la información que les envía el módulo que los invoca) y otros de envío de información (la transfieren a quien invoca el subprograma).

Ejercicio resuelto No. 39

Hacer un subprograma que ordene tres números distintos en forma ascendente.

Análisis

La parte de análisis en la construcción de un subprograma es semejante a la que se ha desarrollado hasta ahora con los algoritmos que se han venido tratando en los capítulos anteriores.

Consiste entonces en detectar cuáles son los parámetros de recibo, cuáles los de envío y el proceso que es necesario hacer para llegar a los parámetros de envío, partiendo de los de recibo.

Parámetros de recibo

Son las variables que van a recibir información del módulo que los invoca, similares a los datos de entrada que se han detectado en los algoritmos anteriores, con la diferencia de que a los datos de entrada se les transfiere información a través de una instrucción de lectura y a los parámetros de recibo se les da información a través del módulo que invoca el subprograma; o sea, que es un asignamiento de información automática desde las variables que envían información, desde el módulo llamante, hacia los parámetros de recibo del subprograma. Estos parámetros son:

- Valor del primer número
- Valor del segundo número
- Valor del tercer número

Parámetros de envío

Son los parámetros que envían información al módulo que invoca al subprograma, semejante a los datos de salida de los algoritmos, con la diferencia de que los parámetros de envío le entregan la información a las variables que la reciben dentro del módulo llamante.

- Menor valor
- Valor medio
- Mayor valor

Proceso

Para determinar el orden ascendente de los tres números, primero se encuentra el menor de todos a través de comparaciones entre sí y se asigna al menor valor, luego se encuentra el segundo menor y se asigna al valor medio y el restante, por defecto, es el valor mayor.

Definición de variables

ORDEN:	Nombre del procedimiento
PV:	Primer valor
SV:	Segundo valor
TV:	Tercer valor
MENOR:	Valor menor
MEDIO:	Valor medio

MAYOR: Valor mayor

Algoritmo

```
PROCEDIMIENTO ORDEN (PV, SV, TV, MENOR, MEDIO, MAYOR)
    SI (PV<SV) ∧ (PV<TV) ENTONCES
            MENOR=PV
            SI SV<TV  ENTONCES
                    MEDIO=SV
                    MAYOR=TV
              SINO
                    MEDIO=TV
                    MAYOR=SV
            FIN_SI
       SINO
            SI  SV<TV ENTONCES
                    MENOR= SV
                    SI PV<TV ENTONCES
                            MEDIO =PV
                            MAYOR =TV
                    FIN_SI
              SINO
                    MENOR=TV
                    SI PV<SV ENTONCES
                            MEDIO= PV
                            MAYOR= SV
                      SINO
                            MEDIO= SV
                            MAYOR= PV
                    FIN_SI
            FIN_SI
    FIN_SI
FIN_ORDEN
```

b. Activación de un procedimiento

El subprograma no inicia ni termina por sí mismo; por lo tanto, necesita ser activado (hacerlo funcionar) desde un modulo externo. Esto se hace cuando el módulo llamante necesita la ayuda para una labor específica que hace el módulo subordinado. La activación se hace a través de la siguiente instrucción.

Formato

Nombre (Argumentos)

Los argumentos son nombres de campos (constantes o variables), que usa el modulo amplio y que tienen una correspondencia biunívoca con los parámetros con los cuales se construyó el subprograma; por lo tanto, estos deben ser iguales en número y en tipo de dato que los parámetros, ya que a cada argumento le corresponde un parámetro ó viceversa.

Los argumentos también son de dos clases: unos que envían información al subprograma y otros que reciben información.

Cuando el subprograma es activado, los argumentos de envío son copiados en los parámetros de recibo, (parámetros de recibo argumentos de envío). Y cuando el subalgoritmo deja de

ejecutarse los parámetros de envío le asignan información a los argumentos de recibo (argumentos de recibo parámetros de envío).

Por ejemplo, elaborar la instrucción que active el procedimiento ORDEN. Este subprograma tiene 6 parámetros (tres de recibo y tres de envío), por lo tanto, la instrucción que lo invoque debe tener 6 argumentos (tres de envío y tres de recibo).

Si las variables a utilizar como argumentos son en su orden: N1, N2, N3, MEN, MED y MAY la instrucción sería.

```
ORDEN (N1, N2, N3, MEN, MED, MAY)
```

Lo que significa lo siguiente:

a. Cuando se activa el subprograma (empieza a ejecutarse).

```
PV=N1
SV=N2
TV=N3
```

Por lo tanto N1, N2 y N3 deben tener valor.

b. Cuando ORDEN deja de ejecutarse (regresa información)

```
MEN=MENOR
MED=MEDIO
MAY=MAYOR
```

Ejercicio resuelto No. 40

Elaborar un algoritmo que active el procedimiento ORDEN

Análisis

Datos de entrada

• Los valores de los tres números a ordenar.

Datos de salida

• Los tres números ordenados.

Proceso

Se leen los tres números, se invoca el subprograma que hace el ordenamiento y se muestran los valores ordenados.

Definición de variables

N1:	Valor del primer número.
N2:	Valor del segundo número.
N3:	Valor del tercer número.
MEN:	Valor del número menor.
MED:	Valor del número intermedio.
MAY:	Valor del número mayor.

Algoritmo

```
INICIO
    LEA: N1, N2, N3
    ORDEN (N1,N2,N3, MEN, MED, MAY)
    ESCRIBA: "VALOR MENOR:", MEN, "VALOR MEDIO:", MED,
             "VALOR MAYOR:", MAY.
FIN_INICIO
```

Prueba de escritorio

Si los valores a entrar son: 20,14,10, en el algoritmo pasaría lo siguiente:

Se ejecuta la instrucción 1:

N1	N2	N3	MEN	MED	MAY
20	14	10			

Al ejecutarse la instrucción 2 se le cede el control de ejecución al subprograma ORDEN, para que ejecute sus instrucciones y devuelva los resultados pedidos; por tanto, le es dado a cada parámetro su respectiva dirección de memoria.

N1	PV	N2	SV	N3	TV	MEN, MENOR	MED, MEDIO	MAY, MAYOR
20	20	14	14	10	10	10	14	20

La anterior gráfica muestra cómo las variables del subprograma sólo son llevadas a memoria en el momento en que éste se activa; y que los parámetros PV, SV y TV tienen una dirección de memoria diferente a la de los argumentos N1, N2 y N3, y los parámetros MENOR, MEDIO y MAYOR, comparten la misma dirección de memoria con los argumentos MEN, MED y MAY respectivamente

Al devolverse el control de ejecución al algoritmo, las variables de ésta tienen los siguientes contenidos

N1	N2	N3	MEN	MED	MAY	Desaparecen las variables
20	14	10	10	14	20	del subprograma

Al ejecutarse la instrucción 3 la salida será:

```
VALOR MENOR: 10
VALOR MEDIO: 14
VALOR MAYOR: 20
```

7.1.2. Funciones

Son subprogramas que le devuelven al programa o subprograma que los invoca un único valor. El valor que éstas devuelven lo hacen a través de una variable involucrada en una instrucción de retorno, dentro del conjunto de instrucciones, o en el nombre de ésta; por lo tanto, este tipo de subprogramas en su encabezamiento (prototipo) sólo tienen parámetros de recibo. Las funciones devuelven el control de ejecución a la misma instrucción desde donde se llaman.

a. Representación

```
FUNCIÓN Nombre (PARAMETROS DE RECIBO)
    Instrucciones
    Retorne valor
FIN_Nombre
```

Ejercicio resuelto No. 41

Elaborar una función que encuentre el valor mayor en un conjunto de 3 números reales diferentes.

Análisis

Parámetros de recibo
- Valor del primer número.
- Valor del segundo número.
- Valor del tercer número.

Valor de retorno (Parámetros de envío)
- El valor mayor de los tres números.

Proceso
Se comparan los tres valores entre sí; el valor que resulte mayor será llevado a una variable y este será el valor que retorna la función.

Definición de variables
MAYOR: Nombre de la función
N1: Primer número.
N2: Segundo numero.
N3: Tercer número.
VALMAY: Valor mayor entre N1, N2 y N3.

Subalgoritmo

```
FUNCIÓN MAYOR (N1, N2, N3)
    SI (N1 > N2) ∧ (N1 > N3) ENTONCES
            VALMAY = N1
      SINO
            SI (N2 > N3) ENTONCES
                  VALMAY = N2
              SINO
                  VALMAY = N3
            FIN_SI
    FIN_SI
    RETORNE VALMAY
FIN_MAYOR
```

b. Activación de una función

Recuerde que las funciones, por no tener en su encabezamiento parámetros de envío, la instrucción que active a la función tampoco tendrá argumentos de recibo. El valor que devuelve la función debe ser asignado a una variable, la cual debe ser del mismo tipo de dato del valor que devuelve la función (no puede existir incompatibilidad de tipos de datos).

Formato

Variable = Nombre de la función (Argumentos)
Por ejemplo, elaborar una instrucción que invoque a la función MAYOR.

Como la función tiene tres parámetros, se debe invocar con tres argumentos y recibir el valor que ella devuelve en una variable. Si los valores a enviar son NUM1, NUM2 y NUM3 y se recibe el valor mayor en la variable NUM_MAYOR la instrucción sería:

NUM_MAYOR = MAYOR (NUM1, NUM2, NUM3)

Ejercicio resuelto No. 42

Elaborar un algoritmo que active la función MAYOR.

Análisis

Datos de entrada
- Los valores de los tres números.

Datos de salida
- El valor mayor entre los tres números.

Proceso
Se leen los tres valores, se invoca la función y se imprime el valor mayor.

Definición de variables

NUM1:	Valor del primer número.
NUM2:	Valor del segundo número.
NUM3:	Valor del tercer número.
NUM_MAYOR:	Variable que recibe de la función el valor mayor.

Algoritmo

```
INICIO
    LEA: NUM1, NUM2, NUM3
    NUM_MAYOR = MAYOR (NUM1, NUM2, NUM3)
    ESCRIBA: "EL VALOR MAYOR ENTRE:", NUM1, ",", NUM2, "y", NUM3,
             "ES:",NUM_MAYOR
FIN_INICIO
```

Una ventaja de las funciones es que pueden estar involucradas en expresiones, como se muestra en el siguiente algoritmo.

En algunos casos particulares, cuando el programa lo requiera, se pueden enviar valores constantes como argumentos, cuando se invoca un subprograma (procedimiento ó función)

Ejercicio resuelto No. 43

Utilizar la función MAYOR para encontrar el valor mayor entre 90.5 y dos valores que entrará el usuario.

Algoritmo

```
INICIO
    LEA:NUM1, NUM2
    ESCRIBA: " EL VALOR MAYOR ENTRE:", NUM1, ",", NUM2, "y 90.5
             ES:", MAYOR (NUM1, NUM2, 90.5)
FINAL_INICIO
```

7.2. Documentación de subprogramas

Una de las grandes utilidades que tienen los subprogramas es que se construyen para que sean utilizados por diferentes módulos, cada que se requiera la tarea específica que realiza el módulo subordinado.

Debido a esto, cada que se construye un subprograma se debe indicar la forma de su empleo. Esta documentación debe incluir:

Nombre: es el nombre escogido por quien lo construye, para diferenciarlo de los demás subprogramas; por lo tanto, debe ser único.

Función: es la redacción de lo que hace el subprograma.

Parámetros: Se deben describir cada uno y en el orden en que se encuentran en la lista; no importa el nombre del parámetro, pero sí el tipo de dato de cada uno de ellos.

En algunos casos específicos es necesario describir otras características tales como:

Método usado: esta parte se describe en caso de que la solución de la tarea que hace el subprograma tenga más de un modelo de solución; en consecuencia, se debe decir cuál de los métodos de solución se empleó.

Otros subprogramas usados: si el módulo activa otros subprogramas, es conveniente incluir la documentación de éstos.

Ejemplo: elaborar la documentación del subprograma construido en el ejercicio resuelto numero 39.

Nombre: orden (procedimiento)

Función: entrega ordenados ascendentemente tres valores diferentes.

Los valores ordenados los transfiere en tres variables distintas.

Parámetros:

1. Nombre que recibe el primer valor numérico a ordenar.
2. Variable que recibe el segundo valor numérico.
3. Campo que recibe el tercer valor numérico.
4. Campo sencillo que entrega el valor menor de los tres datos.
5. Campo sencillo que entrega el valor intermedio de los tres datos.
6. Campo sencillo que entrega el valor mayor de los tres valores entregados.

Métodos: entrega los valores ordenados en variables distintas a las recibidas.

Otros subprogramas utilizados: Ninguno.

Cuando se utiliza un subprograma no interesa conocer el nombre de las variables utilizadas como parámetros, pero sí el orden en que se encuentran en la lista y el tipo de dato de cada una de ellas.

Ejercicio resuelto No. 44

Elabore un subprograma que encuentre la suma y el promedio de un conjunto de N números, lo mismo que el programa que lo invoque.

Análisis

Parámetros de recibo

- Cantidad de números N.

Parámetros de envío

- Suma de los N números SUMA.
- Promedio de los números PROM.

Proceso

Clase de subprograma: PROCEDIMIENTO
Nombre: SUMPROM

Subalgoritmo

```
PROCEDIMIENTO SUMPROM (N, SUMA, PROM)
    SUMA = 0
```

```
    PARA I = 1, N, 1 HAGA
        LEA: NUM
        SUMA = SUMA + NUM
    FIN_PARA
    PROM = SUMA / N
FIN_SUMPROM
```

Algoritmo llamante

Análisis

Datos de entrada

- Cantidad de números CN

Datos de salida

- Suma de los números SUM
- Promedio de los números PROMEDIO

Algoritmo

```
INICIO
LEA: CN
SUMPROM (CN, SUM, PROMEDIO)
ESCRIBA: "LA SUMA DE LOS", CN, "NUMEROS ES:", SUM
ESCRIBA: "EL PROMEDIO ES:", PROMEDIO
FIN_INICIO
```

Ejercicio resuelto No. 45

Elaborar un subprograma que encuentre la suma y el promedio de un conjunto de números enteros, lo mismo que el subprograma que lo utilice.

Análisis

Parámetros de recibo.
No tiene

Parámetros de envío
• La suma del conjunto de números: S.
• El promedio de los números: P.

Proceso
Clase de subprograma: PROCEDIMIENTO
Nombre: SPROM

Como no se sabe cuantos números son, una solución es leer el conjunto de números en el subprograma a través de un esquema cualitativo, usando una bandera para controlar el ciclo de entrada e ir contando y sumando los números leídos.

Deben usarse las siguientes variables locales:

N: Contador de números, necesario para calcular el promedio.

SW: Bandera o switche para controlar al ciclo de entrada de datos, tomará los valores S si hay más valores o N cuando no existen más datos.

NUM: Número que se lee.

Subalgoritmo :

```
PROCEDIMIENTO SPROM (S, P)
    SW = ´S´
```

```
    N, S = 0
    MIENTRAS SW = 'S' HAGA
              LEA: NUM
            S = S + NUM
            N = N+ 1
            ESCRIBA: "HAY MAS DATOS S/N "
            LEA: SW
    FIN_MIENTRAS
    P = S / N
FIN_SPROM
```

Algoritmo llamante

Análisis

Datos de entrada

No tiene

Datos de salida (argumentos para llamar a SPROM)

- Suma de los números: SUM
- Promedio de los números: PROM

```
INICIO
SPROM (SUM, PROM)
ESCRIBA: "LA SUMA ES:", SUM
ESCRIBA: "EL PROMEDIO ES:", PROM
FIN_INICIO
```

Ejercicio resuelto No. 46

Elaborar un subprograma que encuentre el salario bruto, deducción, subsidio y salario neto, de un empleado que labora X horas a Y pesos.

La retención para el empleado se determina así: para salarios menores de $400.000 si el número de hijos es mayor de 6 no hay retención; si el número de hijos es menor o igual a 6, se le retiene un porcentaje igual a 6 menos el número de hijos dividido por 2. Para salarios iguales o mayores de $400.000 si el número de hijos es menor de 3, se le retiene un 3%; si el número de hijos es mayor o igual a 3 se le retiene un porcentaje igual a 10 dividido por el número de hijos; por cada hijo, al empleado se le da un subsidio de $2000.

Análisis

Parámetros de recibo

- Numero de horas trabajadas: X
- Valor hora trabajada: Y
- Numero de hijos: NH

Parámetros de envío

- Salario bruto: SB
- Deducción: DED
- Subsidio: SUB
- Salario neto: SN

Proceso

SB = X * Y

La deducción se obtiene comparando el salario bruto con 400.000 y de acuerdo al número de hijos.

El subsidio será 2.000 * NH

SN= SB - DED + SUB

Clase de subprograma: PROCEDIMIENTO

Nombre: CAL_SALARIO

Variables
PR: Porcentaje de retención (variable local)

Subalgoritmo

```
PROCEDIMIENTO CAL_SALARIO (X, Y, NH, SB, DED, SUB, SN)
    SB = X * Y
    SI SB < 400000 ENTONCES
            SI NH > 6 ENTONCES
                    PR = 0
              SINO
                    PR = (6-NH)/2
            FIN_SI
      SINO
            SI NH < 3 ENTONCES
                    PR = 3
              SINO
                    PR = 10/NH
            FIN_SI
    FIN_SI
    DED = SB * PR/100
    SUB = 2000 * NH
    SN = SB - DED + SUB
FIN_CAL_SALARIO
```

Ejercicio resuelto No. 47

Se tiene la siguiente información por cada uno de los empleados de una empresa:

Código
Nombre
Número de hijos
Salario por hora
Número de horas trabajadas por mes

La deducción y el subsidio de cada empleado se calcula de acuerdo con lo estipulado en el subprograma anterior (ejercicio resuelto No 46).

Elaborar un algoritmo que muestre: código, nombres, salario bruto, retención, subsidio y salario neto.

Análisis

Datos de entrada

- Código: COD
- Nombres: NOM
- Número de hijos: NH
- Salario por hora: SH
- Número de horas trabajadas al mes: NHT

Datos de salida

- Código

- Nombres
- Salario bruto: SBR
- Deducción: DEDUC
- Subsidio: SU
- Salario neto: SNETO

Proceso
Se usará el subprograma CAL_SALARIO, para obtener por cada trabajador: salario bruto, retención, subsidio y salario neto.

Como no se conoce la cantidad de empleados de la empresa, se debe implementar un esquema cualitativo usando el código del empleado.

Cada vez que se encuentre un código diferente de cero, se entrará la información del empleado y se invocará el subprograma CAL_SALARIO para obtener los resultados pedidos.

Algoritmo

```
INICIO
    LEA: COD
    MIENTRAS COD <> 0 HAGA
            LEA: NOM, NH, SH, NHT
            CAL_SALARIO (NHT, SH, NH, SBR, DEDUC, SU, SNETO)
            ESCRIBA: COD, NOM, SBR, DEDUC, SU, SNETO
            ESCRIBA: "ENTRE CODIGO O CERO PARA TERMINAR"
            LEA: COD
    FIN_MIENTRAS
FIN_INICIO
```

Ejercicio resuelto No. 48

Elaborar un subprograma que encuentre el factorial de un número entero mayor o igual a cero.

Análisis

Parámetros de recibo
- El número: N

Parámetros de envío
- El factorial del número: FACTO

Proceso
Clase del subprograma: FUNCIÓN
Nombre: FAC

El factorial de N se define como 1: si N es igual a cero, o 1*2*3*...N-1*N si N es mayor que cero. El proceso usado será el mismo que se implementa en el ejercicio resuelto numero 33.

Subalgoritmo

```
FUNCIÓN FAC (N)
    FACTO=1
    PARA I= 1, N, 1 HAGA
            FACTO= FACTO*I
    FIN_PARA
```

```
    RETORNE FACTO
FIN_FAC
```

Ejercicio resuelto No. 49

Elaborar un algoritmo que utilice el subprograma FAC para encontrar el factorial de un número.

Análisis

Datos de entrada

- El número a calcularle el factorial: NUM

Datos de salida

- El factorial del número: FACTOR

Algoritmo

```
INICIO
    LEA: NUM
    FACTOR = FAC (NUM)
    ESCRIBA: "EL FACTORIAL DE", NUM, "ES:", FACTOR
FIN_INICIO
```

Ejercicio resuelto No.50

Usando la función FAC, elaborar un algoritmo que encuentre el factorial de los primeros N números naturales.

Análisis

Datos de entrada

- La cantidad de números a calcularle el factorial: N

Datos de salida

- Cada uno de los factoriales de los números comprendidos entre 1 y N.

Algoritmo

```
INICIO
    LEA: N
    PARA I = 1, N, 1
            ESCRIBA: "EL FACTORIAL DE", I, "ES:", FAC (I)
    FIN_PARA
FIN_INICIO
```

Ejercicio resuelto No. 51

Elaborar un subprograma que determine si un número es primo o no.

Análisis

Parámetros de recibo

- El número a evaluar si es primo o no: N

Parámetros de envío

- Una respuesta que diga si el número es primo o no: RES (Almacenara S, si el número es primo o N si no lo es).

Proceso

Clase de subprograma: FUNCIÓN

Nombre: PRIMO

Un número es primo si sólo es divisible por la unidad y por sí mismo. Lo anterior implica chequear si el número N tiene o no divisores, fuera de 1 y N; esos posibles divisores estarán entre 2 y N/2, por eso no se chequeara si N es divisible por un número mayor a N/2.

Para darnos cuenta si N es divisible por un número cualquiera k, usaremos el operador módulo. Este operador devuelve el residuo de una división entera. Por ejemplo 11 módulo 3 = 2, que resulta de dividir 11 por 3 y sobran 2. 8 módulo 2 = 0. Quiere decir entonces, que si un número cualquiera N es divisible por k, entonces al dividir N por k el residuo es cero. Si se encuentra que N sí es divisible por cualquier número entre 2 y la mitad de N, quiere decir que N no es primo, por lo tanto, se romperá el ciclo.

Variables locales a la función

MITAD: Parte entera de dividir N por 2.
I: Variable de control del ciclo, tomará valores entre 2 y MITAD.

Subalgoritmo

```
FUNCION PRIMO (N)
    MITAD = N/2
    I = 2
    MIENTRAS (I <= MITAD) ∧ (N MOD I <> 0)
            I = I + 1
    FIN_MIENTRAS
    SI I > MITAD ENTONCES
            RES = 'S'
      SINO
            RES = 'N'
    FIN_SI
    RETORNE RES
FIN_PRIMO
```

Ejercicio resuelto No. 52

Elaborar un algoritmo que mediante la función PRIMO, determine si un número es primo o no.

Análisis

Datos de entrada

- El número a evaluar: NUM

Datos de salida

- Un mensaje que diga si el número chequeado es primo o no.

Algoritmo

```
INICIO
    LEA: NUM
    SI PRIMO (NUM) = 'S' ENTONCES
            ESCRIBA: NUM, "ES PRIMO"
      SINO
            ESCRIBA: NUM, "NO ES PRIMO"
    FIN_SI
FIN_INICIO
```

Ejercicio resuelto No. 53

Usando la función PRIMO, elaborar un algoritmo que encuentre los primeros N números primos.

Análisis

Datos de entrada

- La cantidad de números primos a buscar: CNP

Datos de salida

- Los N primeros números primos.

Proceso

Mediante un ciclo que tome valores entre uno y CNP, se chequearan los números naturales en forma consecutiva de 1 en adelante. La variable de control de este ciclo sólo se incrementara cuando al llamar a la función PRIMO, la respuesta sea S, pero el contador que esté generando los números de 1 en adelante se incrementará sea cual fuere el valor que retorne la función PRIMO.

Variables

N: Contador que genera los números de uno en adelante.
I: Variable de control del ciclo.

Algoritmo

```
INICIO
    LEA: CNP
    N, I = 1
    MIENTRAS I<=CNP HAGA
            SI PRIMO (N) = 'S' ENTONCES
                    ESCRIBA: N, "ES EL PRIMO", I
                    I = I + 1
            FIN_SI
            N = N + 1
    FIN_MIENTRAS
FIN_INICIO
```

Aspectos a tener en cuenta

a. Los subprogramas deben ser lo más simples que se pueda; deben representar tareas que sólo se ejecutan una vez, o posiblemente en tareas repetitivas que son de uso muy frecuente dentro de la programación.
b. Un programa se simplifica mediante su división en módulos que se pueden considerar independientemente, lo que más interesa es qué hace el módulo y cómo lo hace.
c. Los subprogramas, como representación de módulos, tienen todas las características de un algoritmo.
d. Los subprogramas no sólo son útiles para evitar la repetición de instrucciones dentro de un algoritmo, sino que son aplicables en la implementación de la programación modular. Esta se basa en la descomposición descendente (arriba - abajo) o jerarquía de módulos, que se enlazan mediante un módulo principal o raíz.
e. Los módulos realizan siempre una tarea determinada y constan de un conjunto de instrucciones y un nombre por el cual son llamados o invocados desde un algoritmo u otro subprograma.
f. Cuando se invoca un procedimiento, el control de ejecución es transferido a la instrucción del nombre del subprograma, y cuando retorna devuelve el control a la instrucción siguiente

del módulo que lo activó y si es una función el control retorna a la misma instrucción que la invoca.

g. Los subprogramas, por lo general, deben ser declarados antes del algoritmo que lo invoca.
h. Los argumentos deben ser iguales en número y en tipo de datos a los parámetros.
i. Los subprogramas pueden ser construidos sin parámetros, lo cual es una práctica poco recomendable. Si esto sucede, se dice que trabajan con variables globales que son aquéllas definidas en el módulo que invoca al subprograma.
j. Los parámetros de un subprograma no siempre sirven en forma exclusiva para recibir y entregar información. En muchos casos los mismos parámetros cumplen ambos propósitos; como parámetros de recibo transfieren información hacia el módulo invocante.

7.3. Ejercicios propuestos

Elaborar un subprograma y su documentación para los siguientes 10 ejercicios.

127. Encontrar X^y por multiplicaciones sucesivas.
128. Obtener X^y utilizando sumas únicamente, donde X puede ser un número real y Y positivo, cero o negativo.
129. Ordenar tres caracteres en orden ascendente.
130. Determinar si un número es perfecto o no. Un número N es perfecto si la suma de todos sus divisores, excepto por el mismo, da N.
131. Convertir un número decimal a cualquier base r.
132. Encontrar el valor de e^x para una valor de x dado, mediante la serie: $1+X+\frac{X^2}{2!}+\frac{X^3}{3!}+\cdots$ teniendo en cuenta los términos que sean mayores que 10^{-3}
133. Encontrar un término de la sucesión de Fibonacci.
134. Encontrar el perímetro y área de un triángulo, cuando sólo se conocen sus lados.
135. Encontrar la suma y el promedio de los valores enteros comprendidos entre N y M, donde N<M
136. Encontrar los divisores de N.

Para los siguientes 10 ejercicios elaborar el subprograma y el algoritmo que los activa.

137. Encontrar las primeros N números perfectos.
138. Obtener los primeros 40 múltiplos de N
139. Encontrar A^B y luego obtener:
    (((A**B) **C) **D) **E
140. Dibujar un triángulo equilátero de lado L, con asteriscos.
141. Chequear cuántos de N puntos del plano cartesiano están en el cuadrante 1, 2, 3, y 4.
142. Obtener el interés generado por X pesos que se invierte a p porciento mensual en un período de N meses.
143. Encontrar la hora a los 40 segundos siguientes de la hora actual.
144. Obtener el promedio crédito de un estudiante que cursa N materias, con número de créditos distintos.
145. Determinar si con 4 valores que representan patas de mesas se puede construir una mesa de 4 patas no coja o de 3 patas no coja, en caso de que no se pueda construir una de 4 patas no coja.
146. Encontrar su edad en años, meses y días que tendrá dentro de N años.

Capítulo 8

ARREGLOS

En los capítulos anteriores se ha manejado el concepto de campo variable como uno o más caracteres, a los que se le asigna un espacio en memoria donde se puede guardar en forma temporal un único valor, bien sea numérico, carácter o lógico. En este capítulo el concepto de variable se extenderá a un grupo o colección de ellas, agrupadas bajo un mismo nombre.

Frecuentemente se manejan conceptos relacionados como una colección de datos. Por ejemplo, en las montañas de Antioquia vemos un conjunto de acémilas manejadas por un arriero y no importa conocer el nombre, sexo, o color de cada una de ellas para mencionarlas: simplemente decimos mulas.

Los datos siempre que estén relacionados se pueden organizar en estructuras, de tal manera que podemos tener un conjunto de datos numéricos, lógicos, o caracteres manejados a través de un mismo nombre de variable. Una estructura de datos es la organización que reciben los datos para que sean tratados como una unidad.

Existen varias formas de organizar los datos o valores que maneja un algoritmo, los más comunes son los arreglos y los archivos de datos.

Un arreglo es un conjunto finito de componentes del mismo tipo, los cuales se diferencian o relacionan a través de un subíndice.

Todas las componentes pertenecientes, llamadas también *elementos* del arreglo, están una a continuación de otra, tienen el mismo tamaño o espacio en memoria, son todas de un mismo tipo de dato y, por lo tanto, tienen igual forma de almacenamiento.

Para manejar en forma independiente cada componente del arreglo se usa un *índice*, que es una expresión de tipo entero (sin decimales) que indica cuál de los elementos del arreglo queremos relacionar. Siempre que se quiere trabajar o mencionar una componente es necesario adicionar el índice al nombre genérico del arreglo; es éste el que hace independiente una de otra.

Ejemplos de índices:

1 I IND k + I I * jota - 1

Clasificación de los arreglos

8.1. Arreglos de una dimensión o vectores

Un *vector* es un arreglo donde los elementos que lo conforman están dispuestos bajo un mismo concepto de clasificación (fila o columna), es decir, los datos están organizados de una manera lineal, por lo que para referenciar un elemento del arreglo es necesario un índice, que indique la posición relativa del elemento en el arreglo.

Cuando al nombre del arreglo se le adiciona el índice, bien sea entre paréntesis o corchetes, la componente al cual hace referencia el índice es tomada como una variable simple, igual a las

que en capítulos anteriores se han venido tratando; por lo tanto, puede estar involucrada en una expresión, en una entrada o salida de datos, o en una asignación.

Gráficamente un vector puede ser representado en cualquiera de las siguientes formas:

<NOMBRE>

Como fila **FALTA GRÁFICO PAG. 202 LIBRO**

Como columna <NOMBRE>

FALTA GRÁFICO PAG. 202 LIBRO

Donde el nombre es el que se ha escogido para todos los elementos del arreglo, y cada rectángulo representa las componentes o elementos que lo conforman.

Si los valores 30, 15, 20, 60 y 80 representan las edades de cinco personas y queremos tener almacenada esta información en memoria para ser utilizada en un proceso por computadora, perfectamente podemos guardarla en cinco nombres de variables distintas como EDAD1, EDAD2, EDAD3, EDAD4 y EDAD5; pero ¿qué pasa si no son cinco sino mil edades, o más de mil?, sin duda que se convierte en un proceso engorroso y difícil de manejar, pero que se puede solucionar almacenando esa información en un arreglo, que en este caso sería de una dimensión. Si se escoge el nombre EDAD como nombre del arreglo para identificar el conjunto de las 5 edades, internamente en memoria se separan cinco espacios (direcciones de memoria) adyacentes para almacenar en el momento oportuno la información requerida. Si un programa utiliza la variable tipo arreglo EDAD, después de la compilación sucederá lo siguiente:

	MEMORIA	
EDAD[1]	<BASURA>	Dirección X
EDAD[2]	<BASURA>	Dirección X + 1
EDAD[3]	<BASURA>	Dirección X + 2
EDAD[4]	<BASURA>	Dirección X + 3
EDAD[5]	<BASURA>	Dirección X + 4

Donde cada elemento del vector EDAD se puede procesar como si fuese una variable simple, al ocupar cada uno una posición de memoria diferente; esto implica que en los arreglos el acceso a un elemento se hace en forma directa, es decir, que cualquier elemento se puede referenciar sin necesidad de accesar los anteriores a él.

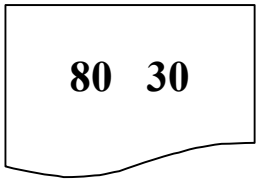
Las siguientes dos instrucciones muestran la independencia que tiene cada una de las variables que pertenecen al arreglo, al adicionar el índice.

```
LEA: EDAD[1], EDAD[3], EDAD[5]
ESCRIBA: EDAD[5], EDAD[1]
```

Si los valores de las tres variables descritas en la instrucción de la lectura son en su orden 30, 20 y 80, al ejecutarse las instrucciones anteriores sucederá lo siguiente:

	MEMORIA	
EDAD[1]	30	Dirección X
EDAD[2]	<BASURA>	Dirección X + 1
EDAD[3]	20	Dirección X + 2
EDAD[4]	<BASURA>	Dirección X + 3
EDAD[5]	80	Dirección X + 4

Salida



En general se puede decir que EDAD[l] hace referencia al contenido que tiene la componente, o elemento ubicado en la posición i del arreglo EDAD, donde $1 \leq i \leq 5$.

Los arreglos pueden ser utilizados como acumuladores o contadores, inicializando cada uno de sus elementos. Por ejemplo, llevar cero a cada uno de los elementos del arreglo EDAD.

Solución utilizando MIENTRAS

```
INICIO
    INDICE = 1
    MIENTRAS INDICE <= 5 HAGA
            EDAD [INDICE] = 0
            INDICE = INDICE + 1
    FIN_MIENTRAS
FIN_INICIO
```

Solución utilizando PARA

```
INICIO
    PARA INDICE = 1, 5, 1  HAGA
            EDAD [INDICE] = 0
    FIN_PARA
FIN_INICIO
```

Solución utilizando REPETIR

```
INICIO
    INDICE = 1
            HAGA
                    EDAD [INDICE] = 0
                    INDICE = INDICE + 1
            MIENTRAS INDICE <= 5
FIN_INICIO
```

Ejercicio resuelto No. 54

Hacer un algoritmo que forme un vector de N elementos, cuyos valores son numéricos que entran de a uno por registro, y luego obtenga la suma y el promedio de los elementos ubicados en las posiciones impares del vector.

Análisis

Datos de entrada

- La cantidad de valores que conformarán el vector.
- Cada uno de los elementos del vector.

Datos de salida

- Suma de los elementos impares.
- Promedio de los elementos impares.

Proceso

Como se conoce la cantidad de valores que se almacenarán en el vector (número de elementos), se puede implementar un esquema cuantitativo. Se hará uso de la estructura PARA en el proceso de entrada y la estructura MIENTRAS para el proceso de cálculo de la suma y el promedio; esto sólo por ilustrar su empleo, ya que indistintamente puede implementarse cualquiera de ellas para los dos procesos. Recuerde: todo lo que hacen las estructuras PARA y REPETIR puede hacerse con la estructura MIENTRAS. Para obtener los elementos impares se inicializa el contador en 1 y se incrementa de dos en dos, hasta que tome un valor que haga falsa la expresión lógica.

Definición de variables

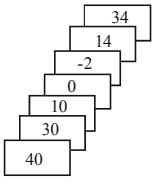
N: Número de elementos del arreglo.
K: Variable de tipo contador, utilizada como controladora de los ciclos PARA y MIENTRAS.
V: Nombre del vector.
SUMA: Sumatoria de los elementos impares del arreglo V.
PROM: Promedio de los elementos impares.
NEI: Número de elementos impares del arreglo

Algoritmo

```
INICIO
    LEA: N                                  (1)
    PARA K = 1, N, 1
        LEA: V[K]                           (2)
    FIN_PARA
    K = 1                                   (3)
    SUMA, NEI = 0                           (4)
    MIENTRAS K <= N HAGA
        SUMA = SUMA + V[K]                      (5)
        NEI = NEI + 1                       (6)
        K = K + 2                           (7)
    FIN_MIENTRAS
    PROM = SUMA / NEI                       (8)
    ESCRIBA: "LA SUMA ES:", SUMA,           (9)
    "Y EL PROMEDIO:", PROM
FIN_INICIO
```

Prueba de escritorio

Miremos el algoritmo, paso a paso, suponiendo un N igual a 7 y los siguientes valores para cada uno de los elementos del arreglo V.



Se ejecuta la instrucción 1, N
7

Se activa el ciclo PARA, K. Compara: es 1 ≤ 7: sí,

1

Se ejecuta la instrucción 2, LEA: V[1],

V

40						
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]

Se incrementa K (K = 1+1) K. Compara: es 2 ≤ 7: sí,
2

Se ejecuta la instrucción 2, LEA: V[2],

V

40	30					
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]

Se incrementa K (K = 2+1); K. Compara: es 3 ≤ 7: sí,
3

Se ejecuta la instrucción 2, LEA: V[3],

V

40	30	10				
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]

Se incrementa K (K = 3+1); K. Compara: es 4 ≤ 7: sí,
4

Se ejecuta la instrucción 2, LEA: V[4],

V

40	30	10	0			
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]

Se incrementa K (K = 4+1); K. Compara: es 5 ≤ 7: sí,
5

Se ejecuta la instrucción 2, LEA: V[5],

V

40	30	10	0	-2		
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]

Se incrementa K (K = 5+1); K. Compara: es 6 ≤ 7: sí,
6

Se ejecuta la instrucción 2, LEA: V[6],

V

40	30	10	0	-2	14	
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]

Se incrementa K (K = 6+1) K. Compara: es 7 ≤ 7: sí,
7

Se ejecuta la instrucción 2, LEA: V[7],

V

40	30	10	0	-2	14	34
V[1]	V[2]	V[3]	V[4]	V[5]	V[6]	V[7]

Se incrementa K (K = 7+1); K. Compara: es 8 ≤ 7: no,
8

Sale del ciclo PARA y quedan almacenados en memoria los 7 valores de los elementos del vector V.

Se ejecuta la instrucción 3, K
1

Se ejecuta la instrucción 4, SUMA NEI
0 0

Activa el ciclo MIENTRAS.

Se compara: ES 1 ≤ 7: sí, se ejecutan las instrucciones 5, 6 y 7

(5) se lleva SUMA + V[1] a SUMA (0 + 40) SUMA
40

(6) se lleva NEI + 1 a NEI, (0 + 1) NEI
1

(7) se lleva K + 2 a K, (1 + 2) K
3

Se compara: ES 3 ≤ 7: sí, se ejecutan las instrucciones 5, 6 y 7

(5) se lleva SUMA + V[3] a SUMA (40 + 10) SUMA
50

(6) se lleva NEI + 1 a NEI, (1 + 1) NEI
2

(7) se lleva K + 2 a K, (3 + 2) K
5

Se compara: ES 5 ≤ 7: sí, se ejecutan las instrucciones 5, 6 y 7

(5) se lleva SUMA + V[5] a SUMA (50 + (-2)) SUMA
48

(6) se lleva NEI + 1 a NEI, (2 + 1) NEI
3

(7) se lleva K + 2 a K, (5 + 2) K
7

Se compara: ES 7 ≤ 7: sí, se ejecutan las instrucciones 5, 6 y 7

(5) se lleva SUMA + V[7] a SUMA (48 + 34) SUMA
82

(6) se lleva NEI + 1 a NEI, (3 + 1) NEI
4

(7) se lleva K + 2 a K, (7 + 2) K
9

Se compara: ES 9 ≤ 7: no, se sale del ciclo MIENTRAS y continúa hacia abajo.

Se ejecuta la instrucción 8, se lleva a PROM 82/4 PROM
20.5

Se ejecuta la instrucción 9 y la salida será:

LA SUMA ES 82 Y EL PROMEDIO: 20.5

Ejercicio resuelto No. 55

Hacer un algoritmo que calcule el valor promedio, la varianza y la desviación típica de un grupo de datos positivos, dispuestos de a uno por registro.

Análisis

Datos de entrada

- Cada uno de los datos del grupo.

Datos de salida

- El promedio.
- La varianza.
- La desviación típica.

Proceso

Analizando el enunciado vemos que no se dice cuántos son los datos que hay en el grupo, por lo tanto es necesario implementar un esquema cualitativo y contar los valores del grupo, ya que es un valor necesario para poder calcular los datos de salida. Como los datos de entrada son positivos se hará un ciclo MIENTRAS que repita la lectura y proceso de cada dato, siempre que el valor leído sea mayor que cero. No se podrá utilizar un ciclo PARA ya que se desconoce el límite final, y la cantidad de valores del grupo.

El promedio es igual a $\sum_{i=1}^{N} X_i/N$

Donde Xi son cada uno de los valores y N la cantidad de éstos. Por lo tanto es necesario, primero, calcular la suma de los valores y, como ya se dijo, la cantidad de éstos.

La varianza es igual a $\sum_{i=1}^{N} (X_i - \text{promedio})^2 / (N-1)$

Como se ve en la fórmula, para calcular la varianza se obtiene previamente el promedio, para lo cual es necesario sumar cada uno de los datos. Para el cálculo de la varianza también se necesitan los datos del grupo. Para no volverlos a entrar, éstos se guardarán en un arreglo lineal cuando se esté calculando el promedio, de tal manera que cuando se calcule la varianza éstos están en memoria y no haya necesidad de leerlos sino de utilizarlos directamente. Al formar el vector conocemos la cantidad de valores; luego para utilizar los elementos de éste, basta con implementar un ciclo desde 1 hasta el tamaño del vector, que en este caso puede ser un ciclo PARA.

Cuando se forme el vector hay que tener cuidado, para que el último valor leído, cero en este caso, no haga parte del arreglo, ya que no es un valor del grupo; esto se puede hacer de dos formas: disminuyendo en uno el tamaño del arreglo si éste se encuentra directamente en la instrucción de lectura, o utilizando una variable que almacene cada dato leído y luego pasando su contenido al vector y que cuando tome el valor cero termine el ciclo de entrada.

Definición de variables

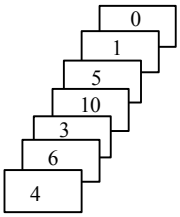
DATO: Cada uno de los valores del grupo.
N: Contador de valores e índice del vector, que al final del ciclo de entrada de datos tendrá almacenado el tamaño del vector.
X: Nombre del vector.
SUMA1: Sumatoria de los valores para el cálculo del promedio.
PROM: Promedio del grupo de datos.
I: Variable de control del ciclo PARA.
SUMA2: Sumatoria para calcular la varianza. Puede utilizarse la misma variable SUMA1?.
VAR: Varianza del grupo de valores.
D T: Desviación típica.

Algoritmo

```
INICIO
    N, SUMA1, SUMA2 = 0
    LEA: DATO
    MIENTRAS DATO <> 0 HAGA
            SUMA1 = SUMA1 + DATO
            N = N +1
            X[N] = DATO
            LEA: DATO
    FIN_MIENTRAS
    PROM = SUMA1 / N
    PARA I = 1, N, 1 HAGA
            SUMA2 = SUMA2 + (X[I] - PROM)**2
    FIN_PARA
    VAR = SUMA2 / (N -1)
    DT = √VAR
    ESCRIBA: "PROMEDIO =", PROM, "VARIANZA =", VAR,
                        "DESVIACION TIPICA =", DT
FIN_INICIO
```

Prueba de escritorio

La prueba se hará teniendo en cuenta el siguiente archivo de datos:



Registro centinela

N	SUMA1	SUMA2	DATO	I	X					PROM	VAR	DT	
					X[1]	X[2]	X[3]	X[4]	X[5]	X[6]			
~~0~~	~~0~~	~~0~~	~~4~~	~~1~~	~~4~~	6	3	10	5	1	4.83	9.37	3.06
~~1~~	~~4~~	~~0.6889~~	~~6~~	~~2~~									
~~2~~	~~10~~	~~2.0578~~	~~3~~	~~3~~									
~~3~~	~~13~~	~~5.4067~~	~~10~~	~~4~~									
~~4~~	~~23~~	~~32.1556~~	~~5~~	~~5~~									
~~5~~	~~28~~	~~32.1645~~	~~1~~	~~6~~									
6	29	46.8334	0	7									

Salida

PROMEDIO: 4.83 VARIANZA: 9.37 DESVIACIÓN TÍPICA: 3.06

Ejercicio resuelto No. 56

Elaborar un algoritmo que forme dos vectores A y B de N y M elementos respectivamente y forme un nuevo arreglo C, con los elementos que correspondan a la unión de A y B. La unión de A y B son los elementos del vector A que no están en el vector B. Los vectores no tienen elementos repetidos. A y B tienen máximo 20 elementos.

Análisis

Datos de entrada

- El tamaño del arreglo A
- El tamaño del arreglo B
- Los elementos del vector A
- Los elementos del vector B

Datos de salida

- Los elementos del vector C

Proceso

Como se conoce la cantidad de elementos de cada uno de los dos vectores, para su lectura se utiliza un esquema cuantitativo. Luego se pasan los elementos del vector A al nuevo vector C y se hace un ciclo externo que recorra los elementos del vector B y uno interno que recorra los elementos del vector A, buscando cada elemento de B en A, si un elemento de B no se encuentra en A entonces se pasa al vector C.

Definición de variables.

N: Número de elementos del vector A.
M: Número de elementos del vector B.
A, B y C: Nombre de los tres vectores.
NEC: Número de elementos del vector C.
I, K: Variables de control de ciclos.

Algoritmo

```
INICIO
    LEA: N, M
    PARA I = 1, N, 1 HAGA
        LEA: A[I]
    FIN_PARA
    PARA K = 1, M, 1  HAGA
        LEA: B[K]
    FIN_PARA
    PARA I = 1, N HAGA
        C[I] = A[I]
    FIN_PARA
        NEC = N
        PARA K = 1, M, 1  HAGA
        I = 1
        MIENTRAS (I <= N) ∧ (B[K] <> A[I]) HAGA
                I = I + 1
        FIN_MIENTRAS
        SI I > N ENTONCES
            NEC = NEC + 1
                C[NEC] = B[K]
        FIN_SI
        FIN_PARA
    PARA I = 1, NEC, 1 HAGA
        ESCRIBA: C[I]
    FIN_PARA
FIN_INICIO
```

Ejercicio resuelto No. 57

Elaborar un subprograma que encuentre la posición del mayor elemento de un vector de tamaño N.

Análisis

Parámetros de recibo

- El tamaño del vector.
- Los elementos del vector.

Parámetros de envío

- La posición que ocupa el mayor de los elementos del arreglo.

Proceso

Para encontrar la posición que ocupa el mayor elemento del vector es necesario recorrer todo el arreglo, por lo cual el subprograma se le debe enviar por parte del módulo que lo invoque el tamaño y todos los valores del vector; este envío se debe hacer a través del nombre del vector sin subíndice. En el caso de que el nombre del arreglo se le acompañe de un índice quiere decir que sólo es transferido el valor del elemento que relacione el índice.

Para encontrar esta posición se asume que el primer elemento es el mayor y se recorre el vector a partir de la posición 2 en adelante, cambiando el valor de la posición cada vez que se encuentre un elemento de mayor peso.

Clase de subprograma: FUNCIÓN

Nombre: POSMAY

Definición de variables

N: Tamaño del vector.
VEC: Nombre asignado al vector.
POSM: Lugar dentro del arreglo donde se encuentra el valor mayor.
MAYOR: Variable que almacena el valor mayor del arreglo
I: Variable de control de ciclo.

Subalgoritmo

```
FUNCIÓN POSMAY (N, VEC)
    MAYOR = VEC[1]
    POSM = 1
    PARA I = 2, N, 1 HAGA
        SI MAYOR < VEC[I] ENTONCES
            MAYOR = VEC[I]
        POSM = I
        FIN_SI
  FIN_PARA
  RETORNE  POSM
FIN_POSMAY
```

Ejercicio resuelto No. 58

Elaborar un algoritmo que invoque el subprograma anterior.

Análisis

Datos de entrada

- El tamaño del vector.
- El vector.

Datos de salida

- La posición que ocupa el mayor elemento del vector.

Proceso

Mediante un ciclo cuantitativo se lee el vector, se llama a la función POSMAY para que busque la posición del valor mayor y se imprime dicho valor.

Definición de variables

TV: Tamaño del vector.
A: Nombre del vector.
POSMAYOR: Posición que ocupa el valor mayor.
I: Variable de control del ciclo.

Algoritmo

```
INICIO
    LEA: TV
    PARA I = 1, TV, 1 HAGA
```

```
    LEA: A[I]
    FIN_PARA
    POSMAYOR = POSMAY (TV, A)
    ESCRIBA: "EL VALOR MAYOR ESTÁ EN LA POSICIÓN:",
FIN_INICIO
```

Ejercicio resuelto No. 59

Elaborar un subprograma que lea un vector de N elementos.

Análisis

Parámetros de recibo

- Tamaño del vector (debe ser proporcionado por el usuario): N

Parámetros de envío

- El vector leído: VEC

Proceso

Como se conoce el número de elementos del vector, se emplea un esquema cuantitativo.

Clase de subprograma: PROCEDIMIENTO (Envía más de un valor).

Nombre: LEER_VEC

Variables locales:

H: Variable de control del ciclo de entrada.

Subalgoritmo

```
PROCEDIMIENTO LEER_VEC (N, VEC)
    PARA H = 1, N, 1 HAGA
        LEA: VEC[H]
    FIN_PARA
FIN_LEER_VEC
```

Ejercicio resuelto No. 60

Elaborar un subprograma que imprima un vector de tamaño N.

Análisis

Parámetros de recibo

- El tamaño del vector: N
- El vector: VEC

Parámetros de envío

No hay. (Los valores no los devuelve a quien lo llama, sino que los envía a un medio externo de salida)

Proceso

Los elementos se muestran mediante un esquema cuantitativo.

Clase de subprograma: PROCEDIMIENTO

Nombre: IMPRIMIR_VEC

Variables locales

H: Variable de control del ciclo.

Subalgoritmo

```
PROCEDIMIENTO IMPRIMIR_VEC (N, VEC)
    PARA H = 1, N, 1 HAGA
        ESCRIBA: VEC[H]
    FIN_PARA
FIN_IMPRIMIR_VEC
```

8.1.1. Operaciones básicas con arreglos

- Búsqueda
- Ordenación
- Inserción
- Borrado

Búsqueda

La búsqueda consiste en encontrar un determinado valor dentro de un conjunto de datos, para recuperar alguna información asociada con el valor buscado.

Existen diferentes formas de hacer esta operación:

- Búsqueda Secuencial o lineal.
- Búsqueda Binaria.
- Búsqueda HASH.
- Árboles de búsqueda.

Búsqueda secuencial. Es una técnica sencilla para buscar un elemento dentro de un arreglo, y consiste en recorrer todo el vector elemento a elemento desde el primero hasta el ultimo. Si se encuentra el valor buscado, el proceso devolverá la posición donde se encuentra dicho elemento, de lo contrario se enviará un mensaje que informe que el valor buscado no se encuentra dentro del arreglo.

Ejercicio resuelto No. 61

Elaborar un subprograma que busque en forma secuencial un VALOR dentro de un arreglo de N elementos.

Análisis.

Parámetros de recibo.

- Tamaño del vector: N.
- El vector donde se hará la búsqueda: VEC
- El valor a buscar: VALORB.

Parámetros de envío.

- La posición donde se encuentra el valor buscado o cero si no se encuentra: I

Proceso.

Mediante un ciclo cuantitativo se buscará el valor dentro del vector elemento a elemento y si se da una coincidencia se rompe el ciclo.

Clase de subprograma: FUNCIÓN.

Nombre: BÚSQUEDASEC

Variables locales

I: Variable de control del ciclo de búsqueda, será el valor enviado por la función, si el valor buscado se encuentra.

Subalgoritmo

```
FUNCIÓN BUSQUEDASEC (N, VEC, VALORB)
     I = 1
    MIENTRAS(I<=N) ∧ (VALORB<> VEC[I]) HAGA
          I = I + 1
     FIN_MIENTRAS
    SI I <= N ENTONCES
     RETORNE  I
       SINO
     RETORNE 0
     FIN_SI
FIN_BÚSQUEDA_SEC
```

Ejercicio resuelto No. 62

Elaborar un algoritmo que use el subprograma anterior para determinar si se encuentra o no un determinado VALOR dentro de los elementos de un vector de tamaño N.

Análisis

Datos de entrada

- El tamaño del vector: N
- El vector: VEC
- El valor a buscar: VALORB.

Datos de salida

- Un mensaje que diga si el valor buscado se encuentra o no dentro del vector.

Proceso

Se usarán los subprogramas LEER_VEC Y BÚSQUEDASEC para leer el vector y efectuar la búsqueda del valor.

Definición de variables

I: Variable de control del ciclo.
POSICIÓN: Variable donde se recibe el valor devuelto por la función.

Algoritmo

```
INICIO
     LEA: N
     LEER_VEC (N, VEC)
     LEA: VALORB
     POSICION = BUSQUEDASEC (N, VEC, VALORB)
     SI POSICION <> 0 ENTONCES
          ESCRIBA: "EL VALOR", VALORB, "ESTA EN LA POSICION",
                              POSICIÓN, "DEL VECTOR"
       SINO
          ESCRIBA:  "EL VALOR ", VALORB, " NO SE ENCUENTRA EN
                    EL VECTOR"
     FIN_SI
FIN_INICIO
```

Búsqueda binaria. Note que en la búsqueda secuencial, cuando no se encuentra el valor buscado, en el caso peor, hay que recorrer todo el vector. Esto hace que el proceso no tenga gran eficiencia, sobre todo cuando el tamaño del vector es muy grande. El método de *búsqueda binaria*, consiste en dividir el vector en dos partes y comparar el valor buscado con el elemento que está en la mitad del arreglo, si no hay coincidencia entonces se determina si el valor buscado está en la primera mitad del arreglo o en la segunda mitad, eliminando con ello la búsqueda en cualquiera de las dos mitades. Se coge la mitad donde posiblemente se encuentra el valor buscado y se repite la misma comparación arriba mencionada. Con esta nueva comparación se elimina nuevamente la mitad de los datos que quedaron. El proceso continúa hasta que se dé una coincidencia, o se pueda decir que el valor buscado no está en el vector. Para poder hacer búsqueda binaria primero hay que ordenar el vector; si el vector no está ordenado, no se puede determinar en cuál de las mitades se encuentra el valor que se desea buscar.

Ejercicio resuelto No. 63

Elaborar un subprograma que haga búsqueda binaria de un VALOR en un vector previamente ordenado.

Análisis

Parámetros de recibo

- Tamaño del vector: N.
- El vector ordenado A.
- El valor a buscar dentro del arreglo: VALORBUSCADO.

Parámetro de envío

- La posición donde se encuentra el valor buscado o cero si no se encuentra.

Proceso

El arriba mencionado.

Clase de subprograma: FUNCIÓN.

Nombre: BUSQUEDABIN

Variables locales

P: Límite inicial de la sección del vector donde se hace la búsqueda.
U: Límite final de la sección del vector donde se hace la búsqueda.
SW: Bandera usada para romper el ciclo.
MITAD: Parte central del vector.

Subalgoritmo

```
FUNCION BUSQUEDABIN (N, A, VALORBUSCADO)
    P = 1
    U = N
    SW = 0
    MIENTRAS (P <= U) ∧ (SW =0) HAGA
            MITAD = [[(P + U) / 2]]     //Toma la parte entera de (P + U) / 2
            SI A[MITAD] = VALORBUSCADO ENTONCES
                    SW = 1
              SINO
                    SI VALORBUSCADO < A[MITAD] ENTONCES
                            U = MITAD - 1
```

```
                    SINO
                        P = MITAD + 1
                    FIN_SI
            FIN_SI
        FIN_MIENTRAS
        SI VALORBUSCADO = A[MITAD] ENTONCES
            RETORNE MITAD
          SINO
            RETORNE 0
        FIN_SI
    FIN_BUSQUEDA_BIN
```

Ordenación

La ordenación se refiere a la operación de organizar los elementos de un vector en algún orden dado: ascendente o descendente. Si el ordenamiento es creciente o ascendente, significa que hay que organizar sus elementos de tal manera que:

A[1] < A[2] < A[3] < A[4] .....< A[N].

Si el ordenamiento es ascendente significar que:

A[1] > A[2] > A[3] > A[4] .....> A[N].

Existen diferentes modelos para organizar los elementos de un arreglo:

Los métodos mas comunes son:

- MÉTODO DE BURBUJA.
- MÉTODO DE BURBUJA MEJORADO.
- ORDENACIÓN POR SELECCIÓN.
- INSERCIÓN O MÉTODO DE LA BARAJA.
- SHELL.
- BINSORT O POR URNAS.
- POR MONTICULOS O HEAPSORT.
- POR MEZCLA O MERGESORT.
- METODO DE LA SACUDIDA O SHACKERSORT.
- RAPID SORT O QUICK SORT.
- POR ÁRBOLES.

Vamos a desarrollar algunos de los principales métodos de organización de los elementos de un arreglo.

Ordenacion por burbuja. Es un método poco eficiente. Se basa en comparar elementos adyacentes del vector e intercambiar sus valores, si están desordenados. Los valores más pequeños burbujean hacia el inicio del vector (ordenación ascendente), mientras que los valores más grandes comienzan a subir hacia la parte final del vector. Por ejemplo, se compara A[1| con A[2]; si están desordenados se intercambian A[1] con A[2] y así sucesivamente a lo largo del vector. Al terminar la primera pasada, el elemento mayor se encuentra al final y el menor ha ido descendiendo hacia el principio del vector. Se vuelve a hacer una nueva exploración, comparando elementos consecutivos e intercambiando. Pero esta vez el elemento mayor no se compara (está en su posición correcta). El vector quedará ordenado cuando se hayan hecho N-1 pasadas. Si los valores del vector son:

80 55 35 21 15 VECTOR ORIGINAL DE TAMAÑO 5 (N)

55 80 35 21 15
55 35 80 21 15
55 35 21 80 15
55 35 21 15 80

Se necesitaron 4 comparaciones (N-1) para colocar el valor mayor de último.

En la segunda pasada ocurrirá lo siguiente:
35 55 21 15 80
35 21 55 15 80
35 21 15 55 80

Tercera pasada:
21 35 15 55 80
21 15 35 55 80

Cuarta pasada:
15 21 35 55 80

En total se necesitaron cuatro pasadas para ordenar un vector de 5 elementos.

Ejercicio resuelto No. 64

Elaborar un subprograma que ordene un vector por el método de la BURBUJA.

Análisis

Parámetros de recibo

- Tamaño del vector: N.
- El vector: A.

Parámetros de envío

- Vector A ordenado. (Será parámetro de recibo y de envío)

Proceso

El arriba explicado.

Clase de subprograma: PROCEDIMIENTO.

Nombre: ORDENBURBUJA

Variables locales

I, J: Variables de control de ciclos.

AUX: Variable que almacena temporalmente uno de los dos valores a intercambiar.

Subalgoritmo

```
PROCEDIMIENTO ORDENBURBUJA (N,  A)
    PARA I =1, N-1, 1 HAGA
        PARA J =1, N-I, 1 HAGA
            SI A[J] > A[J+1] ENTONCES
                AUX = A[J]
                A[J] = A[J+1]
                A[J+1] = AUX
            FIN_SI
        FIN_PARA
    FIN_PARA
FIN_ORDENBURBUJA
```

Ordenación por burbuja mejorado. Este método es eficiente cuando los valores del vector se encuentran ordenados o semiordenados. Se basa en el mismo proceso anterior, pero si en una pasada no ocurren intercambios quiere decir que el vector esta ordenado, por lo tanto hay que implementar un mecanismo para detener el proceso cuando ocurra una pasada sin intercambios.

Ejercicio resuelto No. 65

Elaborar un subprograma que ordene un vector por el método de burbuja mejorado.

Análisis

Parámetros de recibo.

- Tamaño del vector: TV.
- El vector: A.

Parámetros de envío

- Vector A ordenado. (Será parámetro de recibo y de envío)

Proceso

El subprograma se hace con la instrucción REPETIR y una variable tipo bandera que rompa el ciclo si no se detectan intercambios en una pasada.

Clase de subprograma: PROCEDIMIENTO

Nombre: ORDENBURBUJA_MEJORADO.

Variables locales

I, J: Variables de control de ciclos.
AUX: Variable que almacena temporalmente uno de los dos valores a intercambiar.
BAN: Variable tipo bandera que se usa para romper el ciclo.

Subalgoritmo

```
PROCEDIMIENTO ORDENBURBUJA_MEJORADO (TV,  A)
    I = 1
    REPITA
            BAN = 1
            PARA J = 1, TV - I, 1 HAGA
                    SI A[J] > A[J+1] ENTONCES
                            AUX = A[J]
                            A[J] = A[J+1]
                            A[J+1] = AUX
                            BAN = 0
                    FIN_SI
            FIN_PARA
            I = I + 1
    MIENTRAS BAN < > 1
FIN_ORDENBURBUJA_MEJORADO
```

Ordenamiento por selección. Este método se basa en buscar el menor valor de todo el vector e intercambiarlo con el elemento que se encuentra en la posición uno. Para encontrar este valor se asume que el menor de todo del grupo es el que está en la posición uno, y se recorre todo el vector a partir de la posición dos en adelante; cuando se encuentre un valor menor que él, se le cambia el valor a la variable donde se ha guardado el supuesto menor. Luego se busca el segundo menor asumiendo que es el elemento ubicado en la posición dos, y se recorre todo el vector a partir de la posición tres en adelante, haciendo el cambio respectivo cuando se

encuentre un valor menor que él. Una vez encontrado el segundo menor se intercambia con el valor ubicado en la posición dos del vector. El proceso de búsqueda de menores se hace N-1 veces.

Por cada menor buscado hay que guardar en una variable la posición donde se encuentra ese menor.

Ejemplo. Si el vector a ordenar es: 10 20 1 3 0 sucedería lo que se muestra en el siguiente gráfico.

PASO	ACCIÓN	A[1]	A[2]	A[3]	A[4]	A[5]
1	Se intercambia el primer menor con la posición uno	10*	20	1	3	0*
2	Se intercambia el segundo menor con la posición dos	0	20*	1*	3	10
3	Se intercambia el tercer menor con la posición tres	0	1	20*	3*	10
4	Se intercambia el cuarto menor con la posición cuatro	0	1	3	20*	10*
	Queda ordenado el vector	0	1	3	10	20

Note que después del cuarto intercambio el elemento ubicado en la posición cinco, de por sí queda ordenado; además no hay más elementos con quien compararlo se termina el proceso, después de hacer N-1 intercambios, o buscado N-1 menores.

Ejercicio resuelto No. 66

Elaborar un subprograma que ordene un vector por el método de selección.

Análisis

Parámetros de recibo.

- Tamaño del vector: N.
- El vector: A.

Parámetros de envío

- Vector A ordenado. (Será parámetro de recibo y de envío)

Proceso

Se necesitan dos estructuras tipo ciclo: un ciclo externo que busque los N-1 menores; esta búsqueda se hace utilizando una variable que almacene ese valor menor y otra que guarde la posición donde está ubicado. El otro ciclo interno busca los menores. Para la búsqueda del primer menor se supone que el valor ubicado en la posición uno es el menor, es decir, se asigna este valor a la variable que almacenará el menor y, se implementa el segundo ciclo o ciclo interno desde la posición siguiente donde se supone está el menor hasta N, y se empieza a comparar la variable que contiene el menor supuesto con los elementos siguientes, actualizando su contenido cada que se encuentre un valor inferior al valor que contiene la variable. Cuando se termina de recorrer el arreglo a través del ciclo interno, se tiene localizado el valor menor de todos y el lugar donde está ubicado; entonces se hace el intercambio con la posición uno. Luego se supone que el segundo menor es el ubicado en la posición dos, se compara éste con los elementos subsiguientes hasta encontrar este segundo menor y su posición y se intercambia con la posición dos, y así sucesivamente. Cada que se necesite hacer un intercambio se guarda el contenido del elemento que va a ser reemplazado por el menor en una variable auxiliar, para luego trasladarlo al lugar respectivo.

Clase de subprograma: PROCEDIMIENTO

Nombre: ORDENSELECCIÓN.

I, k: Variables de control de ciclos.

POS: Guarda la posición donde está el menor.

MENOR: Guarda el menor buscado.

AUX: Variable que almacena temporalmente uno de los dos valores a intercambiar.

Subalgoritmo

```
PROCEDIMIENTO ORDENSELECCION (N,  A)
    PARA I = 1, N-1, 1 HAGA
            MENOR = A[I]
            POS = I
            PARA K = I+1, N, 1
                    SI MENOR > A[K] ENTONCES
                            MENOR = A[K]
                            POS = K
                    FIN_SI
            FIN_PARA
            AUX = A[I]
            A[I] = MENOR
            A[POS] = AUX
    FIN_PARA
FIN_ORDENSELECCION
```

Ordenamiento por inserción directa (baraja). Este método se basa en la técnica utilizada por los jugadores de cartas. El jugador va ubicando cada carta en la posición correcta. El jugador empieza a ordenar sus cartas a partir de la segunda carta que le llega. Si tiene las cartas Q, J, las coloca como J, Q (hace el intercambio). La técnica usada para el ordenamiento es ir insertando cada uno de los elementos a partir del elemento dos en el lugar que le corresponde por su valor, lo que implica correr todos los valores de la derecha una posición para dejar espacio. El valor que se inserta se hace en el lugar que tenía el ultimo valor desplazado, y para no borrar su contenido se guarda en una variable auxiliar.

Ejercicio resuelto No. 67

Elaborar un subprograma que ordene un vector por el método de la baraja.

Análisis

Parámetros de recibo.

- Tamaño del vector: TV.
- El vector: A.

Parámetros de envío

- Vector A ordenado. (Será parámetro de recibo y de envío)

Proceso

El arriba mencionado.

Clase de subprograma: PROCEDIMIENTO

Nombre: ORDENBARAJA.

Variables locales

I, K: Variables de control de ciclos.

AUX: Variable que almacena temporalmente uno de los dos valores a intercambiar.
BAN: Variable tipo bandera que se usa para romper el ciclo.

Subalgoritmo

```
PROCEDIMIENTO ORDENBARAJA (TV,  A)
    PARA K = 2, TV, 1 HAGA
            AUX = A[K]
            I = K
            BAN = 0
            MIENTRAS (I > 1) ∧ (BAN = 0) HAGA
                    SI A[I-1] > AUX ENTONCES
                            A[I] = A[I-1]
                            I = I - 1
                      SINO
                            BAN = 1
                    FIN_SI
            FIN_MIENTRAS
            A[I] = AUX
    FIN_PARA
FIN_ORDEN_BARAJA
```

Inserción

Esta operación consiste en adicionar un nuevo elemento al arreglo. Se debe tener en cuenta que no se sobrepase el tamaño dado en la definición del arreglo. La operación puede darse con un arreglo desordenado u ordenado. Si el arreglo esta desordenado la operación es simple, se incrementa en uno el número de elementos y en esa posición (N+1), se inserta el nuevo elemento.

Ejercicio resuelto No. 68

Elaborar un subprograma que inserte un VALOR en un vector desordenado.

Análisis

Parámetros de recibo.

- Tamaño del vector: TV.
- El vector: A.
- El valor a insertar: VIN

Parámetros de envío

- El vector A con un elemento más.
- El tamaño del vector TV incrementado en uno.

Proceso

El arriba mencionado.
Clase de subprograma: PROCEDIMIENTO
Nombre: INSERTAR_DESORDENADO.

Variables locales

No tiene.

Subalgoritmo

```
PROCEDIMIENTO INSERTAR_DESORDENADO (TV,   A,   VIN)
    TV = TV+1
    A[TV] = VIN
FIN_INSERTAR_DESORDENADO.
```

Si el arreglo esta ordenado, hay que:

- Buscar el lugar dentro del arreglo donde debe insertarse el nuevo valor para que el arreglo continúe ordenado.
- Correr todos los elementos del vector una posición hacia la izquierda, para abrirle espacio al nuevo elemento, a partir del lugar donde debe insertarse el nuevo dato.
- Insertar el nuevo elemento del vector en el espacio que le corresponde.

Ejercicio resuelto No. 69

Elaborar un subprograma que inserte un nuevo elemento en un vector ordenado ascendentemente.

Análisis

Parámetros de recibo.

- Tamaño del vector: TV.
- El vector: A.
- El valor a insertar: VALIN

Parámetros de envío

- Vector A con un elemento más.
- Tamaño del vector TV incrementado en uno.

Proceso

El arriba mencionado.

Clase de subprograma: PROCEDIMIENTO

Nombre: INSERTAR_ORDENADO.

Variables locales

I: Variable de control del ciclo de búsqueda.

K: Variable que controla el ciclo que hace el desplazamiento de los elementos.

Subalgoritmo

```
PROCEDIMIENTO INSERTAR_ORDENADO (TV,   A,  VALIN)
    I = 1
    MIENTRAS (I <= TV) ∧ (VALIN > A[I]) HAGA
            I = I + 1
    FIN_MIENTRAS
    TV = TV + 1
    PARA K = TV, I + 1, -1 HAGA
            A[K] = A[K-1]
    FIN_PARA
    A[I] = VALIN
FIN_INSERTAR_ORDENADO
```

Borrado

Es eliminar un elemento del arreglo; puede darse cuando el arreglo esta desordenado u ordenado. Para hacer el proceso hay que:

- Verificar que el arreglo no esté vacío.
- Buscar la posición donde se encuentra el elemento a borrar.
- Correr los elementos una posición hacia la izquierda, a partir de la posición siguiente donde se encuentra el valor a borrar.
- Disminuir el número de elementos del vector en uno.
- Enviar un mensaje en caso de que el elemento a borrar no esté dentro del arreglo.

Ejercicio resuelto No. 70

Elaborar un subprograma que borre un elemento en un vector ordenado o desordenado.

Análisis

Parámetros de recibo.
- Tamaño del vector: TV.
- El vector: A.
- El valor a borrar: VALORB

Parámetros de envío
- Vector A con un elemento menos.
- Tamaño del vector TV disminuido en uno.

Proceso
El arriba mencionado.
Clase de subprograma: PROCEDIMIENTO
Nombre: BORRAR_ELEMENTO

Variables locales
I: Variable de control del ciclo de búsqueda.
K: Variable que controla el ciclo que hace el desplazamiento de los elementos.

Subalgoritmo

```
PROCEDIMIENTO BORRAR_ELEMENTO (TV,    A,   VALORB)
    I = 1
    MIENTRAS (I <= TV) ∧ (VALORB <> A[I] HAGA
            I = I + 1
    FIN_MIENTRAS
    SI I<= TV ENTONCES
            PARA K = I, TV-1, 1 HAGA
                    A[K] = A[K+1]
            FIN_PARA
            TV = TV - 1
      SINO
            ESCRIBA: VALORB, " NO ESTA EN EL ARREGLO "
    FIN_SI
FIN_BORRAR_ELEMENTO
```

Ejercicio Resuelto No. 71

Elaborar un algoritmo que lea un vector de N elementos con N<100 y construya un MENÚ que permita:

1. Leer un vector de tamaño N.
2. Imprimir el vector.

3. Insertar un valor en el vector.
4. Ordenar el vector en forma ascendente por cualquiera de los métodos de ordenación expuestos.
5. Insertar un nuevo elemento sin dañar el ordenamiento.
6. Buscar un valor en el arreglo por el método de búsqueda binaria.
7. Borrar un elemento del vector.

Análisis

Datos de entrada

- El tamaño del vector.
- Los elementos del vector.
- El valor a insertar cuando el vector está desordenado.
- El valor a insertar cuando el vector está ordenado.
- El valor a buscar.
- El valor a borrar.

Datos de salida

- El vector, cada vez que el usuario lo quiera ver.

Proceso

Todo lo pedido en el algoritmo ya se ha hecho en los subprogramas anteriores, de tal manera que se hará uso de ellos. (Todos están codificados en la librería LIBVEC.H)

Variables

N:	El tamaño del vector.
A:	El nombre del vector.
VIN1:	El primer valor a insertar.
VIN2:	El segundo valor a insertar.
VALBUS:	El valor a buscar.
VALBO:	El valor a borrar.
OPCIÓN:	Selección a tomar dentro del MENÚ.
OPCIÓN1:	Método de ordenamiento seleccionado.
RESP:	Respuesta que devuelve la función BUSQUEDABIN
SIGA:	Toma de decisión sí regresa o no al MENÚ principal.

Algoritmo

```
INICIO
    LEA: N
    REPITA
        ESCRIBA: "1: LEER EL VECTOR"
        ESCRIBA: "2: IMPRIMIR EL VECTOR"
        ESCRIBA: "3: INSERTAR DESORDENADO"
        ESCRIBA: "4: ORDENAR EL VECTOR"
        ESCRIBA: "5: INSERTAR ORDENADO"
        ESCRIBA: "6: BUSCAR UN VALOR"
        ESCRIBA: "7: BORRAR UN ELEMENTO"
        ESCRIBA: "8: SALIR"
        ESCRIBA: "ELIJA SU OPCIÓN"
        LEA: OPCIÓN
        CASOS DE OPCIÓN:
            CASO 1: LEER_VEC (N, A)
            CASO 2: IMPRIMIR_VEC (N, A)
            CASO 3: LEA: VIN1
                    INSERTAR_DESORDENADO(N,A,VIN1)
            CASO 4:
```

```
                    REPITA
                        ESCRIBA: "1: BURBUJA"
                        ESCRIBA:"2: BURBUJA_MEJORADO"
                        ESCRIBA: "3: SELECCIÓN"
                        ESCRIBA: "4: BARAJA"
                        ESCRIBA: "5: VOLVER AL MENÚ PPAL"
                        ESCRIBA: "ELIJA MÉTODO DE
                                ORDENAMIENTO"
                        LEA: OPCIÓN1
                            CASOS DE OPCION1
                                CASO 1:
                                (N, A)
                                CASO 2:
                                MEJORADO (N, A)
                                CASO 3:
                                CASO 4:
                                (N, A)
                                CASO 5: ESCRIBA:
                                                PRINCIPAL"
                            FIN_CASOS
                        MIENTRAS OPCIÓN1 <> 5
            CASO 5: LEA VIN2
                        INSERTAR_ORDENADO (N, A, VIN2)
            CASO 6: LEA: VALBUS
                            RESP = BÚSQUEDABIN
                            (N, A, BALBUS)
                            SI RESP <> 0 ENTONCES
                                ESCRIBA:VALBUS,"
                                RESP, "DEL VECTOR"
                              SINO
                                ESCRIBA: VALBUS."
                            FIN_SI
            CASO 7: LEA: VALBO
                            BORRAR_ELEMENTO(N.A,VALBO)
            CASO 8: ESCRIBA: "ESTÁ SEGURO QUE DESEA
                            LEA: SIGA
                            SI SIGA = 'N' ENTONCES
                                OPCIÓN = 1
                            FIN_SI
        FIN_CASOS
    MIENTRAS OPCIÓN <> 8
FIN_INICIO
```

Ejericicio resuelto No. 72

Diseñar un subprograma que dado un vector que pueda contener elementos duplicados diferentes de cero, le reemplace cada elemento repetido por un cero y encuentre el número de modificaciones realizados.

Análisis

Parámetros de recibo

- El tamaño del vector.
- El vector con elementos repetidos.

Parámetros de envío

- El vector con los elementos repetidos reemplazados por ceros, será un parámetro tanto de entrada como de salida.
- El número de modificaciones hechas.

Proceso

Mediante un ciclo externo se recorre el arreglo desde la posición uno hasta la N-1 y se compara el elemento uno con las posiciones comprendidas entre dos y N. Si algún elemento entre dos y N es igual al elemento uno, se cambia su valor por cero y se incrementa el número de modificaciones hechas; lo mismo se hace desde el elemento dos hasta el elemento N-1 y así sucesivamente.

Clase de subprograma: PROCEDIMIENTO
Nombre: CAMBIO

Variables locales

N:	Tamaño del vector.
VEC:	Nombre del vector.
NM:	Número de modificaciones hechas.
K, L:	Variables controladoras de los ciclos.

Subprograma

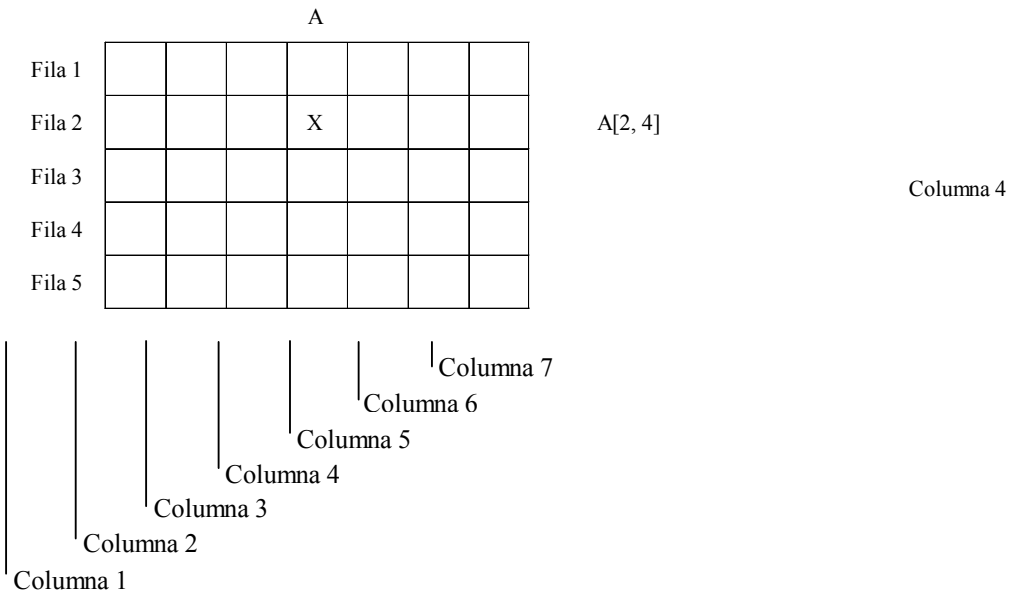
```
PROCEDIMIENTO CAMBIO (N, VEC, NM)
    NM = 0
    PARA K = 1, N-1, 1 HAGA
        PARA L = K+1, N, 1 HAGA
            SI (VEC[K] = VEC[L]) ∧ (VEC[K] <> 0)
                VEC[L] = 0
                NM = NM+1
            FIN_SI
        FIN_PARA
    FIN_PARA
FIN_CAMBIO
```

8.2. Arreglos de dos dimensiones o matrices

Las matrices son estructuras de datos que organizan su información en forma de tablas; es decir, los elementos que la conforman están dispuestos bajo dos conceptos de clasificación (*fila y columna*). Para poder indicar el lugar donde se encuentra un determinado elemento, es necesario utilizar dos índices: uno para indicar el renglón o fila y otro para indicar la columna.

Puede mirarse una matriz como un vector de vectores; por lo tanto, es un conjunto de componentes en el que se necesitan dos subíndices para identificar un elemento que pertenezca al arreglo.

Un arreglo bidimensional N * M tiene N filas y M columnas; por lo tanto, tiene N * M elementos dispuestos interiormente en memoria en forma sucesiva. Gráficamente se puede representar como una tabla de valores. Si A es una matriz de 5 * 7 se representará así:

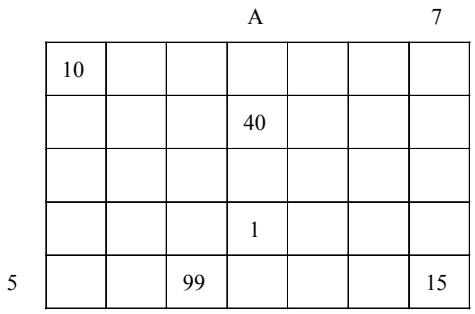


El arreglo anterior tiene 35 elementos, todos ellos con el nombre A y se diferencian entre sí por dos índices; el primero indica la fila donde está ubicado el elemento y el segundo la columna. El elemento A[2, 4] hace referencia al contenido del elemento ubicado en la fila 2 columna 4.

En general, se puede decir que el elemento i, j de un arreglo bidimensional hace referencia al contenido del elemento ubicado en la fila i, columna j, donde:

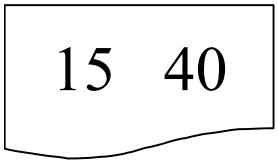
$1 \leq i \leq$ número de filas y $1 \leq j \leq$ número de columnas.

Cualquier elemento del arreglo puede referenciarse en forma directa, siempre que al nombre del arreglo se le adicionen los dos índices. La instrucción LEA: A[1, 1], A[5, 7], A[2, 4], A[5, 3], A[4, 4] tiene como efecto entrar 5 valores al arreglo A; si estos valores son: 10, 15, 40, 99 y 1, y el almacenamiento interno será:



Los demás elementos no tendrán información.

La instrucción ESCRIBA: A[5, 7], A[2, 4] producirá la siguiente salida:



Para llevar información a todo el arreglo se puede hacer de dos formas: una será llenando primero la fila uno, luego la fila 2, hasta la fila N; y la otra sería llenando primero la columna uno, luego la dos y así sucesivamente hasta llenar la columna M, todo dependerá de la forma como se encuentren dispuestos los datos en el archivo de entrada. Por ejemplo: llevar información a una matriz A de 3 * 2, por filas.

Para resolver el problema se necesitan dos ciclos, uno externo que controle las filas y otro interno que por cada valor que tome la variable de control del ciclo externo recorra todas las columnas.

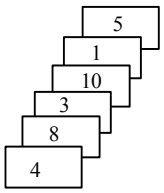
El algoritmo sería:

```
PARA I = 1, 3, 1 HAGA
    PARA J = 1, 2, 1 HAGA
        LEA: A[I, J]
    FIN_PARA
FIN_PARA
```

Si se utiliza el ciclo PARA, o el siguiente si se utiliza el ciclo MIENTRAS.

```
I = 1                                   1
MIENTRAS I <= 3 HAGA
    J = 1                               2
    MIENTRAS J <= 2 HAGA
        LEA: A[I, J]                    3
        J = J+1                         4
    FIN_MIENTRAS
    I = I+1                             5
FIN_MIENTRAS
```

Veamos lo que sucede siguiendo cada uno de los pasos del anterior algoritmo, suponiendo que los valores a entrar son:



FALTA COMPLETAR GRÁFICO PAG. 238 LIBRO

Se ejecuta la instrucción 1 I,

1

Activa el ciclo MIENTRAS externo, compara es I ≤ 3: sí; ejecuta la instrucción 2; J,

1

Activa el ciclo MIENTRAS interno, compara es J ≤: 2 sí; ejecuta las instrucciones 3 y 4.

(3) Lee el elemento A[I, J] (A[1, 1])

A

4

(4) Incrementa J en 1, o sea 1+1; J,

2

Regresa a evaluar la condición del ciclo interno, es 2 ≤ 2: sí; se ejecutan las instrucciones 3 y 4

(3) Lee el elemento A[I, J] (A[1, 2])

A

4 8

(4) Incrementa J en 1, o sea 2+1; J,

3

Regresa a evaluar la condición del ciclo interno, es 3 ≤ 2: no; como la condición es falsa ejecuta la instrucción 5.

(5) Incrementa I en 1, o sea 1+1; I,

2

Regresa a evaluar la condición del ciclo externo, es 2 ≤ 3: sí, ejecuta la instrucción 2; J,

1

Activa el ciclo MIENTRAS interno, compara es 1 ≤ 2: sí; se ejecutan las instrucciones 3 y 4,

(3) Lee el elemento A[I, J], (A[2, 1])

A

4 8

3

(4) Incrementa J en 1, o sea 1+1; J,

2

regresa a evaluar la condición del ciclo interno, es 2 ≤ 2: sí; se ejecutan las instrucciones 3 y 4.

(3) Lee el elemento A[I, J], (A[2, 2])

A

4 8

3 10

(4) Incrementa J en 1, o sea 2+1; J,

3

Regresa a evaluar la condición del ciclo interno, es 3 ≤ 2: no; como la condición es falsa, ejecuta la instrucción 5.

(5) Incrementa I en 1, o sea 2+1; I,

3

Regresa a evaluar la condición del ciclo externo, es 3 ≤ 3: sí, ejecuta la instrucción 2; J,

1

Activa el ciclo MIENTRAS interno, compara es 1 ≤ 2 sí; se ejecutan las instrucciones 3 y 4,

(3) Lee el elemento A[I, J], (A[3, 1])

A

4 8

3 10

1

(4) Incrementa J en 1, o sea 1+1; J,

2

Regresa a evaluar la condición del ciclo interno, es 2 ≤ 2: sí; se ejecutan las instrucciones 3 y 4.

(3) Lee el elemento A[I, J], (A[3, 2])

A

4 8

3 10

1 5

(4) Incrementa J en 1, o sea 2+1; J,

3

Regresa a evaluar la condición del ciclo interno, es 3 ≤ 2: no; como la condición es falsa, ejecuta la instrucción 5.

(5) Incrementa I en 1, o sea 3+1; I,

4

Regresa a evaluar la condición del ciclo externo, es 4 ≤ 3: no; sale del ciclo externo y termina el proceso.

¿Qué características no cumple el anterior algoritmo?

Existen varios tipos de matrices.

- *Matriz cuadrada.* Una matriz es cuadrada cuando el número de filas es igual al número de columnas. Por lo tanto, los elementos ubicados en la diagonal principal cumplen que la fila es igual a la columna.
- *Matriz identidad (I)*. Es también una matriz cuadrada donde los elementos que están ubicados en la diagonal principal son 1 y el resto de elementos tienen un valor de 0.

- *Matriz transpuesta.* La matriz transpuesta A^T de la matriz A de cualquier orden es aquélla en la cual las filas de una matriz son las columnas de la otra.
- *Matriz simétrica.* La matriz simétrica debe cumplir la condición de que el elemento X[I, K] = X[K, I] para una matriz N * M donde; $1 \leq I \leq N$ y
  $1 \leq K \leq M$.
- *Matriz inversa.* Es una matriz A que se representa por A^{-1} y tal que $X * X^{-1} = I$, donde I es la matriz identidad.

Ejercicio resuelto No. 73

Elaborar un algoritmo que entre los elementos por columnas de una matriz A de orden N y luego encuentre la suma de los elementos ubicados en la diagonal principal y la suma de los elementos ubicados en la diagonal secundaria.

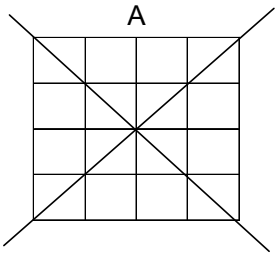
Datos de entrada
- El orden de la matriz (Número de filas = número de columnas).
- Los elementos de la matriz.

Datos de salida
- La suma de los elementos de la diagonal principal.
- La suma de los elementos de la diagonal secundaria.

Proceso

Para entrar los datos del arreglo, primero se debe conocer el orden de la matriz y luego hacer dos ciclos (MIENTRAS O PARA), uno externo que recorra las columnas y uno interno que por cada valor de las columnas recorra todas las filas.

Como ya se dijo, los elementos de la diagonal principal cumplen la condición de que la fila es igual a la columna; veamos esquemáticamente qué característica cumplen los elementos ubicados en la diagonal secundaria.



De acuerdo con la figura, los elementos ubicados en la diagonal secundaria son: el A[1, 4], A[2, 3], A[3, 2], y el A[4, 1], donde la fila crece desde 1 hasta N y la columna decrece desde N hasta 1.

Esto nos permite ubicar el lugar donde se encuentra cada elemento, analizando sus características. Por ejemplo, son los elementos que cumplen que la fila más la columna es igual a N+1. También se puede generar utilizando una variable que en un principio tome el valor máximo de la columna, activando un ciclo que recorra las filas desde 1 hasta N; en la medida que crezcan las filas se decrementa dicha variable en 1. Otra manera de encontrarlos es representando las columnas en términos de las filas. Por ejemplo, cuando la fila vale 1 la columna (N-1+1) = N, cuando la fila vale 2 la columna vale (N-2+1) = N-1 y, cuando la fila vale N la columna vale (N-N+1) = 1; si se utiliza este método, sólo será necesario un ciclo y no se necesita de una variable adicional. También son necesarios dos acumuladores donde se sumen, por separado, los elementos ubicados en cada una de las diagonales.

Definición de variables

N: Tamaño del arreglo.
A: Nombre de la matriz.

I, K: Variables controladoras de ciclo.
SUMAP: Suma de los elementos de la diagonal principal.
SUMAS: Suma de los elementos de la diagonal secundaria.

Algoritmo

```
INICIO
    LEA: N
    PARA K = 1, N, 1 HAGA
        PARA I = 1, N, 1 HAGA
            LEA: A[I, K]
        FIN_PARA
    FIN_PARA
    SUMAP, SUMAS = 0
    PARA I = 1, N, 1 HAGA
        SUMAP = SUMAP + A[I, I]
        SUMAS = SUMAS + A[I, N-I+1]
    FIN_PARA
    ESCRIBA: "SUMA DE LOS ELEMENTOS DE LA DIAGONAL PRIN-
    CIPAL", SUMAP, "Y DE LA SECUNDARIA", SUMAS
FIN_INICIO
```

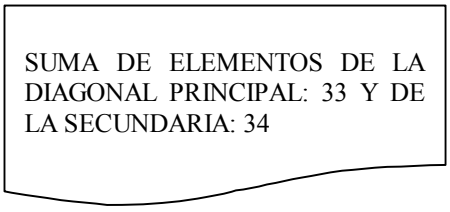
Prueba de escritorio

Si el orden de la matriz es 4 y los valores de los elementos los números: 0, 1, 4, 6, 7, 8, 9, 10, 11, 12, 5, 12, 7, 3, 11, y 20, el proceso de formación de la matriz (entrada de datos), sería el siguiente:

Luego del proceso de entrada sucedería lo siguiente:

N	SUMAP	SUMAS	I
4	~~0~~	~~0~~	~~1~~
	~~0~~	~~7~~	~~2~~
	~~8~~	~~19~~	~~3~~
	~~13~~	~~28~~	~~4~~
	33	34	5

Salida



Ejercicio resuelto No. 74

Hacer un algoritmo que encuentre e imprima la matriz transpuesta de una matriz MAT.

Análisis

Datos de entrada

- Número de filas de la matriz.

- Número de columnas de la matriz
- Los elementos del arreglo.

Datos de salida
- Los elementos de la matriz transpuesta.

Proceso
La matriz transpuesta de la matriz MAT se encuentra intercambiando las filas por las columnas y las columnas por las filas. Si TMAT es la matriz transpuesta de MAT, implica entonces que TMAT [J, I] MAT [I, J].

Si el contenido de MAT es:

MAT =

5	8	1	9
15	11	9	4
16	4	3	0

TMAT =

5	15	16
8	11	4
1	9	3
9	4	0

Como se puede ver, se invierte el orden de la matriz; es decir, el número de filas de MAT es igual al número de columnas de TMAT y el número de columnas se invierte por el número de filas de TMAT.

Definición de variables
NF: Número de filas de la matriz.
NC: Número de columnas de la matriz.
MAT: Nombre de la matriz.
I, J: Variables de control de ciclos e índices de la matriz.
TMAT: Matriz transpuesta de MAT.

El algoritmo hará uso del procedimiento LEER–MAT que lee una matriz por filas, usando los argumentos de envío NF (número de filas), NC (número de columnas), y el de recibo MAT (matriz). Construyendo como ejercicio el procedimiento. Si lo cree necesario básese en el ejercicio resuelto No. 73.

Algoritmo

```
INICIO
    LEA: NF, NC
    LEER_MAT (NF, NC, MAT)
    PARA I = 1, NF, 1 HAGA
        PARA J = 1, NC, 1 HAGA
            TMAT[J, I] = MAT[I, J]
        FIN_PARA
    FIN_PARA
    PARA I = 1, NC, 1 HAGA
        PARA J = 1, NF, 1 HAGA
            ESCRIBA: TMAT[I, J]
        FIN_PARA
    FIN_PARA
FIN_INICIO
```

Ejercicio resuelto No. 75

Hacer un algoritmo que lea una matriz de NF filas y NC columnas y calcule la suma de las filas y de las columnas, dejando los resultados en dos vectores, uno con la sumas de las filas y otro con la suma de las columnas; imprima los dos vectores.

Análisis

Datos de entrada

- Número de filas de la matriz.
- Números de columnas de la matriz.
- Los elementos de la matriz.

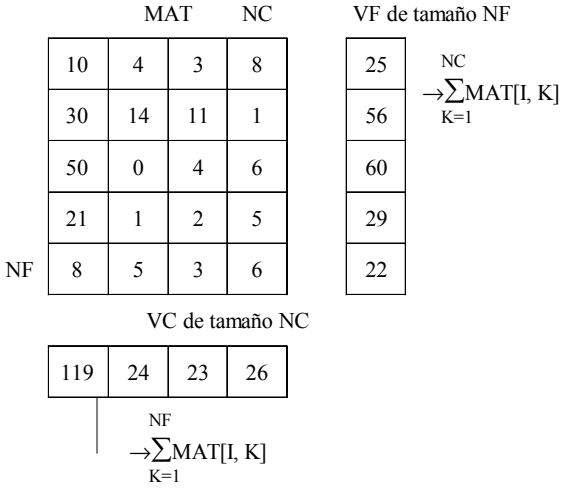
Datos de salida

- El vector que contienen la suma de las filas.
- El vector que contiene la suma de las columnas.

Proceso

Los elementos del arreglo se entrarán por filas; para encontrar la suma de los elementos de una fila o de una columna es necesario un acumulador, que inicie en cero cada que se cambia de una fila a otra o de una columna a otra. Este proceso de acumulación de los valores de los elementos se puede hacer de dos formas: una es utilizando una variable auxiliar como acumulador y una vez que se sume una fila el valor del acumulador se lleva a una posición del vector y se inicializa en cero para obtener la suma de la próxima fila; otra manera es utilizando la posición del vector como si fuera un acumulador, por tanto el valor inicial de cada elemento del vector debe partir de cero cada que se inicie la suma. En el algoritmo se plantearán las dos alternativas: la primera se utilizará cuando se estén sumando los elementos ubicados en cada fila, y la segunda cuando se estén sumando los elementos pertenecientes a las columnas.

Si la matriz es MAT y tiene los siguientes elementos, los contenidos de los vectores se muestran a continuación.



Definición de variables

NF:	Número de filas de la matriz.
NC:	Número de columnas.
MAT:	Nombre de la matriz.
I, K:	Variables controladoras de ciclos.
SUMAF:	Acumulador para sumar los elementos de las filas.
VF:	Vector que contiene la suma de las filas.
VC:	Vector que contiene la suma de las columnas.

Algoritmo

```
INICIO
    LEA: NF, NC
    LEER_MAT(NF, NC, MAT)
    PARA I = 1, NF, 1 HAGA
            SUMAF = 0
            PARA K = 1, NC, 1 HAGA
                    SUMAF = SUMAF + MAT[I, K]
            FIN_PARA
            VF[I] = SUMAF
    FIN_PARA
    PARA K = 1, NC, 1 HAGA
            VC[K] = 0
            PARA I = 1, NF, 1 HAGA
                    VC[K] = VC[K] + MAT[I, K]
            FIN_PARA
            ESCRIBA: "COLUMNA", K, VC[K]
    FIN_PARA
    PARA I = 1, NF, 1 HAGA
            ESCRIBA: "FILA", I, VF[I]
    FIN_PARA
FIN_INICIO
```

Ejercicio resuelto No. 76

Un tablero de damas es un arreglo de 8 filas por 8 columnas. Un uno (1) representa la presencia de una ficha roja en el tablero, un dos (2) representa la presencia de una ficha negra en el tablero y un cero (0) representa la ausencia de ficha. Hacer un algoritmo que calcule:

- El número de fichas rojas.
- El número de fichas negras.
- El número total de fichas.

Análisis

Datos de entrada

- Los elementos de la matriz de 8 * 8 ó estado del tablero.

Datos de salida

- Total de fichas rojas.
- Total de fichas negras.
- Número total de fichas en el tablero.

Proceso

Para saber la clase de ficha que existe en un cajón del tablero se compara el estado de cada elemento con 1, 2 ó 0, y de acuerdo con el resultado se incrementa el contador escogido para cada caso. Un posible estado del tablero de damas sería el siguiente:

			TAB					8
	0	0	1	1	0	0	0	0
	2	0	1	0	0	0	0	1
	0	0	0	0	0	0	0	0
	2	2	0	0	0	2	0	1
	0	0	0	0	0	0	0	0
	0	0	0	2	0	0	1	0
	0	0	0	0	1	0	0	0
8	2	0	0	2	0	0	0	2

Donde existen siete (7) fichas rojas y 8 fichas negras para un total de 15 fichas en el tablero.

Definición de variables

TAB: Matriz que contiene las clases de fichas.
NFR: Número de fichas rojas.
NFN: Número de fichas negras.
TFT: Total de fichas en el tablero.
I, K: Variables controladoras de ciclos.

Algoritmo

```
INICIO
    PARA = 1, 8 1 HAGA
        PARA K = 1, 8, 1 HAGA
            LEA: TAB[I, K]
        FIN_PARA
    FIN_PARA
    NFR, NFN = 0
    PARA I = 1, 8, 1 HAGA
        PARA K = 1, 8, 1 HAGA
            SI TAB[I, K] = 1 ENTONCES
                NFR = NFR + 1
              SINO
                SI TAB[I, J] = 2 ENTONCES
                    NFN = NFN + 1
                FIN_SI
            FIN_SI
        FIN_PARA
    FIN_PARA
    TFT = NFR + NFN
    ESCRIBA: NFR, NFN, TFT
FIN_INICIO
```

Ejercicio resuelto No. 77

Un examen tiene 100 preguntas de selección múltiple. Cada pregunta, dentro de sus alternativas, sólo tiene una respuesta correcta. Las respuestas correctas están en un vector VCR de 100 posiciones, donde VCR[K] representa la respuesta correcta de la pregunta K.

Por cada estudiante se elabora un registro que contiene: nombre del estudiante y a continuación 100 valores que corresponden a las respuestas dadas por el estudiante para cada una de las 100 preguntas.

Elaborar un algoritmo que entre el vector de respuestas correctas, forme un vector con los nombres de los estudiantes, forme una matriz de N * 100 con las respuestas dadas por los estudiantes e imprima los nombres de los estudiantes que aprobaron el examen (se requiere una calificación mínima de 60 puntos para aprobar) y, el número de estudiantes que lo perdieron.

Análisis

Datos de entrada

- Los 100 elementos del vector de respuestas correctas.
- El número de estudiantes que presentaron el examen.
- Los nombres de los N estudiantes.
- Las 100 respuestas dadas por cada uno de los 100 estudiantes.

Datos de salida

- Los nombres de los estudiantes que ganaron el examen.
- La cantidad de perdedores.

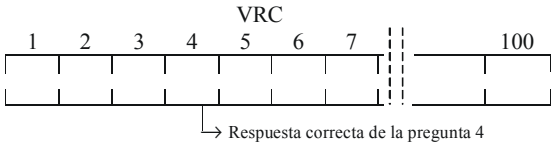
Proceso

Para almacenar la información requerida en el proceso se necesitan tres arreglos.

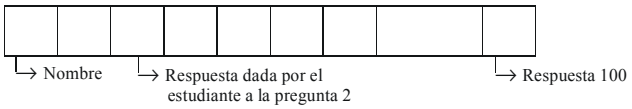
Un vector de 100 posiciones que contiene las respuestas correctas del examen, un vector de N posiciones con los nombres de los estudiantes y la matriz que representa las respuestas dadas por los N estudiantes a las 100 preguntas del examen.

Para obtener el puntaje de un estudiante se compara la respuesta dada a la pregunta K con la posición K del vector de respuestas correctas y si coinciden se incrementa en uno el contador de respuestas correctas.

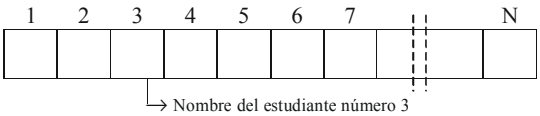
El diseño de los arreglos sería el siguiente:



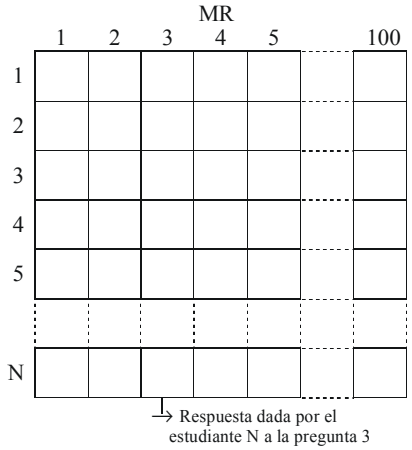
Registro de entrada por estudiante:



Vector de nombres de los estudiantes:



Matriz de respuestas dadas por los estudiantes.



Definición de variables

VRC: Vector de respuestas correctas.
N: Número de estudiantes que presentaron el examen.
NOM: Vector con los nombres de los estudiantes de orden N.
MR: Matriz de respuestas de orden N.
PUN: Variable utilizada para contar las respuestas correctas de cada estudiante.
NP: Variable usada para contar el número de perdedores.
L, K: Variables controladoras de ciclos.

Algoritmo

```
INICIO
    PARA K = 1, 100, 1 HAGA
            LEA: VRC[K]
    FIN PARA
    LEA: N
    PARA L = 1, N, 1 HAGA
            LEA: NOM[L]
            PARA K = 1, 100, 1 HAGA
                    LEA: MR[L, K]
            FIN_PARA
    FIN_PARA
    NP = 0
    ESCRIBA: "NOMBRES DE LOS GANADORES"
    PARA L = 1, N, 1 HAGA
            PUN = 0
            PARA K = 1, 100, 1 HAGA
                    SI MR[L, K] = VRC[K] ENTONCES
                            PUN = PUN + 1
                    FIN_SI
            FIN_PARA
            SI PUN >= 60 ENTONCES
                    ESCRIBA: NOM[L]
              SINO
                    NP = NP + 1
```

```
            FIN_SI
      FIN_PARA
      ESCRIBA: "NÚMERO DE PERDEDORES", NP
FIN_INICIO
```

Ejercicio resuelto No. 78

Elaborar un algoritmo que lea para un arreglo bidimensional la información sobre ventas de NP productos, grabando en la primera columnas el código del artículo y en la siguiente el número de unidades vendidas cada mes (meses de 30 días).

Cada fila corresponde a los datos para un artículo; utilizar las tres columnas siguientes para calcular: promedio de ventas diarias, promedio de ventas en los días que las hubo y número de días que se vendió; imprima la matriz.

Análisis

Datos de entrada

- La cantidad de productos.
- Las ventas de cada producto (información para el arreglo).

Datos de salida

- La información que contiene la matriz luego del proceso.

Proceso

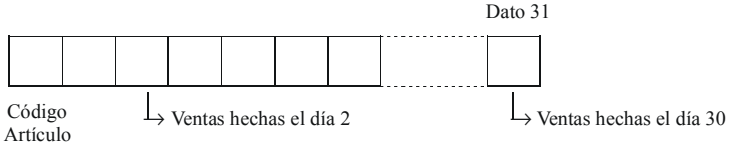
Como el arreglo tiene en la primera columna el código del artículo, inicialmente tendrá 31 columnas (una del código y 30 correspondientes a las ventas hechas cada día del mes). En caso de que en un día no se haga venta, esa posición tendrá valor de cero. La columna 32 del arreglo almacenará el promedio de ventas diarias por artículo, que se calcula sumando cada fila desde la columna dos hasta la treinta y uno y, dividiendo el resultado sobre los treinta días del mes, así:

El promedio de ventas diarias del artículo 1 será:

$$\sum_{k=2}^{31} \text{Ventas } 1,k \ / 30$$

El promedio de ventas en los días que las hubo (columna 33) se obtiene sumando las ventas del producto y dividiendo por el número de días en que hubo ventas, o sea la cantidad de ventas diferentes de cero.

Cada registro de entrada corresponde a una fila de la matriz inicial, éste se diseñaría de la siguiente manera:



Definición de variables

NP:	Cantidad de productos.
VEN:	Matriz de ventas diarias.
NDHV:	Número de días en que hubo ventas.
I, K:	Indices de los ciclos.
SUMA:	Suma de ventas en los 30 días del mes.

El algoritmo hará uso del procedimiento IMPRIMIR–MAT que imprime la matriz por filas, usando los argumentos de envío NP (número de filas), 33, (número de columnas) y la matriz VEN. Construya el procedimiento. Si lo cree necesario vea el ejercicio resuelto No. 74.

Algoritmo

```
INICIO
    LEA: NP
    LEER_MAT (NP, 31, VEN)
    PARA I = 1, N, 1 HAGA
        SUMA, NDHV = 0
        PARA K = 2, 31, 1 HAGA
            SUMA = SUMA + VEN[I, K]
            SI VEN[I, K] <> 0 ENTONCES
                NDHV = NDHV + 1
            FIN_SI
        FIN_PARA
        VEN[I, 32] = SUMA / 30
        VEN[I, 33] = SUMA / NDHV
    FIN_PARA
    IMPRIMIR_MAT (NP, 33, VEN)
FIN_INICIO
```

Ejercicio resuelto No. 79

Se requiere hacer un subprograma que calcule el promedio aritmético de las columnas de una matriz y los almacene en un vector.

Análisis

Parámetros de recibo

- El número de filas de la matriz.
- El número de columnas.
- Los elementos del arreglo.

Parámetros de envío

- Un vector que contiene en cada posición el promedio aritmético de cada columna de la matriz.

Proceso

Se deben emplear dos ciclos: uno externo que recorra las columnas y uno interno que recorra las filas para sumar los elementos de cada columna y luego dividir esta suma por el número de filas de la matriz para obtener el promedio.

Clase de subprograma: PROCEDIMIENTO
Nombre: PROMC

Variables

A:	Nombre de la matriz.
N:	Número de filas de la matriz.
M:	Número de columnas.
VPC:	Vector que contiene el promedio de las columnas, de tamaño M.
SUMA:	Variable auxiliar para totalizar los elementos de cada columna.
I, K:	Variables auxiliares controladoras de los ciclos.

Subprograma

```
PROCEDIMIENTO PROMC (A, N, M, VPC)
        PARA I = 1, M, 1 HAGA
            SUMA = 0
```

```
                PARA K = 1, N, 1 HAGA
                    SUMA = SUMA + A[K, I]
                FIN_PARA
                VEC[I] = SUMA / N
        FIN_PARA
FIN_PROMC
```

Ejercicio resuelto No. 80

Elaborar un subprograma que calcule la suma de los elementos de las filas de una matriz, las almacene en un vector y, además, encuentre la posición del elemento mayor de dicho vector.

Análisis

Parámetros de recibo

- El número de filas de la matriz.
- El número de columnas.
- Los elementos del arreglo.

Parámetros de envío

- Un vector que contenga las sumas de los elementos de las filas de la matriz.
- La posición que ocupa el elemento mayor del vector resultante.

Proceso

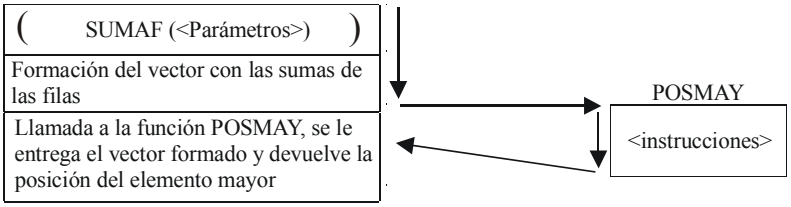
Para encontrar el vector que contenga la suma de las filas de la matriz se implementa un ciclo externo que recorra las filas, y otro interno que recorra las columnas, cada que la variable que controla el ciclo externo se ubica en una fila determinada. Una vez que se forma el arreglo unidimensional se debe encontrar la posición que ocupa el elemento mayor del arreglo. Pero, en el ejercicio resuelto No. 57 se construyó la función POSMAY que hace esta tarea, por lo tanto, este subprograma la invocará.

Si el subprograma a construir se llama SUMAF, esquemáticamente se miraría así:

SUMAF (<Parámetros>)

Formación del vector con las sumas de las filas

Llamada a la función POSMAY, se le entrega el vector formado y devuelve la posición del elemento mayor



Clase de subprograma: PROCEDIMIENTO
Nombre: SUMAF

Variables

NF: Número de filas de la matriz.
NC: Número de columnas.

MA: Nombre del arreglo bidimensional.
VSF: Vector que contiene la suma de las filas de la matriz de tamaño NF.
POSMA: Posición que ocupa el elemento mayor de VSF.
K, L: Variables controladoras de los ciclos.

Subprograma

```
PROCEDIMIENTO SUMAF (MA, NF, NC, VSF, POSMA)
    PARA K = 1, NF, 1 HAGA
            VSF[K] = 0
            PARA L = 1, NC, 1 HAGA
                    VSF[K] = VSF[K] + MA[K, L]
            FIN_PARA
    FIN_PARA
    POSMA = POSMAY ( NF, VSF)
FIN_SUMAF
```

Ejercicio resuelto No. 81

Construir un subprograma que lea el número de filas y los elementos de una matriz de 10 columnas.

Análisis

En este caso, el módulo subordinado no recibe información del módulo que lo invoque ya que los valores a entrar los va a recibir desde un medio externo de entrada de datos, pero sí le devuelve información.
Puede usarse LEER–MAT, que sería lo mismo.

Parámetros de recibo

No hay.

Parámetros de envío

- Número de filas de la matriz.
- Los elementos de la matriz.

Definición de variables

LECTURA: Nombre del subprograma.
F: Número de filas del arreglo bidimensional.
MATRIZ: Nombre de la matriz.
I, J: Variables controladoras de ciclos.

Subprograma

```
PROCEDIMIENTO LECTURA (F, MATRIZ)
    LEA: F
    PARA I = 1, F, 1 HAGA
            PARA J = 1, 10, 1 HAGA
                    LEA: MATRIZ[I, J]
            FIN_PARA
    FIN_PARA
FIN_LECTURA
```

Ejercicio resuelto No. 82

Elaborar un subprograma que escriba:

a. Los elementos de un vector que contiene el promedio de estudiantes por nivel de una universidad.
b. Los elementos de un vector cuyos elementos son los estudiantes por facultad de una universidad.
c. La facultad que tiene mayor número de estudiantes.

Análisis

Parámetros de recibo

- Número de elementos del vector que contienen los promedios.
- El vector de promedios.
- El número de elementos del vector de estudiantes por facultad.
- El vector de número de estudiantes por facultad.
- La facultad que tiene mayor número de estudiantes.

Parámetros de envío

La información no la envía al módulo que lo invoca sino a un medio externo de salida, por lo tanto, no tiene parámetros de envío.

Clase de subprograma: PROCEDIMIENTO

Nombre: SALIDA

Variables

NP: Tamaño del vector que contiene los promedios.
VPROM: Vector que contiene los estudiantes por nivel de la universidad.
NEF: Tamaño del vector de estudiantes por facultad.
VFAC: Vector de estudiantes por facultad.
MAYF: Variable que contiene el número de la facultad con mayor número de estudiantes.
I: Variable controladora de ciclo.

Subprograma

```
PROCEDIMIENTO SALIDA (NP, VPROM, NEF, VFAC, MAYF)
    ESCRIBA: "UNIVERSIDAD DE ANTIOQUIA"
    ESCRIBA: "PROMEDIO DE ESTUDIANTES POR NIVEL"
    ESCRIBA: "NIVEL          PROMEDIO"
    PARA I = 1, NP, 1 HAGA
        ESCRIBA: I, VPROM[I]
    FIN_PARA
    ESCRIBA: "TOTAL DE ESTUDIANTES POR FACULTAD"
    ESCRIBA: "FACULTAD ESTUDIANTES"
    PARA I = 1, NEF, 1 HAGA
        ESCRIBA: I, VFAC[I]
    FIN_PARA
    ESCRIBA: "FACULTAD CON MAYOR NÚMERO DE ESTUDIANTES",
MAYF
FIN_SALIDA
```

Ejercicio resuelto No. 83

Hacer un algoritmo que utilice el subprograma LECTURA para leer el número de facultades y el número de estudiantes de los 10 niveles de cada facultad; utilice el subprograma PROMC para calcular el promedio de estudiantes por nivel en la

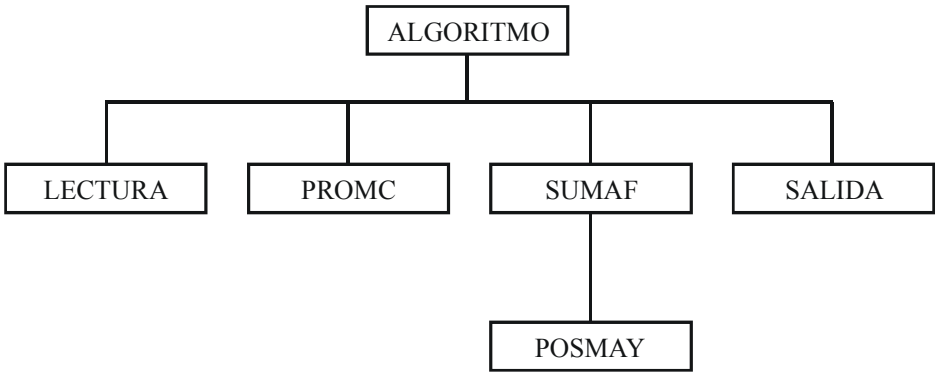
universidad; utilice el subprograma SUMAF para calcular el total de estudiantes por facultad, y localizar la facultad que tiene mayor número de estudiantes; utilice el subprograma SALIDA para imprimir la información requerida.

Análisis

Como se puede ver en el enunciado, toda la labor que debe realizar el algoritmo está ya hecha por los subprogramas; por tanto, el trabajo del algoritmo será invocar a los módulos respectivos para que entreguen la información requerida.

El algoritmo debe definir sus propias variables para enviar y recibir información de las estructuras subordinadas.

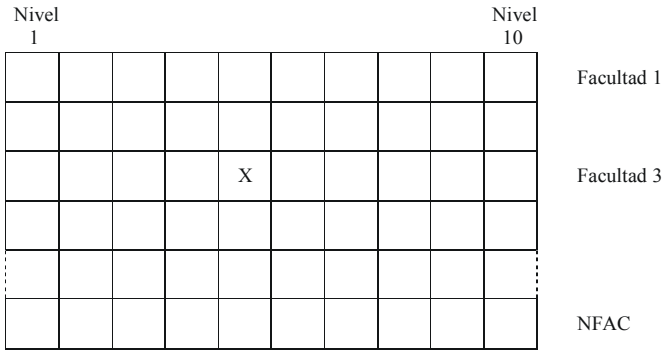
Gráficamente se puede mirar de la siguiente manera:



Definición de variables

NFAC: Número de facultades de la universidad.

NIVEL: Matriz de NFAC filas y 10 columnas que almacenará, en cada elemento, el número de estudiantes por nivel en la universidad. Su estructura será:



Donde X será el número de estudiantes que tiene la facultad 3 en el nivel 5.

VPROM: Vector de promedios de estudiantes por nivel en la Universidad.
VEPF: Vector que contiene el total de estudiantes por facultad.
FMAYOR: Facultad que tiene el mayor número de estudiantes.

Transferencia de información entre módulos

Módulo 1 (LECTURA)

Este módulo devuelve al algoritmo la información de entrada que es almacenada en la variable NFAC y en la matriz NIVEL.

Módulo 2 (PROMC)

Se le entrega información a través de los argumentos NIVEL, NFAC, 10, devuelve el vector de promedios VPROM.

Módulo 3 (SUMAF)

Se le entrega información a través de las variables NIVEL, NFAC, 10, devuelve el vector que contiene el total de estudiantes por facultad VEPF y la facultad de mayor número de estudiantes FMAYOR.

Módulo 4 (SALIDA)

Se le entregan los argumentos 10, VPROM, NFAC, VEPF y FMAYOR y muestra la información a través de un medio externo de salida.

Algoritmo

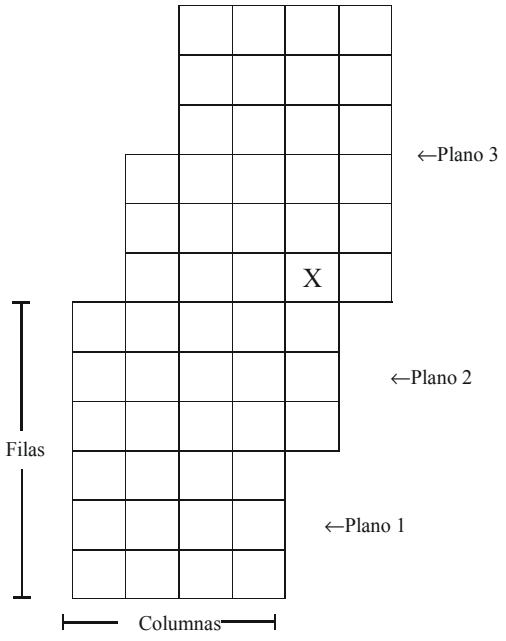
```
INICIO
    LECTURA (NFAC, NIVEL)
    PROMC (NIVEL, NFAC, 10, VPROM)
    SUMAF (NIVEL, NFAC, 10, VEPF, FMAYOR)
    SALIDA (10, VPROM, NFAC, VEPF, FMAYOR)
FIN_INICIO
```

8.3. Arreglos multidimensionales

Pueden existir arreglos de tres y más dimensiones, pero la mayoría de las aplicaciones o procesos por computadora no utilizan arreglos con más de tres dimensiones porque los arreglos gastan bastante memoria principal, y muchas veces existe gran cantidad de memoria separada y no utilizada; también porque visualizar un arreglo con más de tres dimensiones no es fácil.

Un arreglo de tres dimensiones puede visualizarse como una caja: un conjunto de tablas, una tras otra, donde los elementos que conforman el arreglo están dispuestos bajo tres conceptos de clasificación [PLANO, FILA, COLUMNA]; por lo tanto, la referenciación de un elemento cualquiera está direccionada por tres índices.

Por ejemplo: representar un arreglo tridimensional, PLAN de 3 planos, 6 filas y 4 columnas.



Donde X hace referencia al contenido del elemento PLAN[2, 3, 4] o sea el elemento del arreglo PLAN ubicado en el plano 2, fila 3, columna 4. En general, se puede decir que PLAN[P, F, C] hace referencia al elemento del arreglo PLAN ubicado en el plano P, fila F, columna C donde:

1 <= P <= No de planos, 1 <= F <= No de filas y 1 <= C <= No de columnas.

Ejercicio resuelto No. 84

Hacer un algoritmo que entre los elementos de un arreglo tridimensional de NF filas, NC columnas y NP planos; encuentre el valor mayor de los elementos del arreglo y el lugar que ocupa.

Análisis

Datos de entrada

- El número de planos.
- El número de filas.
- El número de columnas.
- Los elementos del arreglo.

Datos de salida

- El valor mayor y la posición que ocupa dentro del arreglo.

Proceso

El arreglo se recorrerá por planos y dentro de éstos por filas; esto implica tener tres ciclos: uno externo que recorra los planos, uno intermedio que recorra las filas, y uno interno que recorra las columnas. Para encontrar el elemento mayor dentro del arreglo asumimos que el valor mayor es el elemento [1, 1, 1], o sea, el elemento ubicado en el plano 1, fila 1, columna 1; luego se continúa comparando este mayor con los demás elementos del arreglo, efectuando cambios cada vez que se encuentre un valor mayor.

Definición de variables

ARTD:	Nombre del arreglo tridimensional.
NP:	Número de planos del arreglo.
NF:	Número de filas del arreglo.
NC:	Número de columnas del arreglo.
F, C, P:	Variables controladoras de ciclos.
MAYOR:	Valor mayor de los elementos del arreglo.
FM:	Fila donde se encuentra MAYOR.
CM:	Columna donde se encuentra MAYOR.
PM:	Plano donde se encuentra MAYOR.

Algoritmo

```
INICIO
    LEA: NP, NF, NC
    PARA P = 1, NP, 1 HAGA
            PARA F = 1, NF, 1 HAGA
                    PARA C = 1, NC, 1 HAGA
                            LEA: ARTD[P, F, C]
                    FIN_PARA
            FIN_PARA
    FIN_PARA
    MAYOR = ARTD[1, 1, 1]
    FM, CM, PM = 1
    PARA P = 1, NP, 1 HAGA
```

```
            PARA F = 1, NF, 1 HAGA
                PARA C = 1, NC, 1 HAGA
                    SI MAYOR < ARTD[P, F, C] ENTONCES
                        MAYOR = ARTD[P, F, C]
                        FM = F
                        CM = C
                        PM = P
                    FIN_SI
                FIN_PARA
            FIN_PARA
        FIN_PARA
        ESCRIBA: "EL MAYOR VALOR DEL ARREGLO ES:", MAYOR, "Y ESTE
UBICADO EN EL PLANO", PM, "FILA", FM, "COLUMNA", CM
    FIN_INICIO
```

Ejercicio resuelto No. 85

Una empresa aérea ofrece 5 vuelos a los clientes. Cada avión de la línea tiene 50 filas de asientos y 6 columnas. Para efectos de sistematización, un asiento ocupado se representa mediante un uno y un asiento libre con un cero. Hacer un algoritmo que muestre el número de pasajeros que lleva cada vuelo.

Análisis

Datos de entrada

- La información o estado de los vuelos. Arreglo tridimensional de unos y ceros.

Datos de salida

- El número de pasajeros de cada vuelo.

Proceso

La información se almacenará en un arreglo tridimensional de tamaño 5 * 50* 6, que a su vez representa la cantidad de pasajeros que puede transportar la línea aérea. Se hará uso de un contador que totalice la cantidad de pasajeros que lleva cada vuelo, el cual debe comenzar en cero cada que se procese la información perteneciente a cada avión.

Definición de variables

PUESTO: Nombre del arreglo.
F, C, P: Variable controladoras de los ciclos.
NPV: Contador de pasajeros por vuelo.

Algoritmo

```
INICIO
    PARA P = 1, 5, 1 HAGA
        PARA F = 1, 50, 1 HAGA
            PARA C = 1, 6, 1 HAGA
                LEA: PUESTO[P, F, C]
            FIN_PARA
        FIN_PARA
    FIN_PARA
    PARA P = 1, 5, 1 HAGA
        NPV = 0
        PARA F = 1, 50, 1 HAGA
            PARA C = 1, 6, 1 HAGA
                SI PUESTO[P, F, C] = 1 ENTONCES
                    NPV = NPV + 1
```

```
                    FIN_SI
                FIN_PARA
            FIN_PARA
            ESCRIBA: "EL VUELO", P, "LLEVA", NPV, "PASAJEROS"
        FIN_PARA
    FIN_INICIO
```

Aspectos a tener en cuenta

a. El nombre asignado a los arreglos debe seguir las mismas normas para la conformación de nombres de variables simples.
b. Si a un elemento de un arreglo se le asigna un valor, éste debe ser compatible con el tipo de dato de los elementos del mismo.
c. Los arreglos sólo existen en memoria principal, y los datos internamente están organizados de manera lineal.
d. Los índices de un arreglo pueden ser variables, constantes o expresiones que den como resultado un valor entero.
e. Los elementos individuales de un arreglo pueden ser utilizados en expresiones, asignación, entrada y salida de datos; para su utilización es necesario acompañar el nombre del arreglo con los índices que ubiquen plenamente el elemento que se requiere referenciar.
f. Todos los elementos de un arreglo deben ser del mismo tipo (enteros, reales, boléanos o caracteres).
g. Un arreglo se debe implementar cuando existe toda una serie de datos pertenecientes al mismo tipo y cuando sus valores son requeridos en memoria para el proceso por computadora.
h. Se debe tener cuidado en no acceder un elemento de un arreglo que no esté dentro del rango de índices preestablecidos.
i. Los procesos que utilizan arreglos multidimensionales requieren ciclos, en lo general un ciclo por dimensión.

8.4. Ejercicios propuestos

147. Hacer un algoritmo que entre 5 valores y los almacene en un vector e imprimir el vector.
148. Elaborar un algoritmo que genere un arreglo de N elementos y encuentre el valor y la posición del mayor elemento.
149. Usando el arreglo anterior, escribir los elementos cuyos valores sean múltiplos de 3.
150. Hacer un algoritmo que entre una serie de valores, los almacene en un vector y sume los elementos de las posiciones pares, lo mismo que las posiciones impares por separado.
151. Hacer un algoritmo que forme un vector del que no se sabe el número de elementos. Calcule el promedio de los elementos y forme dos nuevos arreglos, uno con los elementos menores o iguales al promedio y otro con los superiores. Imprima los dos nuevos arreglos.
152. Elaborar un algoritmo que lea dos arreglos unidimensionales de igual tamaño y forme un tercer arreglo mediante el producto de los elementos de los dos arreglos, tomados en orden inverso, es decir, productos del primer elemento del primer arreglo con el último del segundo; del segundo del primer arreglo con el penúltimo del segundo arreglo; hasta llegar al último del primer arreglo con el primero del segundo arreglo. Imprimir el arreglo formado.
153. Elaborar un algoritmo que forme un vector B de M elementos, luego entre el valor de la variable CLAVE y la busque secuencialmente dentro del arreglo, suspendiendo la búsqueda cuando la encuentre.
154. Hacer el mismo algoritmo anterior aplicando BÚSQUEDA BINARIA DE LA CLAVE.
155. Se tienen dos arreglos unidimensionales A y B de M elementos. Partiendo de los dos arreglos, elaborar un algoritmo que forme tres nuevos arreglos. El primero con la suma de los elementos respectivos, el otro con el producto y el último con la diferencia.

156. Elaborar un algoritmo que suma los elementos de un vector de tamaño N, excepto el primero, último y el elemento del centro.
157. Elaborar un algoritmo que forme dos arreglos. El primero contiene los M códigos de los estudiantes que perdieron COMPUTADORAS y el segundo los N códigos de los estudiantes que perdieron ALGORITMOS. Partiendo de los dos arreglos formar uno nuevo con los códigos de los estudiantes que perdieron ambas materias.
158. Elaborar un subprograma que encuentre el elemento menor en un vector.
159. Elaborar un subprograma que encuentre las posiciones que ocupan el elemento mayor y menor en un vector.
160. Elaborar un subprograma que inserte un valor en el lugar que le corresponda en un vector ordenado descendentemente.
161. Elaborar un subprograma que verifique si una línea de texto que termina en punto es un PALINDROMO. Un texto es palíndromo si se lee lo mismo de izquierda a derecha o de derecha a izquierda. Ejemplo: AMOR A ROMA, OTTO.
162. Elaborar un subprograma que elimine todas las veces que aparece un valor en un arreglo unidimensional, sin destruir la información de entrada.
163. Elaborar un subprograma que sume los elementos correspondientes de dos vectores, dando como resultado un nuevo vector.
164. Elaborar un algoritmo que encuentre la intersección de dos vectores A y B, de N y M elementos respectivamente.
165. Una cadena de almacenes tiene la siguiente información de sus empleados: código, nombre y cargo; los datos de los empleados se deben almacenar en vectores paralelos. Elabore un algoritmo que muestre los datos de los empleados ordenados alfabéticamente.
166. Se tienen dos grupos de datos de tamaño M y N. Elabore un algoritmo que inserte uno de los dos grupos en el otro a partir de una posición dada.
167. Si X y Y son vectores de igual tamaño, elaborar un algoritmo que utilice tres subprogramas para obtener los parámetros a y b de la ecuación de la recta
    Y = a + bx, mediante las fórmulas:

$$a = \frac{\sum_{i=1}^{N} Xi^2 \sum_{i=1}^{N} Yi^2 - \sum_{i=1}^{N} Xi^2 \sum_{i=1}^{N} XiYi}{N\sum_{i=1}^{N} Xi^2 - (\sum_{i=1}^{N} Xi)^2} \quad b = \frac{N\sum_{i=1}^{N} XiYi - \sum_{i=1}^{N} Xi \sum_{i=1}^{N} Yi}{N\sum_{i=1}^{N} Xi^2 - (\sum_{i=1}^{N} Xi)^2}$$

168. Elaborar un algoritmo que forme dos vectores de NA y NB elementos, los cuales, se sabe, están ordenados ascendentemente cada uno. Forme un tercer arreglo que quede ordenado en la medida que pasan los elementos de los dos arreglos iniciales. Imprimir el nuevo arreglo.
169. Elaborar un algoritmo que forme un vector con una serie de códigos y otro con una serie de nombres asociados a los códigos. Luego entre un conjunto de códigos y determine si están o no en el arreglo y, si es así, escriba el código y el nombre.
170. Escribir un algoritmo que dado un vector de tamaño N elimine de éste cuantas veces aparezca un valor. Escriba el arreglo y su nuevo tamaño.
171. Escribir un algoritmo que dado un arreglo unidimensional y su tamaño, elimine sus elementos repetidos, dejando un elemento en cada caso.
172. Dado un arreglo unidimensional de tamaño N, encuentre la moda, es decir, el valor que se presenta más número de veces entre sus elementos.
173. Escriba un algoritmo con las mismas características que el anterior, en el caso de que exista más de una moda.
174. Si tres elementos cualesquiera de un vector V se relacionan por la ecuación:

V[I+2] = V[I+1] + V[l], si V[1] = 1, V[2] = 2, encontrar los primeros N elementos del vector.

175. Elaborar un algoritmo que entre un arreglo unidimensional, calcule e imprima:

- Número de datos repetidos en el arreglo.
- El número de valores impares.
- El número de valores pares.

176. Elaborar un algoritmo que forme dos arreglos relacionados que almacenen los códigos de los N artículos que se venden en un almacén y la existencia (número de, unidades existentes) de cada uno de los artículos, así:

Código	Existencia
101	30
200	40

Del artículo 101 hay 30 unidades en existencia; del artículo 200, hay 40 unidades y así sucesivamente.

Por cada proveedor o cliente que llega al almacén se genera un registro con los siguientes datos:

- Tipo de transacción [1 para proveedor (recibo); 2 para cliente (venta)].
- Código del artículo transado.
- Número de unidades transadas (recibidas o vendidas).

Se requiere calcular lo siguiente para cada transacción:

- Si es recibo, se suma a las existencias actuales de este artículo el número de unidades transadas.
- Si es venta, se resta a las unidades en existencia de ese artículo las unidades vendidas.

Al final se deben mostrar los códigos de los artículos y las existencias de cada uno de ellos, es decir, el arreglo de códigos y de existencias ya actualizado.

Asuma que en ningún momento la cantidad vendida es mayor que la cantidad, en existencia, y por lo tanto no le presentan inconsistencias.

177. Elaborar un algoritmo que forme tres vectores de N elementos cada uno: relacionados entre sí, los cuales contienen la renta líquida, el patrimonio y el valor crédito que corresponde a esa renta líquida o a ese patrimonio, así:

RENTA LIQUIDA	PATRIMONIO	VALOR CREDITO
25.000	100.000	130
30.000	150.000	200
40.000	210.000	280

Para una renta líquida entre 0 y $25.000 o un patrimonio no mayor de $100.000, el valor del crédito correspondiente es $130; para una renta líquida mayor de $25.000 y menor de $30.000 o un patrimonio mayor de $100.000 y menor o igual a $150.000, el valor crédito correspondiente es de $200, etc. Por cada estudiante que se matricula se forma un registro con los siguientes datos:

- Código del estudiante.
- Renta líquida.
- Patrimonio.
- Número de créditos a cursar.

Se debe mostrar el código, lo que debe pagar cada estudiante y el total recaudado por matrícula.

178. Se tienen dos arreglos unidimensionales. Uno de ellos con N elementos y, el otro con M elementos. Los elementos de los dos arreglos se encuentran ordenados en forma ascendente. Elabore un algoritmo que entre los dos vectores se forme uno nuevo de N + M elementos, el cual contendrá los elementos de los dos arreglos ordenados de menor a mayor.
179. En una universidad se efectúa un examen de admisión que consta de dos pruebas: aptitud matemática y aptitud verbal. Cada pregunta tiene 5 opciones numeradas del 1 al 5. Se prepara un registro con 60 campos de una sola posición que contiene, cada uno, la respuesta correcta a la pregunta correspondiente. Las 30 primeras posiciones corresponden al examen de aptitud matemática y las restantes a las de aptitud verbal.

Se presentaron al examen N estudiantes y para cada uno de ellos se preparó un registro con los siguientes datos:

- Número de la credencial.
- Respuestas al examen de aptitud matemática.
- Respuestas al examen de aptitud verbal.

Se requiere saber:

- Puntaje obtenido por cada estudiante en cada examen.
- El puntaje total por estudiante.
- El puntaje promedio de cada examen.
- El puntaje promedio total.
- La credencial y el puntaje correspondiente a los estudiantes que obtuvieron un puntaje superior o igual al promedio.
- El mayor puntaje y el número de la credencial del estudiante que la obtuvo.

180. Hacer un algoritmo que lea dos arreglos A y B y forme un nuevo arreglo M, cuyos elementos corresponden a la unión de los conjuntos definidos por A y B. La unión de A y B son los elementos del conjunto A más los elementos del conjunto que no están en A.
181. El desempeño usual de un grupo de ventas está en registros (archivo), dos valores por registro; el primero es el número de identificación de un vendedor particular y el segundo el número de ventas que ha hecho. Elaborar un algoritmo que muestre el número de identificación y el número de ventas, donde éstas aparezcan ordenadas en forma ascendente.
182. Elaborar un algoritmo que lea una lista de números enteros positivos y ejecute las siguientes operaciones:

1. Mostrar los números de la lista en orden creciente.
2. Calcular e imprimir la mediana (valor central).
3. Determinar el número que ocurre más frecuentemente.
4. Imprimir una lista que contenga:

4.1. Números menores que 30.
4.2. Números mayores que 70.
4.3. Números que no pertenezcan ni al grupo 4.1 ni al 4.2.

183. Dados dos vectores ordenados alfabéticamente con M y N nombres de los empleados de una empresa, hacer un algoritmo para obtener un nuevo vector, que no contenga nombres repetidos y esté ordenado.
184. Una cadena de almacenes tiene sucursales en cuatro ciudades diferentes y un total de N empleados. El registro preparado para cada empleado tiene los siguientes campos: nombre, ciudad, código.

    Los datos de los clientes están en vectores paralelos. Diseñar un algoritmo que ordene todos los datos de los empleados de modo que se pueda visualizar, por sucursal, la lista de los empleados en orden alfabético creciente.
185. Elaborar un algoritmo que forme dos vectores que contengan NOMBRES y SEXO de los alumnos de la materia *Algoritmos*. Generar dos nuevos arreglos: uno que contenga los nombres de los hombres y otro el de las mujeres, ambos ordenados alfabéticamente.

186. Elaborar un algoritmo que lea una matriz por filas y la imprima por columnas.
187. Elabore un subprograma que determine si un número es automórfico. Un número automórfico es aquél que al elevarlo sistemáticamente a una potencia mayor que cero, las últimas cifras del resultado corresponden al mismo número. Por ejemplo:

    $76^1 = 76$; $76^2 = 5776$; $76^3 = 438976$; $76^4 = 33362176$, pruebe con las 3 primeras potencias.

188. Elaborar un algoritmo que lea el orden de una matriz y sus elementos y encuentre los elementos mayor y menor y sus posiciones.
189. Hacer un algoritmo que lea el orden de un arreglo bidimensional y sus elementos; convierta la matriz en un arreglo unidimensional. La conversión debe hacerse por columnas, es decir, mover la primera columna al vector, a continuación la segunda columna y así sucesivamente. Imprima el arreglo formado.
190. Elaborar un algoritmo que lea el número de filas y de columnas de una matriz y sus elementos. Luego intercambie el contenido de la primera y la última columna, de la segunda y la penúltima y así hasta completar cambios que den una matriz con columnas invertidas. Imprima la matriz.
191. Resuelva el mismo problema anterior, haciendo intercambio de filas.
192. Elaborar un algoritmo que lea el orden y la matriz misma, luego la invierta por filas, así:

    El elemento (1, 1) se intercambie con el (N, M); el (1, 2) con el (N, M –1); el (1, 3) con el (N, M–2) y así sucesivamente. Imprimir la matriz invertida.

193. Elaborar un algoritmo que lea el orden y los elementos de una matriz cuadrado y averigüe si es simétrica, es decir, si todos los pares de elementos equidistantes perpendicularmente de la diagonal principal son iguales.
194. Elaborar un algoritmo que forme una matriz con datos provenientes de una serie de registros, sujeto a lo siguiente:

- El contenido de cada registro es un elemento de la matriz.
- La matriz tiene 5 columnas y un máximo de 50 filas.
- El almacenamiento debe hacerse por filas, es decir, los primeros 5 registros forman la fila uno y así sucesivamente.

195. Elaborar un algoritmo que forme una matriz con las siguientes características:

a. La primera fila y primera columna tienen como elementos los números del 0 al 20 consecutivos.
b. Los demás elementos se obtienen de multiplicar cada elemento de la fila uno por cada elemento de la columna uno. Imprimir la matriz. Ejemplo:

0	1	2	3	4	5	_____	20
1	1	2	3	4	5	_____	.
2	2	4	6	8	10	_____	.
3	3	6	9	12	15	_____	.
4	4	8	12	16	20	_____	.
5	5	10	15	20	25	_____	.
.							.
.							.
.							.
.							.
20							400

196. Una fábrica produce N artículos diferentes y utiliza M sucursales para su distribución, variando el precio de venta de cada artículo según la sucursal que lo distribuye. Para esto la fábrica tiene un cuadro que muestra el precio de cada artículo según la sucursal que lo distribuye. Al final de cada período cada sucursal envía a la fábrica la cantidad vendida de

cada articulo, formándose un nuevo cuadro. Elaborar un algoritmo que encuentre las ventas totales de la fábrica para cada uno de los artículos y para cada una de las sucursales.

197. Dos tahúres se han ideado el siguiente juego:

Elaboran una tabla de 6 filas por 6 columnas.

El tahúr A dice: tiro los dados y cada uno de ellos me indica una fila de la tabla, luego multiplico entre sí los elementos de las dos filas y ése es mi puntaje; el tahúr B dice: tiro los dados y cada uno de ellos me representa una columna, luego sumo los elementos correspondientes y ése es mi puntaje.

¿Se podrá decir a simple vista quién es el ganador?

Elabore un algoritmo que entre la tabla y los valores dados por los dados y compruebe quién es el ganador, sabiendo que el tahúr B tira los dados cuando conoce los resultados del tahúr A.

198. Elaborar un algoritmo que entre una matriz de M * N y ordene en forma creciente los elementos de las columnas del arreglo.
199. Dada una matriz simétrica, elaborar un algoritmo que la convierta a triangular inferior (el triángulo superior derecho debe contener ceros, exceptuándose la diagonal principal).
200. Elaborar un algoritmo que llene una matriz de orden N con ceros, excepto las dos diagonales que deben contener unos.
201. Dada una matriz ordenada ascendentemente con base en la primera columna, insertarle los elementos de un vector, de tal forma que el primer elemento de éste no destruya el ordenamiento.
202. Elabore un algoritmo que forme una matriz de N * M elementos con blanco y números del cero al nueve, dispersos en el cuerpo de la matriz y luego busque las posiciones que contienen números y los coloque en forma consecutiva desde la fila uno en adelante, por ejemplo:

		4		
	6			7
	8		9	1

4	6	7	8	9
1				

203. Dadas dos matrices elaborar un algoritmo que obtenga la matriz producto.
204. Elaborar un algoritmo que entre una matriz N * M y un valor y encuentre el producto de éste por la matriz.
205. Elaborar un algoritmo que forme una matriz de N * M elementos. Cada elemento del arreglo representa las ventas atribuibles a cada uno de los N vendedores de una empresa, para cada uno de los M años de operaciones que ha tenido la misma y, luego calcular:

- El total de ventas de cada vendedor en los M años.
- El total de ventas en cada año.
- El gran total de ventas de la empresa.

206. Elaborar un algoritmo que actualice la tabla de clasificación del Campeonato Profesional de Fútbol de su país; luego de efectuada una fecha, el algoritmo debe leer:

- Un registro por equipo que contenga los siguientes datos:
  - Código del equipo.
  - Partidos jugados.
  - Partidos ganados.
  - Partidos empatados.

- Partidos perdidos.
- Goles a favor.
- Goles en contra.
- Puntos.

- Un registro por partido que contenga los siguientes datos:
  - Código del equipo local.
  - Goles del equipo local.
  - Código del equipo visitante.
  - Goles del equipo visitante.

El algoritmo debe mostrar la tabla leída, ordenada de acuerdo a la puntuación, luego de haber calculado los puntos para cada equipo.

207. Dada una matriz de N filas por M columnas que debe ser leída, elaborar un algoritmo que ordene los elementos del arreglo en forma descendente por filas.
208. Un caballo está ubicado en una de las posiciones de un tablero de ajedrez. Elaborar un algoritmo que averigüe qué nueva posición podrá tomar después de una jugada, conociendo el lugar inicial donde se encuentra.
209. Se desea obtener el grado de eficiencia de N operarios de una fábrica productora de tornillos, de acuerdo con las siguientes condiciones que se les impone para un período determinado. Condiciones:
    1. Ausencia del trabajo ≤ 3,5 horas.
    2. Tornillos defectuosos < 300.
    3. Tornillos producidos > 10000.

Los grados de eficiencia del trabajador son asignados de la siguiente forma:

- Si no cumple ninguna condición: Grado = 5
- Si sólo cumple la primera condición: Grado = 7
- Si sólo cumple la segunda condición: Grado = 8
- Si sólo cumple la tercera condición: Grado = 9
- Si cumple la primera y la segunda: Grado = 12
- Si cumple la primera y la tercera: Grado = 13
- Si cumple la segunda y la tercera: Grado = 15
- Si cumple las tres condiciones: Grado = 20

Por cada trabajador se tiene un registro con los siguientes datos:
- Código del trabajador.
- Horas de ausencia.
- Tornillos defectuosos.
- Tornillos producidos.

Elaborar un algoritmo que calcule para cada trabajador su correspondiente grado de eficiencia e imprima toda su información.

210. Un campo de golf consta de 18 hoyos, en ellos debe introducirse, sucesivamente, una pelota a base de golpes con un bastón. En una tarjeta van anotándose el número de golpes requeridos para llegar a cada uno de los hoyos. En la misma tarjeta pueden anotarse los golpes de varios jugadores, ya que ésta tiene la forma de una tabla (matriz): los renglones corresponden a los jugadores y las columnas a cada hoyo del campo. Por ejemplo, si en un juego participan 4 jugadores la tarjeta tendrá 4 renglones y 18 columnas. El juego lo gana el participante que llegue al hoyo 18 con el menor número de golpes.

Suponga que después de concluido el partido se tiene la tarjeta con todos los golpes de los N jugadores.

Elaborar un algoritmo que entre los datos y obtenga:

- La formación de un vector con los nombres de los participantes.
- El nombre de la persona que ocupó el primer lugar, el segundo, etc., hasta el enésimo.

211. Considere un minitablero de ajedrez de 4 * 4 y trate de situar cuatro reinas, de tal modo, que no se *coman* entre sí. Recuerde que una reina come sobre su fila, su columna y las dos diagonales que pasan por su posición.
212. Un avión dispone de 180 plazas, de las cuales 60 son de no fumador y numeradas del 1 al 60 y 120 plazas numeradas del 61 al 180. Diseñar un algoritmo que permita reservar plazas del avión y diga cuántos pasajeros de cada clase ocupan asientos en el vuelo.
213. Estudiantes provenientes de 50 municipios han decidido presentar examen de ingreso a la universidad. Se requiere de una matriz que indique los resultados de las personas de cada colegio que han presentado el examen. La entrada está compuesta por el número de la credencial, el código del colegio (de 1 a 50) y el resultado del examen. La salida debe tener los puntos de los estudiantes procedentes de cada colegio.
214. Elabore un algoritmo que entre los elementos de una matriz cuadrada de orden N y determine si es un CUADRADO MAGICO. Se considera cuadrado mágico aquél en el que la suma de cada una de las filas, suma de cada una, de las columnas y suma de cada una de las diagonales son la misma cantidad.
215. En las elecciones para Alcalde del PUEBLITO PAISA se han presentado tres candidatos (A, B, C). El Pueblito está dividido en 5 zonas de votación.

El reporte de votos de la zona se recibe en orden: primero la zona 1, la 2, la 3, la 4 y por último la zona 5.

Elabore un algoritmo que:

- Forme una matriz de 5 filas y 3 columnas que contenga, en cada fila, los votos reportados por las zonas para cada uno de los tres candidatos.
- Encuentre el total de votos obtenidos por cada candidato y el porcentaje que éste representa.
- Escriba un mensaje declarando ganador a un candidato, si éste obtuvo más del 50% de la votación.

216. Elabore un algoritmo que lea el número de filas y de columnas de una matriz y la matriz misma. Se asegura que los valores de los elementos están entre 1 y 6. Forme 6 vectores con los elementos iguales a 1, 2, 3, 4, 5 y 6, muestre el tamaño de los vectores y diga cuál es la moda, es decir, el elemento que más veces se repite en la matriz.
217. Elabore un algoritmo que para una matriz A de orden N * M obtenga:

- Suma de cada fila impar.
- Productoria de la diagonal secundaria.
- Número de valores iguales a cero y
- Suma de todos los elementos de las columnas pares.

218. A es un arreglo unidimensional de N elementos.
Elaborar un algoritmo para colocar el vector A como diagonal secundaria de un arreglo B de N * N.
219. Elaborar un algoritmo que lea una matriz A de orden N * N y construya otra matriz B, así:

- Los elementos de la diagonal principal y secundaria son los mismos de A.
- En los demás lugares deben ir los múltiplos de 5 de izquierda a derecha y de arriba hacia abajo.

220. Hacer un algoritmo que construya y escriba la siguiente matriz:

126	120	114	108	102	96
	90	84	78	72	66
		60	54	48	42
			36	30	24
				18	12

221. Se tiene un grupo de N estudiantes identificados como 1, 2, 3... N; cada uno de ellos toma las mismas 8 materias: 1, 2, 3... 8; semestralmente, el profesor de cada materia reporta a la oficina de registros el código del estudiante, el número de la materia y la calificación obtenida. Elaborar un algoritmo que obtenga:

- Las calificaciones promedio obtenidas por cada estudiante, mostradas en forma ascendente.
- El promedio de notas en cada uno de 8 cursos.
- El estudiante con mayor promedio.

222. Leer una matriz B de 38 * 38 y efectuar los siguientes pasos:

- Intercambiar los elementos de la diagonal principal con los elementos de la diagonal secundaria.
- Intercambiar la fila 17 con la columna 14.
- Escribir la nueva matriz.

223. Una universidad tiene 12 carreras codificadas del 1 al 12 y cada carrera tiene 10 semestres numerados del 1 al 10. Semestralmente cada carrera elabora un registro que contiene: código de la carrera y, a continuación, los 10 costos correspondientes a los 10 semestres. Elaborar un algoritmo que calcule e imprima:

- Costos totales y promedio por semestre de todas las carreras.
- Costos por carrera en forma descendente.
- Costo global de la universidad.
- El semestre menos cuantioso.

224. Escribir un algoritmo que lea una ficha con los síntomas de un enfermo y de acuerdo con la siguiente tabla muestre la enfermedad del paciente:

ENFERMEDAD	SINTOMAS
1	FFFFT
2	TTTTT
3	TTTFF
4	FTTFF
5	FFTFT
.	.
.	.
30	TFTTF

Si los síntomas leídos no están en las tablas, se debe mostrar un mensaje de error.

225. Elaborar un algoritmo que lea un arreglo tridimensional y encuentre la suma de sus elementos.
226. Hacer un algoritmo que encuentre la suma de tres arreglos tridimensionales.
227. Preparar un algoritmo que inserte una matriz al final de un arreglo de tres dimensiones y lo imprima.
228. Dado un arreglo tridimensional cúbico, calcular e imprimir la suma de los elementos ubicados en su diagonal principal.
229. Dado cualquier arreglo tridimensional del cual se sabe que sus elementos son ceros, elaborar un algoritmo que forme sus aristas con unos.
230. Elaborar un algoritmo que entre un arreglo tridimensional cúbico y encuentre la suma de los elementos que pertenezcan a cualquiera de las diagonales.
231. Hacer un algoritmo que forme un arreglo tridimensional, cuyos elementos estén ordenados con base en las posiciones fila y columna dadas.
232. Un arreglo tridimensional contiene la cantidad de hombres y mujeres que hay en 6 cursos de cada una de las 10 facultades de una universidad. Donde SEXO [I, J, K] representa el número de estudiantes del curso I, de sexo J, de la facultad K. Encontrar:

- Cantidad de hombres en cada facultad.
- Cantidad de mujeres por facultad.
- La facultad con mayor número de mujeres.

233. Elaborar un algoritmo que sume dos arreglos tridimensionales, elemento a elemento.
234. Dado un arreglo de cuatro dimensiones llenarlo con ceros, excepto los primeros elementos de la cuarta dimensión que deben ser unos.

Capítulo 9

REGISTROS Y ARCHIVOS

9.1. Registros

Un registro es una estructura de datos compuesta. Se puede decir que un registro es un conjunto de campos variables relacionadas que, en general, pueden pertenecer a tipos de datos diferentes, llamadas componentes del tipo registro, donde todas las componentes pueden manipularse bajo un solo nombre de variable. Por ejemplo, si se tiene un registro de datos compuesto por los campos: cédula, nombres, deducción y salario, podemos representarlo de la siguiente forma:

32506321 SANDRA FONNEGRA 40000 630000

Como puede notarse, dentro del registro sólo hay datos (información); es el, programador quien le coloca nombres a cada dato para poderlos almacenar en la memoria de la computadora.

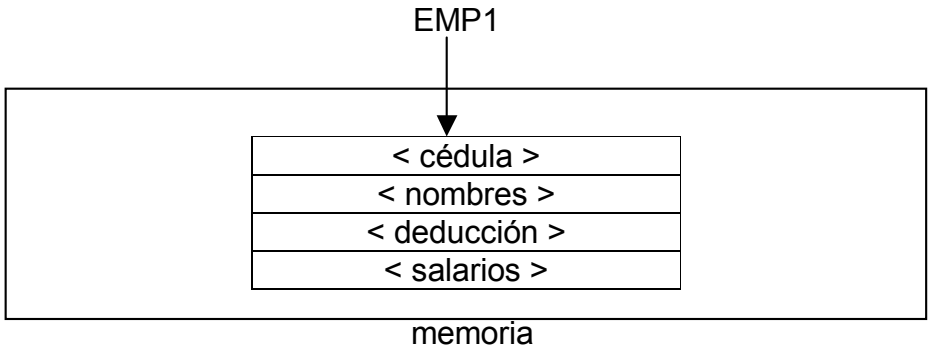
Si la información del empleado se fuera a tratar en forma individual y los nombres de variables seleccionados para cada campo fueran: CC, NOM, DEDUC y SAL, al ejecutarse la instrucción:

LEA: CC, NOM, DEDUC, SAL

Ocurrirá en memoria lo siguiente:

FALTA GRÁFICO PAG. 280 LIBRO

La información de la empleada está dispersa en la memoria. La idea de usar registros es agrupar toda la información en una misma área de memoria bajo un solo nombre. Si se toma la determinación de que EMP1 es una variable de *tipo registro*, o sea, puede almacenar en el área asignada toda la información de un empleado, gráficamente se puede mirar de la siguiente manera.



Para que EMP1 sea tomada como una variable tipo registro, el programador debe definirla como tal, de acuerdo al lenguaje de programación que esté utilizando; es decir, el *tipo de dato*

registro no está predefinido, ya que sus componentes van a depender de cada programa en especial. A modo de ejemplo, el anterior registro en C++ se definiría así:

```
struct empleado
        {
          long cc;
          char nom[36];
          double deduc;
          double sal;
        };
```

Lo anterior quiere decir que el programador ha definido un tipo de dato registro llamado *empleado* y que a cualquier variable que se le asigne el tipo de dato empleado, puede almacenar los valores de las componentes: cc, nom, deduc y sal. En C++ se haría de la siguiente forma:

```
empleado emp1, emp2;
```

En la anterior instrucción se define a emp1 y emp2 como variables tipo registro, por lo tanto, pueden almacenar todos los valores de los campos que conforman al tipo de dato empleado, y su representación interna sería:

FALTA GRÁFICO PAG. 281 LIBRO

Las variables tipo registro emp1 y emp2 son variables compuestas; para referenciar a cada una de sus componentes hay que cualificarlas, o sea, decir en forma explícita a cual de las componentes del registro se quiere uno referir. La cualificación se hace así: **variables_tipo_registro.componente.**

Por ejemplo:

emp1.cc	hace referencia al valor de la cédula almacenada en el área de datos (registro) emp1.
emp2.nom	hace referencia al nombre que hay en el área de dato emp2.
emp1.nom[4]	referencia al carácter cuarto del nombre almacenado en el área de datos emp1, (en C++ sería el quinto).

La instrucción:

LEA: emp1.cc, emp1.nom, emp1.deduc, emp1.sal, lleva información a la variable tipo p registro emp1, como lo muestra la figura siguiente:

FALTA GRÁFICO PAG. 281 LIBRO

Si hay necesidad de mover la información de una variable tipo registro a otra, se puede hacer componente a componente o moviendo toda la información al mismo tiempo. Por ejemplo:

```
emp2.cc = emp1.cc
emp2.nom = emp1.nom
emp2.deduc = emp1.deduc
emp2.sal = emp1.sal
```

sería lo mismo que tener: emp2 = emp1.

Ejercicio resuelto No. 86

Formar un registro con los campos: cédula, nombre, pago, luego pasarlo a otro registro e imprimir la información.

Análisis

Datos de entrada
- Cédula.
- Nombre.
- Pago.

Datos de salida
- Cédula.
- Nombre.
- Pago.

Proceso
Hay que definir las dos variables tipo registro *est1 y est2,* se leen los valores de las componentes est1, luego se pasa a est2 y se imprimen.

Definición de variables
est2 y est2: Variables tipo registro.
cc: Componente cédula.
nom: Componente nombre.
pago: Componente pago.

Algoritmo

```
INICIO
    LEA: est1.cc, est1.nom, est1.pago
    est2 = est1
    ESCRIBA: est2.cc, est2.nom, est2.pago
FIN_INICIO
```

9.1.1. Arreglo de registros

Es una estructura de datos de gran utilidad e importancia en la programación de aplicaciones, ya que muchas veces es deseable, desde el punto de vista de la lógica del programa en particular, mantener disponible en memoria principal una serie de registro con información del mismo tipo para su proceso; dichos registros no estarán dispersos; harán parte de un arreglo, lo que facilita su manejo.

Un arreglo de registros es muy parecido a un archivo de datos (tema próximo a tratar) se diferencian en que los arreglos de registros residen en memoria principal y los archivos en memoria auxiliar, por lo tanto, estos últimos tiene existencia permanente.

Otra importancia que tienen los arreglos de registro es que cada uno de sus elementos está formado por varios campos pertenecientes a diferentes tipos de datos, a diferencia de los otros tipos de arreglos donde sus elementos son de un solo tipo.

Ejercicio resuelto No. 87

Formar un arreglo de máximo 100 registros donde cada elemento contenga: cédula, nombres y pago, luego ordenarlo por el campo nombre, insertar un nuevo registro sin dañar el ordenamiento e imprimirlo.

Análisis

Datos de entrada
- Los campos cédula, nombre y pago.

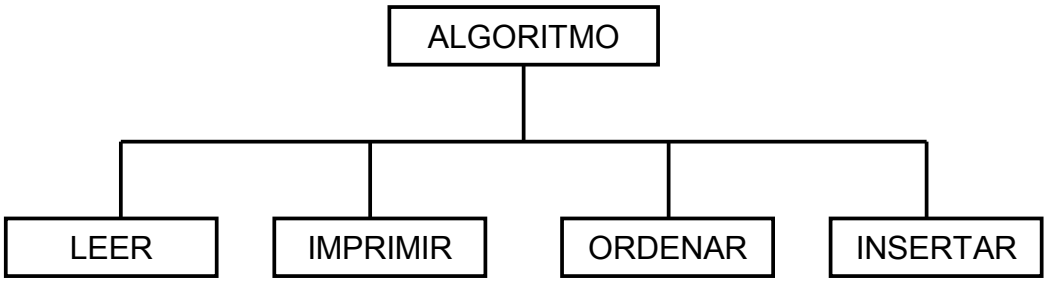
Datos de salida

- La información del arreglo de registros ordenada alfabéticamente.

Proceso
La información cédula, nombre y pago será un registro y cada registro será un elemento del arreglo. Si el arreglo lo llamamos V y sus componentes: cc, nom y P, gráficamente se vería así:

V									
V[1]			V[2]			. . . . .	V[100]		
V[1].cc	V[1]. nom	V[1].p	V[2].cc	V[2]. nom	V{2].p	. . . . .	V[100]. cc	V[100]. nom	V[100]. p

El problema se resolverá a través de módulos (subprogramas)



El subprograma LEER recibe la cantidad de elementos del arreglo (número de estudiantes) N y envía el vector V.

IMPRIMIR. Recibe N y V.
ORDENAR. Recibe N y V y envía V ordenado.
INSERTAR. Recibe N y V y envía N y V.

Otras variables

I, K: Variables de control de ciclos.
MENOR: Almacena el nombre menor en el ordenamiento.
POS: Almacena la posición donde está el nombre menor.
AUX: Variable de tipo registro que guarda temporalmente un elemento del vector mientras se hace el ordenamiento.
RI: Registros a insertar.

```
PROCEDIMIENTO LEER (N, V)
    PARA I = 1, N, 1 HAGA
            LEA: V[I].CC, V[I].NOM, V[I].P
    FIN_PARA
FIN_LEER

PROCEDIMIENTO IMPRIMIR (N, V)
    PARA I = 1, N, 1 HAGA
            ESCRIBA: V[I].CC, V[I].NOM, V[I].P
    FIN_PARA
FIN_IMPRIMIR

PROCEDIMIENTO ORDENAR (N,   V)
    PARA I = 1, N-1, 1 HAGA
```

```
            MENOR = V[I].NOM
            POS = I
            PARA K = I+1, N, 1 HAGA
                  SI MENOR > V[I].NOM ENTONCES
                        MENOR = V[K].NOM
                        POS = K
                  FIN_SI
            FIN_PARA
            AUX = V[I]
            V[I] = V[POS]
            V[POS] = AUX
      FIN_PARA
FIN_ORDENAR


PROCEDIMIENTO INSERTAR ( N,    V,    RI )
      I = 1
      MIENTRAS (I <= N) ∧ (RI.NOM > V[I].NOM) HAGA
            I = I+1
      FIN_MIENTRAS
      N = N+1
      PARA K = N, I+1, -1 HAGA
            V[K] = V[K-1]
      FIN_PARA
      V[K] = RI
FIN_INSERTAR
```

Algoritmo principal

Datos de entrada

- La cantidad de elementos del vector.
- La información del vector.
- Cédula, nombre y pago a insertar.

Datos de salida

- La información del vector registro.

Definición de variables

N:	Cantidad de elementos.
A:	Nombre del arreglo.
R:	Variable tipo registro.
R.CC:	Cédula a insertar.
R.NOM:	Nombre a insertar.
R.P:	Pago a insertar.
OP:	Opción a seleccionar en el menú.
SIGA:	Opción para insertar nuevos elementos.

Algoritmo

```
INICIO
      LEA: N
      REPITA
            ESCRIBA: «1: LEER EL ARREGLO»
            ESCRIBA: «2: IMPRIMIR ARREGLO»
            ESCRIBA: «3: ORDENAR ARREGLO»
```

```
            ESCRIBA: «4: INSERTAR ELEMENTO»
            ESCRIBA: «5: SALIR»
            ESCRIBA: «ELIJA OPCIÓN»
            LEA OP
            CASOS DE OP
                    CASO1: LEER (N, A)
                    CASO2: IMPRIMIR (N, A)
                    CASO3: ORDENAR (N, A)
                    CASO4: REPITA
                            LEA: R.CED, R.NOM, R.P
                            INSERTAR (N, A, R)
                            ESCRIBA: «DESEA INSERTAR OTRO
                            LEA: SIGA
                            MIENTRAS SIGA = 'S'
                    CASO5: ESCRIBA: «FIN DEL PROGRAMA»
    MIENTRAS OP <> 5
FIN_INICIO
```

9.2. Archivos

Un archivo es un dispositivo lógico donde se pueden almacenar en forma permanente grandes volúmenes de información, dividido en estructuras de datos (registros) todas del mismo tipo y relacionadas.

El manejo de información a través de archivos es de gran utilidad en la programación, ya que permite la captura de información almacenada en ellos para su posterior utilización en diversos programas, evitando tener que entrar en forma manual la información que se quiere procesar; y almacenar gran cantidad de información en espacios reducidos y con una alta precisión en el manejo de los datos.

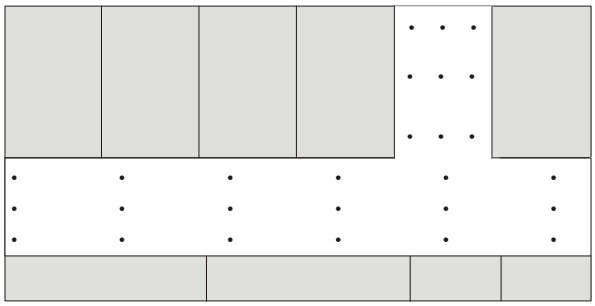
En general los archivos se pueden agrupar en tres grandes categorías:

- *Archivos de programa:* son los programas fuentes que se escriben en un determinado lenguaje de programación. Estos programas son guardados en medios magnéticos auxiliares para su posterior utilización. Estos archivos se pueden cargar del medio magnético a memoria, compilarlos, ejecutarlos, imprimirlos y volverlos a guardar donde estaban o en otro medio de almacenamiento distinto.
- *Archivos de texto:* estos almacenan letras, palabras, frases, párrafos y se suelen crear y mantener mediante programas que procesan o editan texto.
- *Archivos de datos:* son la clase de archivos que se tratarán en este capítulo y, como su nombre lo indica, éstos almacenan valores. Un archivo de datos contiene información que un programa crea y posteriormente procesa; de igual forma puede usarse para almacenar los resultados de la ejecución de un programa.

9.2.1. Componentes de un archivo

Un archivo es un conjunto de datos relacionados entre sí, con una organización jerárquica, de modo que cada archivo esté formado por un conjunto de registros y estos, a su vez, formados por una serie de campos. Un registro es la unidad de medida de un archivo, que contiene la información que se utiliza en los diferentes programas y son tratados de manera independiente.

Un registro puede contener dentro del conjunto de campos que lo conforman, uno que hace que el registro en particular sea único, es decir, lo diferencie de los demás llamado campo clave. Gráficamente se puede mirar un archivo de datos de la siguiente forma:



Como se desprende del gráfico, los campos corresponden a las columnas y los registros a las filas.

El tamaño o longitud de un registro se mide en bytes y será la resultante de sumar las longitudes de los campos que lo conforman. Es importante tener en cuenta que los campos que conforman un registro deben referenciar una misma información y de igual manera los registros deben estar relacionados con la razón de ser del archivo.

Empleados

Codigo	Depto	Nombre	Salario
114	01	Elías José	180 000
121	01	Graciela del Carmen	253 423
211	01	Luisa Berta	312 516
099	02	Guillermo Segundo	222 604
117	02	Albertina Isabel	191 221
212	02	Corina Francisca	823 514
119	03	Ruth de Jesús	119 325
118	03	Florencia Maria	425 000
311	04	Jairo Antonio	388 734

Fin del archivo

Como se puede ver, en un archivo de datos lo que se guarda es información, es el programador quien decide qué nombre le coloca al registro y a cada uno de los campos individuales que integran el registro. Otros ejemplos:

Clientes

Cédula	Nombre	Apellido	Dirección	Teléfono
14879563	Fredy	Jaramillo	Cra. 10 No. 10-17	2310145
47896321	Leonel	Osorno	Cll. 70 No. 11-18	4569874
65874123	Fabián	Ríos	Cll. 20 No. 12-9	5896201
73284562	Gisela	Gil	Cra. 43 No. 10-21	3214568

EOF

Inventario

Código	Descripción	Unidades	Valor-unidad
702110	Bujía	200	7500
802010	Válvula	50	850
902110	Manguera	5	35750

.	.	.	.
.	.	.	.
.	.	.	.

EOF

Libros

Código-libro	**Título**	**Código-autor**	**Unidades**
502115	Antes del fin	8462148	12
602017	Doña Bárbara	7985621	21
902117	El túnel	4568932	32
.	.	.	.
.	.	.	.
.	.	.	.

EOF

EOF significa End Of File (Fin del Archivo), es un indicador que aparece luego del último registro del archivo, denotando el final del mismo.

9.2.2. Clasificación de los archivos según su uso

- *Archivos permanentes:* son archivos rígidos, ya que la información que almacenan cambia muy poco; son usados para extraer información que se utiliza en otros archivos o procesos. Por ejemplo, el archivo que contenga toda la información sobre los salarios de los empleados de una empresa
- *Archivos maestros:* la información almacenada en esta clase de archivos maneja el estado o situación de una entidad o algún aspecto de ella, y su actualización o cambios de los datos de los registros se hace periódicamente. Por ejemplo, un archivo que almacene el inventario de los bienes de una empresa.
- *Archivos históricos:* son archivos maestros que ya no se utilizan y que su función es sólo de consulta para quienes desean obtener información sobre el pasado. Por ejemplo, la información contenida en las hojas de vida académica de los estudiantes que terminaron su carrera en la década pasada.
- *Archivos de movimiento:* son temporales y su función principal es captar información para actualizar los archivos maestros. Sus registros muestran las transacciones o movimientos que se producen durante un determinado período. Cuando el archivo maestro se actualiza, usando los archivos de movimiento, éstos últimos pierden su validez y se pueden destruir. Por ejemplo: los cambios de aumento de salarios, deducciones y sobresueldos producidos durante un determinado periodo.

9.2.3. Concepto de clave

La clave o llave principal es el campo o la combinación de campos del archivo que permiten identificar o diferenciar plenamente cada registro de los demás. Cada archivo debe tener una clave o llave principal, con el fin de poder encontrar y recuperar la información de un registro particular en un momento dado. Para los ejemplos anteriores analicemos cuál o cuáles campos conforman las claves de los archivos y porqué:

Clientes

Campo	**Clave?**	**Explicación**
Teléfono	NO	Dos clientes diferentes pueden tener el mismo número telefónico. Por ejemplo, una pareja de esposos.

Dirección	NO	Dos clientes diferentes pueden compartir la misma dirección.
Apellido	NO	Dos clientes diferentes pueden tener el mismo apellido.
Nombre	NO	Dos clientes diferentes pueden tener el mismo nombre.
Cédula	SI	La cédula es única y diferente para cada cliente.

Inventario

Campo	Clave?	Explicación
Valor-unidad	NO	Dos o más repuestos pueden tener el mismo precio.
Unidades	NO	Es posible que las existencias de dos o más repuestos coincidan.
Descripción	NO	Aunque es poco frecuente, es posible que dos repuestos diferentes tengan la misma descripción.
Código	SI	A cada repuesto corresponde un código que lo diferencia de los demás. Lógicamente este código es diferente para cada tipo de repuesto.

Libros

Campo	Clave?	Explicación
Unidades	NO	Es posible que de dos o más libros se tengan las mismas existencias.
Título	NO	Dos libros diferentes pueden tener el mismo título.
Código-autor	NO	Un autor puede escribir dos o más libros.
Código-libro	SI	A cada libro corresponde un código que lo diferencia de los demás. Este código es único y diferente para cada libro.

¿Cuál sería el campo clave en el archivo de empleados?

a. *Clave simple:* un archivo tiene clave simple si ésta está conformada por un solo campo. En los casos anteriores, todas las claves son simples. Para diferenciarlas de los demás atributos, utilizaremos el carácter especial @ como prefijo. Por ejemplo:

   @Cédula (Clientes), @Código (Inventario), @ Código-libro (Libros) son claves simples.

b. *Clave compuesta:* es aquella que está conformada por dos o más campos del archivo. Por ejemplo: supóngase un archivo de préstamos bibliotecarios con la siguiente estructura:

Préstamo-Blibliotecario = {Cédula-alumno, Código-libro, Fecha-préstamo, Fecha-devolución}

Analicemos ahora cuál o cuáles atributos son candidatos a clave del archivo y porqué:

Campo	Clave?	Explicación
Fecha-devolución	NO	Dos o más libros diferentes pueden ser regresados en la misma fecha.
Fecha-préstamo	NO	Dos o más libros diferentes pueden ser prestados en la misma fecha.
Código-libro	NO	Si consideramos que el archivo debe almacenar la historia de los préstamos, entonces el mismo libro pudo haber sido prestado a diferentes estudiantes.
Cédula-alumno	NO	Un alumno puede hacer préstamos de dos o más libros diferentes.

Agotada la posibilidad de establecer una clave simple, debemos ahora procurar establecer una clave compuesta. Luego de descartar opciones como:

Campos	Clave?	Explicación
Cédula-alumno + Código-libro	NO	Un alumno puede prestar el mismo libro en ocasiones diferentes.
Cédula-alumno + fecha-préstamo	NO	Un alumno puede prestar diferentes libros en la misma fecha.
Cédula-alumno + fecha-devolución	NO	Un alumno puede devolver diferentes libros en la misma fecha

Llegamos al siguiente análisis:

Campos	Clave?	Explicación
Cédula-alumno + Código-libro + Fecha-préstamo	SI	Sólo existe una ocurrencia del registro en todo el archivo. Se asume que no es posible que un libro sea prestado dos veces en un mismo día.

Por lo tanto podemos afirmar que:

Préstamo-Blibliotecario = {@Cédula-alumno, @Código-libro, @Fecha-préstamo, Fecha-devolución}

Los campos que están dentro de las llaves son los componentes del registro.

9.2.4. Almacenamiento de archivos y su acceso a la información

Existe una relación directa entre la forma como se almacena la información en un archivo y la manera como se recupera su información posteriormente. El medio de almacenamiento determina el modo y velocidad de acceso a los datos. En forma general se puede decir que hay tres métodos para recuperar o acceder a la información de un archivo: secuencial, secuencial indexado y aleatoria o al azar.

a. *Acceso secuencial:* En este caso los registros se procesan en el orden en que se guardaron en el archivo. Esta clase de archivos se utiliza cuando el volumen de información es grande y sus datos no cambian continuamente, los registros ocupan posiciones de memoria consecutivas y la forma de acceder a ellos es de uno en uno a partir del primero, no se puede ir en forma directa como en los arreglos o en la otra clase de organización a un determinado registro. Los registros se almacenan en forma ordenada, ascendente o descendente por su campo clave. La clave en este caso no guarda ninguna relación con su posición física que ocupa en el medio donde se almacena. En síntesis, éste método hace que los registros se examinen uno después de otro (si se quiere buscar la información del registro doce, hay que examinar la información de los primeros once registros). Aunque los archivos secuenciales ocupan poco espacio, su proceso es más lento comparado con otros métodos de almacenamiento, como el aleatorio por ejemplo. La actualización de estos archivos puede hacerse rescribiendo un nuevo registro en el lugar que ocupa un registro que ya no se necesita o sacando una nueva copia del archivo, donde sólo se tienen en cuenta los registros que se necesitan.

   Para crear un nuevo archivo actualizado se deberá leer del archivo existente cada uno de los registros, cambiar la información en memoria principal y luego almacenarlos en un nuevo archivo. Los registros que no se necesitan modificar se pasarán al nuevo archivo, tal como están en el archivo antiguo. Por ejemplo, si se tiene el siguiente archivo maestro de los clientes de una empresa.

Clientes

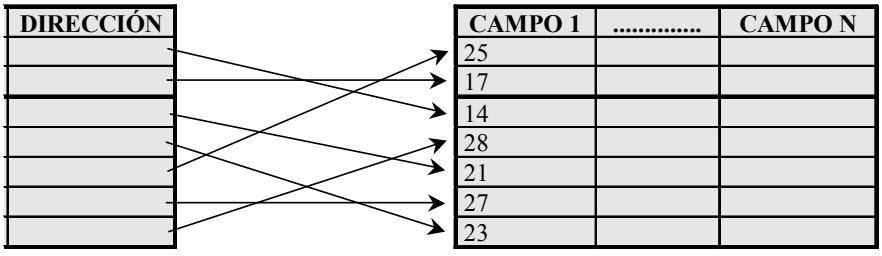
Cédula	Nombre	Apellido	Dirección	Teléfono
14879563	Fredy	Jaramillo	Cra. 10 No. 10-17	2310145
47896321	Leonel	Osorno	Cll. 70 No. 11-18	4569874
65874123	Fabián	Ríos	Cll. 20 No. 12-9	5896201
73284562	Gisela	Gil	Cra. 43 No. 10-21	3214568
	EOF			

Si los clientes Leonel y Gisela cambian de teléfono, es necesario actualizar esos cambios en el archivo de clientes. Como se dijo, una de las formas de hacer estos cambios en el acceso secuencial es pasar la información del archivo a uno nuevo, o sea, hacer lo siguiente: leer el registro de Fredy (pasarlo del medio de almacenamiento donde se encuentra el archivo clientes a memoria principal); como no hay que hacerle cambios se pasa tal como está al nuevo archivo. Luego se lee el registro de Leonel, se cambia el teléfono por 4213825 y se pasa al nuevo archivo; a continuación se lee el registro de Fabián y se pasa tal como está; y por último se lee el registro de Gisela, se cambia el número telefónico por 2867767 y se pasa al nuevo archivo. Este nuevo archivo se guarda y se borra el antiguo, si se quiere se cambia el nombre nuevo por clientes.

Nuevo

Cédula	Nombre	Apellido	Dirección	Teléfono
14879563	Fredy	Jaramillo	Cra. 10 No. 10-17	2310145
47896321	Leonel	Osorno	Cll. 70 No. 11-18	4213825
65874123	Fabián	Ríos	Cll. 20 No. 12-9	5896201
73284562	Gisela	Gil	Cra. 43 No. 10-21	4518720
	EOF			

b. *Acceso secuencial indexado:* es una variante en la organización y recuperación de información en forma secuencial. En este caso los registros se encuentran ordenados por un campo clave y éste determina su posición en el archivo. Cada registro se reconoce por su campo clave; para esto utiliza estructura auxiliar conocida como índice. Los índices se presentan de una manera cotidiana en nuestras vidas. Por ejemplo: la búsqueda de una palabra en un diccionario se hace más rápida, gracias al índice que éste presenta. Los índices son una estructura formada con los valores de indexación, o clave, mediante la cual se hace la búsqueda, las direcciones que corresponden a la localización física y el método de indexación usado para realizar la búsqueda. Gráficamente se pueden ver de la siguiente manera:



Las búsquedas se realizan a través del índice, lo cual resulta sustancialmente más eficiente que una búsqueda secuencial. En el ejemplo anterior podría pensarse en una búsqueda binaria,

aunque realmente existen métodos de búsquedas aún más eficientes utilizando estructuras un poco más complejas, como árboles, entre otras.

A diferencia de los archivos secuenciales no es necesario leer los n-1 registros precedentes para leer el enésimo. Los métodos de indexamiento, generalmente, no son conocidos por el programador, es decir, la búsqueda es llevada a cabo gracias a funciones de los lenguajes de programación, por lo tanto éstos son transparentes para el usuario.

c. *Acceso aleatorio:* Este tipo de organización permite que exista una relación especial entre la clave de un registro y la dirección real de la clave; esto hace que se pueda leer y escribir un registro en forma aleatoria.

La forma de acceder a un registro es mediante su posición en el medio de almacenamiento, es decir indicando el lugar que ocupa en el conjunto de registros, debido a que cada clave tiene una única dirección. Primero se localiza la dirección y luego se accede a la información del registro. De todos modos, la relación clave-dirección dependerá de los datos del archivo y del dispositivo de almacenamiento.

Una ventaja de los archivos de acceso directo es la rapidez con que se llega a un determinado registro. Si se desea acceder al registro noventa y nueve, se va directamente a ese registro sin necesidad de recorrer los noventa y ocho registros anteriores; esta ventaja se descompensa con la gran cantidad de espacio que consume.

El acceso directo se utiliza cuando las modificaciones son continuas. Por ejemplo: Hacer las consignaciones o retiro de una entidad bancaria.

9.2.5. Procesamiento de archivos

Antes de entrar en detalle, es necesario, tener claros los siguientes conceptos:

- *Apuntador:* es un indicador lógico que los sistemas usan para identificar el registro que se quiere referenciar en un momento determinado. Cuando se inicia el proceso, el apuntador se coloca al principio del primer registro del archivo; y cada vez que se haga un proceso de lectura o escritura sobre el archivo, el apuntador se mueve al principio del registro siguiente.

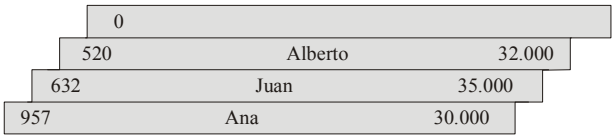
- *Marca de fin de archivo (End Of File o simplemente EOF):* cuando se termina de hacer un proceso con un archivo, es necesario hacer la operación de cerrar el archivo; ésta hace que al archivo se le adicione un nuevo registro donde se coloca una marca, conocida como marca de fin de archivo, utilizada para saber cuándo se llega al final de los registros que contienen información. Esta marca se comporta como un registro centinela y puede interpretarse como que se llegó al final del archivo o no hay más datos. La marca permite implementar un ciclo cualitativo para recorrer el archivo que finalizará cuando ésta sea detectada.

a. *Crear el archivo.* Permite relacionar una variable tipo archivo, llamada nombre interno del archivo, con el nombre que el archivo tendrá en el medio de almacenamiento, llamado nombre externo. La instrucción:
   CREAR (A, ”C:\DATOS.DAT”): genera una conexión entre el nombre interno A y el nombre externo del archivo, donde: A es la variable con la que se manejará el archivo en el programa y DATOS.DAT es el nombre que se le dará al archivo en el disco duro.
b. *Abrir el archivo.* ABRIR(A): consiste en desproteger el área donde residirá o reside el archivo, para poder tener acceso a la información; es decir, el archivo queda a disposición del programa que lo ha abierto y el apuntador se ubica al principio del primer registro.
c. *Escribir en el archivo.* ESCRIBA(A,R): pasa la información que hay en la variable tipo registro R en memoria al archivo cuyo nombre externo está relacionado con A, o sea, adiciona un registro al archivo.
d. *Leer de un archivo.* LEA(A,R): pasa la información de un registro del archivo a memoria principal, el registro cuya información se transporta a la memoria puede ser: el referenciado

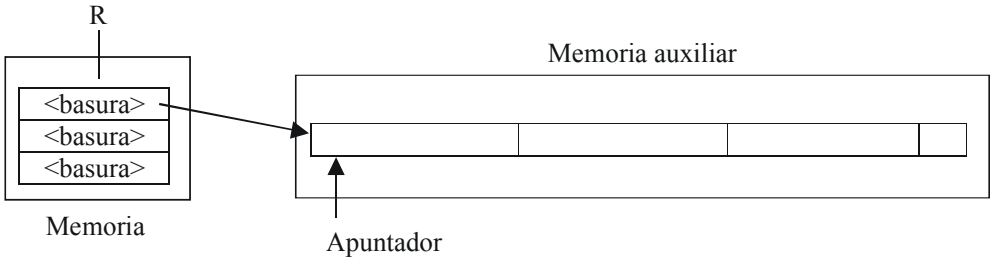
por el apuntador en el caso de un archivo de acceso secuencial, o el encontrado mediante algún método de búsqueda implementado por cada lenguaje, en el caso de un archivo indexado, o el registro hallado mediante un método de búsqueda implementado por el programador en el caso de un archivo directo. Si el archivo es secuencial cada vez que se lee un registro salta y posiciona el apuntador en el registro siguiente.

e. *Sobreescribir en el archivo.* SOBREESCRIBA(A,R): actualiza o modifica la información del registro que en el momento esté en memoria y lo lleva a la misma posición donde estaba.
f. *Cerrar el archivo.* CERRAR(A): esta instrucción hace lo siguiente:
   - Le da protección al área donde reside el archivo y actualiza el directorio del medio de almacenamiento reflejando el nuevo estado del archivo.
   - Destruye las conexiones físicas y lógicas que se construyeron cuando se abrió el archivo.
   - Adiciona un registro después del último grabado donde coloca la marca de fin de archivo.

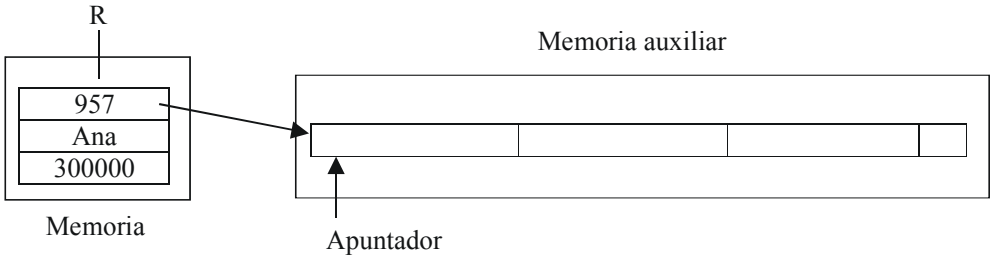
Ejemplo: si R es una variable tipo registro, cuyos componentes son: cédula, nombre y salario y A es el nombre interno de un archivo secuencial, donde se quiere guardar la siguiente información:



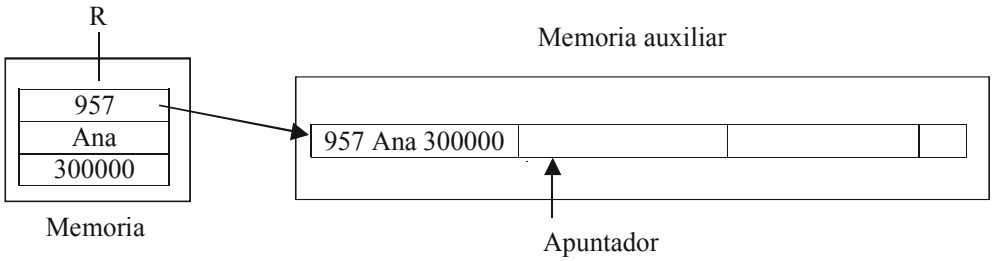
Al ejecutar ABRIR(A), ocurrirá lo siguiente:



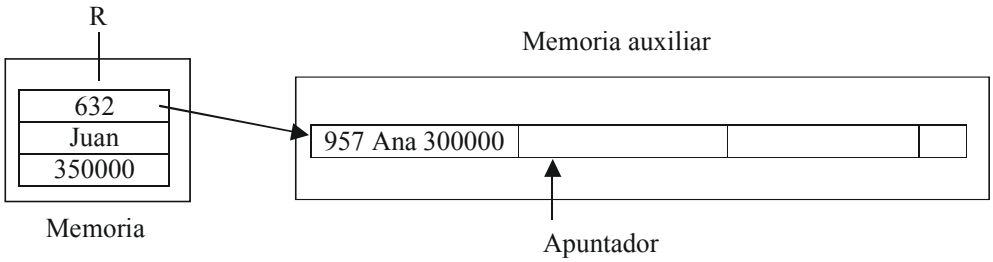
La instrucción: LEA: R.CC, R.NOM. R.SAL, pasa información desde un medio externo de entrada a la dirección de memoria donde reside la variable R.



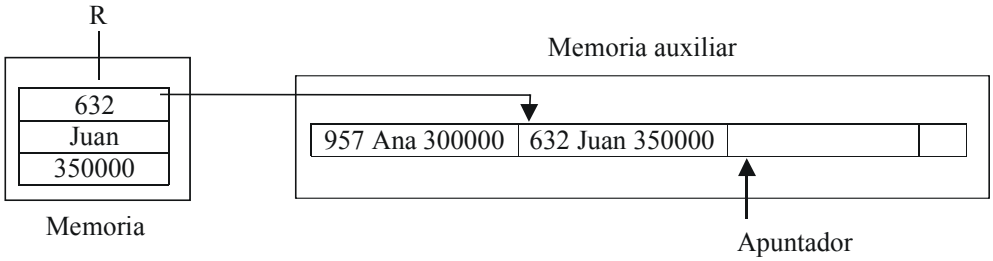
ESCRIBA(A, R), pasa la información almacenada en la variable R al archivo.



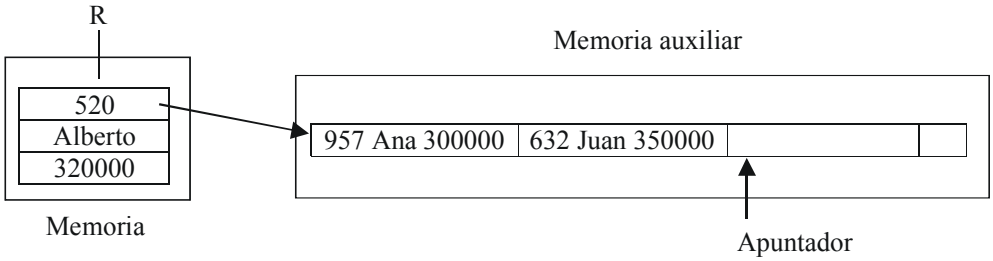
LEA: R.CC, R.NOM, R.SAL, pasa la información del segundo registro a memoria.



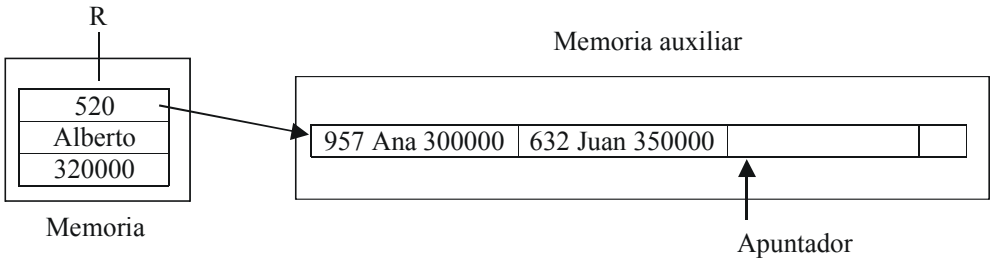
ESCRIBA(A, R).



LEA: R.CC, R.NOM. R.SAL.



ESCRIBA(A.R).



Al ejecutarse la instrucción: CERRAR(A), ocurre lo siguiente:

957 Ana 300000	632 Juan 350000	520 Alberto 320000	/*
Registro 0	Registro 1	Registro 2	Registro con la marca

Sobre la marca EOF, se puede además decir lo siguiente:

- Es manejada como una constante lógica manejada internamente por la computadora, cuando el archivo es abierto su valor es FALSO y cuando se detecta la marca de fin de datos cambia a VERDADERO.
- El valor colocado en el registro depende de la arquitectura de la computadora y es transparente para el programador, sólo importa el valor que en un momento determinado tenga la constante EOF.
- Aunque el registro donde se coloca la marca hace parte del archivo, no se tiene en cuenta al encontrar el tamaño del archivo en cantidad de registros; por lo tanto, el archivo anterior tiene un tamaño de tres registros.

9.2.6. Operaciones básicas sobre archivos

a. *Adición:* ingresa nuevos registros al archivo. Antes de escribir un nuevo registro es preciso validar que éste no exista, utilizando la clave del archivo.
b. *Modificación:* actualiza la información de un registro existente.
c. *Retiro:* elimina un registro del archivo. Si el archivo es secuencial, generalmente se utiliza un "borrado lógico" (no físico, es decir, físicamente el registro sigue ocupando espacio de almacenamiento, lo cual podría considerarse como una desventaja). El borrado lógico consiste en la destinación de un campo del archivo para guardar el estado lógico de cada registro (Activo ó Inactivo), de tal forma que eliminar un registro sea tan simple como cambiar su estado de Activo a Inactivo (o de 1 a 0). Con archivos Indexados debe reorganizase o reindexarse el índice, lo cual a veces resulta una tarea compleja, pero afortunadamente es realizada por herramientas de los lenguajes de programación (los aspectos relacionados con métodos de indexación y búsqueda se encuentran fuera del alcance de este libro).

Operaciones sobre archivos secuenciales

Ejercicio resuelto No. 88

Dado el siguiente archivo secuencial:

ESTUDIANTE = {@CÓDIGO, NOMBRE, DIRECCIÓN, ESTADO}

Donde:
@: Clave
Código: Número de identidad del alumno
Nombre: Nombre del alumno
Estado: Indicador numérico del estado de cada registro 1 para activo ó 0 para inactivo.

a. Escribir un algoritmo que adicione un estudiante al archivo.
b. Un algoritmo que modifique la información de un estudiante en el archivo.
c. Algoritmo que retire un estudiante del archivo

Consideraciones:

- Archivo secuencial
- Clave: CÓDIGO
- Antes de adicionar un nuevo estudiante debemos verificar que no exista en el archivo
- Debemos tener presente el estado de cada registro, ya que es posible que el código exista, pero en caso de que el estado sea 0 (cero) esto nos supone el equivalente a ignorar el registro, es decir, es como si el registro no existiera.

a. Algoritmo

```
INICIO
    ESCRIBA: "DIGITE EL CÓDIGO DEL ESTUDIANTE A ADICIONAR:"
    LEA: CODIGOAD
    ENCONTRÓ = 0
    ABRIR (ESTUDIANTE)
    LEA: (ESTUDIANTE, R)
    MIENTRAS (!EOF(ESTUDIANTE)) ∧ (ENCONTRÓ = 0) HAGA
        SI (CODIGOAD = R.CÓDIGO) ∧ (R.ESTADO= 1) ENTONCES
            ENCONTRÓ = 1
        FIN_SI
        LEA: (ESTUDIANTE, R)
    FIN_MIENTRAS
    SI ENCONTRÓ = 1 ENTONCES
        ESCRIBA: " EL ESTUDIANTE YA EXISTE "
      SINO
        ESCRIBA: " DIGITE EL NOMBRE Y LA DIRECCIÓN DEL
        LEA: R.NOMBRE, R.DIRECCIÓN
        R.CÓDIGO = CODIGOAD
        R.ESTADO = 1
        ESCRIBA (ESTUDIANTE, R)
    FIN_SI
    CERRAR (ESTUDIANTE)
FIN_INICIO
```

Anotaciones importantes:

- En el algoritmo se asume que el archivo existe y tiene registros, de lo contrario debería ser creado con la operación Crear ().
- Tenga presente la diferencia entre la lectura de una variable y la lectura de un archivo (LEA(nombre archivo, variable tipo registro)) ambas tienen efectos completamente diferentes.
- EOF(nombre de archivo): Indicador que nos devuelve un valor lógico, indicando si se encontró o no el fin del archivo.
- !: Negación
- !EOF(): No Fin del archivo

b. Algoritmo que modifica la información de un estudiante.

```
INICIO
    ESCRIBA: "DIGITE EL CÓDIGO DEL ESTUDIANTE A MODIFICAR»
```

```
    LEA: CODMOD
    ENCONTRÓ = 0
    ABRIR (ESTUDIANTE)
    LEA (ESTUDIANTE, R)
    MIENTRAS (!EOF(ESTUDIANTE)) ∧ (ENCONTRÓ = 0)  HAGA
        SI (CODMOD = R.CÓDIGO) ∧ (R.ESTADO = 1) ENTONCES
            ENCONTRÓ = 1
        FIN_SI
        SI ENCONTRÓ = 0)  ENTONCES
            LEA: (ESTUDIANTE, R)
        FIN_SI
    FIN_MIENTRAS
    SI ENCONTRÓ = 0 ENTONCES
        ESCRIBA: "EL ESTUDIANTE NO EXISTE "
      SINO
        ESCRIBA: "DIGITE EL NUEVO NOMBRE Y LA NUEVA
        LEA: R.NOMBRE, R.DIRECCIÓN
        R.CÓDIGO = CODMOD
        R.ESTADO = 1
        SOBREESCRIBA: (ESTUDIANTE, R)
    FIN_SI
    CERRAR (ESTUDIANTE)
FIN_INICIO
```

c. Algoritmo que retira la información de un estudiante.

```
INICIO
    ESCRIBA: "DIGITE EL CÓDIGO DEL ESTUDIANTE A ELIMINAR"
    LEA: CODELIM
    ENCONTRÓ = 0
    ABRIR (ESTUDIANTE)
    LEA: (ESTUDIANTE, R)
    MIENTRAS (!EOF(ESTUDIANTE)) ∧ (ENCONTRÓ = 0) HAGA
        SI (CODELIM = R.CÓDIGO) ∧ (R.ESTADO = 1) ENTONCES
            ENCONTRÓ = 1
        FIN_SI
        SI (ENCONTRÓ = 0) ENTONCES
            LEA(ESTUDIANTE, R)
        FIN_SI
        FMQ
        SI (ENCONTRÓ = 0)  ENTONCES
            ESCRIBA: "EL ESTUDIANTE NO EXISTE"
          SINO
            ESCRIBA: "EL ESTUDIANTE SERÁ ELIMINADO DEL
            R.ESTADO = 0
            SOBREESCRIBA (ESTUDIANTE.R)
        FIN_SI
    FIN_MIENTRAS
    CERRAR (ESTUDIANTE)
FIN_INICIO
```

Ejercicio resuelto No. 89

Elaborar un subprograma que genere el archivo A3 = {@COD, NOM, PAGO, ESTADO}, que quede ordenado ascendentemente por el campo COD, cada vez que se le adicione un registro.

```
PROCEDIMIENTO ADICIONARO (A3)
        ABRIR (A3)
        LEA: CODIGOAD
        R.COD = CODIGOAD
        LECTURA (R) // LEE R. NOM Y R. PAGO
        ESCRIBA (A3, R)
        CERRAR (A3)
        LEA:CODIGOAD
        MIENTRAS CODIGOAD <> 0  HAGA
                ABRIR (A3)
                LEA: (A3, R)
                ENCONTRÓ = 0
                MIENTRAS (!EOF(A3) ∧ (ENCONTRÓ = 0) HAGA
                        SI (CODIGOAD = R.COD) Ù (R.ESTADO =
                                ENCONTRÓ = 1
                          SINO
                                LEA: (A3, R)
                        FIN_SI
                FIN_MIENTRAS
                CERRAR (A3)
                SI ENCONTRÓ = 1 ENTONCES
                        ESCRIBA:"EL ESTUDIANTE YA EXISTE"
                  SINO
                        ABRIR (A3)
                        R.COD = CODIGOAD
                        LECTURA (R)
                        ESCRIBA (A3, R)
                        NR = TAMAÑO DE (A3)
                        I = NR   SW = 0
                        CERRAR (A3)   ABRIR (A3)
                        MIENTRAS (I > 0) ∧ ( SW = 0)
                                POSICIÓN (A3, I-1)
                                LEA: (A3, R1)
                                SI CODIGOAD < R1.COD ENTONCES
                                        POSICIÓN (A3, I)
                                        ESCRIBA: (A3, R1)
                                  SINO
                                        SW = 1
                                FIN_SI
                        FIN_MIENTRAS
                        POSICIÓN (A3, I)
                        ESCRIBA: (A3, R)
                        CIERRE (A3)
                FIN_SI
                LEA: CODIGOAD
        FIN_MIENTRAS
FIN_ADICIONARO
```

La función TAMAÑO, encuentra el tamaño de un archivo en cantidad de registros y POSICIÓN(A,I), ubica el apuntador que recorre el archivo A al inicio del registro I, hay que tener en cuenta que los registros están numerados de 0 a N-1

Ejercicio resuelto No. 90

Elaborar un subprograma que cree el archivo A3 = {@ COD, NOM, PAGO, ESTADO}, ordene alfabéticamente el archivo A3, dejando su contenido en el nuevo archivo A4.

```
PROCEDIMIENTO ORDENALFA (A3, A4)
    ABRIR (A3)   ABRIR (A4)
    ADICIONARO (A3)
    LEA: (A3, R)
    MIENTRAS (!EOF (A3))
            ESCRIBA (A4, R)
            LEA (A3, R)
    FIN_MIENTRAS
    CERRAR (A3)   CERRAR(A4)
    ORDENAR (A4)
FIN_ORDENALFA

PROCEDIMIENTO ORDENAR (A4)
    N = TAMAÑO DE (A4)
    ABRIR (A4)
    PARA I = 0, N-1, 1  HAGA
            POSICIÓN (A4, I)
            LEA: (A4, EST1)
            POS = I   NOMEN = EST1.NOM
            PARA K = I+1, N, 1  HAGA
                    POSICIÓN (A4, K)
                    LEA: (A4, EST2)
                    SI NOMEN > EST2.NOM  ENTONCES
                            POS = K   NOMEN = EST2.NOM
                    FIN_SI
            FIN_PARA
            POSICIÓN (A4, POS)
            LEA (A4, EST2)
            POSICIÓN (A4, I)
            ESCRIBA: (A4, EST2)
            POSICIÓN (A4, POS)
            ESCRIBA (A4, EST1)
    FIN_PARA
FIN_ORDEN
```

Operaciones sobre archivos secuenciales indexados

Ejercicio resuelto No. 91

Si se tiene el archivo secuencial ESTUDIANTE, indexado por el atributo código.

ESTUDIANTE = {@CÓDIGO, NOMBRE, DIRECCIÓN}

a. Elaborar un algoritmo que adicione un estudiante al archivo.
b. Algoritmo que modifique la información de un estudiante
c. Algoritmo que retire a un estudiante del archivo

Consideraciones:

- Archivo secuencial Indexado
- Clave: Código
- Índice: Código
- Antes de adicionar un nuevo estudiante debemos verificar que no exista en el archivo.
- No es necesario considerar el estado de cada registro, ya que el borrado de los registros, a diferencia de los archivos secuenciales es físico, no lógico, lo cual ofrece una ventaja puesto que se puede utilizar el espacio de almacenamiento disponible.

a. Algoritmo

```
INICIO
    ESCRIBA: "DIGITE EL CÓDIGO DEL ESTUDIANTE"
    LEA: CODIGOAD
    ENCONTRÓ = 0
    ABRIR (ESTUDIANTE)
    ENCONTRÓ = BUSCAR (ESTUDIANTE, CODIGOAD)
    SI ENCONTRÓ = 1 ENTONCES
            ESCRIBA: "EL ESTUDIANTE YA EXISTE"
      SINO
            R.CÓDIGO = CODIGOAD
            ESCRIBA: "DIGITE NOMBRE Y DIRECCIÓN"
            LEA: R.NOMBRE, R.DIRECCIÓN
            ESCRIBA (ESTUDIANTE,R)
    FIN_SI
    CERRAR (ESTUDIANTE)
FIN_INICIO
```

Anotaciones importantes:

- Asumimos que el archivo existe y tiene registros, de lo contrario deberá ser creado con la operación Crear ().
- En ningún momento es necesario hacer referencia al indicador, EOF, de fin de archivo.
- Buscar es una función que recibe como argumentos el nombre del archivo y el valor a buscar. Ésta devuelve verdadero (1) si el valor se encuentra en el archivo o falso (0) en caso contrario. Tenga presente que la función es una caja negra y su eficiencia puede variar dependiendo del método de indexamiento utilizado por el lenguaje.

b. Algoritmo que modifica la información de un estudiante en el archivo.

Algoritmo

```
INICIO
    ESCRIBA: "DIGITE EL CODIGO DEL ESTUDIANTE A MODIFICAR"
    LEA: CODMOD
    ENCONTRÓ = 0
    ABRIR (ESTUDIANTE)
    ENCONTRÓ = BUSCAR (ESTUDIANTE, CODMOD)
    SI ENCONTRÓ = 0 ENTONCES
            ESCRIBA: "EL ESTUDIANTE NO EXISTE EN EL ARCHIVO"
      SINO
            ESCRIBA: "DIGITE NOMBRE Y DIRECCIÓN DEL NUEVO
            LEA: R.NOMBRE, R.DIRECCIÓN
```

```
            R.CÓDIGO = CODMOD
            SOBREESCRIBA (ESTUDIANTE, R)
        FIN_SI
        CERRAR (ESTUDIANTE)
    FIN_INICIO
```

c. Algoritmo que retira un estudiante del archivo.

```
    INICIO
        ESCRIBA: "DIGITE EL CÓDIGO DEL ESTUDIANTE A ELIMINAR"
        LEA: CODELIM
        ENCONTRÓ = 0
        ABRIR (ESTUDIANTE)
        ENCONTRÓ = BUSCAR (ESTUDIANTE, CODELIM)
        SI ENCONTRÓ = 0 ENTONCES
            ESCRIBA: "EL ESTUDIANTE NO EXISTE"
          SINO
            ESCRIBA: "EL ESTUDIANTE SERÁ ELIMINADO"
            ELIMINE (ESTUDIANTE, CODELIM)
        FIN_SI
        CERRAR (ESTUDIANTE)
    FIN_INICIO
```

Al igual que BUSCAR, ELIMINAR es una función disponible en los lenguajes de programación que permite liberar físicamente el espacio que ocupaba el registro. Esta función, además, debe encargarse de la reorganización o reindexamiento del índice o de los índices después de la operación, lo cual a veces resulta complejo pero necesario, teniendo en cuenta los tiempos de respuesta de los métodos de búsqueda que utilizan estructuras complejas.

Aspectos a tener en cuenta:

a. Los arreglos son creados en memoria principal; por lo tanto, es bueno controlar su tamaño.
b. Los arreglos de registros permiten tener elementos con diferentes tipos de datos
c. El entendimiento de los arreglos de registros, por su parecido, son base fundamental para el manejos de archivos.
d. Cuando se crea un archivo, hay que tener en cuenta que si existe otro archivo con el mismo nombre en el medio de almacenamiento, será destruido.
e. La instrucción LEA es diferente para archivos y para entrada de datos desde teclado o cualquier otro medio de entrada.
f. Cuando se trabaja con archivos secuenciales indexados, no es necesario hacer referencia a la constante interna EOF.
g. El borrado de un registro en un archivo secuencial es lógico, es decir, aunque el registro no se tiene en cuenta en el proceso aún hace parte del archivo, mientras que en los archivos secuenciales indexados el borrado es físico.
h. El uso de las funciones BUSCAR y ELIMINAR son transparentes para el programador, ya que de una u otra forma los lenguajes de programación las tienen implementadas.

9.3. Ejercicios propuestos

235. Elaborar un algoritmo que forme un arreglo de registros que contenga en cada elemento: cédula, nombre, horas trabajadas, valor hora trabajada, porcentaje de deducción. El

algoritmo debe producir un informe con la cédula, nombre y salario devengado (ordenado ascendentemente por el campo cédula)

236. Crear un arreglo de registros con la siguiente información por elemento: cédula, apellidos y nombre y cuatro notas con sus respectivos porcentajes. Luego de creado el vector se pide mostrar listado ordenado por apellido, insertar un nuevo estudiante, la calificación definitiva de cada estudiante y visualizar el contenido del arreglo.
237. En un campeonato de fútbol se cuenta con un número máximo de veinte equipos participantes y un mínimo de diez. Para la realización del campeonato los equipos se dividen en dos grupos. De cada grupo se sacan los dos equipos que tengan los mayores puntajes y éstos serán los que van a la final. Para obtener los dos equipos de la final se hace de igual manera, mediante números aleatorios, lo mismo se hace para obtener el campeón.

    Realice un algoritmo que represente en vectores de registros los equipos participantes en cada una de las rondas, donde cada equipo debe conservar el puntaje obtenido en cada fase; el algoritmo debe mostrar en cualquier momento los participantes en cada una de las etapas del campeonato. Usted les da los nombres a los equipos.
238. Elabore un arreglo de registros con la siguiente información por elemento: apellidos nombres, edad, estatura, color de los ojos, medida del busto, medida de la cintura y medida de la cadera, de las candidatas a un reinado de belleza. El algoritmo debe proporcionar un MENU con los siguientes ítems: candidatas ordenadas por apellido, candidatas que tienen medidas perfectas, candidatas con ojos azules, y un listado ordenado descendentemente por estatura.
239. Se tiene la siguiente información de un grupo de personas: cédula, apellidos nombres, dirección, teléfono, sexo (M: masculino; F: femenino), estado laboral (1: trabaja; 2: no trabaja). Elabore un algoritmo que forme dos vectores de registros con la anterior información; uno debe almacenar las personas que trabajan y el otro las que no trabajan, ordenados ascendentemente por cédula.

    Elabore un algoritmo que forme y desarrolle el siguiente MENU: adicionar una nueva persona, retirar una persona, cambio de estado laboral, buscar una persona, listar personas que trabajan, listar personas que no trabajan, listar mujeres que no trabajan, listar hombres que trabajan.

    Cuando se adicione una nueva persona se deben pedir los datos de ella e intercalarla en el respectivo vector sin dañar el ordenamiento. Para retirar una persona, se le debe pedir la cédula y su estado laboral y borrarla físicamente sin dañar el ordenamiento en el vector donde se encuentre. Para el cambio de estado laboral se le pide la cédula y su estado laboral y se adiciona y se borra del respectivo vector. Cuando se busca una persona, se le pide la cédula y el estado laboral y se muestra su información.
240. Un grupo económico almacena información de sus empresas en un archivo secuencial con la siguiente composición:

Grupo = (Código-Empresa, Código-Departamento, Código-empleado, SexoA, Total-Ventas-Empleado, Estado)

¿Cuál es su clave primaria?

Sabiendo que la información se encuentra ordenada ascendentemente por cada código, escriba los respectivos algoritmos y programas en C++ que calculen:

1. Total de ventas del grupo económico
2. Código de la empresa con mayores ventas
3. Código del departamento con menores ventas
4. Código del empleado con mayores ventas por cada departamento de cada empresa.

241. Dado el siguiente archivo secuencial indexado:

Préstamo = {Cédula-Prestatario, Nombre-Prestatario, Tipo, Descripción, Valor, Fecha}

Escriba los algoritmos y programas que:

2.1. Adicionen préstamos.
2.2. Modifiquen préstamos.
2.3. Retiren préstamos.
2.4. Imprima todos los nombres de los clientes que deben entre 5 y 15 millones.
2.5. Permita consultar los préstamos de una persona.

Nota Elija con cuidado la clave e índice del archivo, tenga presente que una persona puede tener varios préstamos, pero sólo uno de cada tipo.

242. Las universidades A y B almacenan información de sus estudiantes en los siguientes archivos:

A = {@CódigoA, NombreA, DirecciónA, SexoA, EdadA}

B = {@CódigoB, NombreB, DirecciónB, SexoB, EdadB}

Escriba los algoritmos que permitan consultar:

1. A - B
2. A ∩ B
3. A ∪ B
4. A - (A - B)

Realice cada consulta para archivos secuenciales y archivos secuenciales indexados. Compare los tiempos de ejecución para cada una de ellas dependiendo de los tipos de archivos utilizados. ¿Qué podemos concluir?

Para cada uno de los siguientes ejercicios construya una solución usando archivo secuencial y otra con secuencial indexado.

243. El banco EL ANCLAR S A, desea conocer el estado de sus clientes partiendo de un archivo que tiene el número de la cuenta, nombre del cliente, teléfono del cliente, saldo actual y número de cheques girados en la última semana. Con los datos anteriores hallar el teléfono del cliente que menos cheques giró, el número de clientes con saldo en rojo, el número de clientes con saldo mayor a $ 500.000, el número de clientes con saldo mayor que cero y el acumulado de saldos de todos los clientes.
244. La oficina de deportes de un politécnico desea conocer personas para integrar un equipo de baloncesto, y para ello abrió inscripciones y recolectó un archivo con el nombre, teléfono, edad, sexo (‘F’ - femenino, ‘M’ - masculino), estatura (en centímetros) y peso (en kilos) de personas interesadas. Se requiere averiguar e imprimir el nombre y teléfono de las personas que tengan entre 18 y 30 años, que midan 1,80 metros o más y que pesen entre 80 y 110 kilos; además, la edad promedio de las personas que cumplieron con los requisitos exigidos, el número de hombres con un peso mayor a 110 kilos, el número de mujeres menores de 18 años y el nombre de la persona más alta.
245. Un almacén de cadena tiene un archivo de artículos con el código, nombre, cantidad existente, precio unitario de compra, precio unitario de venta y número de unidades vendidas hasta la fecha. El Administrador desea saber el código del artículo y el precio unitario de compra, de los artículos que se vendieron totalmente; también, el nombre y la cantidad restante de los que no se vendieron totalmente, y la ganancia bruta del almacén.
246. La empresa de construcción MONTELIBANO S. A , desea conocer terrenos cuya área esté entre 4800 y 5400 metros cuadrados, y que su frente y fondo no sea menor de 60 y 80 metros respectivamente. Para ello recolectó medidas de terrenos con el nombre del propietario, teléfono, dirección, longitud de frente y de fondo en metros, y desea saber la información completa de los terrenos que cumplan los requisitos, además de cuántos terrenos cumpliendo con el área, no cumplieron con las medidas de frente y fondo.
247. Se tiene un archivo de personal de una empresa con la cédula del trabajador, nombre, edad, tiempo de servicio, asignación mensual, estado civil (1: casado; 2: soltero; 3: viudo)

y número de hijos. Se requiere hallar e imprimir la siguiente información según el caso: la cédula y nombre de los trabajadores con más de 20 años de servicio y 50 ó más años de edad; la cédula, el número de hijos, y el 20 por ciento de la asignación mensual de aquellos trabajadores que tengan más de 5 hijos y menos de 10 años de servicio; el número total de trabajadores de la empresa; la cédula y el valor a pagar por subsidio familiar para los trabajadores casados, sabiendo que se pagan $20.000 de subsidio por hijo; el promedio de asignación mensual de los trabajadores de la empresa.

248. Una agencia de compra, venta y arrendamiento de propiedad raíz en la ciudad dispone de un archivo cuyo registro está compuesto de los siguientes campos: dirección de la propiedad, teléfono, número de alcobas, área total, valor de compra, valor de venta, valor arrendamiento, indicador de estado (1: arrendada: 2: desocupada). También tiene un archivo de novedades de la semana, con los siguientes campos: dirección de la propiedad, teléfono, número de alcobas, área total, valor de compra/venta/arrendamiento, tipo de transacción (1: compra; 2: venta; 3: se arrendó). Hacer un algoritmo para actualizar el archivo de propiedades, sabiendo que:

- Cuando la propiedad se compra, siempre está desocupada, su precio de venta es del 15% adicional a su precio de compra, y su precio de arrendamiento en ese momento es cero.
- Cuando la propiedad se vende, debe estar desocupada.

Además se debe imprimir el valor total de las propiedades compradas, el valor total de las propiedades vendidas y cuántas propiedades se arrendaron.

249. Una empresa de transporte aéreo tiene un archivo con el siguiente diseño:

Descripción del campo	**Tipo campo**	**Observaciones**
Nombre de la ciudad de destino	Alfanumérico	Ciudades del Ecuador
Valor del pasaje	Numérico	
Número de pasajeros que viajaron en cada mes del año	Numérico	Arreglo de doce (12) elementos

Hacer un algoritmo para hallar el total recaudado por transporte en la empresa. ¿En cuál mes se recaudó más dinero en la empresa, en diciembre o en junio?. Nombre de la ciudad de destino, número de pasajeros que viajaron en el año y valor recaudado por transporte a cada una de las N ciudades de destino, sabiendo que la ciudad de origen es Bogotá. ¿A cuál ciudad viajaron menos pasajeros en el año?.

250. Una Empresa Textil tiene un archivo de rollos de tela terminada en fondo entero y con un ancho estándar de 1,50 metros, cuyo registro está compuesto del número de rollo de tela, color (1: blanco; 2: negro;, 3: verde), tipo de tela (A: algodón; L: lino; S: seda), largo en metros exactos y valor del rollo. Hacer un algoritmo para hallar el número de rollos de tela, su valor total y el promedio de longitud de los rollos; el porcentaje (%) de tela en lino respecto de toda la producción; cantidad de vestidos para hombre en algodón de color negro, sabiendo que para confeccionar cada vestido se requiere de 4 metros de largo; cuántas banderas de la paz se pueden confeccionar si cada una requiere un metro de tela.

251. Una librería dispone de un archivo de ventas de la semana con el siguiente diseño de registro:

Descripción campo	Nom.Nemot.	Long.	Tipo	Codificación
Título del libro	TITL	50	Alfanumérico	
Editorial	EDIT	40	Alfanumérico	
Valor unitario de venta	VUNI	7	Numérico	
Área	AREA	1	Numérico	1 - Matemáticas 3 - Español 5 - Sociales
Cantidad vendida	CANT	4	Numérico	

Hacer un algoritmo para hallar la siguiente información solicitada por el administrador:

- Título del libro del área de sociales, de menor cantidad vendida.
- De que área entre matemáticas y español, se vendieron más libros.
- Título del libro, editorial y total de ventas de cada libro.
- Total de ventas en la semana y promedio de venta de cada libro.

BIBLIOGRAFIA

JOYANES, Luis: *Fundamentos de programación*, McGRAW-HILL, Madrid, 1988.

JOYANES, Luis: *Problemas de la metodología en la programación*, McGRAW-HILL, España, 1990.

JOYANES, Luis, ZAHONERO, Ignacio: *xEstructura de datos, algoritmos abstractos y objetos,* McGRAW-HILL, España, 1998.

RIOS C. Fabian: *Soluciones secuenciales*, Editorial U de A, Medellín, 1995

CORREA, Guillermo: *Desarrollo de algoritmos y sus aplicaciones en BASIC, PASCAL, COBOL Y C. McGRAW-HILL*, Bogotá, 1992.

BORES, Rosario y ROSALES, Ramon: *Computación-metodología, lógica computacional y programación*, McGRAW-HILL, México, 1993.

SANCHIS, J y MORALES A: *Programación con lenguaje Pascal*, McGRAW-HILL. España, 1980.

SHAUM: *Programación en Pascal*, McGRAW-HILL, España, 1985.

LOZANI, R. Letvin: *Diagramación y programación*, McGRAW-HILL, Bogotá, 1988.

BECERRA, Cesar: *Lenguaje C,* Kimpres Ltda, Bogotá, 1998

CAIRÓ/GUARDATI: *Estructura de datos*, McGRAW-HILL, México, 1993.

H. PAUL, Haidux: *Turbo Pascal Orientado a Objetos*, McGRAW-HILL, México, 1994.

VILLALOBOS, Jorge : *Diseño y manejo de estructuras de datos en C++*, McGRAW-HILL, Bogotá, 1996.

DEITEL & DEITEL: *Como programar C++*, Pearson, México, 1999.

JO ANN SMITH: *C++ Desarrollo de proyectos*, Ingramex, México, 2001.

INDICE ANALITICO

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V