Fundamentos de programación,
Guía de autoenseñanza

Fundamentos de programación, Guía de autoenseñanza

Carlos Andrés Zapata Ospina

Alfaomega Ra-Ma

Fundamentos de programación, guía de autoenseñanza.

ISBN 75-089-060, edición original publicada por RA-MA editorial,
Caldas, Colombia.

Primera edición: Alfaomega grupo editor, Colombia, septiembre 2006.

ISBN 75-089-060

Impreso en Colombia – Printed in Colmbia.

A mí dedicada, comprensiva y amorosa familia,
A mis maestros, Víctor Hugo Angel y Marlon Manrique.
A todos mis antiguos, presentes y futuros alumnos.

Contenido

1

Introducción
a las Computadoras,
y lenguajes de programación

2. Fundamentos de programación, guía de autoenseñanza.

Plan general.

1.1 Introducción.
1.2 Historia.
1.3 Hardware.
1.4 Software.
1.5 Programación.

OBJETIVOS

- Realizar un recorrido resumen acerca de la evolución de los computadores personales (PC).
- Presentar de manera general el hardware y del software de un computador.
- Explicar los diferentes dispositivos de un ordenador.
- Conocer que es la programación de computadores.
- Entender el concepto de lenguaje de programación.

1.1. INTRODUCCIÓN.

Una computadora u ordenador es un dispositivo electrónico desarrollado para ejecutar un conjunto de instrucciones, facilitar el manejo de la información y procesar datos agrandes velocidades.

El avance de la electrónica hizo posible desarrollar este tipo de maquina, que es utilizada por toda la sociedad en innumerables usos. Son herramientas esenciales prácticamente en todos los campos de investigación y en tecnología aplicada. Pero es para destacar que aun faltan muchos usos por dársele y sacar su máximo provecho.

El objetivo de este capitulo pretende ambientar al lector sobre conceptos concisos acerca de la computación, abordando ítems de su evolución desde dos puntos de vista, a nivel físico o también llamado Hardware (componentes de un ordenador, tipos de ordenadores, etc.) y en la capa lógica o software (sistemas operativos, herramientas aplicadas, lenguajes de programación) encargado de administrar la primera. Ya que es imposible hablar de una sin tenerse en cuenta la otra, y los avances en ambas ramas repercuten directamente en la otra. Además de proporcionar unos conceptos claros para que el lector pueda afianzar sus conocimientos y generar unas sólidas bases.

1.2. HISTORIA.

1.2.1. Historia del hardware.

600-500	El ábaco fue la primera maquina desarrollada para ayudar y realizar cálculos matemáticos, su origen data entre el año 600 y 500 A.C; la estructura consistía en una serie de bolas de madera que se deslizaban sobre una varilla o cuerda, avanzando o retrocediendo según la operación.
1617	El inventor de los logaritmos John Napier, construyo unas piezas de calculo con huesos o marfil denominadas “Huesos de Napier”, teniendo mucha influencia en la construcción de la regla de calculo.
1642	El francés Blaise Pascal, matemático y filosofo, invento la primera calculadora automática denominada “maquina de Pascal” capaz de sumar y restar, creada originalmente para apoyar la recaudación de impuestos de su padre.
1666	El maestro de la mecánica de la corte del rey Calos II de Inglaterra, sir Samuel Morland, crea la maquina de multiplicar compuesta por una serie de ruedas que representan las unidades, decenas, centenas, entre otras.
1769	El húngaro Barón Kempelen, crea el jugador de ajedrez autómata. En manos de Johann Nepomuk viaja por toda Europa y estados unidos presentándose como el jugador de ajedrez “robotizado”.

4. Fundamentos de programación, guía de autoenseñanza.

1804	Joseph Marie Jacquard inspirado en las cajas de música que emplean papel perforado para producir sonido, inventa el telar Jacquard.
1822	Charles Babbage presenta la maquina diferencial, capaz de realizar cálculos de tablas simples. Por este desarrollo se le considera el “padre de los ordenadores modernos”.
1854	El ingles George Boole desarrollo el álgebra de boole, generando una reducción de combinaciones mediante el uso de los operadores algebraicos básicos: (y) and, “or” o y “not” no. Por este desarrollo se le considera el padre de la teoría de la información.
1906	El estadounidense Lee De Forest creo el tubo de vació. Tenia tres elementos en una bombilla de cristal vacía. En los años 30 se empezó a utilizar el tuvo al vació para el desarrollo de ordenadores.
1919	Los estadounidenses W. H. Eccles y F.W. Jordán desarrollaron el primer Flip – Flop, dispositivo capaz de cambiar entre dos estados. Este permitió a los circuitos electrónicos tener dos estados fijos, forjando los modos de almacenamiento de dígitos binarios para los computadores modernos.
1946	J. Presper Eckert y John Maucly construyeron el ENIAC, el primer ordenador electrónico digital. Contenía 18000 tubos de vació.
1947	John Bardeen, Walter Brattain y William shockley inventaron el transistor, dispositivo electrónico capaz de regular el flujo de la corriente eléctrica, permitiendo la reducción en el tamaño de los ordenadores. Por este desarrollo ganaron el premio Nóbel en 1956.
1958	Jack Kilby, construyo el primer circuito integrado, conformado por una serie de componentes de silicona individuales, ensamblados en conjunto.
1964	IBM presento el System/360, reemplazando los transistores por tecnología de circuitos integrados.

1969	Data General Corporation distribuyo el primer ordenador a 16 BIT. Este ordenador denominado NOVA aumento la velocidad y potencia.
1970	Corning Glass Works, inc, comercializó el primer cable de fibra óptica. A diferencia que de otros cables de naturaleza de metal, este trasmite as de luz a grandes velocidades, permitiendo la transmisión de grandes volúmenes de datos.
1971	Intel Corporation desarrollo el primer chip microprocesador, el 4004.
1975	Micro Instrumentation Telemetry System (MITS) introdujeron al mercado el Altair. Este ordenador no tenia teclado, monitor o un dispositivo básico de memoria. Tenia un procesador Intel 8080 de 8 BIT.
1980	Laboratorio Bell introdujeron el primer microprocesador a 32 BIT, llamado Bellmac-32.
1981	IBM introdujo su primer ordenador basado de un procesador Intel 8088, un disco flexible y alrededor de 128k o 256k en memoria Ram. Utilizaba el sistema operativo D.O.S.
1984	Apple computer, inc. Comercializo el primer ordenador personal Macintosh. Este ordenador tenia un monitor integrado, ratón y una capacidad de memoria de 128k. Además era el primer sistema de computo que poseía un entorno grafico para el trabajo y administración.
1984	IBM distribuyo el PC-AT con procesador Intel 80286. sobre este ordenador se introdujo un sistema de gráficos EGA, permitiendo gráficos de 16 colores y un bus de datos de 16 BIT.
1992	Intel desarrollo el procesador 80486 igualando la calidad de su homólogo 68040 de motorota.

6. Fundamentos de programación, guía de autoenseñanza.

1.2.2. Historia del software.

1957	IBM desarrollo el primer lenguaje de programación de alto nivel llamado FORTRAN (traductor de formulas). Es muy usado aun actualmente por muchos científicos, ingenieros y matemáticos.
1960	Un equipo de la universidad de Pensilvania, apoyado por el departamento de defensa de los estados unidos, desarrollo un primer lenguaje de programación llamado COBOL (lenguaje común orientado a los negocios)
1962	Ivan Sutherland de M.I.T implemento el primer software de gráficos, permitiendo dibujar interactivamente en la pantalla.
1964	Thomas Kurt y John Kemendy desarrollaron el lenguaje de programación BASIC, muy popular y usado hasta los días de hoy.
1967	Richard Greenblatt implemento el primer programa de ajedrez con éxito llamado MacHack.
1970	E.F Codd publico el primer diseño de bases de datos relacional.
1978	Brian Kernighan publico una descripción del lenguaje C, confinando desde principios de los años 60 en Bell Telephone. C fue y continua siendo el lenguaje de programación más popular, en los años 80 dio origen al lenguaje C++, basado sobre el lenguaje pero con el soporte al paradigma de programación orientado a objetos.
1991	Sun Microsystem bajo el proyecto de James Gosling desarrollaron el lenguaje Java. Java es el lenguaje mas usado actualmente y empleado en diferentes ambientes como: paginas web, ordenadores de escritorio, servidores de alto rendimiento, equipos móviles y entre otros.
1997	IBM desarrollo el sistema de juego de ajedrez llamado Deep Blue. En la imagen se muestra al campeón mundial de ajedrez Gari Kaspárov durante una partida con Deep Blue en Nueva York.

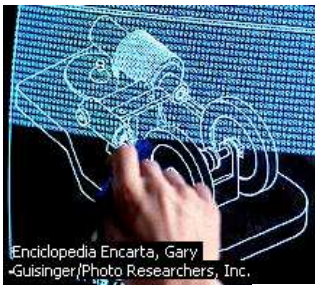
1.3. HARDWARE

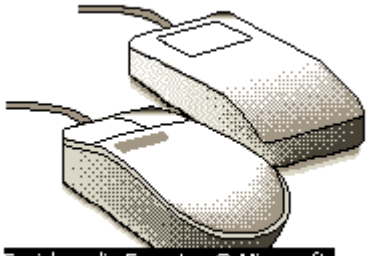
Corresponde a la parte física o tangible del ordenador y esta compuesta por los siguientes elementos: dispositivos de entrada, salida y almacenamiento.

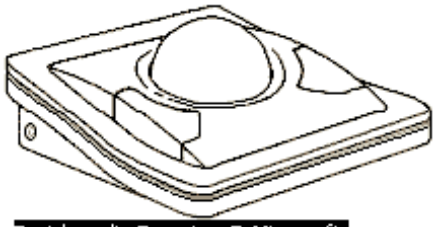
1.3.1. Elementos de un computador según utilización.

1.3.1.1. Dispositivos de entrada.

Generalmente estos componentes se encuentran ubicados fuera de la torre del computador y permiten ingresar datos para ser procesados por la CPU. Entres estos se encuentran: el mouse, teclado, lápiz óptico, tablas digitalizadoras, control de juegos o joystick, pantallas sensibles al tacto que permiten el usuario señalar acciones como emplean algunos cajeros electrónicos, cámaras, scanner, micrófonos, entre otros.







8. Fundamentos de programación, guía de autoenseñanza.

1.3.1.2. Dispositivos de salida.

Estos permiten la visualización de la información en una amplia gama de formatos o modos. Entre estos se encuentran: las impresoras, las pantallas o monitores, parlantes, entre otros. Con ellos se busca la representación de la información de una manera más amena y entendible.

1.3.1.3. Dispositivos de almacenamiento.

A este grupo pertenecen todas las unidades de lectura / escritura (leer / guardar) de datos. Existen unidades internas y externas a la torre del computador, creadas para dar solución a las diferentes necesidades del usuario, aplicando diferentes tecnologías de almacenamiento.

Las unidades CD-ROM, CD-RW, DVD son tanto internas como externas, utilizan discos compactos para almacenar datos utilizando tecnologías ópticas para el almacenamiento.

1.3.1.3.1. Internas.

La principal unidad de almacenamiento interna es el disco duro, donde se almacenan los programas y datos que van a interactuar con el hardware como: sistemas operativos, aplicaciones de escritorio y de servicios. La unidad de disquete ha existido por mucho tiempo tendiendo a desaparecer, dadas sus limitaciones en cantidad de almacenamiento y las innumerables variables como temperatura, radiación, entre otras. Esta unidad es conocida como la unidad de tres un medio (3 1/2) o la unidad A. Estas unidades emplean tecnologías magnéticas para almacenar información.

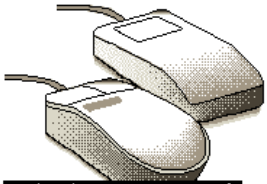
1.3.1.3.2. Externas.

Son las unidades que se conectan con el exterior de la torre y generalmente son transportables. Las unidades USB son muy usadas actualmente permitiendo no solo almacenar datos, sino también música en el formato mp3.

1.3.2. Partes del computador.

1.3.2.1. Mouse o Ratón.

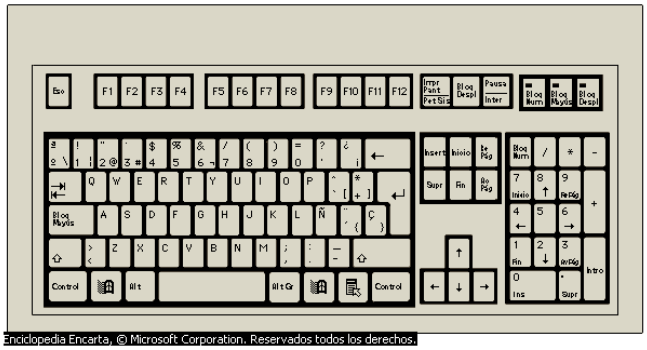
Es un dispositivo apuntador o señalador que le permite a un usuario de un equipo de computo navegar sobre la interfaz grafica, facilitando el acceso a acciones u operaciones dentro del ordenador. Algunos mouse actuales emplean aun la tecnología mecánica de rodillos y bolilla para ubicar el cursor del ratón dentro de una interfaz grafica. La tecnología nueva emplea ratones infrarrojos que al no depender de un sistema mecánico permiten tener una gran exactitud requerida para el trabajo de diseño y creación visual de imágenes computarizadas.



1.3.2.2. El teclado.

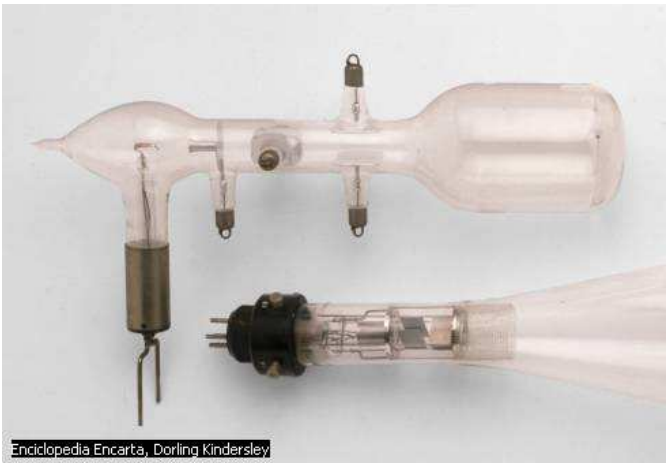
Es el dispositivo básico de entrada de acciones hacia un ordenador presionando una serie de teclas o mediante una combinación de las mismas. Cuenta con teclas de función numeradas como F1 al F12, un teclado numérico, un teclado alfanumérico, teclas de desplazamiento del cursor, indicadores de paneles activos.

10. Fundamentos de programación, guía de autoenseñanza.



1.3.2.3. Monitor.

Es el dispositivo primario de salida donde se visualizan las imágenes generadas por el ordenador, conectado mediante un adaptador de video, el cual responde por una gran parte en la calidad de la imagen generada y otra proporcionada por el monitor. Una de las tecnologías de monitores que se cuentan actualmente es la del tubo de rayos catódicos con el también cuentan televisores, osciloscopios, radares. En este las imágenes se representan con un haz de electrones que barren una superficie fosforescente y proyectan imágenes.



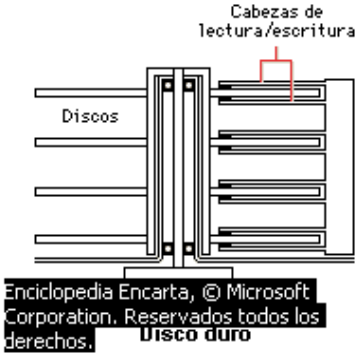
Otra nueva tecnología presenta monitores de pantalla liquida LCD, usada por los ordenadores portátiles.

1.3.2.4. Torre del computador.

La gran mayoría de los usuarios de ordenadores la llaman “CPU”. El termino no esta bien empleado ya que CPU corresponde a la Unidad Central de Proceso de la computadora, y la torre es simplemente el soporte para una serie de componentes albergados dentro. Explorando con detenimiento los elementos contenidos en la torre se encuentran:

1.3.2.5. Disco duro.

Es la principal unidad de almacenamiento de los datos dentro de un computador. Su forma interna corresponde a un apilamiento de discos o laminas rígidas de forma circular, albergando los datos mediante procesos magnéticos. Estas laminas giran alrededor de 3600 revoluciones por minuto. Cuentan con cabezas de lectura escritura muy similares a las agujas de los equipos de sonido para el manejo de acetatos, donde este cada uno se posiciona sobre un área del disco para guardar o leer datos.



Disco duro

1.3.2.6. CPU.

Unidad central de proceso (UCP de sus siglas en ingles), es el dispositivo electrónico encargado controlar y manejar los datos en los ordenadores. Esta compuesta por la ALU (unidad aritmética lógica) que realiza todo tipo de operaciones lógicas mediante el álgebra de bolee, y una serie de registro de almacenamiento temporal para efectuar operaciones. Una CPU esta contenida en un microprocesador que es un chip.

12. Fundamentos de programación, guía de autoenseñanza.

1.3.2.7. La memoria Ram.

RAM (memoria de acceso aleatorio) es la memoria de trabajo del ordenador, es decir, el disco duro almacena datos de forma permanente, mientras la Ram se emplea para labores de procesos que se efectúan con el procesador y otros periféricos, guardado datos mientras se estén usando, cuando se finaliza cualquier tarea los espacios de memoria son usados por otra tarea de procesamiento. Además es la memoria que mantiene los datos mientras usted desarrolla tareas sobre aplicaciones como: procesadores de texto, juegos, hojas de calculo.

Es una memoria que guarda una replica mínima de los datos que contiene el disco duro y que se están usando, ya que el ordenador se toma un tiempo mientras accede al disco duro (recuerde la analogía con el toca discos de acetato y la "aguja" para este caso la cabeza de lectura / escritura se posiciona en el lugar indicado), razón por la cual emplea una memoria de acceso rápido.

Cuando en capítulos posteriores se este estudiando la programación básica, denominada algoritmia, va encontrase una interacción constante con la memoria Ram, es por eso que importante conocerla y conceptualizar su uso dentro de un sistema de computo.

1.3.2.8. Tarjeta Madre, Main board o Placa Base.

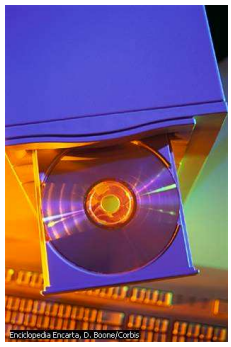
Este componente electrónico contiene el procesador, la memoria Ram, los buses de datos y otros elementos, es por eso que se le llama comúnmente tarjeta madre, es una gran tarjeta que interconecta todos los elementos contenidos dentro de una torre del computador.

1.3.2.9. Buses de datos.

Son los encargados de conectar cada dispositivo dentro de la torre del computador con la tarjeta madre. Las labores de envió y recibo de datos a través de los buses o correas de datos, son administradas por el microprocesador.

1.3.2.10. Unidades de CD-ROM.

Es un estándar de solo lectura en un disco compacto empleando tecnología de láser ópticos.



1.3.2.11. Unidades de CD-rw.

Es un estándar de lectura / escritura en un disco compacto con una capacidad de almacenamiento de alrededor de 650 megas bytes.

1.3.2.12. Unidades de disquete.

También conocidas unidades flexibles de almacenamiento de baja capacidad, pueden contener 1.44Megbytes. Son totalmente potables y todos los ordenadores contienen esta unidad básica de almacenamiento.

1.4. SOFTWARE.

El software como se menciono antes es la parte lógica de un sistema de computo, que no es tangible pero se hace visible cuando se saca algún provecho a un ordenador. El software abarca todo tipo aplicaciones (aplicaciones o también denominados programas) que sirven para controlar, manipular, capturar datos, que luego son mostrados como información entendible para el usuario.

Para conocer de una manera diferente todos los tipos de programas que tiene en su ordenador, se desarrollará etapa por etapa el simple proceso desde encender su ordenador y dejarlo listo para ejecutar cualquier aplicación.

Cuando se presiona el botón de encendido de su sistema de computo suceden las siguientes operaciones: existe un programa residente en un chip que se encarga de verificar y chequear la configuración “hardware” de su maquina, por esto, recién

enciende su maquina se muestra en el monitor ciertas textos que muestran información de su computadora. Este chip se conoce como la **Bios.**

Finalizada la tarea de chequeo se inicia la ejecución del sistema operativo. Este programa es el encargado de administrar todos los recursos[1] de su maquina. El sistema operativo más popular mundialmente es Windows, desarrollado por la empresa Microsoft Corporation. Pero existen otras alternativas como esta el sistema operativo Linux apoyado por la organización de software libre Gnu[2], en cada una de sus diferentes distribuciones y para los usuarios de equipos mac esta MacOs2. La labor de cualquier sistema operativo en este proceso de arranque es preparar y verificar el estado del hardware ya combinado con algunas aplicaciones para el usuario pueda empezar a interactuar y sacar el máximo provecho.

Finalmente, cuando se ingresa el sistema se puede empezar a utilizar software para escribir documentos usando Microsoft Office o su alternativa en software libre Open Office. Para navegar por Internet emplea un brouser o navegar web como Internet Explorer para sistemas Windows o Mozilla, FireWorks, opera, entre otros.

En resumen, para utilizar y disfrutar de todo el hardware de un ordenador se hace necesario el uso de programas que permitan interactuar directamente de una manera discreta.

1.5. PROGRAMACIÓN.

Un programa para computadora es un conjunto de instrucciones que le indican al hardware que tipo de acciones debe efectuar con los datos. La programación es la acción de codificar o escribir esas instrucciones sobre un entorno llamado lenguaje de programación, para generar aplicaciones software o simplemente programas. Existen lenguajes de programación para casi todas las necesidades, desde aplicaciones para manejar directamente un circuito electrónico, sistemas de nomina, hasta software para registrarse y realizar compras por Internet. Incluso los equipos móviles que son actualmente tan cotidianos emplean programas para buscar un nombre en la lista telefónica, para jugar, y muchas mas opciones según el tipo de equipo con el que se cuente.

Con la programación se puede hacer lo que se ocurra y se necesite, es un amplio mundo con muchísimos campos de aplicación.

---

[1] Cuando se menciona recursos se refiere a todo el hardware del ordenador.

[2] Gnu / Open Source es una organización que lidera a nivel mundial para el desarrollo y aprovechamiento de software libre.

1.5.1. Lenguajes.

Como se menciono anteriormente, un lenguaje de programación es el entorno donde se escribe una serie de instrucciones para que el hardware del ordenador efectué cualquier operación. Esas instrucciones deben estar entonces dentro de una serie de reglas que define el lenguaje. Los lenguajes informaticos no están lejos de ser incluso tan estrictos como las normas cuando se habla otro idioma, cada uno tiene sus parámetros de uso y de escritura.

1.5.1.1. Lenguaje máquina

Este es lenguaje nativo del computador, como se menciono su naturaleza eléctrica puede oscilar entre dos estados: encendido y apagado. Los cuales se representan como 1 (encendido) y 0 (apagado). Por consiguiente la maquina genera cadenas extensas de unos y ceros, que representan un documento de texto, una imagen, un correo electrónico, entre otros.

Para interactuar con la maquina era necesario tener un dominio completo de las cadenas de unos y ceros, y así poderse generar unas nuevas o realizar modificaciones. Una falla en el arreglo de unos y ceros podía representarse con la aparición de fallas en el sistema de computo, incluso con la reprogramación de los componentes y en el peor de los casos la adquisición de nuevo hardware.

1.5.1.2. Lenguaje ensamblador.

Este es lenguaje que se invento para manejar el hardware de una manera un poco más amena. Su estructura corresponde a la identificación total de cada uno de los elementos del ordenador y representarlos sobre una serie de registros o campos en la memoria. Para este tipo de lenguaje se necesita un conocimiento extenso de cada uno de los registros para evitar borrar o modificar alguno sin verse afectado la integridad del sistema.

1.5.1.3. Lenguajes de alto nivel

Este tipo de lenguajes emplean términos mas cercanos a las estructuras gramaticales de los lenguajes del hombre. Es decir, existen palabras y gramática que son usadas comúnmente por las diferentes lenguas de habla, por ejemplo: si usted desea mostrar un mensaje por pantalla que diga **Hola**, usted diría

simplemente muestre o escriba **Hola**, ahora vera similitud que tienen algunos lenguajes.

Lenguaje	**Expresión usada**
Lenguaje algorítmico	Escribir “Hola”
C	print(“Hola”);
C++	cout>>”Hola”;
Java	System.Out.Print(“Hola”);
Pascal	Write(“Hola”);
Basic	Print(“Hola”)

Un lenguaje de alto nivel se hace más fácil y cercano para el uso de los usuarios de sistemas de computo.

RESUMEN.

- Una computadora u ordenador es un dispositivo electrónico desarrollado para ejecutar un conjunto de instrucciones, facilitar el manejo de la información y procesar datos agrandes velocidades.

- El avance de la electrónica hizo posible desarrollar del computador.

- El ábaco fue la primera maquina desarrollada para ayudar y realizar cálculos matemáticos.

- El inventor de los logaritmos John Napier, construyo unas piezas de calculo con huesos o marfil denominadas “Huesos de Napier”.

- El francés Blaise Pascal, matemático y filosofo, invento la primera calculadora automática denominada “maquina de Pascal”.

- El maestro de la mecánica de la corte del rey Calos II de Inglaterra, sir Samuel Morland, crea la maquina de multiplicar.

- El húngaro Barón Kempelen, crea el jugador de ajedrez autómata.

- Joseph Marie Jacquard inspirado en las cajas de música que emplean papel perforado para producir sonido, inventa el telar Jacquard.

- Charles Babbage presenta la maquina diferencial, capaz de realizar cálculos de tablas simples.

- El ingles George Boole desarrollo el álgebra de boole, generando una reducción de combinaciones mediante el uso de los operadores algebraicos básicos: (y) and, “or” o y “not” no.

- El estadounidense Lee De Forest creó el tubo de vació.

- Los estadounidenses W. H. Eccles y F.W. Jordán desarrollaron el primer Flip – Flop, dispositivo capaz de cambiar entre dos estados.

- J. Presper Eckert y John Maucly construyeron el ENIAC, el primer ordenador electrónico digital. Contenía 18000 tubos de vació.

- John Bardeen, Walter Brattain y William shockley inventaron el transistor, dispositivo electrónico capaz de regular el flujo de la corriente eléctrica.

18. Fundamentos de programación, guía de autoenseñanza.

- Jack Kilby, construyo el primer circuito integrado.
- IBM presento el System/360, reemplazando los transistores por tecnología de circuitos integrados.
- Data General Corporation distribuyó el primer ordenador a 16 BIT.
- Corning Glass Works, inc, comercializó el primer cable de fibra óptica.
- Intel Corporation desarrollo el primer chip microprocesador, el 4004.
- Micro Instrumentation Telemetry System (MITS) introdujeron al mercado el Altair.
- Laboratorio Bell introdujeron el primer microprocesador a 32 BIT, llamado Bellmac-32.
- IBM introdujo su primer ordenador basado de un procesador Intel 8088.
- Intel desarrollo el procesador 80486 igualando la calidad de su homólogo 68040 de motorota.
- IBM desarrollo el primer lenguaje de programación de alto nivel llamado FORTRAN (traductor de formulas).
- Ivan Sutherland de M.I.T implemento el primer software de gráficos, permitiendo dibujar interactivamente en la pantalla.
- Thomas Kurt y John Kemendy desarrollaron el lenguaje de programación BASIC.
- Richard Greenblatt implementó el primer programa de ajedrez con éxito llamado MacHack.
- E.F Codd publico el primer diseño de bases de datos relacional.
- Brian Kernighan publico una descripción del lenguaje C, confinando desde principios de los años 60 en Bell Telephone.
- Sun Microsystem bajo el proyecto de James Gosling desarrollaron el lenguaje Java.

- IBM desarrollo el sistema de juego de ajedrez llamado Deep Blue.

- Hardware Corresponde a la parte física o tangible del ordenador y esta compuesta por los siguientes elementos: dispositivos de entrada, salida y almacenamiento.

- Los dispositivos de entrada se encuentran ubicados fuera de la torre del computador y permiten ingresar datos para ser procesados por la CPU.

- Los dispositivos de salida permiten la visualización de la información en una amplia gama de formatos o modos.

- Los dispositivos de salida son todas las unidades de lectura / escritura (leer / guardar) de datos.

- El Mouse un dispositivo apuntador o señalador que le permite a un usuario de un equipo de computo navegar sobre la interfaz grafica, facilitando el acceso a acciones u operaciones dentro del ordenador.

- El teclado es el dispositivo básico de entrada de acciones hacia un ordenador presionando una serie de teclas o mediante una combinación de las mismas.

- El monitor es el dispositivo primario de salida donde se visualizan las imágenes generadas por el ordenador, conectado mediante un adaptador de video, el cual responde por una gran parte en la calidad de la imagen generada y otra proporcionada por el monitor.

- El disco duro es la principal unidad de almacenamiento de los datos dentro de un computador. Su forma interna corresponde a un apilamiento de discos o laminas rígidas de forma circular, albergando los datos mediante procesos magnéticos.

- CPU: Unidad central de proceso (UCP de sus siglas en ingles), es el dispositivo electrónico encargado controlar y manejar los datos en los ordenadores.

- Tarjeta madre componente electrónico que contiene el procesador, la memoria Ram, los buses de datos y otros elementos.

- Los Buses de datos son los encargados de conectar cada dispositivo dentro de la torre del computador con la tarjeta madre.

20. Fundamentos de programación, guía de autoenseñanza.

- El software como se menciono antes es la parte lógica de un sistema de computo, que no es tangible pero se hace visible cuando se saca algún provecho a un ordenador.

- Un programa para computadora es un conjunto de instrucciones que le indican al hardware que tipo de acciones debe efectuar con los datos.

- Un lenguaje de programación es el entorno donde se escribe una serie de instrucciones para que el hardware del ordenador efectué cualquier operación.

EJERCICIOS DE AUTO EVALUACIÓN.

1. ¿Cuál fue el componente que hizo posible la evolución de la computación?
2. ¿Cuál fue la primera maquina para realizar cálculos?
3. ¿Quién es el constructor de la maquina llamada "huesos de Napier"?
4. ¿Cuál el nombre del constructor de primera calculadora automática?
5. ¿El álgebra de boole es un invento de?
6. John Bardeen, Walter Brattain y William shockley inventaron el?
7. Fortran significa:
8. El lenguaje Java es el producto de?
9. Que significa CPU

RESPUESTAS A LOS EJERCICIOS DE AUTO EVALUACIÓN.

1. La electrónica
2. Ábaco
3. John Napier
4. Blaise Pascal
5. George Boole
6. Transistor
7. Traductor de formulas
8. Sun microsystem
9. Unidad central de proceso Unidad central de proceso

2

Ejercicios de Lógica

Plan general.

2.1 Introducción.
2.2. Técnicas para solucionar problemas.
2.3. Ejercicios de lógica.

OBJETIVOS

- Presentar una serie de técnicas simples para resolver problemas.
- Enfatizar sobre la importancia de la lógica.
- Proponer una serie de ejercicios de lógica.
- Realizar un acercamiento a la algoritmia.

2.1. INTRODUCCIÓN.

El interés de este capitulo es proporcionar un conjunto de elementos que permitan afrontar cualquier tipo de problema, de una manera más eficaz facilitando el ahorro de energía y tiempo. También se enfoca de una forma didáctica el uso de la lógica, en la manera de pasar un tiempo resolviendo unos ejercicios y comprender la importancia de la constante ejercitación mental.

El primer numeral presenta una serie de trucos que deben tenerse en cuenta en el momento de enfrentar y dar solución a diferentes puntos problemáticos.

2.2. TÉCNICAS PARA LA SOLUCIÓN DE PROBLEMAS.

Cuando se piensa dar solución a un problema, lo más común es seguir una serie de pasos que permitan encontrar respuestas lógicas, que expliquen el echo de la forma más natural para quien lo estudia y de tener la certeza que los procedimientos seguidos son los mas adecuados. Las reglas que se plasman a continuación pretenden proporcionar al lector, una serie de herramientas con las cuáles aborde cualquier tipo de problema, adaptando cada una de ellas según las necesidades.

2.2.1. Leer y entender.

Aunque parezca una regla simple y conocida por todos, el 50% de la solución de un problema se satisface, si se lee a conciencia entendiendo cada palabra, acción; identificando cada unos de los elementos que pertenecen a la problemática, con esto, lograr un manejo global del problema plasmado en unas líneas.

2.2.2. Repasar varias veces el problema y hacer énfasis en los elementos del mismo.

En ocasiones los elementos de un problema se encuentran ocultos, el abordar varias veces el problema conlleva a descubrir dichos elementos, y obtener una visión total de todo el interrogante.

2.2.3. Un problema tiene varias soluciones para resolverlo.

Él aferrarse a lo que primero que se interpreta, limitará la capacidad de razonar otras soluciones partiendo de los mismos elementos. Si la interpretación actual cumple con los requerimientos solicitados y encuentra una solución satisfactoria, fluya con todas esas ideas y plásmelas.

2.2.4. La tinta más pálida es mejor que la memoria más retentiva.

En ocasiones ese mar de ideas genera una oleada de confusión. El escribir en papel todas las ideas, minimiza el número de pensamientos redundantes, resaltando los conceptos precisos y adecuados para encontrar la solución más acertada.

2.2.5. Representar gráficamente un problema facilita la solución del mismo.

La realización de bosquejos (o en algunos casos la utilización de diagramas conceptúales) cuando se pretende dar solución a un problema, proporciona el elemento visual con el cuál se incluye en el proceso de razonamiento un sentido más; que permite pasar de la sola abstracción mental a un concepto gráfico más entendible y ameno.

2.2.6. El orden de los factores no altera el resultado.

El resultado de aplicar esta regla permite observar los elementos relacionados entre sí, desde diferentes puntos de vista, dando una nueva visualización de los elementos del problema, desarrollando una visión más concreta y real del mismo.

2.2.7. El cansancio acaba con la idea más simple.

Resulta demasiado difícil pensar en resolver un problema, si el cerebro requiere un descanso; la acción de razonar genera un poco de cansancio. Lo más ideal es crear secciones o bloques de trabajo donde se descanse por hora de trabajo de cinco (ideal) a diez (máximo) minutos.

2.2.8. La carga compartida no es tan pesada.

Si razonando solos no se encuentra la solución, lo más práctico es discutir con otros el problema, obteniendo diferentes puntos de vista y considerar otros elementos y formas de resolver el mismo.

Nota.

En la realidad los problemas de computación no se encuentran ordenados y desglosados como se plasman aquí, el ideal es que el alumno a medida que practique adquiera la capacidad de crear sus propias técnicas para resolver problemas.

2.3. EJERCICIOS DE LÓGICA.

Lo más importante antes de conocer muchos lenguajes de programación, es tener lógica de programación. Está sin incluirse en la computación (la lógica simplemente), corresponde a la forma de cómo se debe pensar para encontrar soluciones a través de diferentes métodos. La lógica es el pilar más importante en las ciencias de la computación y de vital interés la lógica de programación en el desarrollo de software. El interés con los siguientes ejercicios es incentivar la exploración de la lógica de una forma divertida con una serie de juegos de ejemplo.

La idea de en este numeral es generar un momento de diversión, de disfrute, de reto intelectual frente a los siguientes problemas, buscando las soluciones a los ejercicios a desarrollar en diferentes ambientes, dentro de todo este acompañamiento en los fundamentos de programación de computadores. Goce este momento y desarrolle todos los retos propuestos.

2.3.1. Ejercicios con soluciones.

El objetivo de estos ejercicios es enfrentar al lector a unos cuántos problemas típicos de lógica. La mejor opción es que cada uno de ellos se realice y se busque la solución más apropiada, además se discuta sobre cada uno de ellos.

1. El Almacén. En un almacén puedes conseguir un descuento del 20%, pero, al mismo tiempo, tienes que pagar unos impuesto el 15%, ¿Qué preferirías que calculasen primero, el descuento o el impuesto?.

2. La tira de papel. Imagínate una tira de papel larga y estrecha, extendida ante ti sobre la mesa, de izquierda a derecha. Coge el extremo derecho y colócalo encima del izquierdo. Ahora dobla la tira sobre la mesa aplanándola, de manera que quede plegada por la mitad y presente un doblez. Repite toda la operación dos veces más sobre la nueva tira doblada. ¿Cuántos dobleces se producirán? ¿Cuántos dobleces habrá después de repetir la operación diez veces en total?.

3. Números capicúas. A los números como el 12321, que se leen lo mismo de derecha a izquierda que de izquierda a derecha, se les llama capicúas. Tengo un amigo que asegura que todos los números capicúas de cuatro cifras son divisibles por 11. ¿Es cierto?.

4. Los retazos de colores. Coge un cuadro y traza en él una recta que lo divida en dos partes. Traza a continuación varias otras rectas arbitrariamente distribuidas, de manera que dividan al cuadro en varias regiones o <<retazos>>. La tarea a realizar consiste en colorear todas estas regiones adyacentes (o con frontera común) resulten de distinto color. (Regiones que sólo tengan un punto de frontera común no se considerarán adyacentes.) ¿Cuántos colores distintos serán necesarios, como mínimo, para colorear Cualquier distribución de regiones de este tipo?.

5. Cuadros de Ajedrez. Alguien dijo una vez que el tablero de ajedrez corriente tenía 204 cuadrados. ¿Puedes explicar esta afirmación?.

6. Madre. Una madre es 21 años mayor que su hijo, y en seis años la edad de la madre será cinco veces la del niño. ¿Dónde está el padre?

7. Herencia Caballos. Tres hermanos se reparten la herencia de su padre, esta está formada por 35 caballos y en el testamento el padre dejó escrito que el mayor se quedara con la mitad de la herencia, el mediano con la tercera parte y el más pequeño con la novena parte, como las divisiones no eran exactas estos no se ponían de acuerdo, por lo que decidieron consultar con un viejo matemático que les propuso lo siguiente:

Puesto que 35 caballos no se pueden dividir exactamente por la mitad, ni por la tercera parte ni por la novena, yo os regalo el mío, ahora tenéis 36 caballos por lo que los tres saldréis ganando. Tú por ser el mayor te llevarás la mitad de 36, es decir 18 caballos. Tú por ser el mediano la tercera parte, 12 caballos. Y tú por ser el pequeño según los deseos de tú padre, la novena parte, 4 caballos. Ahora ya tenéis los tres vuestra herencia, y como 18+12+4=34 ahora sobran dos caballos, por lo que yo recupero el mío y me quedo también con el otro por resolver vuestro problema.

¿Cómo es esto posible?

8. Excursionista. Un excursionista es capturado por caníbales y le dicen:
Si dices una mentira te matamos lentamente y si dices una verdad te matamos rápidamente. ¿Que dice para que no lo maten?

9. Entre vacas, ovejas y gallinas. Compra de ganado. El amo le dio al criado 500 pesetas para que fuese al mercado a comprarle 100 cabezas de ganado, teniendo éste que comprar: vacas, ovejas y gallinas y emplear justo las 500 pesetas. Cuando llegó al mercado comprobó que las vacas costaban 25 pesetas, las ovejas 5 pesetas y las gallinas un real. ¿Cuántas cabezas de ganado compró de cada una?

Una peseta equivale a 4 reales.

10. Matadero. En un matadero el jefe le dice al empleado: Hay que matar estas 30 ovejas en 15 días, matando al menos una por día y siempre número impar. ¿Puede el empleado cumplir la orden de su jefe?

11. Valores. ¿Cómo hacemos para que a veinte, agregándole uno nos dé diecinueve?

12. Ovejas. Dos pastores hablaban: - ¿Por qué no me das una de tus ovejas, así tendremos igual cantidad? A lo que su amigo le responde:
- Mejor dame una de las tuyas así yo tendré el doble de ovejas que tú.
¿Cuántas ovejas tenía cada uno?

13. El diablo y el campesino. Iba un campesino quejándose de lo pobre que era, dijo: daría Cualquier cosa si alguien me ayudara. De pronto se le aparece el diablo y le propuso lo siguiente:

Ves aquel puente, si lo pasas en Cualquier dirección tendrás exactamente el doble del dinero que tenías antes de pasarlo. Pero hay una condición debes tirar al rió 24 pesos por cada vez que pases el puente.

Pasó el campesino el puente una vez y contó su dinero, en efecto tenía dos veces más, tiro 24 pesos al río, y pasó el puente otra vez y tenía el doble que antes y tiro los 24 pesos, paso el puente por tercera vez y el dinero se duplicó, pero resulto que tenía 24 pesos exactos y tuvo que tirarlos al río. Y se quedo sin un peso. ¿ Cuánto tenía el campesino al principio?

¿Cuánto tenía el campesino antes de pasar por ultima vez?
Enviado por: Kerbe (Xavi Fernàndez Tejero)

14. La viejecita en el mercado: Una viejecita llevaba huevos al mercado cuando se le cayó la cesta. - ¿Cuántos huevos llevabas? - le preguntaron,
- No lo sé, recuerdo que al contarlos en grupos de 2, 3, 4 y 5, sobraban 1, 2, 3 y 4 respectivamente. ¿Cuántos huevos tenía la viejecita?

15. La botella de vino. Si nos dicen que una botella de vino vale 10 euros y que el vino que contiene cuesta 9 euros más que el envase, ¿cuánto cuestan el vino y el envase por separado?.

16. Llenar la piscina: Para llenar de agua una piscina hay tres surtidores. El primer surtidor tarda 30 horas en llenarla, el segundo tarda 40 horas y el tercero tarda cinco días. Si los tres surtidores se conectan juntos, ¿cuánto tiempo tardará la piscina en llenarse?.

17. En el bar: .Tres amigos van a tomar café. Piden la cuenta y el camarero les dice que son 25 pesetas por los tres cafés. Cada uno pone 10 pesetas, en total 30. Con las 5 que sobran, se queda cada uno 1 peseta, y las otras 2 para el bote del bar. Es decir, cada uno paga 9 pesetas, que por los tres serían 27, más las 2 de la propina, 29. ¿Dónde está la peseta que falta?

18. María y Juan. María tiene un hermano llamado Juan. Juan tiene tantos hermanos como hermanas. María tiene el doble de hermanos que de hermanas. ¿Cuántos chicos y chicas hay en la familia?

19. El vagabundo. Un vagabundo se hace un pitillo con cada siete colillas que encuentra en el suelo. ¿Cuántos pitillos podrá fumarse si encuentra 49 colillas?

20. Juan y Pedro. Juan le dice a dice a Pedro: "si me das una oveja tengo yo el doble que tú" Pedro le contesta: " no seas tan listo, dámela tú a mí, y a si tenemos los dos igual" ¿Cuántas ovejas tiene cada uno?.

21. El tío y el sobrino. Un tío le dice a su sobrino: " Yo tengo el triple de la edad que tú tenías cuando yo tenía la edad que tú tienes. Cuando tú tengas la edad que yo tengo ahora, la suma de las dos edades será de 70 años". ¿Qué edad tienen ahora ambos?

22. Las tres hijas. Dos amigos se encuentran por la calle: el primero le pregunta al otro - qué tal están sus hijas y Cuántos años tienen, el segundo le contesta: - El producto de las tres edades es 36 y la suma el número del portal en el que vives, el primero le dice: - entonces, me falta un dato, y el amigo le contesta - es cierto, la mayor toca el piano. ¿Cuál es la edad de cada hija?

23. La colección de monedas. Un comerciante decide vender una colección de monedas de oro a tres coleccionistas. El primero compra la mitad de la colección y media moneda; el segundo, la mitad de lo que queda y media moneda y el tercero la mitad de lo que queda y media moneda. ¿Cuántas monedas tenía el comerciante?

24. Pies por pulgadas. Cierto individuo ordenó telefónicamente un tramo de cordel de X pies e Y pulgadas, y descubrió que el dependiente se había equivocado con la orden y había intercambiado pies y pulgadas. Como resultado, la cuerda media sólo 30% del tramo que el cliente deseaba. ¿De qué longitúd era la cuerda ordenada?

25. Un problema de balanza sin pesas. Una bolsa contiene 27 bolas de billar que parecen idénticas. Sin embargo, nos han asegurado que hay una defectúosa que pesa más que las otras. Disponemos de una balanza, pero no de un juego de pesas, de manera que lo único que podemos hacer es comparar pesos. Demuestra que se puede localizar la bola defectúosa con solo tres pesadas.

26. La tela de araña. Una araña teje su tela en el marco de una ventana. Cada día duplica la superficie hecha hasta entonces. De esta forma tarda 30 días en cubrir el hueco de la ventana. Si en vez de una araña, fueran dos, ¿Cuánto tardarían en cubrir dicho hueco?

27. La rana obstinada. Buscando agua, una rana cayó en un pozo de 30 metros de hondo. En su intento de salir, la obstinada rana conseguía subir 3 metros cada día, pero por la noche resbalaba y bajaba dos metros. ¿Podrías decir cuántos días tardó la rana en salir del pozo?

28. El lechero ingenioso. Un lechero dispone únicamente de dos jarras de 3 y 5 litros de capacidad para medir la leche que vende a sus clientes. ¿Cómo podrá medir un litro sin desperdiciar la leche?

29. Un problema de peso. Un tendero dispone de una balanza y cuatro pesas distintas, y estas pesas son tales que le permiten pesar Cualquier número exacto de kilogramos desde 1 a 40. ¿Cuánto pesa cada una de las pesas?

30. Siempre diofanto. ¿Cuál es el número de 3 cifras, que cumplen la condición de que el producto de dichas cifras es igual a su suma?

31. Si nos falta la luz. En un cajón hay 12 pares de calcetines negros y doce pares blancos. No habiendo luz en la habitación, usted quiere coger el mínimo número de calcetines que le asegure que obtendrá al menos un par del mismo color. ¿Cuántos calcetines deberá tomar del cajón?

SOLUCIONES

Ahora confronte sus estrategias o formas de resolver cada uno de los ejercicios, con las posibles soluciones que se presentan a continuación. Recuerde que existen un sin numero de soluciones para un mismo problema.

1. El Almacén. Es necesario particularizar los casos y determinar que efectivamente se llega al mismo resultado, pero la parte importante es determinar las ecuaciones que componen la solución.

2. La tira de papel.
Los dobleces siendo siempre dos darían un número exponencial de dos, según el número de dobleces. Es decir: 2^n.

3. Números capicúas. Números capicúas.

4. Los retazos de colores. Cuatro o cinco son los colores necesarios.

5. Cuadros de Ajedrez. La explicación es incluir los cuadrados internos, el caso importante es determinar la ecuación que los gobierna.

N = 1 → 1
N = 2 → 5
N = 3 → 14

N = 4 → 30
N = 5 → 55
...

El resultado se determina a través de la fórmula:

$$\frac{(n)(n+1)(2n+1)}{6}$$

6. Madre.
Sea X la edad del niño, e Y la edad de la madre.

Y = X + 21
(Y + 6) = 5(X + 6)

Son las dos ecuaciones, son las que gobiernan el problema es necesario resolver una en términos de la otra así:

(X + 21 + 6) = 5(X + 6) queda todo en términos de X
X + 27 = 5X + 30 Despejando
X - 5X = 30 - 27
-4X = 3 entonces, es decir - 0,75 años, cuál corresponde a - 9 meses, es decir que el niño acaba de ser concebido. ¿Ahora dónde está el padre?

7. Herencia Caballos. La suma de los porcentajes de la herencia es 1/2 + 1/3 + 1/9 = 17/18 por lo que al hacer el reparto de los 35 caballos habrían sobrado 1/18 de estos, que es el equivalente a un caballo entero y parte de otro. Esta parte incompleta de caballo es la que se reparte de mas entre los hermanos para que se puedan llevar caballos enteros, y el otro caballo de sobra junto con el del matemático son los dos caballos que se lleva este.

8. Excursionista. Si es tomado como verdad habría que matarlo rápidamente, por que la respuesta sería mentira, y si se toma como tal habría que matarlo lentamente, por lo que sería verdad.

9. Entre vacas, ovejas y gallinas.

80 gallinas X 1 real = 80 reales = 20 ptas.
1 oveja a 5 ptas.
19 vacas a 25 ptas. = 475 pesetas

80 gallinas = 20 ptas.

1 oveja	=	5 ptas.
19 vacas	=	475 ptas.
100 animales		500 ptas.

10. Matadero. El problema nos pide matar al menos una por día, partiendo de matar una oveja por día en 15 días se matan 15 ovejas y nos quedan otras 15 por matar. Si en un día cualquiera matamos X ovejas en vez de una, y siendo X un número impar, el número total de ovejas muertas ese día aumenta en un número PAR.

Como no se puede llegar a 15 sumando números pares, no se pueden matar las 15 ovejas restantes aumentando un número par de ovejas muertas por día.

11. Valores. Veinte en número romanos es XX si le agregamos un uno en el medio nos queda XIX.

12. Ovejas. Un pastor tenía 5 ovejas y el otro 7.

13. El diablo y el campesino. Tenía 21 pesos.

14. La viejecita en el mercado. Tenía 59 huevos

15. La botella de vino. El envase cuesta 0,5 y la botella 9,5.

16. Llenar la piscina. 15 horas.

17. En el bar. El problema está en que el lenguaje comete un fallo. Cada uno paga 9 pesetas, en total 27, y dentro de esas, ya están las dos de propina. El razonamiento correcto es: 25 de los cafés, más 2 del bote, serían las 27 que en realidad han pagado.

18. María y Juan. Cuatro chicos y tres chicas.

19. El vagabundo. 8 pitillos.

20. JUAN Y PEDRO: Juan tiene 7 ovejas y Pedro tiene 5.

21. EL TIO Y EL SOBRINO: El tío tenía 30 años y el sobrino 20.

22. LAS TRES HIJAS: 9,2 y 2:
De todas las combinaciones de tres números cuyo producto es 36 sólo existen dos que a su vez tengan el mismo resultado al ser sumadas, teniendo en cuenta

que el personaje al que le ponen el a acertijo sabe en que portal vive, duda entre estas dos combinaciones y es cuando pide un dato más para poder resolverlo, 9x2x2= 36, 9+2+2= 13 y 6x6x1=36, 6+6+1=13 sólo la primera combinación es posible ya que en la segunda existen dos hermanas mayores y el último dato era que la mayor tocaba el piano.

23. LA COLECCIÓN DE MONEDAS: Había 7 monedas

24. PIES POR PULGADAS: 9 pies y 2 pulgadas, con lo que el dependiente le dio 2 pies y 9 pulgadas. Además se pueden encontrar otras soluciones. A ver si eres capaz de encontrarlas.

25. UN PROBLEMA DE BALANZA SIN PESAS: Compara 9 bolas Cualesquiera con otras 9 y deja los 9 restantes en la caja. Si la balanza se equilibra, la bola más pesada estará entre las 9 bolas que han quedado en la caja y si no, estará entre las 9 del platillo que se incline hacia su lado la balanza. Dividamos en 3 grupos de tres este conjunto y repitamos la operación. De esta forma, con dos pesadas habremos aislado la bola más pesada en un grupo de tres bolas. Si repetimos la operación una tercera vez, habremos aislado la bola más pesada de las otras.

26. LA TELA DE ARAÑA: 29 días: Cuando una tenga cubierto medio hueco en el día 29, la otra araña también lo tendrá, y entre las dos tendrán la ventana completa.

27. LA RANA OBSTINADA: 28 días.

28. EL LECHERO INGENIOSO: Primero llena la jarra de 3 litros. Luego vierte el contenido en la jarra de 5 litros. Vuelve a llenar la jarra de 3 litros y vuelve a verter su contenido en la jarra de 5 litros que ya está medio llena. Lo que quede en la jarra de 3 litros será un litro de leche.

29. UN PROBLEMA DE PESO: Las pesas son de 1, 3, 9 y 27 Kg. Con estas pesas siempre encontraremos una combinación. Por ejemplo, para pesar 23 es 27 - 3 - 1, y así cualquier otra combinación.

30. SIEMPRE DIOFANTO: 1, 2 y 3

31. SI NOS FALTA LA LUZ: Tres.

2.3.2. Ejercicios adicionales, propuestos

Estos ejercicios típicos pretender retar al lector, basando en la experiencia adquirida con los anteriores, con ejercicios clásicos matemáticos y algunos más cotidianos.

- Un reloj se atrasa un cuarto de minuto durante el día, pero debido al cambio de temperatura se adelanta un tercio de minuto durante la noche. ¿Al cabo de cuántos días se habrá adelantado el reloj dos minutos, sabiendo que hoy al atardecer marca la hora exacta?

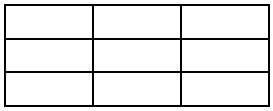
- Entre las ciudades M y C hay cinco caminos, desde la ciudad C hasta la ciudad P hay cuatro caminos. ¿De cuántas maneras diferentes se puede ir de la ciudad M hasta la ciudad P?

- En cierto país legendario había tres ídolos que hablaban: el dios de la verdad, que siempre decía la verdad; el dios de la mentira, que siempre decía mentiras; el dios de la picardía, a veces decía la verdad y a veces decía mentiras. Los tres ídolos eran iguales.

  Un niño muy hábil le preguntó al ídolo de la mitad quien era él, y el ídolo le contestó: “Yo soy el dios de la picardía”; le preguntó al de la izquierda quien era el que estaba a su lado, y el ídolo le contesto: “El dios de la verdad”; le preguntó al de la derecha quién estaba a su lado, y el ídolo le contestó: “El dios de la mentira”. Con estas respuestas pudo descifrar el nombre de cada ídolo.

- Tres hombres recibirán como pago de un servicio 21 vasos de vino iguales; siete llenos, siete medio llenos y siete vacíos. ¿De qué manera se deben dividir los vasos de forma que cada uno reciba la misma cantidad de vino?.

- Colocar los números del 1 al 9 en la siguiente figura de forma tal que todas las líneas sumen 15.



- Cómo medir 6 litros de agua si se tienen dos recipientes de 4 y 9 litros respectivamente.

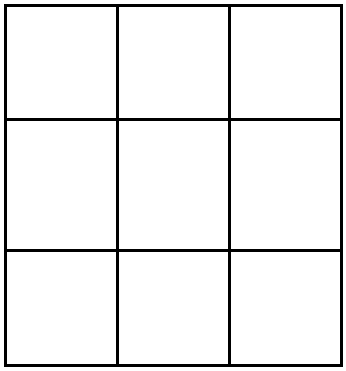
- Cuál es el tiempo mínimo requerido para asar 3 arepas, sabiendo que en la parrilla únicamente se pueden calentar 2 arepas a la vez; el tiempo de calentamiento para que se ase la arepa por uno de sus lados es de 30 segundos.

- Como intercambiar el contenido de dos vasos con líquidos diferentes (Leche - Refresco.

- Encuentre la solución más adecuada para resolver el siguiente problema: Un camión de carga pretendía atravesar un túnel a alta velocidad. Por equivocación del conductor el automóvil se atascó con el túnel, y por la inercia de la velocidad se introdujo un tercio de su longitud. La diferencia de altura entre el túnel y el camión es de unos pocos centímetros. El camión no tiene otro camino para llegar a su objetivo que no sea atravesando el túnel. Tampoco puede retroceder porque podría terminar de dañar la carga y además no se puede destrozar el túnel. Partiendo de esta problemática encuentre una solución práctica.

2.3.3. Ejercicios con Sudoku.

Sudoku es un interesante crucigrama lógico japonés, basado en los cuadrados latinos desarrollados por el matemático suizo Leonhard Euler en el siglo 18. El objetivo del juego es rellenar una tabla compuesta por 9 cuadriculas de 9 elementos donde van ubicados los números del 1 al 9 sin repetir ninguno por cuadricula.



Ejercicios.

1.

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

2.

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

3.

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

4.

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

5.

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

6.

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

7.

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

8.

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

9.

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

10.

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

11.

4 . 6 . 1 7 . 8 .	5 . . 3 4	8 . . . 5 . . . 7
7 . 8 . 6 . . 5 .	. . . .	. 4 . . 9 . 1 . 6
. 9 3	7 . . 4 . 1 . . 2	. . . 2 6 3 5 . .

12.

5 . . . 2 . . 9 .	7 . 4 3 8 . . . 6	. 1 . 6 4
9 . . . . 8 . 1 6	2 . . . 5 . . . 7	8 3 2 1 . . . . .
1 . . . . 5 . 4 .	5 . . . 7 2 6 . .	. 9 . . 8 . . . 3

13.

6 . . . . 8 5 . .	. . 2 . . . 9 6 1	. . 5 2 7
. . 9 . 8 . . . 3	5 . 7 . 3 . 8 4 6	3 . . . 7 . 1 . .
3 . . . . 2 4 . .	6 . 4 . . . . 9 .	. . 8 9 4 . . . 2

14.

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

15.

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

16.

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

17.

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

18.

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

19.

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

20.

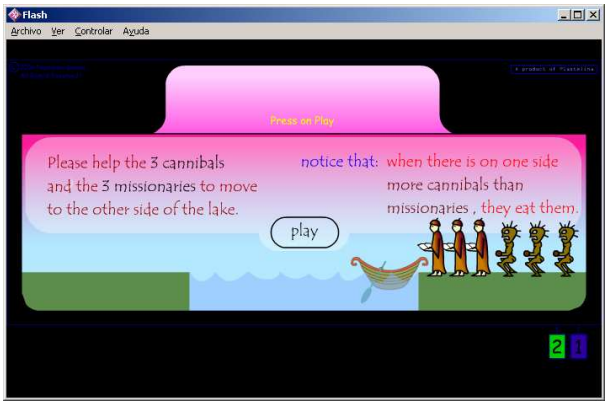
			5				7	9
8		7		1	6			
3	4		9					2
					7	4	3	6
		6		8		5		
4		2	3					
	1				2		5	3
		3	6	5		7		1
6								4

2.3.4. Ejercicios con el acompañamiento computacional.

La idea con esta serie de juegos es explorar un nuevo ambiente de desarrollo de habilidades lógicas, empleado el ordenador como medio.

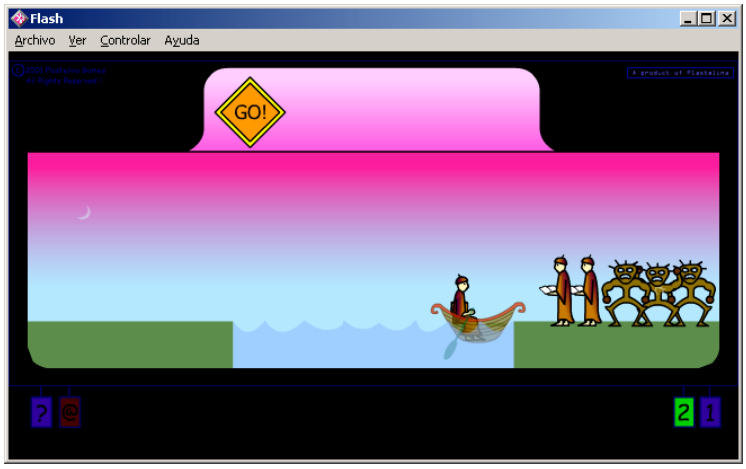
Los juegos están contenidos en CD en la carpeta **Juegos de lógica** que acompaña el libro, además se adicionan direcciones web (url) para acceder a juegos sobre Internet. Por respeto a los derechos de autor, cada grupo de juegos esta contenido en una carpeta que con el nombre del sitio donde se descargo. Cada juego esta acompañado de una pagina web con la explicación del juego y los enlaces para ingresar al sitio web.

Para realizar una corta demostración, ingrese a la carpeta **plastelina.net** y ejecute el juego que se llama **Padres_e_canibais**. Es un divertido juego desarrollado en Flash, compatible tanto para la ejecución en el computador como sobre paginas Web.

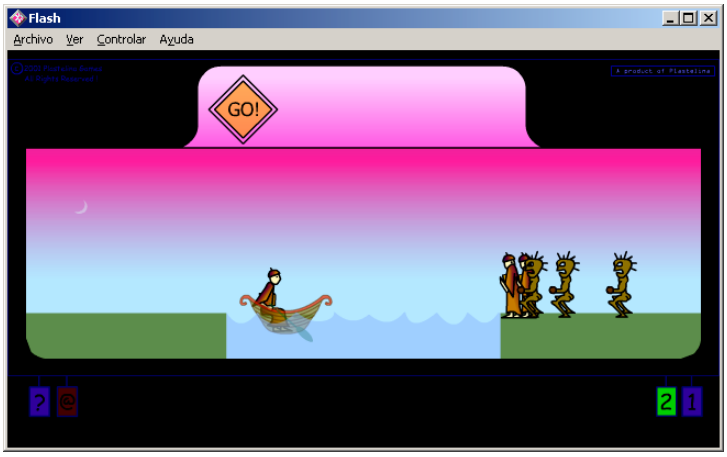


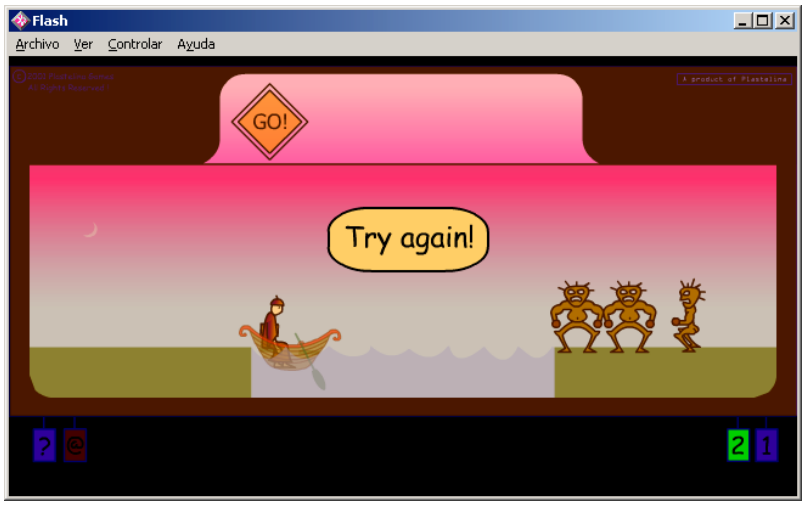
La idea del juego es pasar al otro lado del lago a los padres o misioneros y a los caníbales. El juego tiene una simple regla, donde hay más caníbales que misioneros los caníbales se comen a los misioneros, por eso hay la misma cantidad de personajes. En la barca pueden subir máximo dos personajes, sean 2 misioneros, 2 caníbales o un misionero y un caníbal.

Para iniciar la ejecución del juego de clic sobre la palabra **Play**. Para hacer una muestra de cómo los caníbales se comen a los misioneros, de clic sobre cualquier misionero y automáticamente se subirá a la barca.



En la parte superior de la pantalla se habilita el símbolo **GO!** para que el personaje o personajes que están sobre la barca se desplacen al otro lado. De clic para desplazar al misionero al otro lado. Como a la derecha quedan menos misioneros que caníbales, los caníbales se los comen, así como se muestra a continuación.



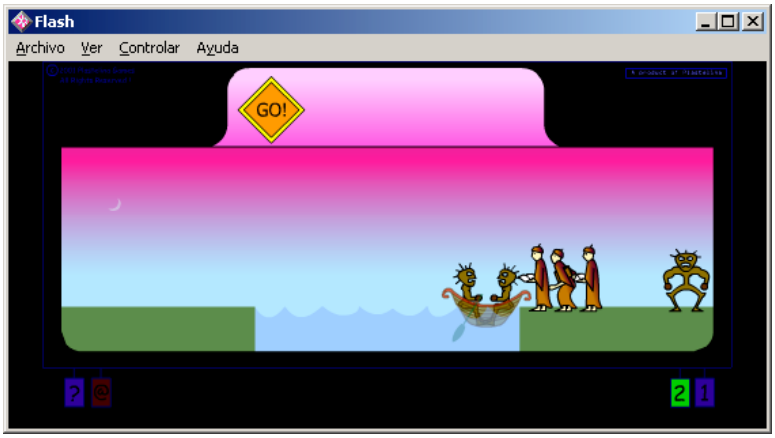


Para reiniciar el juego de clic sobre la opción **Try again!**.

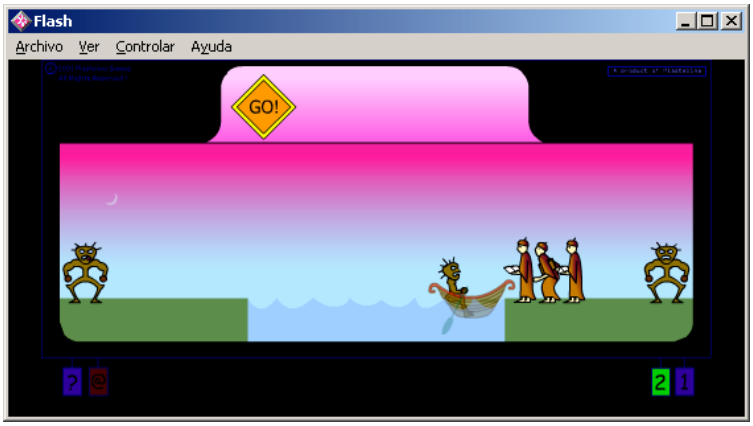
Ahora rete a su imaginación y capacidad de abstracción dándole solución al juego, piense en que es lo mas adecuado para llevar a todos los personajes al otro lado del lago sin que los misioneros sean devorados por los caníbales.

SOLUCION AL JUEGO

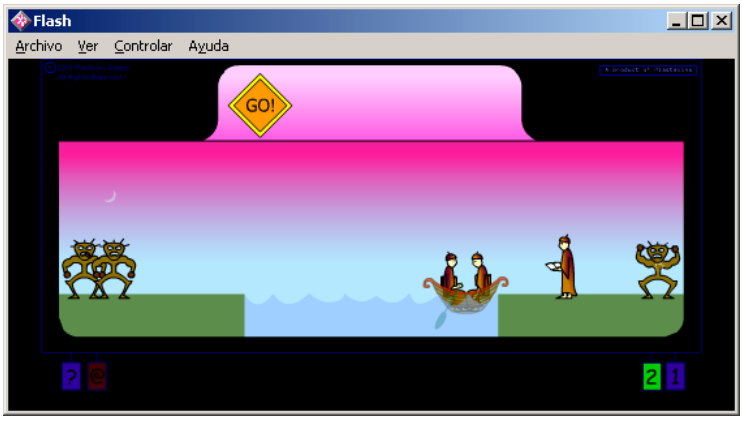
El primer paso consiste en llevar al otro lado a los caníbales, por eso se suben 2 a la embarcación y se llevan al otro lado.



Descargue uno de los caníbales y regrese por el otro.

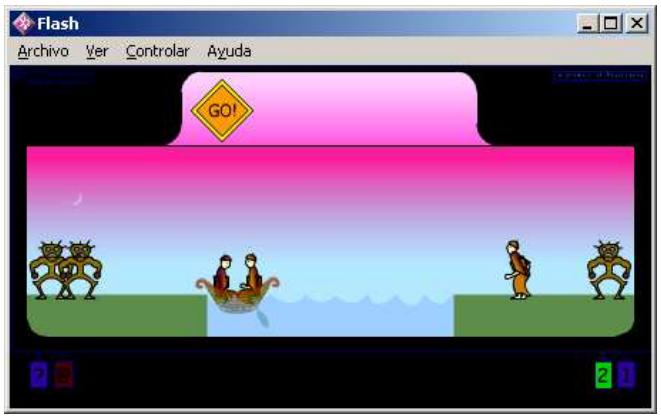


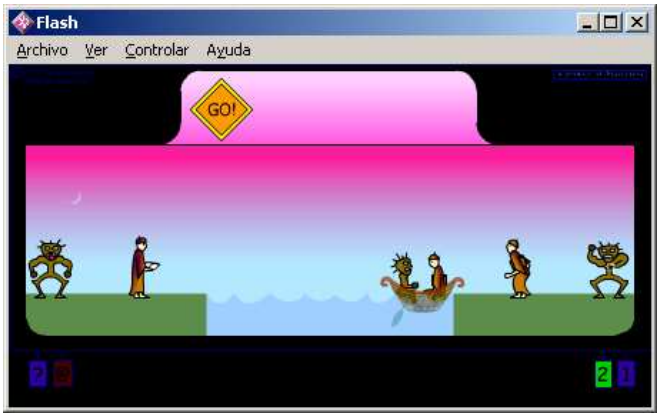
Ahora como la barca no puede devolverse sola tendrá que ir con un caníbal al otro lado, descárgalo y subir dos misioneros.



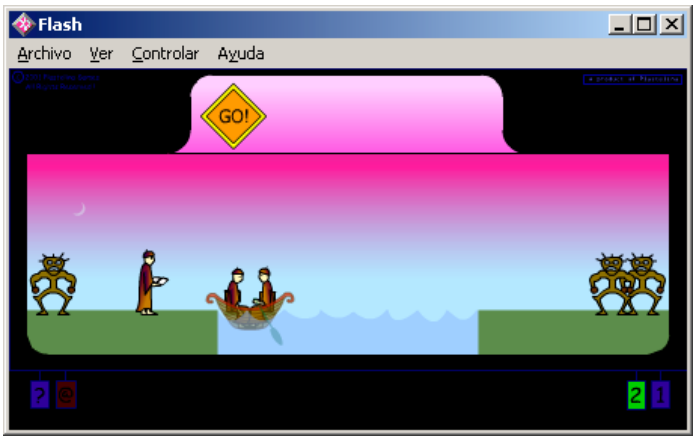
Si observa con detenimiento ninguno de los caníbales procede a devorar a los misioneros por que de cada lado hay la misma cantidad de misioneros y de caníbales.

Para que no altere las cantidades, descargue a un misionero y suba a un caníbal a la embarcación y desplácelos al otro lado del lago.

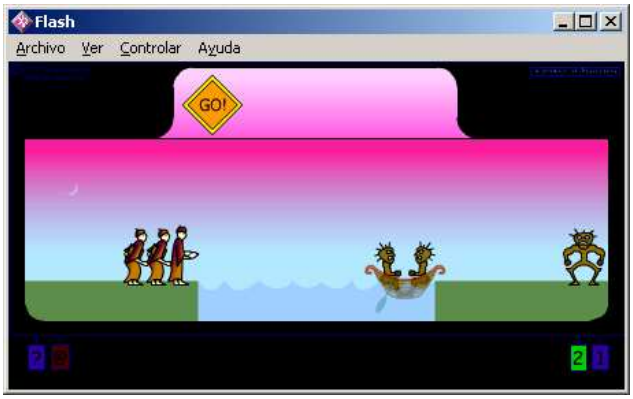


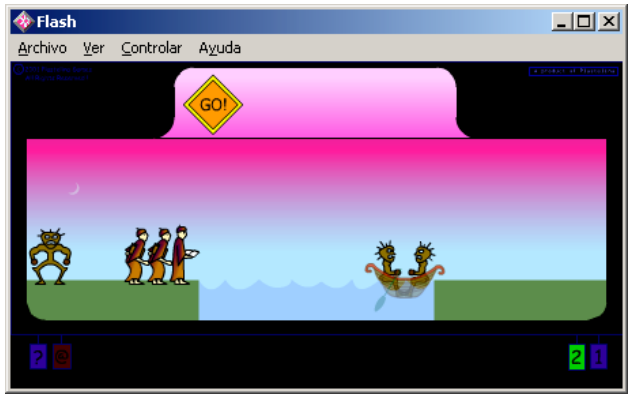


En este caso descargue al caníbal, y suba al otro misionero y diríjalos al otro lado del lago.



Descargue ahora a los misioneros, suba a la embarcación al caníbal y con este, traiga a la orilla de la izquierda a los otros de caníbales.





Cuando se descarga al ultimo caníbal se gana el juego. La clave era asegurar del otro lado a los misioneros y luego pasar a los caníbales. Pruebe otras formas de resolverlo, recuerde que un problema tiene múltiples soluciones.

Ahora abra la carpeta **pabsolutist.comindex_es** donde se encuentra la versión digital del juego Sudoku y un juego digital de preguntas en ingles.

De doble clic sobre la pagina web **index_es** para que pueda ver la siguiente pagina web.

De manera similar todos los juegos son versiones libres, tríales y demos. Ponga a prueba de una manera interactiva sus conocimientos y su forma de solucionar problemas.

RESUMEN.

- Existen una serie de formas para solucionar problemas de cualquier índole. Así como los científicos usan el método científico, en el ámbito computacional existen una serie de métodos para analizar un problema y generar una solución informática.

- Los juegos de lógica son maneras de ejercitar la mente para adquirir mas capacidad intelectual.

- Desde que se emplean las computadoras para educar, se han ampliado las posibilidades en la generación de innumerables cantidades de ambientes como juegos de rol, estrategia, en primera persona, en red y con dispositivos especiales como las gafas en 3D (tercera dimensión) entre otros.

EJERCICIOS DE AUTO EVALUACIÓN.

Explore el contenido del CD y observe con detenimiento las especificaciones y reglas de cada juego. Instale en su ordenador los de su preferencia y rete entonces su intelecto.

3

Fundamentos

Plan general.

OBJETIVOS

- Presentar el concepto de algoritmo.
- Conocer los tipos de representación de algoritmos.
- Conocer la representación de los datos de la naturaleza en datos computacionales.
- Profundizar en el manejo de la memoria en sistemas de computo.
- Entender la representación de formulas matemáticas en formulas matemáticas computaciones.

3.1. INTRODUCCIÓN.

El propósito de este capitulo es proporcionar al lector los conceptos básicos de fundamentos de programación a partir de una primera técnica denominada algoritmo. De una manera progresiva se irán conociendo los elementos de la algoritmia hasta desarrollar programas básicos computacionales.

3.2. ALGORITMO.

Cuando se quiere dar solución a un problema a nivel computacional se emplean algoritmos. Estos corresponden a los bosquejos de posibles formas de solución, sobre una serie de modelos de interpretación fácil y rápida, como son: el diagrama de flujo y el pseudocodigo (que se explicaran con mas detenimiento más adelante. Entiéndase algoritmo como una secuencia de pasos coherentes y finitos para resolver un problema. En un algoritmo se fijan una serie de acciones precisas y lógicas que conllevan a resolver cualquier problema.

Los algoritmos están conformados por tres elementos: Entrada, Proceso y Salida. En algunos casos un algoritmo puede requerirlos en cualquier orden, o emplear uno o varios, según la temática que se plantee para resolver el problema. Para fijar estos conceptos, se interpretará con el siguiente ejemplo:

Suponga que se desea sumar dos números, y saber el valor de esa suma.

Entrada: es conocer los dos números, por ejemplo 5 y 9.

Proceso: ejecutar la operación.

```
    5
  + 9
--------
   14
```

Salida: es obtener el valor de la suma que es igual a 14. Ya se ha realizado el primer algoritmo, se creó una secuencia de pasos coherentes y dependientes.

Coherentes en la medida que se obtuvo un valor de la forma correcta como siempre se efectúa una suma.

Dependiente en que no se puede sumar dos números que no se conocen, y obtener el resultado de la suma sin sumar los números.

Ahora observe otro ejemplo.

Suponga que alguien le dice un número y a ese número le suma 15.

Entrada: imagine que el número que se da es 10. el número 15 no corresponde a la entrada por que ya es un número conocido. El número 15 es un valor fijo o un valor constante.

Proceso: sumar 10 y 15.

```
    10
  +15
--------
   25.
```

La salida: el resultado de la suma es 25.

Nota.

A medida que se plantean los ejercicios y se analizan con cierto detenimiento se comprenderán de forma más profunda estos elementos (datos entrada, proceso y datos de salida). Por el momento es importante que se conozca su definición y se obtenga una idea personal de ellos.

3.3. TIPOS DE DATOS

El objetivo principal de todo programa para computador es procesar información, la cuál es clasificada según su característica o naturaleza, como: datos personales, monetarios, financieros, estadísticos, entre otros. Los tipos de datos proporcionan una variada gama de modelos de datos, más adecuados para albergar esa información dependiendo de su singularidad. Observe la siguiente analogía, no es correcto parquear un camión de carga, en donde se puede parquear un automóvil pequeño como un compacto. De esa misma manera; no es correcto alojar el nombre de una persona en un tipo de dato, en donde se pueden almacenar datos numéricos.

Los tipos de datos se crearon para representar la información que se encuentra en la naturaleza, como: datos de temperatura, estatura de una persona, edad, altura en una manejable por los sistemas de computo. Un programa de computador representa o simula un comportamiento humano, social, matemático.

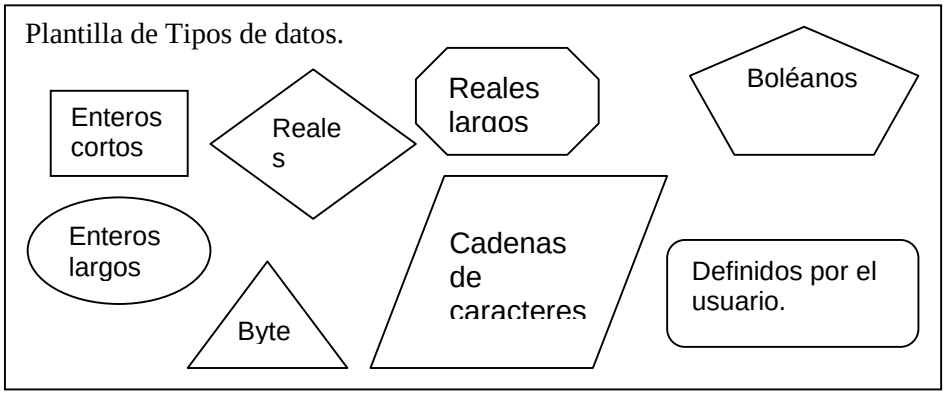
Definiendo lo anterior en otros términos se tiene: cuando se maneja y procesa información lo más adecuado es que dicha información no pierda sus características y su forma. Los sistemas implementados en computadoras deben permitir dicha conservación de características y para ello ofrecen modelos de datos. Entonces, los modelos de datos de forma homóloga a una plantilla de figuras geométricas, se construyeron con la finalidad de proporcionar varias formas de las cuáles el usuario podrá disponer cada vez que las necesite y adaptarlas a la naturaleza de su uso.

Cada figura dentro de la plantilla tiene un uso particular, se pueden construir círculos, óvalos, rectángulos, entre otros. De forma similar los modelos de datos permiten tener números, textos o cadenas de caracteres, entre sus usos más comunes. Además cada uno de ellos puede llegar a tener sus submodelos. Por ejemplo, los números tienen dos modelos principales: los enteros y los reales. Los números enteros corresponden a los números que no poseen valores con decimales, mientras que los números reales si poseen valores con decimales.

Este tipo de subdivisión proporciona modelos más cercanos a la realidad, por ejemplo: las temperaturas por lo general tienen valores decimales (25.6, 32.56 grados

centígrados), la población mundial es un número muy grande, pero en cantidades exactas, nunca se existen medios, tercios, 0.256 personas.

Partiendo de ello, observe la siguiente plantilla de modelos o tipos de datos.



En la siguiente sección se conocerán los tipos de datos y sus usos más comunes.

3.3.1. Datos tipo numérico

Como su nombre lo indica, son tipos de datos en los cuáles se puede almacenar y representar números y sobre ellos se pueden efectuar operaciones o cálculos matemáticos. Los datos de tipo numérico se clasifican en dos categorías distintas: los numéricos enteros y los numéricos reales.

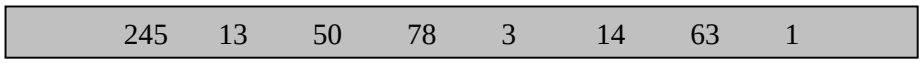
3.3.1.1. Tipo numérico entero

Este tipo de dato permite acumular cantidades enteras o cantidades sin valores decimales, y pueden ser positivos o negativos, excepto los datos de tipo Byte. Existe además, una subclasificación de este tipo de dato en Byte, enteros cortos y enteros largos. Lo anterior con el fin de proporcionar un tipo de dato más al acomodo de las necesidades en el manejo de cantidades enteras.

3.3.1.1.1 Byte

Este tipo de dato puede manejar cantidades en el rango de valores: 0 a 255.
Por lo tanto una cantidad de tipo Byte no puede ser menor de 0, y mayor que 255.

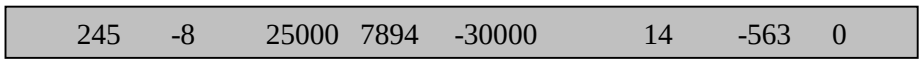
Ejemplos de este tipo de dato son:



3.3.1.1.2 Enteros cortos

Este tipo de dato puede manejar cantidades en el rango de valores: -32768 a 32767.
Por lo tanto una cantidad entera corta no puede ser menor de –32768, y mayor que 32767.

Ejemplos de este tipo de dato son:



3.3.1.1.3 Enteros largos

Este tipo de dato puede manejar cantidades en el rango de valores: -2147483648 a 2147483647. Por lo tanto una cantidad entera larga no puede ser menor de –2147483648, y mayor que 2147483647.

Ejemplos de este tipo de dato son:

3.3.1.2. Tipo numérico real

Este tipo de dato permite acumular cantidades reales o cantidades con valores decimales, y pueden ser positivos o negativos. Existe además, una subclasificación de este tipo de dato en reales cortos y reales largos.

3.3.1.2.1 Reales cortos

Este tipo de dato puede manejar cantidades en el rango de valores: -3.40E+38 a 3.40E+38.

Por lo tanto una cantidad real corta no puede ser menor de –3.40E+38, y mayor que 3.40E+38.

Ejemplos de este tipo de dato son:

245.68	-8.23	25000.1245	7.894	-30000.478

3.3.1.2.2 Reales largos

Este tipo de dato puede manejar cantidades en el rango de valores: -1.79D+308 a 1.79D+308. Por lo tanto una cantidad real larga no puede ser menor de –1.79D+308, y mayor que 1.79D+308.

Ejemplos de este tipo de dato son:

40000.12455665	-8.2654878787	-50000.14898211542
5000000.49887979565	-45792.15499744646	

3.3.2. Datos tipo lógico

Este tipo de dato puede albergar uno de dos valores, Verdadero o Falso. Se emplea a menudo en condiciones, banderas, entre otros usos más comunes.

Ejemplos de este tipo de dato:

Verdadero	**Falso**

3.3.3.Datos de tipo cadena de caracteres.

Se conocen como cadena de caracteres aquellos elementos que albergan uno o más símbolos alfabéticos, numéricos y especiales de uso regular, y se encuentran contenidos entre comillas simples (´´. A continuación se mostrarán ejemplos de cada tipo:

Símbolos alfabéticos: el alfabeto (a,b,...y,z) (A,B,...,Y,Z).
Símbolos numéricos: los números (0,1,2,...,8,9).
Símbolos especiales: (¡, @, #, $,..., &,...)

Entonces, una cadena de caracteres puede estar conformada por solo símbolos alfabéticos, o numéricos o especiales. Además se pueden crear combinaciones entre símbolos como se observa en la figura 1.9, por ejemplo:

Alfabéticos: nombres de personas o elementos.
´Carlos Andrés Zapata Ospina´
´Fundamentos de Programación'

Numéricos: códigos o números con los cuales no se efectúan operaciones.
´75860´ ´5779898´ ´007´

Mixtos: agrupación de elementos de diferentes símbolos que producen un significado.

´Luis V´ ´caz8@correo.com´
´789-562´ ´1 de marzo del 20001´

3.3.4.Tipos de datos definidos por el usuario.

A partir de los anteriores tipos de datos preestablecidos (enteros cortos, enteros largos, reales cortos, entre otros), se pueden definir otros tantos mediante una serie de combinaciones. Este característica brinda una amplia gama de posibilidades para resolver un algoritmo. Estos tipos de datos podrían clasificasen en:

- Arreglos (Vectores, Matrices.)
- Estructuras de datos.
- Punteros.

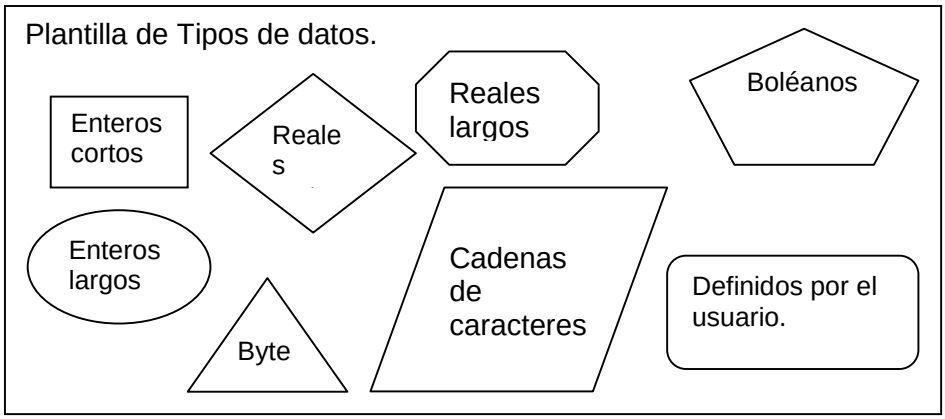
Estos tipos de datos se explicaran con más detenimiento en capítulos posteriores.

Nota.

El propósito de conocer los diferentes tipos de datos pretende orientar al lector, en la forma en la cual la información y los datos son realmente procesados y manejados en el computador mediante programas (software).

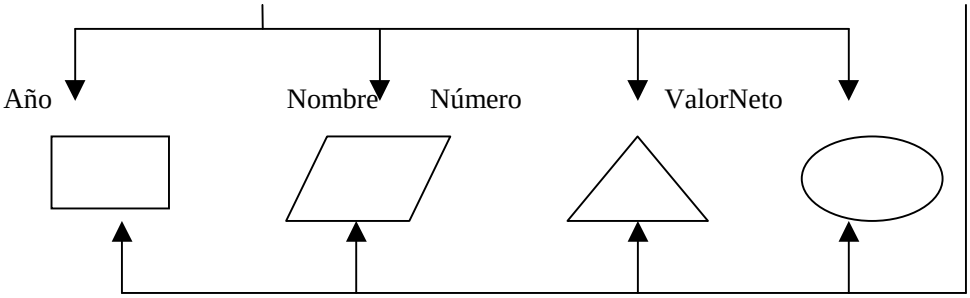
3.4. VARIABLES.

Una variable es un espacio reservado en memoria que recibe un nombre representativo, donde se almacena un dato de cualquier tipo mencionado anteriormente. Las variables poseen la característica de cambiar de contenido sobre la ejecución de un algoritmo. Además una variable solo puede almacenar información de un tipo de dato, en la cuál se almacenan datos numéricos, o cadenas de caracteres, o datos de tipo lógico. Nunca una misma variable puede almacenar datos de diferentes tipos en un propio algoritmo. Definido de otra manera, una variable, es un contenedor (como una caja o recipiente) en donde se puede almacenar datos y siendo estricto solo puede guardar un elemento a la vez.



Retomando la explicación de los tipos datos en la figura 1.3. se observa que las variables están ligadas a un tipo de dato.

- Nombre de la variable.
- Tipo de dato al que corresponde.



3.4.1. Reglas para nombrar variables

3.4.1.1. El nombre de una variable debe comenzar con una letra, seguida de otras letras, números o caracteres. Como ejemplo:

A,	valor,	num1,	valor_num1
Nombre	ciudad	potencia	raíz

3.4.1.2. El nombre de las variables nunca debe emplear caracteres especiales como:

¡,!, |, “, @, ·, #, $, %, /, (,), =, ‘, ¿, ?, +, -, *, {, }, [,], ^, entre otros.

Es incorrecto nombrar:

e-mail	cantidad_$	#1	numero.1	el%interes

∑Sumatoria

3.4.1.3. Cuando se nombre una variable, nunca emplee tildes, ni espacios, ni signos de puntuación. Por ejemplo:

Usos incorrectos

Código	Nombre Empleado	Cantidad.

Usos correctos

Codigo	NombreEmpleado	Cantidad

3.4.1.4. Las variables no pueden tener nombres empleados por palabras reservadas. Estas se encuentran contenidas en la mayoría, por funciones que se verán en el apartado 1.10.1 y 1.10.2, como funciones aritméticas y de cadena de este capítulo.

Con lo anterior se concluye que las variables son espacios reservados en memoria, definidos sobre un modelo de dato que indica que tipo de valor puede ser almacenado en cada una de ellas. También que cada una de ellas recibe un nombre simbólico con el cuál pueda ser usado.

El formato para la declaración de variables es el siguiente:

```
Declaracion Nombre_Variable.
```

Por ejemplo:

```
Declaracion Numero
```

Otra forma de declarar variables es también:

```
Declaracion Nombre, Salario
```

Donde se indica que ***Nombre*** y ***Salario*** son dos variables independientes una de la otra, pero declaradas sobre la misma instrucción de ***Declaración***

Nota.

El nombre que puede recibir una variable es indiferente al valor que pueda albergar. La idea es nombrar las variables con nombres **nemotécnicos** o que indiquen con su nombre que van a almacenar. El manejo de los nombres de variables **nemotécnicos** será un punto que más adelante se explicará.

3.5. CONSTANTES

Como su nombre lo indica son datos que siempre van a tener el mismo valor, y por ninguna razón este valor puede ser modificado. También comparten el mismo concepto de reglas empleado en la declaración de variables. Las constantes se clasifican en:

- Constantes literales.
- Constantes declaradas.

3.5.1. Constantes literales.

Son constantes que expresan el sentido directo de un valor de Cualquier tipo o de una cadena de caracteres. Por ejemplo:

"la programación es fácil"	4895.6	"Carlos"
3.1416	"24 de abril 2000"	

3.5.2. Constantes Declaradas.

Son constantes que deben ser declaras antes de ser usadas. En la declaración debe usarse un nombre representativo que no se repita, seguidamente del valor que se desea asignar. Cabe resaltar que para poder declararse una constante declarada debe usarse una palabra reservada Constante. El formato de declaración es el siguiente:

Constante Nombre: valor o cadena de caracteres.

Por ejemplo:

Constante Pi = 3.1416
Constante Nombre = "Carlos Andrés"
Constante Salario = 150000

El verdadero propósito de las constantes y variables se conocerá más adelante con ejemplos explicativos, por el momento es más importante conocer el concepto.

3.6. REPRESENTACIÓN DE EXPRESIONES ARITMÉTICAS.

De manera frecuente es necesario desarrollar algoritmos en los cuáles, se utilicen expresiones aritméticas para realizar de una manera rápida operaciones complejas.

Para esta sección en especial de libro es importante centrar la atención, en la representación de dichas expresiones empleando los operadores antes vistos, con el objetivo de convertir ese conocimiento teórico en uno práctico. Observe detenidamente cada uno de los siguientes ejemplos.

Ejercicio.

Representar la siguiente expresión **5 (5^2)**.

Expresión aritmética **5 x (5^2).**

El símbolo para representar la multiplicación es ***.**
El símbolo para expresar el exponencial es **^.**

Expresión Algorítmica resultante es **5 * (5 ^ 2)**

Nota.

Todo elemento numérico seguido de un paréntesis, se interpretará como una multiplicación, entonces **5 (5^2)** es igual a **5 x (5^2)**

Ejercicio.

Representar la siguiente expresión.

Expresión aritmética $\frac{(a^2 + b^2)}{2}$

Expresión Algorítmica resultante es **(a^2 + b^2) / 2**

De forma homóloga se representan en general todas las expresiones aritméticas, lo más importante es reconocer el símbolo equivalente de la expresión en el algoritmo.

RESUMEN.

- La lógica corresponde a la forma de cómo se debe pensar para encontrar soluciones a través de diferentes métodos.

- Cuando se quiere dar solución a un problema a nivel computacional se emplean algoritmos. Estos corresponden a los bosquejos de posibles formas de solución, sobre una serie de modelos de interpretación fácil y rápida.

- Los algoritmos están conformados por tres elementos: Entrada, Proceso y Salida.

- Los tipos de datos proporcionan una variada gama de modelos de datos, más adecuados para albergar esa información dependiendo de su singularidad.

- Los tipos de datos se crearon para representar la información que se encuentra en la naturaleza, como: datos de temperatura, estatura de una persona, edad, altura en una manejable por los sistemas de computo.

- Datos de tipo numérico son tipos de datos en los cuáles se puede almacenar y representar números y sobre ellos se pueden efectuar operaciones o cálculos matemáticos.

- Datos de tipo lógico puede albergar uno de dos valores, Verdadero o Falso.

- Se conocen como cadena de caracteres aquellos elementos que albergan uno o más símbolos alfabéticos, numéricos y especiales de uso regular, y se encuentran contenidos entre comillas simples (´´.

- Una variable es un espacio reservado en memoria que recibe un nombre representativo, donde se almacena un dato de cualquier tipo mencionado anteriormente. Las variables poseen la característica de cambiar de contenido sobre la ejecución de un algoritmo.

- El nombre de una variable debe comenzar con una letra, seguida de otras letras, números o caracteres.

- El nombre de las variables nunca debe emplear caracteres especiales.

- Cuando se nombre una variable, nunca emplee tildes, ni espacios, ni signos de puntuación

- Las constantes son datos que siempre van a tener el mismo valor, y por ninguna razón este valor puede ser modificado.

EJERCICIOS DE AUTO EVALUACIÓN.

1. El diagrama de flujo y el pseudocodigo son?
2. Un algoritmo es?
3. Los algoritmos están conformados por tres elementos:
4. Los tipos de datos se crearon para representar:
5. Los datos de tipo numérico entero pueden representar
6. Los datos de tipo byte pueden representar cantidades entre:
7. Los datos de tipo entero corto pueden representar cantidades entre:
8. Los datos de tipo entero corto pueden representar cantidades entre:
9. Los datos de tipo numérico real pueden representar
10. Los datos de tipo lógico pueden representar:
11. Que es una variable?
12. Que es una constante?

RESPUESTAS A LOS EJERCICIOS DE AUTO EVALUACIÓN.

1. Algoritmos
2. Una secuencia de pasos coherentes y finitos para resolver un problema
3. Entrada, Proceso y Salida
4. La información que se encuentra en la naturaleza
5. Cantidades enteras o cantidades sin valores decimales, y pueden ser positivos o negativos
6. 0 a 255
7. -32768 a 32767
8. -2147483648 a 2147483647
9. Cantidades reales o cantidades con valores decimales, y pueden ser positivos o negativos
10. Uno de dos valores, Verdadero o Falso
11. Un espacio reservado en memoria que recibe un nombre representativo, donde se almacena un dato de cualquier tipo
12. Son datos que siempre van a tener el mismo valor, y por ninguna razón este valor puede ser modificado

EJERCICIOS PROPUESTOS.

Representar algorítmicamente las siguientes expresiones:

- $a^2 + b^2 + c$
- $\frac{A}{B} + (C + D)^2$
- $c(d^2 + a)$
- $\frac{D+7}{B-3}$
- $\frac{c+32-1}{5}$
- $A^{L+2} + B^{L+6}$
- $\frac{x^2 + y^2}{5 + r^2}$
- $9\left[(1 - X) - \left(\frac{4}{Y}\right)\right]$
- $T^2 - 2AC$
- $-b + \sqrt{\frac{b^2 - 4ac}{2a}}$

Diseñe la solución para resolver cada uno de los siguientes problemas y trate de refinar sus soluciones mediante algoritmos adecuados.

- Cambiar la llanta de un automóvil.
- Ir al cine.
- Hacer un retiro de un cajero electrónico.
- Llamar a un contacto con un celular.
- Realizar una llamada desde un teléfono publico
- Cocinar una tortilla
- Arreglar un pinchazo en la bicicleta
- Freír un huevo.

4

Algoritmia

Plan general.

OBJETIVOS

- Presentar las diferentes formas de representar un algoritmo.
- Reforzar el concepto de datos de entrada, proceso y datos de salida.
- Mostrar los diferentes operadores que intervienen en un algoritmo.
- Conocer y aplicar la jerarquía de operadores.
- Desarrollar algoritmos básicos.

4.1. INTRODUCCIÓN.

Los algoritmos son los primeros pasos hacia el mundo de la programación de computadores. La importancia de su estudio, permite mejorar la capacidad de abstracción que desarrolla también las matemáticas y además facilita la aplicación de los conocimientos básicos de programación, a cualquier campo de las ciencias de la computación de una manera transparente, es decir con las bases que se proporcionan aquí el lector podrá estudiar mas lenguajes de programación de ultima generación, aprehender sistemas de desarrollo multimedia, trabajar en herramientas de simulación y aplicaciones matemáticas.

Es por lo anterior, que los algoritmos son el campo más importante que todo estudiante de computación o áreas afines debe conocer para garantizar desde un referente lógico.

4.2. REPRESENTACIÓN DE LOS ALGORITMOS.

La codificación de un algoritmo no se debe limitar estrictamente, a la sintaxis o a la forma en el cuál un lenguaje de programación en particular, como (Basic, Cobol, Fortran, Pascal, C, C++, entre otros), pueda llegar a representarlo.

Se podría comparar un algoritmo como el bosquejo que se hace antes de iniciar una obra. En la cual el artista plasma sus ideas y realiza cierta cantidad de modelos con el propósito de generar un resultado óptimo e ideal.

El algoritmo por si mismo contiene una notación fácil de entender, con un lenguaje familiar representativo. Existen en la actualidad, gran cantidad de métodos para la representación de algoritmos, de los cuáles serán motivo de estudio en este libro: **El Pseudocódigo.**

Es de anotar que existe otro elemento para representar algoritmos en forma grafica, llamado diagrama de flujo, del cual se conocerá de forma general.

4.2.1. El Diagrama de Flujo (DF)

El DF es una representación gráfica de un algoritmo, que emplea una serie de figuras o símbolos, para indicar un proceso o instrucción sobre el cuál se ejecuta una acción.

Un diagrama de flujo se comporta de manera similar a la representación de un organigrama dentro de una compañía. En el cuál se muestra el rango de mando u operación, se señala además, el flujo o conducto regular para la toma de decisiones o procesos que se generan dentro de la misma. Además, cuando se realiza la tarea del análisis y diseño de cualquier proceso, unas de las herramientas de modelado emplean características similares al DF, como son: El diagrama de flujo de datos, diagrama entidad – relación, diagramas de transición de estados, para el análisis estructurado moderno.

Las características más sobresalientes de un DF se muestran a continuación, con el sentido de proporcionar una visión conceptual del mismo:

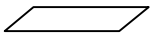
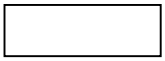
4.2.1.1. Características del DF

- La notación concisa que posee, facilita una interpretación inmediata y un entendimiento consecuente. Simplemente con observarlo se puede comprender.

- Los DF generalmente ocupan una página, logrando captar toda la atención por parte del que lo observa.

La siguiente figura muestra los símbolos representativos empleados por el diagrama de flujo.

SÍMBOLOS PARA DIAGRAMA DE FLUJO.

Símbolos Principales	**Nombre y Función**
	**Inicio**. Representa el comienzo de un programa.
	**Fin**. Representa el final de un programa.
	**Lectura**. Captura de datos introducidos por el teclado.
	**Salida**. Los mensajes o resultados de procesos sobre la información son mostrados por pantalla.
	**Asignación**. Representación de Cualquier tipo de proceso donde cambie el valor de las variables.
	Línea de Flujo. Indica la dirección de ejecución del flujo del programa.

Nota.

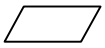
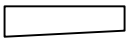
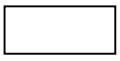
Los símbolos vistos son los modelos estándar en la representación de diagramas de flujo. Por lo general los otros modelos encierran cierta similitud con estos y su uso sobre el diagrama es de forma exacta.

4.2.2. Pseudocódigo

Se conoció anteriormente una serie de símbolos para representar un algoritmo; el Pseudocódigo de forma similar representa un algoritmo pero empleando palabras simbólicas en vez de gráficos. Para complementar esta explicación, se cita a continuación la definición dada por **Luis Joyanes Aguilar**:

> "El Pseudocódigo nació como lenguaje similar al inglés y era un medio de representar básicamente las estructuras de control de programación estructurada que se verán en capítulos posteriores. Se considera un primer borrador, dado que el Pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. El Pseudocódigo no puede ser ejecutado por una computadora. La ventaja del Pseudocódigo es que en su uso, en la planificación de un programa, el programador se puede concentrar en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje especifico. "[3]

En la siguiente figura se representan las palabras simbólicas, de forma similar y en el mismo orden, de los símbolos gráficos:

Símbolo en DF	**Palabra en Pseudocódigo**	**Comentario**
	**Inicio**	Representa el comienzo de un programa.
	**Fin**	Representa el final de un programa.
	**Leer**	Captura de datos introducidos por el teclado.
	**Escribir**	Los mensajes o resultados de procesos sobre la información son mostrados por pantalla.
		Anteriormente se realizaba una asignación u operación en el símbolo proceso. En Pseudocódigo la operación o asignación se efectuara de manera igual, se omite por completo el símbolo.
		En DF La flecha indica el flujo u orden con el cuál se realizan las operaciones, en Pseudocódigo se remplazara por la justificación de cada una de las instrucciones.

[3] JOYANES AGUILAR, Luis. Fundamentos de Programación. Madrid. McGraw Hill. 1997. P.58.

4.2.2.1. Reglas para usar Pseudocódigo.

- El nombre de una variable debe respetar las reglas del apartado **3.4.1 reglas para nombrar variables** del capítulo 3. Además nunca se debe nombra una variable con las palabras que representan los símbolos como: Inicio, Fin, Leer, Escribir, entre otras que gradualmente se conocerán.

- Los mensajes de texto o cadenas de caracteres deben estar contenidos entre comillas dobles para el Pseudocódigo (“”), mientras que en el DF se emplearon comillas simples (‘’).

- A diferencia del DF, el Pseudocódigo estándar requiere de la declaración de las variables que emplee, en uno o varios tipos de dato. Para el caso en particular de esta publicación y las herramientas software con las que se pretende realizar las prácticas y generar conocimiento, no se usará declaración de variables sobre un tipo de dato. Simplemente se declararán variables para reservar los espacios en memoria y realizar la tarea de inicializar las mismas.

- El encabezado que se va a adoptar para indicar las declaraciones de una o más variables será **Declaracion**, de forma seguida se encontrara el nombre de las variables que corresponden a ese tipo de dato. Observe el siguiente prototipo:

Declaración nombre de variable 1, nombre de variable 2, ...

Ejercicio.

Representar en un Pseudocódigo la declaración dos variables.

```
1   'Pseudocódigo Declaración de variables
2
3   Inicio
4       Declaracion Numero1
5       Declaracion Numero2
6   Fin
7
```

EXPLICACIÓN POR INSTRUCCIÓN:

(1). Todo Pseudocódigo necesita de un nombre representativo, en este caso se llama **Declaración de variables**.

(3). La palabra **Inicio** indica que se va iniciar un Pseudocódigo.

(4). Ese fragmento del código indica que la variable **Número1** se va a declarar.

(5). Ese otro fragmento de código indica también que la variable **Número2** se va a declarar.

(6). La palabra **Fin** indica que se terminó un Pseudocódigo.
Es importante recordar los tipos de datos para la declaración de variables; en la sección **3.3. Tipos de datos**, del capítulo 3.

Nota.

Cuando se realiza la operación de declaración de variables, se está reservando directamente en la memoria del computador, espacios que estén disponibles para su uso. El nombre que se les proporciona a cada una de las variables facilita su manejo, ya que por lo general el computador les asigna nombres con valores hexadecimales, lo cual hace que sea difícil manejarlo como lo hace el ordenador. Por tanto se les asigna nombres representativos.

Cuando se declara una variable no posee ningún tipo de dato definido, en el momento que se les asigna por primera vez un valor, automática e internamente se define un tipo de dato más acorde con el dato que se almacena en la variable.

Conocidos ahora los elementos para el desarrollo de algoritmos, se repasará entonces una herramienta básica para el análisis y desarrollo de dichos diagramas y Pseudocódigo (que se conocerán sobre esta sección más adelante)

Como se vio en el capítulo anterior, todo algoritmo es una secuencia de pasos secuencias y coherentes para solucionar un problema. Además que en la mayoría de los casos, todo algoritmo posee: **datos de entrada**, **proceso** y **datos de salida**.

DATOS DE ENTRADA.

Se entiende por datos de entrada, todos los datos de los cuáles se requiere para la ejecución del algoritmo. Para ilustrar esta definición se planteará el siguiente ejemplo: *Cuando se desea retirar dinero del cajero electrónico, se ingresa la tarjeta de crédito a la ranura del cajero, seguidamente la pantalla del cajero solicita el número de la clave para comprobar que el número de clave corresponde a esa tarjeta de crédito y realizar las operaciones que se desean.*

El dato de entrada para esta operación fue el número de clave. Sin ella, la tarjeta de crédito no tendría validez alguna.

Para ser más generales se entiende como datos de entrada todas aquellas acciones que requieren del ingreso de datos, bien sea desde el teclado (como números de clave, valores, nombre, entre otros), como selecciones de menús de opciones (tipo de operación, cantidad de efectivo, entre otros.)

PROCESO.

Corresponde a las diferentes operaciones que se realizan para procesar datos y generar información requerida por el usuario. Continuando con el ejemplo del cajero electrónico, tenemos:

Al ingresarse el número de la clave, el cajero electrónico realiza la operación de validar la tarjeta, como se mencionó anteriormente. Este proceso de validación lo hace revisando el conjunto de datos o registros de usuarios y tratando de ubicar al cliente. Si encuentra al cliente en la lista de clientes, le permite retirar, revisar el saldo, entre otras operaciones. Pero si no lo encuentra le indica que la clave es incorrecta y debe ingresar otro número de clave.
De forma homóloga cuando se presionan teclas en un computador, este genera una cantidad de procesos para capturar la tecla presionada y determinar que tipo de operación se desea realizar.

DATOS DE SALIDA.

Los datos de salida son el objetivo por el cuál se genera un proceso. En otras palabras, es la función por la cuál se implementan aplicativos o sistemas de información (software). Los datos de salida que pueden visualizarse en la pantalla del computador en una página mediante la impresión, es realmente lo que le interesa el usuario del sistema. Continuando con el ejemplo:

Terminada la comprobación del número de clave, el usuario solicita un recibo para conocer su saldo. Selecciona la operación de impresión del recibo y espera la impresión del mismo.

Todos los ejemplos contenidos en este libro, utilizan esta herramienta para analizar y plantear un algoritmo.

Ejercicio.

Diseñar un Pseudocódigo en el cuál se muestre el siguiente mensaje "**Hola Mundo**."

Entrada: no existe ningún dato de entrada. El mensaje es conocido.
Proceso: no se necesita efectuar ningún calculo ni acción.
Salida: mostrar el mensaje **Hola Mundo**.

```
1    'Pseudocódigo mensaje
2
3    Inicio
4            Escribir "Hola Mundo"
5    Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(1). Es recomendado colocar un nombre a los algoritmos, en este caso debe ir siempre presidido de la comilla simple, seguido de la palabra **Pseudocódigo** y luego del nombre del mismo. Como este ejemplo muestra un mensaje, así se llamara el algoritmo.

(3). Todo Pseudocódigo contiene la palabra de **Inicio**, para indicar que se comienza un Pseudocódigo.

(4). La palabra **Escribir** permite mostrar mensajes, siempre y cuando se encuentren entre comillas dobles, como en este caso "**Hola Mundo**". Los mensajes a su vez pueden estar contenidos por paréntesis. Así como: **Escribir("Hola Mundo")**.

(5). Todo Pseudocódigo contiene la palabra **Fin**, para indicar que finaliza el Pseudocódigo.

Ejercicio.

Diseñar un Pseudocódigo en el cuál se lea un número introducido por el usuario, y mostrar el valor del número previamente dado.

Entrada:	un valor numérico dado por el usuario. Se almacenará en una variable llamada **Numero**.
Proceso:	no se necesita efectuar ningún cálculo ni acción.
Salida:	mostrar el valor de la variable **Numero** que contiene el valor dado por el usuario.

```
1       'Pseudocódigo suma
2
3       Inicio
4               Declaracion Numero
5               Numero = Leer("Digite un número")
6               Escribir "El número digitado es:" & Numero
7       Fin
8
```

EXPLICACIÓN POR INSTRUCCIÓN:

(4). La variable llamada **Numero** esta siendo declarada e inicializada.

(6). La palabra simbólica que representa la lectura de datos por teclado **Leer**, se compone de la siguiente estructura:

Nombre_Variable = **Leer**("**mensaje**")

Donde: **Nombre_Variable** corresponde a un nombre de variable hábil o correcto, en la cuál se almacenará el valor introducido desde teclado por el usuario. La palabra reservada **leer** indica que se realizará una lectura desde el teclado y se compone por paréntesis y dentro de los paréntesis un **mensaje** dentro de comillas dobles, que se desea aparezca en la pantalla del computador. Este **mensaje** debe indicar o representar al usuario que tipo de operación o dato se espera ingrese.

Por ejemplo:

Es correcto

A = **Leer(“**ingrese un número**”)**

Es erróneo

A = **Leer()**

Es erróneo

A, B, C = **Leer(“ingrese un valor para a,b,c”)**

Es correcto

A = **Leer(“**ingrese un valor para a**”)**
B = **Leer(“**ingrese un valor para b**”)**
C = **Leer(“**ingrese un valor para c**”)**

La lectura solo debe hacerse para una variable a la vez.

(7). El símbolo de salida **Escribir**, permite también mostrar el contenido de una o más variables. En este caso hay un mensaje entre comillas dobles, seguido del símbolo **&** (que permite concatenar o unir cadenas e incluso variables) y el nombre de la variable que contiene el valor que se desea mostrar.

4.3. OPERADORES

Los operadores son los componentes con los cuáles, las variables pueden realizar operaciones o cálculos. Con el siguiente ejemplo se demostrará la importancia de los operadores.

Observe: 12 18

Simplemente esa línea representa dos números 12 y 18. **ahora explique la siguiente línea:** 12 + 18

el operador + (mas) indica que se debe realizar una suma entre 12 y 18.

Explicación: para poderse interpretar esa expresión matemática, se requería de un operador que indicará que operación debía efectuarse. De igual forma, las computadoras necesitan conocer los componentes y el operador para efectuar la correspondiente expresión.

4.3.1. Operador de asignación (=)

Como su nombre lo indica asigna un valor a una variable.

por ejemplo: **Número1 = 12**

Cuando se expresa **Numero1 = 12** se dice que la variable llamada **Numero1** se le asigna el valor de **12** o toma el valor de **12**.

```
1       'Pseudocódigo asignación
2
3       Inicio
4               Declaracion Numero1
5               Numero1=12
6               Escribir Numero1
7       Fin
```

4.3.2. Operador de Asociación (())

La acción que cumplen los paréntesis es indicar que tipo de operación se debe realizar primero.

por ejemplo:

5 * 2 + 10 = 20.
5 * (2 + 10) = 60.

En el caso anterior, primero se realice la suma y luego el valor total de la suma se multiplica por 5. Cuando se conozcan todos los operadores se explicará las reglas de prioridad.

```
1       'Pseudocódigo asociación
2
3       Inicio
4               Declaracion Numero1
5               Declaración Numero2
6               Numero1=5*2+10
7               Numero2=5*(2+10)
8               Escribir Numero1
9               Escribir Numero2
10      Fin
```

4.3.3. Operadores matemáticos

En esta sección se conocerán los operadores matemáticos más empleados, su símbolo de operador, su funcionamiento y ejemplos explicativos para una mayor compresión y entendimiento.

4.3.3.1. Suma (+)

Este operador ejecuta la suma correspondiente entre dos términos o variables. Por ejemplo:

Con constantes

12 + 18 = 30.

Con variables

A = 8	la variable A se le asigna el valor de 8
B = 3	la variable B se le asigna el valor de 3
C = A + B	la variable C se le asigna el valor de la suma de A + B, que es 11.

```
1   'Pseudocódigo suma
2
3   Inicio
4       Declaracion A, B, C
5       A=8
6       B=3
7       C=A+B
8       Escribir C
9   Fin
```

4.3.3.2. Resta (-)

Este operador ejecuta la resta correspondiente entre dos términos o variables.

Por ejemplo:

Con constantes

18 - 12 = 6.

Con variables

A = 8 la variable A se le asigna el valor de 8
B = 3 la variable B se le asigna el valor de 3
C = A - B la variable C se le asigna el valor de la resta de A - B, que es 5.

```
1   'Pseudocódigo resta
2
3   Inicio
4       Declaracion A, B, C
5       A=8
6       B=3
7       C=A-B
8       Escribir C
9   Fin
```

4.3.3.3. Multiplicación (*)

Este operador ejecuta la multiplicación correspondiente entre dos términos o variables.

Por ejemplo:

Con constantes

12 * 5 = 60.

Con variables

A = 8 la variable A se le asigna el valor de 8
B = 3 la variable B se le asigna el valor de 3
C = A * B la variable C se le asigna el valor de la multiplicación de A * B, que es 24.

```
1    'Pseudocódigo multiplicación
2
3    Inicio
4            Declaracion A, B, C
5            A=8
6            B=3
7            C=A*B
8            Escribir C
9    Fin
```

4.3.3.4. División (/)

Este operador ejecuta la división correspondiente entre dos términos o variables.

Por ejemplo:

Con constantes

12 / 2 = 6.

Con variables

A = 27 la variable A se le asigna el valor de 27
B = 3 la variable B se le asigna el valor de 3
C = A / B la variable C se le asigna el valor de la división de A / B, que es 9.

```
1    'Pseudocódigo división
2
3    Inicio
4            Declaracion A, B, C
5            A=27
6            B=3
7            C=A/B
8            Escribir C
9    Fin
```

4.3.3.5. División entera (\)

Este operador ejecuta la división entera correspondiente entre dos términos o variables.

Por ejemplo:

Con constantes

20 \ 3 = 6.

Con variables

A = 27 la variable A se le asigna el valor de 27
B = 3 la variable B se le asigna el valor de 3
C = A \ B la variable C se le asigna el valor de la división de A \ B, que es 9.

Explicación:
Observe la siguiente división:

```
12 | 5
 20| 2.4
  0|
```

- Este es el valor que se obtiene al escribir 12 / 5 = 2.

```
1  'Pseudocódigo división entera
2
3  Inicio
4      Declaracion A, B, C
5      A=12
6      B=5
7      C=A \ B
8      Escribir C
9  Fin
```

4.3.3.6. Módulo (MOD)

Este operador ejecuta la división correspondiente entre dos términos o variables, obteniendo el residuo.

Por ejemplo:

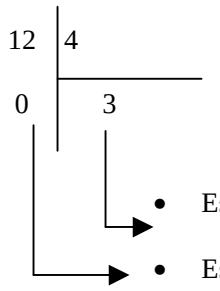
Con constantes

12 MOD 2 = 0.

Con variables

A = 8	la variable A se le asigna el valor de 8
B = 2	la variable B se le asigna el valor de 2
C = A MOD B	la variable C se le asigna el valor del Residuo entre A MOD B, que es 0.

Explicación:
Observe la siguiente división:



- Este es el valor que se obtiene al escribir 12 / 4 = 3.
- Este es el residuo de escribir 12 MOD 4 = 0

```
1    'Pseudocódigo modulo
2
3    Inicio
4        Declaracion A, B, C
5        A=12
6        B=4
7        C=A MOD B
8        Escribir C
9    Fin
```

4.3.3.7. Exponencial (^)

Este operador ejecuta el exponente correspondiente entre dos términos o variables.

Por ejemplo:

Con constantes
2 ^ 3 = 8.

Con variables

A = 8	la variable A se le asigna el valor de 8, la base.
B = 2	la variable B se le asigna el valor de 2, el exponente.
C = A ^ B	la variable C se le asigna el valor del exponente de A ^ B, que es 64.

```
1       'Pseudocódigo exponente
2
3       Inicio
4               Declaracion A, B, C
5               A=8
6               B=2
7               C=A ^ B
8               Escribir C
9       Fin
```

4.3.4. Operadores de cadenas de caracteres

4.3.4.1. Concatenar (&)

El operador **&** concatena o une dos o más cadenas de caracteres.

Por ejemplo:

Con constantes
"la programación " & "es fácil" = "la programación es fácil"
Con variables

A = "Carlos"	la variable A se le asigna la cadena "Carlos"
B = "Andrés"	la variable B se le asigna la cadena "Andrés"
C = A & B	la variable C se le asigna el valor de concatenar A & B, que es "CarlosAndres".

```
1    'Pseudocódigo concatenar
2
3    Inicio
4        Declaracion A, B, C
5        A=”Carlos”
6        B=”Andres”
7        C=A & B
8        Escribir C
9    Fin
```

4.3.5. Reglas de prioridad

Las expresiones matemáticas que contienen dos o más operadores, necesitan una serie de reglas que determinen el orden al ejecutarse sus operaciones, las cuáles son las siguientes:

- Las operaciones que se encuentran encerradas entre paréntesis se ejecutan primero.

  Por ejemplo:
  5 * (2 + 10) 12 + (8 + 3) +(5 - 3) = 25.

- El orden de prioridad de los operadores aritmético, obedece a la siguiente jerarquía que se muestra a continuación:

  Jerarquía de operadores aritméticos.

Operador	Nombre	Jerarquía al efectuar una operación
^	Exponencial	1
*	Multiplicación	2
/	División.	3
Div	División entera	4
Mod.	Residuo	5
+	Suma	6
-	Resta	7

  Lo anterior expresa lo siguiente, si en una expresión aritmética contiene un exponencial por ejemplo; por orden de jerarquía se ejecuta primeramente esa

operación, luego continuaría realizando las siguientes operaciones teniendo en cuenta su nivel de prioridad en la jerarquía.

Ejercicio.

El objetivo de estos ejercicios es retomar los conceptos de reglas de prioridad y representación de expresiones aritméticas.
Obtener el resultado de las siguientes expresiones, recuerde emplear la jerarquía de operadores.

- 3 + 8 * 4 ^ 3
- 6 / 3 * 2
- 2 + 8 + 3 – 9 / 25 + 4 – 3
- –56 + 7 – 2 + 7
- 78 – 6 ^ 3 + 47 / 2 ^ 3
- 45/ 23 * 2 – 7
- 89 – 41 + 12 / (2 * 14)
- 12 + 34 – 29 * 10 / 12
- 14 ^ 2 + (25 DIV 5) – 32 MOD 4
- 56 MOD 13 – 4 ^ 2

Ejercicio.

Diseñar un Pseudocódigo en el cuál se lea un número introducido por el teclado, y a ese valor dado se le suma el valor de 15. Por ejemplo, si el usuario digita el valor de 13, el resultado es ese valor (13) mas 15, obteniendo 28.

Entrada:	un valor numérico dado por el usuario. Se almacenará en una variable llamada **Numero**.
Proceso:	se toma el contenido de la variable **Numero** , a este se le suma con el valor constante **15** se efectúa la operación en una variable llamada **Resultado. Resultado = Numero + 15**
Salida:	mostrar el valor de la variable **Resultado** que contiene el valor de la operación.

```
1       ‘suma mas quince
2
3       Inicio
4               Declaracion Numero, Resultado
5
6               Numero = Leer("un numero")
7               Resultado = Numero+15
8               Escribir Resultado
9       Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(7). La variable **Resultado** contendrá el valor de la operación del valor de **Numero** mas el valor constante de 15.

(8). En esta instrucción se muestra el contenido de la variable **Resultado**.

Nota.

En este Pseudocódigo se emplearon 2 variables, **Numero** almacenaría el valor introducido por el teclado por parte del usuario, y **Resultado** el valor de la operación entre la constante numérica 15 y el valor dado por el usuario contenido en **Numero**. Como el ejercicio únicamente pide sumarle 15 a un valor dado, existe una variante del Pseudocódigo con una sola variable, pero obteniendo el mismo resultado.

Entrada: un valor numérico dado por el usuario. Se almacenará en una variable llamada **Numero**.
Proceso: se toma el contenido de la variable **Numero** , a este se le suma con el valor constante **15** se efectúa la operación en la misma variable **Numero** así: **Numero = Numero + 15**
Salida: mostrar el valor de la variable **Numero** que contiene el valor de la operación.

```
1    'suma mas quince resumido
2
3    Inicio
4            Declaracion Numero
5
6            Numero = Leer("un numero")
7            Numero = Numero+15
8            Escribir Numero
9    Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(6). La variable **Numero** almacenara el valor leído por el teclado.

(7). La operación se esta almacenando en la misma variable, supóngase que el valor dentro de **Numero** es 24, entiendo esta instrucción se tendría:

Numero = Numero + 15 [como **Numero** vale 24 entonces]
Numero = 24 + 15 [en la variable numero se almacena el valor que tiene la misma variable más el valor 15]

(8). Cuando se muestra el valor contenido en **Numero** con los valores por ejemplo del paso 2, se imprimiría en pantalla **39**.

Ejercicio.

Diseñar un Pseudocódigo que realice la suma correspondiente entre dos números enteros dados por el usuario y muestre el valor de la suma.

Entrada: los números introducidos por el usuario. En la variable **Num1** se almacenará el primer dato; en la variable **Num2** se almacenará el segundo dato dado por el usuario.

Proceso: Realizar la suma entre los valores almacenados en **Num1** y **Num2**. El resultado se acumulará en una variable llamada **Suma**; así: **Suma = Num1 + Num2**.

Salida: Enseñar el valor de la suma, contenido en la variable **Suma**.

```
1     'Pseudocódigo suma dos números.
2
3     Inicio
4
5           Declaracion Num1, Num2, Suma
6
7           Num1 = Leer("Ingrese el primer número")
8           Num2 = Leer ("Ingrese el segundo número")
9           Suma = 0
10          Suma = Num1 + Num2
11          Escribir "El valor de la suma es:" & Suma
12
13    Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(5). Se declaran las variables que son requeridas en el programa.

(7). En este paso se efectúa la lectura del primer valor introducido por teclado, almacenado en la variable **Num1**.

(8). Lectura del segundo dato, almacenado en la variable **Num2**.

(9). La variable **Suma** almacenará el valor de la suma entre los dos valores dados anteriormente. La expresión **Suma = 0** corresponde a la inicialización de la variable **Suma** en cero, como un valor neutro que no altere el valor de la suma. Es recomendable inicializar todas las variables, con el objetivo de eliminar basura u otros datos que no se necesiten dentro de ellas.

(10). Como el valor de la variable **Suma** almacenara otro resultado, se emplea el símbolo de proceso. En este la variable **Suma** acumulará el valor de **Num1 + Num2.**

Nota.

Cuando ejecute el programa en la herramienta que acompaña esta obra, quizás el resultado de la operación sea la unión de los dos valores, es decir, si se ingresa por ejemplo los siguientes números **2** y **5** , el resultado sea **25**. El anterior resultado es una suma de cadenas de caracteres. Para realizar la suma de forma numérica ingrese el anterior Pseudocódigo con la siguiente variante.

Entrada: los números introducidos por el usuario. En la variable **Num1** se almacenará el primer dato; en la variable **Num2** se almacenará el segundo dato dado por el usuario.

Proceso: Realizar la suma entre los valores almacenados en **Num1** y **Num2**. El resultado se acumulará en una variable llamada **Suma**; así: **Suma = Num1 + Num2**.

Salida: Enseñar el valor de la suma, contenido en la variable **Suma**.

```
1   'Pseudocódigo suma de dos números mejorada
2
3   Inicio
4
5       Declaracion Num1, Num2, Suma
6
7       Num1 = Leer("Ingrese el primer número")
8       Num2 = Leer ("Ingrese el segundo número")
9       Suma = 0
10      Suma = CEnteroCorto(Num1) + Num2
11      Escribir "El valor de la suma es:" & Suma
12
13  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(10). La función **CEnteroCorto** convierte una expresión de cadena de caracteres a un numero entero corto. Es decir, le indica al programa que efectué la operación como si se tratara de 2 números enteros cortos, y no como dos cadenas de caracteres. Si no se incluye este elemento el resultado de la operación continuara siendo la unión o concatenación de números.

Ejercicio.

Diseñar un Pseudocódigo que realice la suma y la multiplicación correspondiente entre dos números enteros; permita visualizar respectivamente el valor de la suma y la multiplicación.

Entrada: Los números introducidos por el usuario. En la variable **Num1** se almacenará el primer dato; en la variable **Num2** se almacenará el segundo dato dado por el usuario.

Proceso: Realizar la suma entre los valores almacenados en **Num1** y **Num2**. El resultado se acumulará en una variable llamada **Suma**; así: **Suma = Num1 + Num2**

La multiplicación se efectuará en una variable llamada **Producto**; así: **Producto = Num1 * Num2**

Salida: Enseñar el valor de la suma y la multiplicación referentemente contenido en las variables **Suma** y **Producto**.

```
1   Pseudocódigo suma y producto.
2
3   Inicio
4
5           Declaracion Num1, Num2, Suma, Producto
6
7           Num1 = Leer("Ingrese el primer número")
8           Num2 = Leer ("Ingrese el segundo número")
9
10          Suma = 0
11          Suma = CEnteroCorto(Num1) + Num2
12
13          Producto = 0
                        (3)
15          Producto = Num1 * Num2
16
17      Escribir "La suma:" & Suma & "El producto:" & Producto
18
19  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(5). Para la gran mayoría de Pseudocódigos cuando se necesita calcular valores u operaciones diferentes, se hace uso de más variables. Para este caso en una variable almacenamos la suma y en otra el producto.

(11). Como en el ejercicio anterior se usa la función **CEnteroCorto**

(13). La variable producto se inicializa en cero (0).

(15). El valor de la suma y la multiplicación se muestran de manera conjunta.

Ejercicio.

Diseñar un Pseudocódigo que realice la conversión de grados Celsius a grados Fahrenheit; dado por la siguiente fórmula **F = (9/5) C +32**

Entrada: El valor de los grado Celsius.

Proceso: Realizar la correspondiente fórmula, convirtiéndola en una expresión algorítmica; la cual se almacenará en la variable llamada **F**.

Salida: Mostrar el valor de la fórmula, contenido en la variable **F**.

```
1   'Pseudocódigo Fahrenheit.
2
3   Inicio
4
5       Declaracion C, F
6
7       C = Leer("Digite los grados Celsius")
8       F=(9/5)*C + 32
9       Escribir "Los grados Fahrenheit son:" & F
10  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(5). Las variables **C** y **F** son declaradas

(8). Representación de una expresión matemática en una algorítmica. Para recordar los conceptos conviene leer nuevamente “Reglas de Prioridad” apartado 4.3.5. y “Representación de Expresiones aritméticas” apartado 3.6.

Ejercicio.

Calcular el cuadrado y la raíz cuadrada de un número introducido por el teclado.

Entrada: El número leído desde el teclado . Se almacenará en una variable llamada **Numero**.

Proceso: Almacenar el cuadrado en la variable **Ncuadrado** y la raíz en la variable **Nraiz** del valor contenido en la variable **Numero** .

Salida: Mostrar el cuadrado el número.

```
1   ‘Pseudocódigo raíz cuadrada
2
3   Inicio
4
5       Declaración Nrariz, Numero, NCuadrado
6
7       Numero = Leer(“Digite el número”)
8       Ncuadrado = Numero^2
9       Nrariz = Raiz2(Numero)
10
11      Escribir “El cuadrado es:” & NCuadrado
12      Escribir “La raíz cuadrada es:” & Nraiz
13  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(8). El operador ^ (exponente) obtiene la potencia de un número indicándole el valor, es decir 3^2 es 3 a la 2 o 3^3 es 3 a la 3. La variable que recibe el valor puede ser de tipo entero largo, ya que el resultado obtenido puede ser grande.

(9). La función **Raiz2** obtiene la raíz cuadrada de un número. El valor obtenido debe almacenarse en una variable y en la mayoría de los casos la raíz cuadrada de un número contiene decimales.

Ejercicio.

Mostrar el equivalente en mayúsculas y minúsculas, de una cadena dada por el usuario.

Entrada: La cadena introducida por el teclado. Se almacenará en una variable llamada **Mensaje**.

Proceso: Almacenar en la variable **MenMay** la cadena es mayúsculas y en **MenMin** el mensaje en minúsculas.

Salida: Mostrar el contenido de las variables que alberga el mensaje en mayúsculas y en minúsculas.

```
1   'Pseudocódigo cadena en mayúsculas y minúsculas
2
3   Inicio
4
5       Declaracion Mensaje, MenMay, MenMin
6
7       Mensaje  = Leer("Digite el mensaje")
8       MenMay = Mayuscula(Mensaje)
9       MenMin = Minuscula(Mensaje)
10
11      Escribir "En Mayúsculas:" & MenMay
12      Escribir "En Minúsculas:" & MenMin
13  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(8). La función **Mayúscula** convierte una cadena en caracteres mayúsculos.

(9). La función **Minúscula** convierte una cadena en caracteres minúsculos.

Ejercicio.

Mostrar la cantidad de caracteres contenidos en un mensaje dado por el usuario.

Entrada: La cadena introducida por el teclado. Se almacenará en una variable llamada **Mensaje**.

Proceso: Almacenar en la variable **Tamano** la longitud o la cantidad de caracteres que contiene **Mensaje**.

Salida: Mostrar el contenido de la variable **Tamano**.

```
1     'Pseudocódigo tamaño de una cadena
2
3     Inicio
4
5           Declaracion Mensaje, Tamano
6           Mensaje = Leer ("Digite el mensaje")
7           Tamaño=longitud(Mensaje)
8           Escribir "El número de caracteres es:" & Tamaño
9
10    Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(7). La función **Longitud** obtiene el número de caracteres contenidos en una cadena dentro de una variable o una constante.

Ejercicio.

Calcular el área del círculo en función del radio. **S = π R²**

Entrada:	El valor del radio de la circunferencia (R). Se almacenará en una variable llamada **Radio**.
Proceso:	Crear una constante declara para el uso del valor de **π** (3.1416), llamada Pi. Desarrollar la fórmula y obtener el resultado en una variable llamado **AreaC**.
Salida:	Mostrar el contenido de la variable **AreaC**.

```
1   'Pseudocódigo área del circulo
2
3   Inicio
4       Constante Pi = 3.1416
5       Declaración Radio, AreaC
6
7       Radio = Leer("Ingrese el Radio")
8       AreaC = Pi * Radio^2
9       Escribir "El área del circulo es:" & AreaC
10
11  Fin
```

Explicación por instrucción:

(4). Declaración de una constante declarada en donde se asigna el valor de **3.1416** a la constante **Pi**.

(8). En la expresión **AreaC←Pi x Radio^2** se empleó el operador exponencial para elevar el radio al cuadrado. De forma similar pudo haber empleado la función **Cuadrado**, obtener el cuadrado del radio.

A lo largo de este capítulo se han establecido elementos que conforman un algoritmo como son: los tipos de datos, las variables, las constantes, los operadores, entre otros. El capítulo siguiente trata el tema de la representación de los algoritmos mediante el diagrama de flujo(DF) y el Pseudocódigo; conocidas entonces las formas de representación se inicia la parte práctica y dirigida con diferentes ejemplos que abarcan los diversos temas.

RESUMEN.

- Los algoritmos son los primeros bosquejos en la elaboración de un programa de computo.
- Los algoritmos pueden ser representados en: diagramas de flujo y Pseudocódigo.
- El DF es una representación gráfica de un algoritmo, que emplea una serie de figuras o símbolos, para indicar un proceso o instrucción sobre el cuál se ejecuta una acción.
- El Pseudocódigo es la presentación con palabras muy similares al idioma humano de un algoritmo, indicando procesos y acciones.
- Un algoritmo posee tres componentes que pueden ser usados en diferente orden, los cuales son: datos de entrada, proceso y datos de salida.
- Se entiende por datos de entrada, todos los datos de los cuáles se requiere para la ejecución del algoritmo.
- Corresponde a las diferentes operaciones que se realizan para procesar datos y generar información requerida por el usuario.
- Los datos de salida son el objetivo por el cuál se genera un proceso. En otras palabras, es la función por la cuál se implementan aplicativos o sistemas de información (software).
- Los operadores son las herramientas con las cuales se determina el proceso que se va a desarrollar en un algoritmo.
- Los operadores poseen reglas de prioridad que indican cual es su jerarquía de ejecución con respecto a otros, es decir que operador es más importante que otro dentro de una operación matemática.

EJERCICIOS DE AUTO EVALUACIÓN.

1. Los algoritmos se pueden representar en
2. El diagrama de flujo es
3. El Pseudocódigo es
4. Al asignar un valor a una variable se usa
5. Los operadores que pueden agrupar operaciones son
6. Si se quiere obtener la división entera se usa
7. Para obtener el modulo de una división se usa
8. Para elevar un termino a una potencia se usa
9. Para sumar dos o más cadenas de caracteres se usa

RESPUESTAS A LOS EJERCICIOS DE AUTO EVALUACIÓN.

1. Diagrama de flujo y Pseudocódigo
2. Una representación gráfica
3. Una representación escrita
4. El operador de asignación
5. Los paréntesis
6. El operador Div
7. El operador Mod
8. El operador exponencial
9. El operador de concatenar

EJERCICIOS PROPUESTOS.

Desarrolle los siguientes algoritmos

1. Leer un numero por teclado y sumarle el valor de 27.

2. Leer dos números por teclado y realizar las operaciones básicas (suma, resta, multiplicación y división).

3. Dado un numero por teclado calcular: el cuadrado, el cubo, la raíz cuadrada.

4. Dados 2 números por teclado que representan la base y el exponente, calcular la base elevada al exponente.

5. Calcular la hipotenusa dadas las longitudes de los catetos.

6. Dados el nombre de un automóvil y su modelo, mostrar por pantalla un mensaje que diga el modelo y el nombre del automóvil.

7. Desarrolle un programa que calcule la siguiente operación:

$$z = \sqrt{\frac{5a + b3}{c + 8}}$$

Donde a, b y c son leídos por teclado.

8. Calcular la siguiente formula:

$$area = \sqrt{p(p-a)(p-b)(p-c)}$$

donde p es igual a:

$$p = \frac{a + b + c}{2a}$$

9. Dados 3 números por teclado que representan las notas de un estudiante, calcular el promedio de las notas.

10. Dada una cantidad de artículos y el valor de un articulo, imprimir el valor de la compra. El valor final de compra es el impuesto de la compra 10% + el valor de la compra.

11. Escribir un programa que lea 2 valores y que los muestre por pantalla

12. Al problema anterior agréguele el título en pantalla "LECTURA"

13. Modifique el problema anterior para que obtenga la suma de los dos valores, cambie el título por "SUMA" el cual debe aparecer subrayado, y antes del resultado debe aparecer el mensaje "El valor de la suma es XXX" (XXX es el resultado de la operación)

14. Escriba un programa en donde por pantalla se pida que ingrese su nombre, y como salida tenga el siguiente mensaje "Su nombre es HHHHHH" (HHHHH es el nombre ingresado).

15. Modifique el programa anterior de manera que se solicite el nombre a dos personas y aparezca un cartel que diga "Buenos días XXXXX y YYYYY .... ¿Comenzamos a trabajar?

16. Escribir un programa al cual ingrese la velocidad de un móvil expresada en metros por segundo e imprima en pantalla la velocidad en kilómetros por hora.

17. Modifique el programa anterior de manera tal que por pantalla aparezca el siguiente cartel. "Los XXX m/s equivalen a YYY K/H" (Donde XXX es el valor ingresado e YYY es el resultado)

18. Un constructor sabe que necesita 0,5 metros cúbicos de arena por metro cuadrado de revoque a realizar. Hacer un programa donde ingrese las medidas de una pared (largo y alto) expresada en metros y obtenga la cantidad de arena necesaria para revocarla.

19. Desarrollar un programa que dado el largo y el ancho de un campo, permita determinar cuantos metros de alambre serán necesarios para colocar le al perímetro 5 hilos de alambrado. Y que cantidad de Soja se espera obtener, si el rendimiento de la misma es 145 quintales por hectárea.

20. Escriba un programa que pida el ingreso del valor de cada una de las raíces de una ecuación cuadrática. En función de ellos reconstruya la ecuación y la muestre por pantalla.

21. Escriba un programa donde se ingrese el tiempo necesario para un cierto proceso en horas, minutos y segundos. Se calcule el costo total del proceso sabiendo que el costo por segundo es 0,25$. (Debe salir por pantalla el tiempo expresado en horas, minutos y segundos, el costo por segundo y el costo total)

22. Una farmacia aplica al precio de los remedios el 10% de descuento. Hacer un programa que ingresado calcule el descuento y el precio final. Sacando por pantalla la siguiente imagen:

Precio de producto	XXX.XX
-	
Descuento	YY.YY

Valor a pagar	RRR.RR

23. La misma farmacia para la obra social OSZOPAPA, realiza el siguiente descuento: 70% por la obra social, y sobre ese resultado le aplica el 40% por cuenta de la propia farmacia (lo que ellos denominan el 70% + 40%). Cree un programa que calcule el precio final que pagará un afiliado a esa obra social por un remedio, y diseñe una salida equivalente a la del problema anterior.

24. Se necesita un programa que permita conocer el resultado del diseño de un tanque en forma de cilindro. Los datos que debe pedir el programa es el radio de la base y la altura. En función del mismo se calculará. Volumen que puede almacenar. Cantidad de chapa necesaria, cantidad que se debe pedir (ya que chapa circular no viene, viene en chapas rectangulares o cuadradas y el costo de la chapa es 2,25$ el metro cuadrado. Deberá salir por pantalla la siguiente información:

Radio	XXX m
Altura	YYY m
Volumen	ZZZ m cúbicos
Chapa base y techo	RRR * UUU m
Chapa lateral	LLL * JJJ m
Sup. Total de la chapa	SSS.SS m cuadrados
Costo	CCCC.CC $

25. Los propietarios de la pizzería "El Morón Binario" desean que se les haga un programa interactivo que solicite al usuario el diámetro de la pizza en centímetros y la cantidad de ingredientes extras que se quiere agregar. Como resultado de esto el programa deberá mostrar por pantalla el precio ce venta de la misma. Dicho precio se calcula de la siguiente manera.

    a) El precio de venta de la pizza se obtiene recargando un 150% en costo total
    b) El costo básico (pizza sin ingredientes extras) es de 0,016 $/$cm^2$
    c) El costo de cada ingrediente agregado a la pizza base es de 0,003 $/$cm^2$.

Se hace notar que como es un programa de tipo comercial la pantalla deberá tener el nombre de la pizzería en la parte superior de la pantalla y un saludo genérico para el cliente como "Buenos días señor" (puede reemplazarse por uno que sea personalizado, solicitándole el nombre al cliente y luego usándolo), y se le deberá solicitar cada dato "el usuario no es adivino" y mostrar el costo final.

5

Funciones establecidas, Funciones del sistema

Plan general.

OBJETIVOS

- Proporcionar una serie de elementos que permitan acelerar el proceso de desarrollo de algoritmos.
- Establecer funciones o subprogramas para el manejo de operaciones matemáticas, de cadena y de conversión de tipos de datos.
- Presentar una explicación clara y concisa del manejo de funciones, acompañado por un ejemplo funcional.
- Conocer las diferentes funciones especiales de un lenguaje de programación.
- Desarrollar algoritmos más complejos y de tratamiento especial.

5.1. INTRODUCCIÓN.

Los elementos establecidos son componentes de uso interno por un lenguaje de programación que realiza una operación especial. En otras palabras; si se observa con detenimiento los operadores aritméticos, cada uno de ellos realiza tareas específicas, por ejemplo: el operador + (mas) realiza la suma entre dos variables o constantes numéricas de cualquier tipo. El operador – (menos) efectúa la resta entre dos elementos. Partiendo de ese concepto, existen una serie de medios para hacer tareas especificas, como: la raíz cuadrada, el seno, el coseno, la secante, entre otros. Este tipo de medios llamados funciones simplifican las tareas que va a realizar un algoritmo. Una función es un algoritmo que realiza una operación específica, y puede ser invocado cualquier número de veces.

5.2. FUNCIONES.

5.2.1. Funciones matemáticas.

5.2.1.1. Función Abs.

Devuelve el valor absoluto de un numero o una expresión numérica.

Sintaxis. **Abs(**numero**)**

Ejemplos.

```
1   'Pseudocódigo absoluto 1
2
3   Inicio
4       Escribir Abs(32)
5       Escribir Abs(-23)
6   Fin
```

```
1   'Pseudocódigo absoluto 2
2
3   Inicio
4       Declaracion A
5       A=-32
6       A=Abs(A)
7       Escribir A
8   Fin
```

5.2.1.2. función Arctan.

Devuelve el arco tangente de un número o una expresión numérica. La arco tangente es la función inversa a la tangente.

Para convertir de grados a radianes se multiplica el valor por 3.1416/180. De forma similar para pasar de radianes a grados se multiplica el valor por 180/3.1416.

Sintaxis. **Arctan(**numero**)**

Ejemplos

```
1    ‘Pseudocódigo arco tangente 1
2
3    Inicio
4        Declaracion A
5        A=25
6        Escribir ArcTan(A)
7    Fin
```

```
1    ‘Pseudocódigo arco tangente 1
2
3    Inicio
4        Declaracion A
5        A=Leer(“ingrese un numero”)
6        Escribir ArcTan(A)
7    Fin
```

5.2.1.3. función Cos.

Devuelve el coseno de un número o una expresión numérica.

Para convertir de grados a radianes se multiplica el valor por 3.1416/180. De forma similar para pasar de radianes a grados se multiplica el valor por 180/3.1416.

Sintaxis. **Cos(**numero**)**

Ejemplos.

```
1    ‘Pseudocódigo coseno 1
2
3    Inicio
4            Escribir Cos(25)
5    Fin
```

```
1    ‘Pseudocódigo coseno 2
2
3    Inicio
4        Declaracion A
5        A=Leer(“ingrese un numero”)
         Escribir Cos(A)
7    Fin
```

5.2.1.4. función Exp.

Devuelve e elevado a una potencia. El número e tiene el valor de 2.71828182845905.

Sintaxis. **Exp(**potencia**)**

Ejemplos.

```
1    ‘Pseudocódigo exponente 1
2
3    Inicio
4            Escribir Exp(1)
5            Escribir Exp(1.5)
6            Escribir Exp(3)
7    Fin
```

```
1    ‘Pseudocódigo exponente 2
2
3    Inicio
4            Declaracion Numero, Resultado
5            Numero = Leer (“ingrese la potencia de E”)
6            Resultado = Exp(Numero)
7            Escribir Resultado
8    Fin
```

5.2.1.5. función Hex.

Convierte un número decimal (en base 10), a su equivalente en número hexadecimal (en base 16).

Sintaxis. **Hex(**numero decimal**)**

Ejemplos.

```
1   'Pseudocódigo hexadecimal 1
2
3   Inicio
4    Escribir Hex(10) ' 10 decimal es A en hexadecimal
5   Fin
```

```
1   'Pseudocódigo hexadecimal con ciclo
2
3   Inicio
4       Para i=1 Hasta 20
5           Escribir Hex(i)
6       FinPara
7   Fin
```

5.2.1.6. función Ln.

Convierte cualquier expresión numérica en el logaritmo natural de dicha expresión.

Sintaxis. **Ln(**expresión o valor numérico**)**

Ejemplos

```
1   'Pseudocódigo logaritmo 1
2
3   Inicio
4       Escribir Ln(20)
5   Fin
```

```
1   'Pseudocódigo logaritmo 2
2
3   Inicio
4       Declaracion Numero
5       Numero = Leer("ingrese un numero")
6       Escribir Ln(Numero)
7   Fin
```

5.2.1.7. función Oct.

Convierte cualquier numero o expresión numérica en base 10, en su equivalente en octal (base 8).

Sintaxis. **Oct(**número decimal**)**

Ejemplos.

```
1  'Pseudocódigo octal 1
2
3  Inicio
4   Escribir Oct(10) ' 10 decimal es 12 en octal
5  Fin
```

```
1  'Pseudocódigo octal con ciclos
2
3  Inicio
4     Para i=1 Hasta 20
5        Escribir Oct(i) 'convierte el valor de la variable i a octal
6     FinPara
7  Fin
```

5.2.1.8. función Raiz2.

Devuelve el valor absoluto de un numero o una expresión numérica.

Sintaxis. **Raiz2(**expresión o valor numérico**)**

Ejemplos.

```
1  'Pseudocódigo raíz cuadrada 1
2
3  Inicio
4       Declaracion resultado
5       resultado=Raiz2(25) ' a la variable resultado se le asigna 5
6       Escribir resultado
7  Fin
```

```
1    'Pseudocódigo raíz cuadrada de los numero del 1 al 10
2
3    Inicio
4       Para i=1 Hasta 10
5           Escribir Raiz2(i)
6       FinPara
7    Fin
```

5.2.1.9. función Redondeo.

Convierte a un número entero cualquier valor con fracciones decimales, al entero anterior o posterior según sea el valor de la fracción, es decir 5,41 es 5 y 5,51 es 6.

El separador de números decimales es la coma (,)

Sintaxis. **Redondeo (**numero**)**

Ejemplos.

```
1    'Pseudocódigo redondeo 1
2
3    Inicio
4            Escribir Redondeo(5,41) ' escribe 5
5            Escribir Redondeo(5,61) ' escribe 6
6    Fin
```

```
1    'Pseudocódigo redondeo 2
2
3    Inicio
4            Declaracion Numero
5            Escribir "Nota: El separador de decimal es la coma ,"
6            Numero = Leer("ingrese un numero decimal")
7            Escribir Redondeo(Numero)
8    Fin
```

5.2.1.10. función Sen.

Devuelve el seno de un número o una expresión numérica.
Para convertir de grados a radianes se multiplica el valor por 3.1416/180. De forma similar para pasar de radianes a grados se multiplica el valor por 180/3.1416.

Sintaxis. **Sen(**numero**)**

Ejemplos

```
1    'Pseudocódigo Seno 1
2
3    Inicio
4        Declaracion A
5        A=25
6        Escribir Sen(A)
7    Fin
```

```
1    'Pseudocódigo Seno 2
2
3    Inicio
4        Declaracion A
5        A=Leer("ingrese un numero")
         Escribir Sen(A)
7    Fin
```

5.2.1.11. función Signo.

Obtiene un entero que representa el signo del valor o expresión numérica. Los posibles valores son:

1 número positivo.
-1 número negativo.
0 número igual a cero.

Sintaxis. **Signo (**valor o expresión numérica**)**

Ejemplos.

```
1   ‘Pseudocódigo signo de un numero 1
2
3   Inicio
4           Escribir Signo(-25) ' retorna -1
5           Escribir Signo(7)   ' retorna 1
6           Escribir Signo(0)   ' retorna 0
7           Escribir Signo()    ' se origina un error
8   Fin
```

```
1   ‘Pseudocódigo signo de un numero 2
2
3   Inicio
4           Declaracion Numero
5
6           Numero = Leer(“ingrese un numero”)
7           Escribir Signo(Numero)
8   Fin
```

5.2.1.12. función Tan.

Devuelve la tangente de un número o una expresión numérica.

Para convertir de grados a radianes se multiplica el valor por 3.1416/180. De forma similar para pasar de radianes a grados se multiplica el valor por 180/3.1416.

Sintaxis. **Tan(**número o una expresión numérica**)**

Ejemplos.

```
1   ‘Pseudocódigo tangente 1
2
3   Inicio
4           Declaracion A
5           A=25
6           Escribir Tan(A)
7   Fin
```

```
1       ‘Pseudocódigo tangente 2
2
3       Inicio
4           Declaracion A
5           A=Leer(“ingrese un numero”)
            Escribir Tan(A)
7       Fin
```

5.2.2. Funciones de cadena.

5.2.2.1. función Ascii.

La función **Ascii** obtiene el número ASCII del primer carácter sobre una expresión.

Sintaxis. **Ascii (**carácter**)**

Ejemplos.

```
1       ‘Pseudocódigo numero ASCII 1
2
3       'Este ejemplo imprime el número ASCII de la letra A,
4       'el cual es 65.
5
6       Inicio
7               Escribir Ascii("A")
8       Fin
```

```
1       ‘Pseudocódigo numero ASCII 2
2
3       Inicio
4               Declaracion Carácter
5               Carácter =  Leer(“ingrese un carácter”)
6               Escribir Ascii(Carácter)
7       Fin
```

5.2.2.2. función Car.

La función **Car** obtiene el carácter ASCII asociado a un número entero que es pasado como argumento.

Sintaxis. **Car(**numero**)**

Ejemplos.

```
1   ‘Pseudocódigo
2
3   'Este ejemplo imprime el carácter A, ya que su
4   ‘Número Ascii es el 65.
5
6   Inicio
7    Escribir Car(65)
8   Fin
```

```
1   ‘Pseudocódigo
2
3   ‘Este ejemplo imprime los caracteres entre 65 y 80
4
5   Inicio
6           Declaracion i
7                   Para i=65 Hasta 80
8                           Escribir Car(i)
9                   FinPara
10  Fin
```

5.2.2.3. función CadenaDer.

La función **CadenaDer** obtiene un número determinado de caracteres a la derecha de una cadena o expresión de cadena.

Sintaxis. **CadenaDer (**cadena de caracteres**)**

Ejemplos.

```
1    ‘Pseudocódigo cadena de la derecha 1
2
3    Inicio
4        Escribir CadenaDer("Fundamentos de Programación", 5)
5        Escribir CadenaDer("Fundamentos de Programación", 12)
6    Fin
```

```
1    ‘Pseudocódigo cadena de la derecha 2
2
3    Inicio
4        Declaracion mensaje
5
6        Mensaje = Leer(“ingrese un mensaje”)
7        Escribir CadenaDer(Mensaje, 3)
8
     Fin
```

5.2.2.4. función CadenaIzq

La función **CadenaIzq** obtiene un número determinado de caracteres a la izquierda de una cadena o expresión de cadena.

Sintaxis. **CadenaIzq (**cadena de caracteres**)**

Ejemplos.

```
1    ‘Pseudocódigo cadena de la izquierda 1
2
3    Inicio
4        Escribir CadenaIzq("Fundamentos de Programación", 5)
5        Escribir CadenaIzq("Fundamentos de Programación", 12)
6    Fin
```

```
1       ‘Pseudocódigo cadena de la izquierda 2
2
3       Inicio
4           Declaracion mensaje
5
6           Mensaje = Leer(“ingrese un mensaje”)
7           Escribir CadenaIzq(Mensaje, 3)
8
        Fin
```

5.2.2.5. función CompararCadena.

Determina si una cadena es igual o no a otra. Los posibles valores obtenidos son:

-1 sí cadena1 menor que cadena2
0 sí cadena1 igual que cadena2
1 si cadena1 mayor que cadena2

Sintaxis. **CompararCadena (cadena1, cadena2)**

Ejemplos.

```
1       ‘Pseudocódigo compararCadena 1
2
3       Inicio
4       Escribir CompararCadena("Hola", "hola mundo")‘Devuelve-1
5       Escribir CompararCadena("Hola", "Hola") ‘Devuelve 0
6       Fin
7
```

```
1       ‘Pseudocódigo compararCadena 2
2
3       Inicio
4       Declaracion Cadena1, Cadena2
5       Cadena1 = Leer(“ingrese un mensaje”)
6       Cadena2 = Leer(“ingrese una palabra del mensaje anterior”)
7       Escribir CompararCadena(Cadena1, Cadena2)
8       Fin
```

5.2.2.6. función EnCadena.

Obtiene el valor de la posición en la cual se encuentra una cadena con respecto a otra.

Sintaxis. **EnCadena** ([posición], cadena en que se busca, cadena buscada**)**

[Posición] es opcional. Se omite por defecto inicia en el primer carácter.

Ejemplos.

```
1   'Pseudocódigo encadena palabra 1
2
3   'Este ejemplo imprime el número de caracteres en donde se
4   'inicia la cadena buscada sobre la cadena en la cual se busca.
5
6   Inicio
7       Declaracion posicion
8       posicion=EnCadena("hola mundo","mundo")
9       Escribir posicion
10  Fin
```

```
1   'Pseudocódigo encadena palabra 2
2
3   Inicio
4   Declaracion Cadena1, Cadena2
5   Cadena1 = Leer("ingrese un mensaje")
6   Cadena2 = Leer("ingrese una palabra del mensaje anterior")
7   Escribir EnCadena (Cadena1, Cadena2)
8   Fin
```

5.2.2.7. función Invertir.

Invierte una cadena, es decir se obtiene una cadena al revés.

Sintaxis. **Invertir** (cadena o expresión de cadena)

Ejemplos.

```
1    ‘Pseudocódigo invertir cadena 1
2
3    'En este ejemplo se visualiza una cadena al revés.
4
5    Inicio
6        Declaracion cadena
7        cadena="Fundamentos de programación"
8        Escribir Invertir(cadena)
9    Fin
```

```
1    ‘Pseudocódigo invertir cadena 2
2
3    'En este ejemplo se visualiza un mensaje al revés.
4
5    Inicio
6        Declaracion cadena
7        cadena = Leer(“ingrese un mensaje”)
8        Escribir Invertir(cadena)
9    Fin
```

5.2.2.8. función Mayuscula

Convierte una cadena o expresión de cadena a mayúsculas.

Sintaxis. **Mayuscula** (cadena o expresión de cadena**)**

Ejemplos.

```
1    ‘Pseudocódigo mayúscula 1
2
3    Inicio
4        Declaracion texto
5        texto=”hola mundo”
6        texto=Mayuscula(texto)
7        Escribir texto
8    Fin
9
```

```
1     ‘Pseudocódigo mayúscula 2
2     ‘Este ejemplo lee un mensaje y se almacena en una variable
3     ‘llamada texto.
4
5     Inicio
6           Declaracion texto
7           texto=Leer("ingrese un mensaje")
8           texto=Mayuscula(texto)
9           Escribir texto
10    Fin
11
```

5.2.2.9. función Minuscula

Convierte una cadena o expresión de cadena a minúsculas.

Sintaxis. **Minuscula** (cadena o expresión de cadena**)**

Ejemplos.

```
1     ‘Pseudocódigo minúscula 1
2
3     ‘Este ejemplo lee un mensaje y se almacena en una variable
4     ‘llamada texto. A este variable se almacena nuevamente el
5     ‘mensaje pero en minúsculas.
6
7     Inicio
8           Declaracion texto
9           texto=Leer("ingrese un mensaje")
10          texto=Minuscula(texto)
11          Escribir texto
12    Fin
```

```
1    ‘Pseudocódigo minúscula 2
2
3    Inicio
4        Declaracion texto
5        texto=”HOLA MUNDO”
6        texto=Minuscula(texto)
7        Escribir texto
8    Fin
```

5.2.2.10. función Longitud.

Obtiene la cantidad de caracteres que contiene una cadena o expresión de cadena.

Sintaxis. **Longitud** (cadena o expresión de cadena)

Ejemplos.

```
1    ‘Pseudocódigo longitud 1
2
3    Inicio
4        Declaracion texto
5        texto=”hola”
6        Escribir Longitud(texto)
7    Fin
```

```
1    ‘Pseudocódigo longitud 2
2
3    ‘Este ejemplo muestra por pantalla el número de caracteres
4    ‘que contiene el mensaje que escriba el usuario.
5
6    Inicio
7        Declaracion texto
8        texto=Leer("ingrese un mensaje")
9        Escribir Longitud(texto)
10   Fin
```

5.2.2.11. función Reemplazar.

Obtiene una cadena en la cual se busca una cadena o carácter y se reemplaza por otro.

Sintaxis. **Reemplazar** (cadena, cadena buscada, cadena reemplazo)

Ejemplos.

```
1   'Pseudocódigo reemplazar 1 estático
2
3   Inicio
4       Declaracion texto, buscar, reemplazo
5
6       texto="hola mundo"
7       buscar="mundo"
8       reemplazo=" a todos"
9       texto=Reemplazar(texto, buscar, reemplazo)
10      Escribir texto
11  Fin
```

```
1   'Pseudocódigo reemplazar 2 dinámico
2
3   Inicio
4       Declaracion texto, buscar, reemplazo
5
6       texto=Leer("ingrese un mensaje")
7       buscar=Leer("ingrese la cadena  a buscar")
8       reemplazo=Leer("ingrese la cadena que reemplaza")
9       texto=Reemplazar(texto, buscar, reemplazo)
10      Escribir texto
11  Fin
```

5.2.2.12. función SinEspacioDer.

Obtiene una nueva cadena sin espacios en blanco a la derecha de la cadena. El uso de esta función elimina de la cadena o de la expresión de cadena, los espacios que son innecesarios de la expresión afectando por ejemplo el tamaño de la misma.

Sintaxis. **SinEspacioDer** (cadena o expresión de cadena)

Ejemplo.

```
1     ‘Pseudocódigo sin espacios a la derecha
2
3     Inicio
4           Declaracion cadena
5
6           cadena="hola mundo    "
7           Escribir Longitud(cadena) ' se imprime 14 caracteres
8           cadena=SinEspacioDer(cadena)
9           Escribir Longitud(cadena) ' se imprime 10 caracteres
10    Fin
```

5.2.2.13. función SinEspacioIzq.

Obtiene una nueva cadena sin espacios en blanco a la izquierda de la cadena. El uso de esta función elimina de la cadena o de la expresión de cadena, los espacios que son innecesarios de la expresión afectando por ejemplo el tamaño de la misma.

Sintaxis. **SinEspacioIzq** (cadena o expresión de cadena)

Ejemplo.

```
1     ‘Pseudocódigo sin espacios a la izquierda
2
3     Inicio
4           Declaracion cadena
5
6           cadena="     hola mundo"
7           Escribir Longitud(cadena) ' se imprime 15 caracteres
8           cadena=SinEspacioIzq(cadena)
9           Escribir Longitud(cadena) ' se imprime 10 caracteres
10    Fin
```

5.2.2.14. función SinEspacioLados

Obtiene una nueva cadena sin espacios en blanco a la derecha y a la izquierda de la cadena. El uso de esta función elimina de la cadena o de la expresión de cadena, los espacios que son innecesarios de la expresión afectando por ejemplo el tamaño de la misma.

Sintaxis. **SinEspacioLados** (cadena o expresión de cadena)

Ejemplo.

```
1     ‘Pseudocódigo sin espacio a los lados
2
3     Inicio
4           Declaracion cadena
5
6           cadena="     hola mundo     "
7           Escribir Longitud(cadena) ' se imprime 20 caracteres
8           cadena=SinEspacioLados(cadena)
9           Escribir Longitud(cadena) ' se imprime 10 caracteres
10    Fin
```

5.2.2.15. función SubCadena.

Obtiene uno o varios elementos de una cadena generando otra nueva cadena.

Sintaxis. **SubCadena**(cadena, posición, longitud o tamaño)

Ejemplos.

```
1     ‘Pseudocódigo subcadena 1
2
3     Inicio
4           Declaracion texto, i
5           texto=”hola mundo”
6           Escribir SubCadena(texto,2,6)
7     Fin
```

```
1       ‘Pseudocódigo subcadena 2
2
3       Inicio
4               Declaracion texto, i
5               texto=Leer("ingrese cadena")
6               Para i=1 Hasta Longitud(texto)
7                       Escribir SubCadena(texto,i,1)
8               FinPara
9       Fin
```

5.2.3. Funciones de Fecha y hora.

5.2.3.1. función Anno.

La función obtiene el año a partir de una fecha que se entregue como argumento. La fecha debe ser una expresión valida de fecha.

Sintaxis. **Anno**(fecha)

Ejemplos.

```
1       ‘Pseudocódigo año 1
2
3       ‘Para este ejemplo se obtiene el año 1978. El argumento para
4       ‘este ejemplo es una cadena de caracteres que forman una
5       ‘fecha en el formato día, mes, año.
6
7       Inicio
8               Escribir Anno("29/09/1972")
9       Fin
```

```
1   ‘Pseudocódigo año 2
2
3   ‘Para este ejemplo se emplea la función Fecha y se imprime el
4   ‘año de la fecha actual.
5
6   Inicio
7    Escribir Anno(Fecha)
8   Fin
```

5.2.3.2. función Fecha

Obtiene la fecha actual del sistema, en el formato que posee por defecto el equipo.

Sintaxis. **Fecha**()

Ejemplos.

```
1   ‘Pseudocódigo fecha 1
2
3   ‘Este ejemplo muestra por pantalla la fecha actual.
4
5   Inicio
6           Escribir Fecha()
7   Fin
```

```
1   ‘Pseudocódigo fecha 2
2
3   ‘Este ejemplo hace lo mismo, pero asignando la fecha a una
4   ‘variable.
5
6   Inicio
7           Declaracion dato
8           dato = Fecha()
9           Escribir  dato
10  Fin
```

5.2.3.3. función FechaIntervalo

Obtiene una fecha posterior o anterior a partir de una fecha, dados un intervalo y una cantidad de tiempo.

Sintaxis. **FechaIntervalo** (“intervalo”, cantidad de tiempo, fecha)

Donde **intervalo** puede tener los siguientes valores:

Valor	**Descripción**
yyyy	Año
q	Trimestre
m	Mes
y	Día del año
d	Día
w	Día de la semana
ww	Semana
h	Hora
n	Minuto
s	Segundo

Ejemplos.

```
1   ‘Pseudocódigo intervalo de fecha 1
2
3   ‘Este ejemplo imprime la fecha dentro de cinco meses de la
4   ‘fecha actual.
5
6   Inicio
7       Escribir FechaIntervalo("m",5,Fecha())
8   Fin
```

```
1   ‘Pseudocódigo intervalo fecha 2
2
3   Inicio
4       Escribir FechaIntervalo("m",5,"06/11/2002")
5   Fin
```

5.2.3.4. función FechaDiferencia

Obtiene la cantidad de intervalos de tiempo entre dos fechas.
Sintaxis. **FechaDiferencia** (“intervalo”, fecha1, fecha2)

Donde **intervalo** puede tener los siguientes valores:

Valor	Descripción
yyyy	Año
q	Trimestre
m	Mes
y	Día del año
d	Día
w	Día de la semana
ww	Semana
h	Hora
n	Minuto
s	Segundo

Ejemplo.

```
1    ‘Pseudocódigo fecha diferencia
2
3    ‘Este ejemplo imprime el número de días transcurridos entre
4    ‘el primero de enero del 2001 hasta el primero de enero del
5    ‘2002.
6
7    Inicio
8        Escribir FechaDiferencia("d","01/01/2001","01/01/2002")
9    Fin
```

5.2.3.5. función FechaParte

Obtiene la parte especificada de una fecha dada.

Sintaxis. **FechaParte** (“intervalo”, fecha)

Donde **intervalo** puede tener los siguientes valores:

Valor	Descripción
yyyy	Año
q	Trimestre
m	Mes
y	Día del año
d	Día
w	Día de la semana
ww	Semana
h	Hora
n	Minuto
s	Segundo

Ejemplos.

```
1   'Pseudocódigo fecha parte 1
2
3   'Este ejemplo imprime el mes al que corresponde la fecha
4   'dada.
5
6   Inicio
7       Escribir FechaParte("m","14/10/2002")
8   Fin
```

```
1   'Pseudocódigo fecha parte 2
2
3   'Este ejemplo imprime el trimestre al que corresponde la
4   'fecha.
5
6   Inicio
7       Escribir FechaParte("q","14/10/2002")
8   Fin
```

5.2.3.6. función Dia

Obtiene el día del mes que corresponde a la fecha que se le determina. Los valores posibles se encuentran entre 1 y 31.

Sintaxis. **Dia**(fecha)

Ejemplo.

```
1     ‘Pseudocódigo día
2
3     ‘Este ejemplo solicita una fecha al usuario e imprime el día a
4     ‘partir de la fecha.
5
6     Inicio
7           Declaracion FechaUsuario
8           FechaUsuario=Leer("ingrese una Fecha")
9           Escribir Dia(FechaUsuario)
10    Fin
```

5.2.3.7. función Hora.

Obtiene la hora actual del sistema. La hora resultante esta en el formato que tiene el sistema, por lo general es: Hora : minuto: segundo

Sintaxis. **Hora**

Ejemplo.

```
1     ‘Pseudocódigo hora del sistema
2
3     ‘Este ejemplo solicita una fecha al usuario e imprime el día a
4     ‘partir de la fecha.
5
6     Inicio
7           Declaracion HoraUsuario
8           HoraUsuario=Hora
9           Escribir HoraUsuario
10    Fin
```

5.2.3.8. función Mes

Obtiene el número del mes a partir de una fecha dada.

Sintaxis. **Mes**(Fecha)

Ejemplo.

```
1    ‘Pseudocódigo mes
2
3    ‘Este sencillos ejemplo imprime el mes actual a partir de la
4    ‘fecha del sistema.
5
6    Inicio
7        Escribir Mes (Fecha)
8    Fin
```

5.2.3.9. función Minuto

Obtiene un numero entero que representa el minuto actual sobre una hora dada.

Sintaxis. **Minuto**(parámetro de hora)

Ejemplos.

```
1    ‘Pseudocódigo minutos 1
2
3    ‘Hora sobre una cadena constante
4
5    Inicio
6        Escribir Minuto("03:25:58")
7    Fin
```

```
1    ‘Pseudocódigo minutos 2
2
3    ‘Hora sobre la función Hora
4
5    Inicio
6        Escribir Minuto(Hora())
7    Fin
```

5.2.4. Funciones de conversión de tipos.

5.2.4.1. función CBooleano

Devuelve una expresión de tipo booleano.

Sintaxis. **CBooleano** ()

Ejemplo.

```
1     'Pseudocodigo convertir a booleano
2
3     Inicio
4
5           Declaracion A, B, Prueba
6
7           A = 5
8           B = 5                    ' Inicializa variables.
9           Prueba = CBool(A = B)   ' Prueba contiene True.
10          Escribir prueba
11          A = 0   ' Define variable.
12          Prueba = CBool(A)        ' Prueba contiene False
13          Escribir prueba
14    Fin
```

5.2.4.2. función CFecha

Devuelve una expresión de tipo fecha / hora.

Sintaxis. **CFecha**()

Ejemplo.

```
1     Pseudocodigo convertir a Fecha
2
3     Inicio
4           Declaracion MiFec, MiHo
5           Declaracion MiHora, MiHoraCorta
6
7           MiFec = "12 febrero 1969"
8           MiFechaCorta = CFecha(MiFec)
9           Escribir MiFechaCorta
10          MiHo = "4:35:47 PM"
11          MiHoraCorta = CFecha (MiHo)
12          Escribir MiHoraCorta
13    Fin
```

5.2.4.3. función CEnteroCorto

Devuelve una expresión de tipo entero corto.

Sintaxis. **CEnteroCorto** ()

Ejemplo.

```
1     'Pseudocódigo suma con CEnteroCorto
2
3     Inicio
4
5           Declaracion Num1, Num2, Suma
6
7           Num1 = Leer("Ingrese el primer número")
8           Num2 = Leer ("Ingrese el segundo número")
9           Suma = 0
10          Suma = CEnteroCorto(Num1) + Num2
11          Escribir "El valor de la suma es:" & Suma
12
13    Fin
```

5.2.4.4. función CEnteroLargo

Devuelve una expresión de tipo entero largo.

Sintaxis. **CEnteroLargo** ()

Ejemplo.

```
1   'Pseudocódigo suma con CEnteroLargo
2
3   Inicio
4
5           Declaracion Num1, Num2, Suma
6           Num1 = Leer("Ingrese el primer número")
7           Num2 = Leer ("Ingrese el segundo número")
8           Suma = 0
9           Suma = CEnteroLargo (Num1) + Num2
10          Escribir "El valor de la suma es:" & Suma
11  Fin
```

5.2.4.5. función CRealCorto

Devuelve una expresión de tipo real corto.

Sintaxis. **CRealCorto** ()

Ejemplo.

```
1   'Pseudocódigo suma con CRealCorto
2
3   Inicio
4
5           Declaracion Num1, Num2, Suma
6           Num1 = Leer("Ingrese el primer número")
7           Num2 = Leer ("Ingrese el segundo número")
8           Suma = 0
9           Suma = CRealCorto(Num1) + Num2
10          Escribir "El valor de la suma es:" & Suma
11  Fin
```

5.2.4.6. función CRealLargo

Devuelve una expresión de tipo real largo.

Sintaxis. **CEnteroLargo** ()

Ejemplo.

```
1     'Pseudocódigo suma con CRealLargo
2
3     Inicio
4
5           Declaracion Num1, Num2, Suma
6           Num1 = Leer("Ingrese el primer número")
7           Num2 = Leer ("Ingrese el segundo número")
8           Suma = 0
9           Suma = CRealLargo (Num1) + Num2
10          Escribir "El valor de la suma es:" & Suma
11    Fin
```

5.2.4.6. función CCadena

Devuelve una expresión de tipo cadena.

Sintaxis. **CCadena** ()

Ejemplo.

```
1     'Pseudocódigo conversión a cadena
2
3     Inicio
4           Declaracion MiDoble, MiCadena
5
6           MiDoble = 437.324
7           MiCadena = CCadena(MiDoble)
8     Fin
```

RESUMEN.

- Función Abs. Devuelve el valor absoluto de un numero o una expresión numérica.
- Función Arctan. Devuelve el arco tangente de un número o una expresión numérica. La arco tangente es la función inversa a la tangente.
- Función Cos. Devuelve el coseno de un número o una expresión numérica.
- Función Exp. Devuelve e elevado a una potencia. El número e tiene el valor de 2.71828182845905.
- Función Hex. Convierte un número decimal (en base 10), a su equivalente en número hexadecimal (en base 16)
- Función Ln. Convierte cualquier expresión numérica en el logaritmo natural de dicha expresión.
- Función Oct. Convierte cualquier numero o expresión numérica en base 10, en su equivalente en octal (base 8).
- Función Raiz2. Devuelve el valor absoluto de un numero o una expresión numérica.
- Función Redondeo.Convierte a un número entero cualquier valor con fracciones decimales, al entero anterior o posterior según sea el valor de la fracción, es decir 5,41 es 5 y 5,51 es 6.
- Función Sen. Devuelve el seno de un número o una expresión numérica.
- Función Signo. Obtiene un entero que representa el signo del valor o expresión numérica.
- Función Tan. Devuelve la tangente de un número o una expresión numérica.
- Función Ascii. Obtiene el número ASCII del primer carácter sobre una expresión.
- La función Car obtiene el carácter ASCII asociado a un número entero que es pasado como argumento.

- La función CadenaDer obtiene un número determinado de caracteres a la derecha de una cadena o expresión de cadena.

- La función CadenaIzq obtiene un número determinado de caracteres a la izquierda de una cadena o expresión de cadena.

- Función CompararCadena. Determina si una cadena es igual o no a otra.

- La función EnCadena. Obtiene el valor de la posición en la cual se encuentra una cadena con respecto a otra.

- Función Invertir. Invierte una cadena, es decir se obtiene una cadena al revés.

- Función Mayúscula. Convierte una cadena o expresión de cadena a mayúsculas.

- Función Minúscula. Convierte una cadena o expresión de cadena a minúsculas.

- Función Longitud. Obtiene la cantidad de caracteres que contiene una cadena o expresión de cadena.

- Función Reemplazar. Obtiene una cadena en la cual se busca una cadena o carácter y se reemplaza por otro.

- Función SinEspacioDer. Obtiene una nueva cadena sin espacios en blanco a la derecha de la cadena.

- Función SinEspacioIzq. Obtiene una nueva cadena sin espacios en blanco a la izquierda de la cadena.

- Función SinEspacioLados. Obtiene una nueva cadena sin espacios en blanco a la derecha y a la izquierda de la cadena.

- Función SubCadena. Obtiene uno o varios elementos de una cadena generando otra nueva cadena.

- Función Anno. La función obtiene el año a partir de una fecha que se entregue como argumento.

- Función Fecha. Obtiene la fecha actual del sistema, en el formato que posee por defecto el equipo.

- Función FechaIntervalo. Obtiene una fecha posterior o anterior a partir de una fecha, dados un intervalo y una cantidad de tiempo.

- Función FechaDiferencia. Obtiene la cantidad de intervalos de tiempo entre dos fechas.

- Función FechaParte. Obtiene la parte especificada de una fecha dada.

- Función Dia. Obtiene el día del mes que corresponde a la fecha que se le determina. Los valores posibles se encuentran entre 1 y 31.

- Función Hora. Obtiene la hora actual del sistema.

- Función Mes. Obtiene el número del mes a partir de una fecha dada.

- Función Minuto. Obtiene un numero entero que representa el minuto actual sobre una hora dada.

- Función Cbooleano. Devuelve una expresión de tipo booleano.

- Función Cfecha. Devuelve una expresión de tipo fecha / hora.

- Función CEnteroCorto. Devuelve una expresión de tipo entero corto.

- Función CEnteroLargo. Devuelve una expresión de tipo entero largo.

- Función CRealLargo. Devuelve una expresión de tipo real largo.

- Función Cadena. Devuelve una expresión de tipo cadena.

EJERCICIOS DE AUTO EVALUACIÓN.

Desarrolle la gran mayoría de Pseudocódigos anteriores de funciones del sistema para que verifique su correcto funcionamiento y entendimiento.

EJERCICIOS PROPUESTOS.

Desarrolle los siguientes algoritmos

1. Leer un numero por teclado, calcular su valor absoluto y calcular el seno, coseno, tangente y arco tangente.

2. Dados 3 dígitos por teclado, calcule el valor del numero E elevado a esos tres dígitos, usando la función Exp.

3. Dado un numero entero convertirlo en un numero octal y hexadecimal.

4. Dado un carácter por teclado, imprimir su carácter ASCII y luego generar el carácter sumándole 10 e imprimir el carácter generado.

5. Dado un mensaje por el teclado, imprimir el mensaje en mayúsculas, minúsculas e invertido.

6. Dados 2 nombres de personas imprimir él numero de caracteres de cada nombre.

7. Dados 2 fechas por teclado, calcule la diferencia en días, meses y años.

8. Desarrolle un Pseudocódigo que imprima la fecha, la hora, los minutos, segundo y el día del sistema actualmente.

9. Dado un numero entero, calcular la raíz cuadrada, el cuadrado y el cubo.

10. Dado el nombre de una persona, muestre por pantalla: el nombre invertido, la cadena de derecha, la cadena de la izquierda, el nombre en mayúsculas y minúsculas. Además reemplace uno de los apellidos y muestre el nombre original y el actualizado.

11. Mostrar por pantalla la fecha actual de sistema, seguido de la hora. Pedirle al usuario un numero entero que representa el numero de meses par agregarle a la fecha. Visualizar por pantalla la nueva fecha.

12. Calcular el número de días transcurridos entre la fecha de su nacimiento y la fecha actual.

13. Calcular el numero de segundos que demora el usuario para digitar un dato por pantalla.

14. Dada una fecha por teclado obtener: el nombre del día actual, el periodo al que pertenece.

6

Control de Flujo I.
Estructuras condicionales

Plan general.

OBJETIVOS

- Presentar unos componentes básicos para el desarrollo de algoritmos mas dinámicos.
- Desarrollar ejercicios detallados para explicación de las estructuras condicionales.
- Fortalecer los conceptos de algoritmos.
- Presentar herramientas para afrontar problemas computacionales mas complejos.

6.1. INTRODUCCIÓN.

Todos los algoritmos desarrollados hasta el momento se ejecutan de manera secuencial, es decir se ejecutan desde el inicio hasta el fin sin modificaciones y siempre van a desarrollar el mismo proceso con diferentes valores.

Las estructuras condicionales permiten la múltiple ejecución de instrucciones según los contenidos de las variables o datos que ingrese el usuario, es decir, según los datos suministre el usuario pueden ejecutarse una serie u otra de instrucciones.

6.2. CONTROL DEL FLUJO, TOMA DE DECISIONES.

Este tipo de elementos permiten generar un tipo de dinamismo sobre los Pseudocódigos, en la forma que proporcionan varias opciones de ejecución. En otros términos, un sencillo programa puede ejecutar diferentes tipos de acciones según parámetros que se establecen en la construcción del mismo.

Todas las estructuras que se verán a continuación, necesitan de operadores de relación con las cuales generar reglas para que el Pseudocódigo pueda tomar decisiones. Antes

de iniciar el recorrido por cada una de ellas, se repasara los operadores de relación y lógicos (que permiten generar condiciones más complejas a partir de operadores lógicos) con el fin de tener claros los conceptos teóricos y reforzarlos después con la practica.

6.2.1. Operadores de relación

Los operadores que se muestran a continuación pueden realizar acciones con datos o variables del mismo tipo que no sean de tipo lógico.

6.2.1.1. Mayor que (>)

Compara dos términos del mismo tipo.

Por ejemplo:

Con constantes	
5 > 3	la respuesta es verdadera
Con variables	
A = 3	la variable A se le asigna el valor de 3
B = 9	la variable B se le asigna el valor de 9
A > B	la respuesta es falsa.

```
1   'Pseudocodigo mayor que
2
3   Inicio
4           Declaracion A, B, Prueba
5
6           A = 3
7           B = 9
8           Prueba = CBool(A > B)
9           Escribir Prueba
10  Fin
11
```

6.2.1.2. Menor que (<)

Compara dos términos del mismo tipo.

Por ejemplo:

Con constantes	
5 < 3	la respuesta es falsa
Con variables	
A = 3	la variable A se le asigna el valor de 3
B = 9	la variable B se le asigna el valor de 9
A < B	la respuesta es verdadera.

```
1     'Pseudocodigo menor que
2
3     Inicio
4           Declaracion A, B, Prueba
5
6           A = 3
7           B = 9
8           Prueba = CBool(A < B)
9           Escribir Prueba
10    Fin
```

6.2.1.3. Mayor o igual que (>=)

Compara dos términos del mismo tipo.

Por ejemplo:

Con constantes	
5 >= 3	la respuesta es verdadera
Con variables	
A = 3	la variable A se le asigna el valor de 3
B = 9	la variable B se le asigna el valor de 9
A >= B	la respuesta es falsa.

```
1     'Pseudocodigo mayor o igual que
2
3     Inicio
4           Declaracion A, B, Prueba
5
6           A = 3
7           B = 9
8           Prueba = CBool(A >= B)
9           Escribir Prueba
10    Fin
```

6.2.1.4. Menor o igual que (<=)

Compara dos términos del mismo tipo.

Por ejemplo:

Con constantes		
	5 <= 3	la respuesta es falsa
Con variables		
	A = 3	la variable A se le asigna el valor de 3
	B = 9	la variable B se le asigna el valor de 9
	A <= B	la respuesta es verdadera.

```
1     'Pseudocodigo menor o igual que
2
3     Inicio
4           Declaracion A, B, Prueba
5
6           A = 3
7           B = 9
8           Prueba = CBool(A <= B)
9           Escribir Prueba
10    Fin
```

6.2.1.5. Igual a (=)

Compara dos términos del mismo tipo.

Por ejemplo:

Con constantes

5 = 3 la respuesta es falso

Con variables

A = 3 la variable A se le asigna el valor de 3
B = 9 la variable B se le asigna el valor de 9
A = B la respuesta es falsa.

```
1   'Pseudocodigo igual a
2
3   Inicio
4       Declaracion A, B, Prueba
5
6       A = 3
7       B = 9
8       Prueba = CBool(A = B)
9       Escribir Prueba
10  Fin
```

Nota.

Debe hacerse claridad con respecto a este operador. Secciones anteriores se conoció que **=** es una asignación y para este numeral se conoce como una operación de comparación. Ambas tareas son correctas, pero se diferencian según la expresión en la que se están usando. Para expresiones sencillas como por ejemplo:

A =20 se comporta como una asignación, pero en: **S*i A =20* entonces** es una operación de comparación y se diferencia por que esta acompañado por la instrucción **Si**, la cual sirva para comparar dos o más términos.

6.2.1.6. Diferente (<>)

Compara dos términos del mismo tipo.

Por ejemplo:

Con constantes

5 <> 3 la respuesta es verdadera

Con variables

A = 3	la variable A se le asigna el valor de 3
B = 9	la variable B se le asigna el valor de 9
A <> B	la respuesta es verdadera.

```
1   'Pseudocodigo diferente
2
3   Inicio
4       Declaracion A, B, Prueba
5
6       A = 3
7       B = 9
8       Prueba = CBool(A <> B)
9       Escribir Prueba
10  Fin
```

6.2.2. Operadores lógicos

Los operadores que se muestran a continuación pueden realizar acciones con datos o variables del mismo tipo que sean de tipo lógico (Verdadero v. V. – Falso f. F).

6.2.2.1. And [Y] (Conjunción)

El concepto de conjunción corresponde a la unión de dos elementos. Para que el resultado sea verdadero, se necesita que ambos elementos sean verdaderos.

Por ejemplo:

- Con datos boléanos.

Valor1	**Valor2**	Valor1 **and** Valor 2
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Falso
Falso	Verdadero	Falso
Falso	Falso	Falso

con expresiones matemáticas

Expresión1	**Expresión 2**	**Expresión1 and Expresión2**
8 > 4	7 < 10	Verdadero
10 >= 3	14 < 3	Falso
10 <> 10	5 = 5	Falso
5 >= 10	9 <= 3	Falso

```
1     'Pseudocodigo el and
2
3     Inicio
4           Declaracion A, B, C, D, Prueba
5
6           A = 8
7           B = 4
8           C = 7
9           D = 10
10          Prueba = CBool((A>B) y (C<D))
11          Escribir Prueba
12    Fin
```

6.2.2.2. Or (O lógico)

Él **O** lógica realiza la operación de disyunción o separación y análisis de elementos o expresiones. Para que el resultado sea verdadero basta conque una de las expresiones o elementos sea verdadero. Por ejemplo:

Con datos boléanos.

Valor1	**Valor2**	Valor1 **Or** Valor 2
Verdadero	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Falso

con expresiones matemáticas

Expresión1	**Expresión 2**	**Expresión1 Or Expresión2**
8 > 4	7 < 10	Verdadero
10 >= 3	14 < 3	Verdadero
10 <> 10	5 = 5	Verdadero
5 >= 10	9 <= 3	Falso

```
1     'Pseudocodigo el and
2
3     Inicio
4           Declaracion A, B, C, D, Prueba
5
6           A = 8
7           B = 4
8           C = 7
9           D = 10
10          Prueba = CBool((A>B) or (C<D))
11          Escribir Prueba
12    Fin
```

6.2.2.3. NAND ()

Por ejemplo:

Con datos boléanos.

Valor1	**Valor2**	Valor1 **Y** Valor 2
Verdadero	Verdadero	Falso
Verdadero	Falso	Verdadero
Falso	Verdadero	Verdadero
Falso	Falso	Verdadero

con expresiones matemáticas

Expresión1	**Expresión 2**	**Expresión1 Y Expresión2**
8 > 4	7 < 10	Falso
10 >= 3	14 < 3	Verdadero
10 <> 10	5 = 5	Verdadero
5 >= 10	9 <= 3	Verdadero

6.2.2.4. NOR ()

Por ejemplo:

Con datos boléanos.

Valor1	**Valor2**	Valor1 **Y** Valor 2
Verdadero	Verdadero	Falso
Verdadero	Falso	Falso
Falso	Verdadero	Falso
Falso	Falso	Verdadero

con expresiones matemáticas

Expresión1	**Expresión 2**	**Expresión1 Y Expresión2**
8 > 4	7 < 10	Falso
10 >= 3	14 < 3	Falso
10 <> 10	5 = 5	Falso
5 >= 10	9 <= 3	Verdadero

6.2.2.5. NOT (Negación lógica)

El **NO** lógico realiza la operación de negación sobre una expresión. Para obtener verdadero la expresión debe ser falsa y viceversa.

Por ejemplo:

Con datos boléanos.

Valor	**NO** Valor
Verdadero	Falso
Falso	Verdadero

con expresiones matemáticas

Expresión	NO Expresión
8 > 4	Falso
10 >= 3	Falso
10 <> 10	Verdadero
5 >= 10	Verdadero

De manera homóloga se comportan los operadores de relación y los operadores lógicos. Cada uno se encuentra regido por un orden jerárquico para efectuar las diferentes operaciones en una expresión.

JERARQUÍA DE OPERADORES LÓGICOS.

Operador	Nombre	Jerarquía al efectuar una operación
No	Negación	1
Y	Conjunción	2
O	Disyunción	3
XOR		

JERARQUÍA DE OPERADORES DE RELACIÓN.

Operador	Nombre	Jerarquía al efectuar una operación
=	Igual	1
<>	Diferente	2
<	Menor que	3
>	Mayor que	4
<=	Menor o igual	5
>=	Mayor o igual	6

6.3. ESTRUCTURAS CONDICIONALES

6.3.1. Él Sí simple.

Permite ejecutar un grupo de acciones mientras la condición o condiciones sean verdaderas. Véase la definición de esta primera estructura.

```
...
Si Condición entonces
        Acción 1
        Acción 2
        ...
        Acción N
FinSi
...
```

Cabe resaltar que la **Acción 1, Acción 2,..., Acción N** se ejecutaran mientas la condición sea verdadera. De lo contrario si la condición es falsa, dichas acciones no se llevaran a cabo.

Ejercicio.

Dado un numero por teclado, mostrar un mensaje si es un numero positivo. todos los números positivos son mayores que cero (0), por tanto la comparación que se llevara a cabo se realiza con respecto a cero.

Entrada: Un numero, que se almacenara en una variable llamada **Numero**.
Proceso: Comparar el contenido de la variable **Numero** y verificar si es mayor que cero (0).
Salida: Mostrar un mensaje si es mayor que cero

```
1   'Pseudocódigo condición simple
2
3   Inicio
4           Declaracion numero
5
6           numero=Leer("un numero")
7
8           Si numero > 0 Entonces
9                   Escribir numero & " es un numero positivo"
10          FinSi
11  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(8). En esta instrucción se hace efectivo el tema antes visto de operadores de relación. En este caso se compara el contenido de la variable **numero** en pos del valor cero (0), estableciendo la siguiente comparación:

En términos algorítmicos es,
Si numero > 0 Entonces

En términos hablados es,
Si el contenido de la variable **numero** es mayor que cero **entonces**

Suponga los siguientes valores: **-45** y **12**, posibles de la variable **numero**. Cuando **numero** contenga **–45** la instrucción se evaluaría: **Si –45 > 0 entonces**, obteniendo una condición **falsa,** ya que –45 nunca va a ser mayor que cero, en este caso la instrucción **Escribir numero & " es un numero positivo"** no se llevaría a cabo. Ahora si el valor de **numero** es 12, la instrucción se vería: **Si 12> 0 entonces**, y como 12 si es mayor que cero, se imprimiría en pantalla un mensaje similar, **12 es un numero positivo.**

(9). La idea del mensaje en pantalla es que aparezca el numero, seguido del mensaje **es un numero positivo**, por ese caso se muestra el contenido de la variable **numero** y luego se une con la cadena "**es un numero positivo**".

(10). Todas las estructuras condicionales terminan con la palabra reservada **FinSi**.

Ejercicio.

Dado un numero por teclado, mostrar un mensaje si es un numero positivo o negativo.

Para este ejemplo se deben evaluar las condiciones cuando sea mayor que cero y menor que cero. Esto se logra usando dos estructuras condicionales simples.

Entrada: Un numero, que se almacenara en una variable llamada **Numero**.
Proceso: Comparar el contenido de la variable **Numero** , verificar si es mayor que cero (0) y menor que cero.
Salida: Mostrar un mensaje si es mayor que cero o menor que cero.

```
1   'Pseudocódigo múltiple condición simple
2
3   Inicio
4       Declaracion numero
5
6       numero=Leer("un numero")
7
8       Si numero > 0 Entonces
9           Escribir numero & " es un numero positivo"
10      FinSi
11
12      Si numero < 0 Entonces
13          Escribir numero & "es un numero negativo"
14      FinSi
15  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(8). Esta primera estructura condicional evalúa si el numero dado es mayor que cero, de ser verdadera ejecuta la acción de mostrar el mensaje y continua en la instrucción **FinSi**. Ahora si la condición es falsa, salta al **FinSi** en pos de continuar con el flujo del programa. Recuerde que los programas que se realizan en esta sección son secuenciales, es decir una instrucción a la vez.

(12). La segunda condición evalúa si es un numero menor que cero.

6.3.2. El Si compuesto.

Permite ejecutar un grupo de acciones mientras la condición o condiciones sean verdaderas o falsas. Este componente evalúa la condición si es verdadera ejecuta un grupo de acciones y si es falsa ejecuta otro grupo diferente de acciones Véase la definición de esta estructura.

```
...
Si Condición entonces
        Acción verdadera1
        Acción verdadera2
        ...
        Acción verdaderaN
Sino
        Acción falsa1
        Acción falsa 2
        ...
        Acción falsaN
FinSi
...
```

La estructura condicional simple solo tenia en cuenta uno posible estado evaluando la condición. Para este caso la estructura compuesta, tiene en cuenta los 2 resultados de la evaluación de la condición, el **verdadero** y el **falso.**

Entiéndase su funcionamiento el ejemplo que se ha venido desarrollando.

Ejercicio.

Dado un numero por teclado, mostrar un mensaje si es un numero positivo o negativo.

Entrada: Un numero, que se almacenara en una variable llamada **Numero**.
Proceso: Comparar el contenido de la variable **Numero** , verificar si es mayor que cero (0) y menor que cero.
Salida: Mostrar un mensaje si es mayor que cero o menor que cero.

```
1   'Pseudocódigo condicional compuesto
2
3   Inicio
4       Declaración numero
5
6       numero=Leer("un numero")
7
8       Si numero > 0 Entonces
9           Escribir numero & " es un numero positivo"
10      Sino
11          Escribir numero & "es un numero negativo"
12      FinSi
13  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(8). En esta primera parte de la estructura condicional evalúa la condición, si es verdadera ejecuta la acción **Escribir numero & " es un numero positivo".** por lo contrario si la condición es falsa se ejecuta el bloque de acciones contenida entre el **Sino y el FinSi**, para este caso corresponde a **Escribir numero & "es un numero negativo"**

(10). El **Sino** es la otra posibilidad de la condición siempre se ejecuta cuando la condición evaluada en el **Si** no es verdadera.

(12). Como se menciono anteriormente, todas las estructuras condicionales poseen del **FinSi** para indicar el fin de la estructura condicional.

Ejercicio.

Dado un numero por teclado, mostrar un mensaje si el numero pertenece o no al rango, entre 45 y 70.

Entrada: Un numero, que se almacenara en una variable llamada **Num**.
Proceso: Verificar si el contenido de la variable **Num** , se encuentra entre 45 y 70.
Salida: Mostrar un mensaje si pertenece o no al rango.

```
1   'Pseudocódigo condicional compuesto en un rango
2
3   Inicio
4       Declaracion numero
5
6       numero=Leer("un numero")
7
8       Si numero >= 45 and numero<=70 Entonces
9           Escribir " Pertenece el rango"
10      Sino
11          Escribir "No pertenece el rango"
12      FinSi
13  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(8). La condición que evalúa esta estructura condicional constante de lo siguiente: **Si numero >= 45 and numero<=70 Entonces**

Consta primeramente de una condición con el operador relacional **mayor o igual que** en numero >= 45. Seguidamente contiene un operador lógico **and**. Y por ultimo del operador **menor o igual que** en numero<=70.

La construcción de esta condición pertenece al interés del ejercicio planteado, en evaluar si el numero dado se encuentra entre 45 y 70. por eso se evalúa primero si el numero es mayor o igual a 45, luego si el numero es menor o igual a 70 y el operador lógico **and** permite fijar una relación entre ambas

condiciones. Este operador une ambas condiciones y como se menciono anteriormente, si ambas condiciones son verdaderas el resultado es verdadero.

5.2.3. Combinación de Si simples y compuestos.

En la gran mayoría de las implementaciones de algoritmos, se necesita de la combinación de las anteriores estructuras para generar condiciones mas complejas y adecuadas, al problema planteado.

Véase algunas de las posibles combinaciones.

```
...
Si Condición entonces
        Si Condición entonces
                Acción 1
                ...
                Acción N
        FinSi
Sino
        Acción falsa1
        Acción falsa 2
        ...
        Acción fallan
FinSi
```

```
...
Si Condición entonces
        Si Condición entonces
                Acción verdadera1
                Acción verdadera2
                ...
                Acción verdaderaN
        Sino
                Acción falsa1
                Acción falsa 2
                ...
                Acción falsaN
        FinSi
FinSi
...
```

Nota.

Obsérvese que cada vez que se abre una estructura condicional, las siguientes instrucciones que pertenecen a ella se encuentran dentadas (alineadas) a su derecha, con el propósito de identificarlas más rápido, permitiendo la legibilidad del código. Además, cada estructura condicional va acompaña del **FinSi**.

Ejercicio.

Dado un numero por teclado calcular las siguiente operaciones: si el numero es menor que diez, calcular la raíz cuadrada. si esta en el rango entre 15 y 32, calcular el cubo y cuadrado. Si es 20 o 30 mostrar un mensaje que diga que el numero termina en cero.

Entrada: Un numero, que se almacenara en una variable llamada **Num**.
Proceso: Realizar las respectivas comparaciones: si es menor que diez, si es mayor igual a 15 y menor e igual a 32 y si es igual a 20.
Salida: Mostrar los respectivos valores según el tipo de condición sea correcta.

```
1   ‘Pseudocódigo procesos según valor
2
3   Inicio
4      Declaracion numero
5
6      numero=Leer("un numero")
7
8       Si numero < 10 Entonces
9           raiz=Raiz2(numero)
10          Escribir "la raíz cuadrada es " & raiz
11      Sino
12          Si numero >=15 and numero<=32 Entonces
13             cubo=numero^3
14             cuadrado=numero^2
15            Escribir "el cubo es igual a " & cubo & " el cuadrado es "
16  & cuadrado
17
18              Si numero=20 or numero=30 Entonces
19                 Escribir "el numero termina en cero"
20              FinSi
21          FinSi
22     FinSi
23  Fin
24
```

EXPLICACIÓN POR INSTRUCCIÓN:

(8). La primera condición que se evalúa corresponde a si el numero es menor que 10, si la condición es verdadera se calcula el valor de la raíz cuadrada y se imprime el valor. Cuando se terminan de ejecutar estas acciones, se va hacia el **FinSi** de esa instrucción.

(11). Cuando la condición del anterior paso es falsa, se salta al cuerpo del **Sino**, este posee un condicional que evalúa si el numero corresponde al rango entre 15 y 32.

(18). Esta ultima instrucción evalúa que el numero dado sea 20 **O** 30, empleando el operador lógico **OR**.

6.3.4. La estructura SegunSea.

Este tipo de estructura condicional permite evaluar un gran numero de condiciones de las cuales se conocen sus posibles valores sobre una misma variable.

```
...
SegunSea nombreVariable
        Opcion valor1:
                Accion1 Opcion valor1
                ...
                AccionN Opcion valor1
        Opcion 2:
                Accion1 Opcion valor2
                ...
                AccionN Opcion valor2
        SinOpcion:
                Accion1 SinOpcion
                ...
                AccionN SinOpcion
FinSegunSea
...
```

Para entender el funcionamiento de esta estructura, se va a realizar el mismo ejemplo con los elementos vistos hasta ahora.

Ejercicio.

Dado un numero entre 1 y 7 que representa un día de la semana, imprimir el nombre del día correspondiente: 1 lunes, 2 martes, 3 miércoles, ..., 7 domingo.

DESARROLLO CON EL SI SIMPLE.

Usando el si simple basta con realizar una condición por cada valor que se desea evaluar, es decir una condición para el valor 1 que corresponde a lunes, otra para el valor 2 que pertenece a martes y así sucesivamente con todos los valores.

Entrada: Un numero, que se almacenara en una variable llamada **numeroDia**.
Proceso: Realizar las respectivas comparaciones para cada uno de los valores entre 1 y 7.
Salida: Mostrar el nombre del día, según el valor contenido en **numeroDia**.

```
1   Pseudocódigo 4.2.3.1.A
2   Inicio
3           Declaracion numeroDia
4           numeroDia=Leer("ingrese el numero")
5           Si numeroDia=1 Entonces
6                   Escribir "Lunes"
7           FinSi
8           Si numeroDia=2 Entonces
9                   Escribir "Martes"
10          FinSi
11          Si numeroDia=3 Entonces
12                  Escribir "Miercoles"
13          FinSi
14          Si numeroDia=4 Entonces
15                  Escribir "Jueves"
16          FinSi
17          Si numeroDia= 5 Entonces
18                  Escribir "Viernes"
19          FinSi
20          Si numeroDia=6 Entonces
21                  Escribir "Sabado"
22          FinSi
23          Si numeroDia=7 Entonces
24                  Escribir "Domingo"
25          FinSi
26  Fin
```

DESARROLLO CON EL SI COMPUESTO Y COMBINACIÓN CON EL SI SIMPLE.

Cuando se implementan algoritmos con si compuestos debe tenerse en el manejo de las acciones falsas. Estas pueden ser operaciones o mas estructuras condicionales simples o compuestas, según las necesidades para resolver el algoritmo en Pseudocódigo.

Entrada: Un numero, que se almacenara en una variable llamada **numeroDia**.
Proceso: Realizar las respectivas comparaciones para cada uno de los valores entre 1 y 7.
Salida: Mostrar el nombre del día, según el valor contenido en **numeroDia**.

```
1   Pseudocódigo 4.2.3.1.B
2
3   Inicio
4     Declaracion numeroDia
5     numeroDia=Leer("ingrese el numero")
6     Si numeroDia=1 Entonces
7       Escribir "Lunes"
8     Sino
9       Si numeroDia=2 Entonces
10        Escribir "Martes"
11      Sino
12        Si numeroDia=3 Entonces
13          Escribir "Miercoles"
14        Sino
15          Si numeroDia=4 Entonces
16            Escribir "Jueves"
17          Sino
18            Si numeroDia= 5 Entonces
19              Escribir "Viernes"
20            Sino
21              Si numeroDia=6 Entonces
22                Escribir "Sabado"
23              Sino
24                Si numeroDia=7 Entonces
25                  Escribir "Domingo"
26                FinSi
27              FinSi
28
```

```
29              FinSi
30            FinSi
31          FinSi
32        FinSi
33      FinSi
34  Fin
```

DESARROLLO CON LA ESTRUCTURA SEGUNSEA.

Esta estructura facilita el desarrollo de este tipo de ejercicios, donde se conocen los posibles valores. En ella se evalúan por valores y se ejecuta al que pertenece el valor que se tiene sobre la variable.

Entrada: Un numero, que se almacenara en una variable llamada **numeroDia**.
Proceso: Realizar las respectivas comparaciones para cada uno de los valores entre 1 y 7.
Salida: Mostrar el nombre del día, según el valor contenido en **numeroDia**.

```
1   ‘Pseudocódigo con según sea
2
3   Inicio
4           Declaracion numeroDia
5           numeroDia=Leer("ingrese el numero")
6           SegunSea numeroDia
7                   Opcion 1: Escribir "lunes"
8                   Opcion 2: Escribir "martes"
9                   Opcion 3: Escribir "miercoles"
10                  Opcion 4: Escribir "jueves"
11                  Opcion 5: Escribir "viernes"
12                  Opcion 6: Escribir "sabado"
13                  Opcion 7: Escribir "domingo"
14                  SinOpcion: Escribir "valor erroneo"
15          FinSegunSea
16  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(6). Se usa la estructura **SegunSea** seguidamente la variable (para este caso es **numeroDia**) que se quiere evaluar sobre los posibles valores que puede tener o llegar a contener.

(7). La palabra reservada **Opcion** va seguida del valor que pueda tener para al caso. Como el ejercicio pide evaluar sobre los valores del 1 al 7, se examina cada valor de este rango de manera separa. Existen varia formas de indicar esta instrucción:

La primera indica que se evalúa la opción y de ser verdadera se muestra por pantalla el mensaje . **Opcion 1: Escribir "lunes".** Esta es empleada cuando solo se realiza una sola acción cuando la opción es la correcta.

la otra forma de expresarla es:

Opcion 1:
Escribir "Lunes"

Esta ultima se emplea cuando se necesitan ejecutar varias acciones si se cumple esta opción.

RESUMEN.

- El control de flujo permite generar un tipo de dinamismo sobre los Pseudocódigos, en la forma que proporcionan varias opciones de ejecución.

- Todas las estructuras condicionales necesitan de operadores de relación con las cuales generar reglas para que el Pseudocódigo pueda tomar decisiones.

- Los operadores lógicos pueden realizar acciones con datos o variables del mismo tipo que sean de tipo lógico (Verdadero v. V. – Falso f . F).

- El Si simple. Permite ejecutar un grupo de acciones mientras la condición o condiciones sean verdaderas.

- La estructura condicional simple solo tenia en cuenta uno posible estado evaluando la condición.

- El Si compuesto permite ejecutar un grupo de acciones mientras la condición o condiciones sean verdaderas o falsas.

- La Combinación de Si simples y compuestos se encuentran en la gran mayoría de las implementaciones de algoritmos, se necesita de la combinación de las anteriores estructuras para generar condiciones mas complejas y adecuadas

- La estructura SegunSea permite evaluar un gran numero de condiciones de las cuales se conocen sus posibles valores sobre una misma variable.

EJERCICIOS DE AUTO EVALUACIÓN.

Desarrolle la gran mayoría de Pseudocódigos anteriores de estructuras condicionales para que verifique su correcto funcionamiento y entendimiento.

EJERCICIOS PROPUESTOS.

Desarrolle los siguientes algoritmos

1. Leer una persona el nombre y la edad. Si es mayor de edad mostrar por pantalla su nombre.

2. Dado un numero por teclado verificar si es un numero positivo o negativo. Mostrar un mensaje para cada caso.

3. Dados 2 números imprimir el mayor de ellos.

4. Dados 3 números imprimir el menor de los tres.

5. Dados 3 números imprimir el valor intermedio de los tres.

6. Dados 3 números determinar si están ordenados ascendentemente.

7. Realizar las operaciones básicas con 2 números leídos por teclado. Tenga en cuenta que la división por cero es una indeterminación.

8. Calcular el salario de un empleado dado el numero de horas, el valor de la hora y la categoría del empleado

Categoría	Porcentaje de incremento del salario
A	10%
B	15%
C	23%
D	25%

9. Dado el nombre de un numero del 1 al 10 imprimir el valor.

10. Ingresar valores numéricos enteros A, B, C, D, E y decir si su promedio es mayor que o igual a 10.

11. Ingresar valores numéricos reales A, B, C, y decir si su promedio es mayor que o igual a 10.

12. Ingresar valores numéricos reales a, b, c, que son coeficientes de una ecuación cuadrática y obtener los valores X1 y X2 reales. Si la operación dentro de la raíz diera como resultado un valor negativo, imprimir un cartel que diga **"La solución son dos números complejos conjugados"**.

13. Ingresar valores numéricos reales a, b, c, que son coeficientes de una ecuación cuadrática y obtener los valores X1 y X2 reales. Si la operación dentro de la raíz diera como resultado un valor negativo, imprimir el resultado como

    "m + n i; m - n i ".

14. Hacer un programa que permita ingresa un número de 1 a 7 y salga el correspondiente día de la semana (Ej.: 1 → Lunes; 2 → Martes; ...). Si ingresa un valor que no este comprendido entre 1 y 7 deberá imprimir un cartel que diga "ERROR ... valor fuera de rango"

15. Hacer un programa que permita ingresa dos números reales y el símbolo de la operación. Obteniéndose el correspondiente resultado. Si el símbolo no es correcto, deberá imprimir un mensaje que indique "Error en símbolo"

16. La estructura del tipo selectivo (salto condicional) ¿requiere modificaciones en el hardware para que pueda ser utilizada en los programas? ¿Que ventajas presenta?

17. Calcule las raíces de una ecuación cuadrática mostrando las raíces reales y las complejas.

18. Dados dos puntos del plano la recta que los une calcule la distancia entre los puntos y el cuadrante en que se encuentra cada uno. Valide todos los casos posibles.

19. Realice un programa que calcule el interés simple de un monto ingresado al "x" % anual. Si el cliente se atrasa en el pago de la cuota, se le agrega un "y" % mensual. Muestre en pantalla el total adeudado.

20. Decidir si dos puntos del plano son equidistantes al origen de coordenadas. Si lo son calcular el ángulo de desfase.

21. Una empresa se encarga de la venta y distribución de CD vírgenes. Los clientes pueden adquirir los artículos (supongamos un único producto de una única marca) por cantidad. Los precios son:

    • 0.35 U$S si se compran unidades separadas hasta 9.
    • 0.34 U$S si se compran entre 10 unidades hasta 99.
    • 0.30 U$S entre 100 y 499 unidades
    • 0.28 U$S para mas de 500

    El costo del vendedor es de 0.25 U$S por CD. Realizar un programa que dado un número de CDs a vender calcule el precio total para el cliente, el costo total y la ganancia para el vendedor.

7

Control de Flujo II.
Estructuras Repetitivas

Plan general.

7.1. Introducción.
7.2. Control de flujo toma de decisiones.
7.3. Estructuras condicionales.
Resumen.

OBJETIVOS

- Presentar unos componentes básicos para el desarrollo de algoritmos mas dinámicos.
- Desarrollar ejercicios detallados para explicación de las estructuras repetitivas.
- Fortalecer los conceptos de algoritmos.
- Presentar herramientas para afrontar problemas computacionales mas complejos, donde se necesite la ejecución de un grupo de instrucciones una cantidad de veces.

7.1. INTRODUCCIÓN.

Con el capitulo anterior se dio un poco de dinamismo a los algoritmos, pero existen problemas que requieren de la ejecución de instrucciones ya desarrolladas. Con las estructuras repetitivas o ciclos, se da la primera posibilidad de reutilizar código ya generado y usarlo el numero de veces que se necesite.

7.2. CONTROL DEL FLUJO, CICLOS.

En el capitulo anterior, se conoció como generar algo de dinamismo en los Pseudocódigos mediante el uso de estructuras condicionales, estas permiten cambiar el flujo de ejecución dependiendo de una serie de reglas o condiciones. Según el estado de la condición (Verdadero o Falso) tomarían una serie de acciones diferentes.

En este capitulo se aprenderá a desarrollar algoritmos aun mas dinámicos mediante el uso de estructuras repetitivas. Estas permiten reproducir una serie instrucciones unas numero finito o infinito de veces, según el objetivo del algoritmo.

Para entender la importancia de las estructuras repetitivas, obsérvese el siguiente ejemplo.

Ejercicio.

Imprimir los números del 1 al 10.

Nota.

Con lo aprendido hasta ahora, se podría desarrollar así:

a. imprimir la lista de números, con la función **Escribir**
b. inicializar una variable en 1, e ir incrementando su valor y mostrando.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 10.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo números del 1 al 10 con múltiples escribir
2
3   Inicio
4           Escribir "1"
5           Escribir "2"
6           Escribir "3"
7           Escribir "4"
8           Escribir "5"
9           Escribir "6"
10          Escribir "7"
11          Escribir "8"
12          Escribir "9"
13          Escribir "10"
14  Fin
```

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 10.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo números del 1 al 10 con un solo escribir
2
3   Inicio
4       Escribir "1,2,3,4,5,6,7,8,9,10"
5   Fin
```

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 10.
Salida: Mostrar la lista de números.

```
1   Pseudocódigo 1n.
2
3   Inicio
4       Declaracion Numero
5       Numero=1
6       Escribir Numero
7       Numero=Numero + 1
8       Escribir Numero
9       Numero=Numero + 1
10      Escribir Numero
11      Numero=Numero + 1
12      Escribir Numero
13      Numero=Numero + 1
14      Escribir Numero
15      Numero=Numero + 1
16      Escribir Numero
17      Numero=Numero + 1
18      Escribir Numero
19      Numero=Numero + 1
20      Escribir Numero
21      Numero=Numero + 1
22      Escribir Numero
23      Numero=Numero + 1
24      Escribir Numero
25  Fin
```

EXPLICACIÓN POR ALGORITMO:

En el primer Pseudocódigo se imprime separadamente cada numero usando la función **Escribir** y el numero que se desea mostrar. Es una aplicación sencilla que cumple con el enunciado, pero de manera estática y plana.

El siguiente, muestra todos los números separados por comas sobre la misma ventana de salida de datos. Este presenta un comportamiento similar al anterior.

El ultimo, permite inicializar una variable con el valor de 1, se muestra el resultado y se continua incrementando la variable en 1, mostrando los números del 1 al 10 dinámicamente, ya que es una variable que varia entre el rango deseado. Esta solución es bien desarrollada pero continua siendo muy manual, ya que cuando se implementa hay que tener en cuenta el numero total de incrementos o veces que la variable aumenta su valor en 1.

DESARROLLO E IMPLEMENTACIÓN EN ESTRUCTURAS REPETITIVAS.

Si se revisa nuevamente el primer y ultimo Pseudocódigo, se encuentra que para ambos la función **Escribir** se repite diez veces y en el caso del ultimo se hacen 10 cambios de valor sobre la variable **Numero**. Partiendo de allí, se usara una estructura repetitiva, que permita ejecutar diez veces el incremento de la variable y la salida con la función **Escribir**.

7.2.1. El ciclo Para...FinPara.

Esta estructura repetitiva esta diseñada para cuando se conoce el valor inicial (en donde empieza el ciclo) y el final (en donde termina). Obsérvese su sintaxis.

```
...
Para nombreVariable  = valorInicial Hasta valorFinal
        Acciones 1
        Acciones 2
        ...
        Acciones N
FinPara
...
```

Donde **nombreVariable** corresponde al nombre hábil de una variable, que contendrá los valores que se definan en los siguientes 2 parámetros. **valorInicial** pertenece al valor donde se desea iniciar el ciclo y **valorFinal** es el ultimo valor del ciclo.

Para entender mas esta conceptualización, imagine que le dan 2 números cualquiera, por ejemplo 15 y 23, y le piden que diga todos los números contenidos en ese rango incluyendo los que lo determinan, entonces se empezarían a nombrar como: 15, 16, 17, 18, 19, 20, 21, 22, 23. Esa misma labor la cumple este ciclo, ud le determina el rango de valores sobre los cuales quiere trabajar. Además UD mentalmente realiza el incremento uno a uno para obtener el valor siguiente, esta misma operación de incremento interno lo realiza el ciclo cada vez que realiza una iteración o empieza de nuevo a ejecutar el cuerpo del ciclo. En la explicación por instrucción se va a prestar mucha atención en el funcionamiento del ciclo.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 10.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo ciclo para 1
2
3   Inicio
4           Para Numero = 1 Hasta 10
5                   Escribir Numero
6           FinPara
7   Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(4). En esta instrucción se encuentra la definición del ciclo **Para**. La variable **Numero** contendrá los valores entre el valor inicial, que para este caso es 1 y el valor final que es 10, uno cada vez por iteración o vez que se repita el ciclo. Además de verificar los contenidos de la variable **Numero** con el fin de validar que no sobre pase el limite superior; de ser así, termina la ejecución del ciclo y ejecuta la línea siguiente al **FinPara**.

(5). La instrucción **Escribir** mostrara el valor actual que contenga la variable **Numero**.

(6). Esta instrucción en la encargada de las siguiente tareas:

a). Finalizar el ciclo Para.

b). Incrementar la variable para este caso se llama **Numero**, en uno ya que es el incremento por defecto que tiene el ciclo.

c). Dirigir el flujo de ejecución nuevamente a la instrucción 2, a la definición del ciclo para, con el fin de que esta realice sus tareas.

Retomando este ejercicio, observe la inclusión de un nuevo elemento que permite determinar cual es el incremento.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 10.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo ciclo para 2
2
3   Inicio
4           Para Numero = 1 Hasta 10 Incremento 1
5                   Escribir Numero
6           FinPara
7   Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(4). Adicionalmente a las tareas que realiza esta instrucción, se agrega un nuevo elemento que es **Incremento**. Este componente permite definir un valor de incremento por iteración. En este ejemplo se determina que el incremento es 1, valor por defecto cuando no se define, como se observo en el ejemplo anterior. Esto le indica a la variable que su valor va aumentado desde su valor inicial hasta el final de uno en uno.

Para entender mas a fondo el funcionamiento del **incremento** y por ende de la estructura repetitiva **Para**, observe los siguientes ejemplos.

Ejercicio.

Imprimir los números del 10 al 1.

Nota.

En el ejercicio anterior se imprimieron los números ascendentemente, la idea en este es mostrarlos descendentemente, así: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

En este caso la definición del ciclo para seria la siguiente:

a). El valor inicial es 10, ya que se quiere empezar por el valor de 10.
b). El valor final es 1 en donde se termina el ciclo.
c). Para este caso hay que definir el incremento en –1. Esto quiere decir, que la variable va a incrementar su valor en –1, es decir incrementar negativamente o decrementar el valor cada vez en –1, ir restando al valor original con -1.

Es importante definir este incremento negativo o decremento, ya que el ciclo por si mismo no es capaz de detectar que se inicia en un valor mayor a uno menor y por tanto no realiza internamente el decremento.

Imagine definir un ciclo para así:

```
Para Num = 10 Hasta 1 Incremento 1
        Bloque de acciones
FinPara
```

Se le indica que el ciclo inicia en 10 y termina 1, y su incremento es positivo de uno en uno. Este tipo de ciclos se denomina ciclos infinitos ya que nunca la variable Num para este caso, va a llegar a contener el valor de 1. los valores de Num serian: 10, 11, 12, 13, 14,... por tanto el ciclo nunca terminaría, por que se definió iniciando en 10 y finalizando en 1, y sus valores en vez de disminuir aumentan.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 10 al 1.
Salida: Mostrar la lista de números.

```
1    Pseudocódigo 1n.
2
3    Inicio
4        Para Numero = 10 Hasta 1 Incremento -1
5            Escribir Numero
6        FinPara
7    Fin
```

7.2.2. El ciclo Mientrasque...FinMientras.

Esta estructura repetitiva esta diseñada generalmente para cuando no se conoce cuantas iteraciones o veces se debe repetir el ciclo, empleándose también cuando se conocen la cantidad de iteraciones como en el ciclo anterior. La estructura de este ciclo es como se muestra a continuación:

```
...
MientrasQue Condición
        Acciones 1
        Acciones 2
        ...
        Acciones N
FinMientras
...
```

Donde **Condición** corresponde a una condición hábil usando operadores lógicos o relacionales con respecto a una o mas variables. El ciclo mientras no realiza auto incrementos como el ciclo Para, debe tenerse en cuenta para no desarrollar ciclos infinitos o que no nunca se cumple la condición de salida.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 10.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo mientrasQue1
2
3   Inicio
4           Declaracion numero
5
6           numero=1
7           MientrasQue numero<=10
8                   Escribir numero
9                   numero=numero+1
10          FinMientras
11  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(6). Obsérvese que la variable numero se inicializa en un valor.

(7). Lo primero que hace el ciclo es evaluar el contenido de la variable numero y verificar la condición, si esta es verdadera ejecuta las líneas 8 y 9, de ser falsa salta a la instrucción 11.

(9). Observe que dentro del cuerpo del ciclo se realiza el incremento de la variable numero, de no hacerse se convierte en un ciclo infinito, ya que el valor de la variable numero siempre va a ser uno.

(11). La palabra reservado FinMientras devuelve la ejecución a la línea 7 o donde se enciente el MientrasQue, para evaluar de nuevo sentido de la condición.

Ejercicio.

Imprimir los números del 10 al 1.

Nota.

En el ejercicio anterior se imprimieron los números ascendentemente, la idea en este es mostrarlos descendentemente, así: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

En este caso la definición del ciclo para seria la siguiente:

a). El valor inicial es 10, ya que se quiere empezar por el valor de 10.

b). El valor final o la condición de salida es 1 en donde se termina el ciclo.

c). Para este caso hay que definir el incremento en –1. Esto quiere decir, que la variable va a incrementar su valor en –1, es decir incrementar negativamente o decrementar el valor cada vez en –1, ir restando al valor original con -1.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 10 al 1.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo números del 10 al 1
2
3   Inicio
4           Declaracion numero
5
6           numero=10
7           MientrasQue numero>=1
8                   Escribir numero
9                   numero=numero-1
10          FinMientras
11  Fin
```

7.2.3. El ciclo Repita...HastaQue.

Al igual que la estructura repetitiva MientrasQue se diseño para cuando no se conoce cuantas iteraciones o veces se debe repetir el ciclo, se emplea también cuando se conocen la cantidad de iteraciones como en el ciclo anterior. La diferencia con la anterior esta en que por lo menos una vez se hace una iteración, ya que la condición de salida se evalúa al final del bloque de acciones a ejecutar. A continuación se muestra su estructura:

```
...
Repita
        Acciones 1
        Acciones 2
        ...
        Acciones N
HastaQue Condición
...
```

Donde **Condición** corresponde a una condición hábil usando operadores lógicos o relacionales con respecto a una o mas variables. El ciclo Repita no realiza auto incrementos como el ciclo Para, debe tenerse en cuenta para no desarrollar ciclos infinitos o que no nunca se cumple la condición de salida. La condición debe ser falsa para que se ingrese al cuerpo del ciclo, cuando sea verdadera sale de ciclo. Es contrario al ciclo mientras, que se ejecutaba mientras la condición sea verdadera, cuando pasaba a falso salía del cuerpo del ciclo.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 10.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo números del 1 al 10
2
3   Inicio
4           Declaracion numero
5
6           numero=1
7           Repita
8                   Escribir numero
9                   numero=numero+1
10          HastaQue numero>10
11  Fin
```

EXPLICACIÓN POR INSTRUCCIÓN:

(6). Obsérvese que la variable numero se inicializa en un valor.

(7). La palabra reservada repita indica el comienzo del ciclo, aquí no se evalúan condiciones, por eso permite ejecutar por lo menos una vez las acciones dentro del ciclo.

(9). Observe que dentro del cuerpo del ciclo se realiza el incremento de la variable numero, de no hacerse se convierte en un ciclo infinito, ya que el valor de la variable numero siempre va a ser uno.

(10). La palabra reservado HastaQue evalúa la condición de ser falsa ejecuta la línea 7 para realizar otra iteración. De ser verdadera ejecuta la línea siguiente, para esta caso es el fin del programa.

Ejercicio.

Imprimir los números del 10 al 1.

Nota.

En el ejercicio anterior se imprimieron los números ascendentemente, la idea en este es mostrarlos descendentemente, así: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1

En este caso la definición del ciclo para seria la siguiente:

a). El valor inicial es 10, ya que se quiere empezar por el valor de 10.

b). El valor final o la condición de salida es 1 en donde se termina el ciclo.

c). Para este caso hay que definir el incremento en –1. Esto quiere decir, que la variable va a incrementar su valor en –1, es decir incrementar negativamente o decrementar el valor cada vez en –1, ir restando al valor original con -1.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 10 al 1.
Salida: Mostrar la lista de números.

```
1   'Pseudocódigo números del 10 al 1
2
3   Inicio
4           Declaracion numero
5
6           numero=10
7           Repita
8                   Escribir numero
9                   numero=numero-1
10          HastaQue numero<1
11  Fin
```

7.3. USOS DE LOS CICLOS.

Existen muchos problemas para resolver a nivel computacional con la ayuda de un algoritmo, pero todas se pueden agrupar en términos generales en los siguientes casos como: generadores de números y contadores.

7.3.1. Generadores de números.

En el ejercicio anterior se observo como una estructura repetitiva podía generar o producir números, como en el caso de imprimir los números del 1 al 10 o los números del 10 al 1. Se entiende como generador de números, los algoritmos en los cuales cuando se definen los parámetros que indican donde inicia, finaliza e incrementa el ciclo y permiten con esos números o sus rangos efectuar tareas de impresión, calculo, entre otros. Se va a afianzar los conocimientos con una serie de ejercicios para entender la tarea de generación de números.

Ejercicio.

Imprimir los 20 primeros números naturales

Entrada: No existen datos de entrada.
Proceso: Imprimir los números del 1 al 20.
Salida: Mostrar la lista de números.

Con el ciclo Para.

```
1  ‘Pseudocódigo numero naturales con para
2
3  Inicio
4          Para Numero = 1 Hasta 20 Incremento 1
5                  Escribir Numero
6          FinPara
7  Fin
```

Con el ciclo Mientras.

```
1  ‘Pseudocódigo numero naturales con mientras
2
3  Inicio
4          Numero = 1
5          MientrasQue Numero<=20
6                  Escribir Numero
7                  Numero = Numero+1
8          FinMientras
9  Fin
```

Con el ciclo Repita.

```
1  'Pseudocódigo numero naturales con repita
2
3  Inicio
4          Numero = 1
5          Repita
6                  Escribir Numero
7                  Numero = Numero+1
8          HastaQue numero>20
9  Fin
```

Ejercicio.

Calcular la suma de los números del 1 al 10.

Entrada: No existen datos de entrada.
Proceso: Realizar la suma de los números del 1 al 10, así:1+2+3+4+5+6+7+8+9+10.
Salida: Mostrar la suma.

Con el ciclo Para.

```
1   ‘ Pseudocódigo suma del 1 al 10.
2
3   Inicio
4           Declaracion Numero, Suma
5
6           Suma = 0
7           Para Numero = 1 Hasta 10 Incremento 1
8                   Suma = Suma + Numero
9           FinPara
10          Escribir Suma
11  Fin
```

Con el ciclo Mientras.

```
1   ' Pseudocódigo suma del 1 al 10.
2
3   Inicio
4           Declaracion Numero, Suma
5
6           Suma = 0
7           Numero=1
8           MientrasQue numero<=10
9                   Suma = Suma + Numero
10                  Numero = Numero +1
11          FinMientras
12          Escribir Suma
13  Fin
```

Con el ciclo Repita.

```
1   ' Pseudocódigo suma del 1 al 10.
2
3   Inicio
4           Declaracion Numero, Suma
5
6           Suma = 0
7           Numero=1
8           Repita
9                   Suma = Suma + Numero
10                  Numero = Numero +1
11          HastaQue Numero>10
12          Escribir Suma
13  Fin
```

Ejercicio.

Imprimir los números pares existentes entre 1 y 20.

Entrada: No existen datos de entrada.
Proceso: Imprimir los números pares.
Salida: Mostrar los números pares.

Con el ciclo Para.

```
1   'Pseudocódigo pares de 1 a 20.
2
3   Inicio
4           Declaracion Numero
5           Para Numero = 2 Hasta 20 Incremento 2
6                   Escribir Numero
7           FinPara
8   Fin
```

Con el ciclo mientras.

```
1    'Pseudocódigo pares de 1 a 20.
2
3    Inicio
4        Declaracion Numero
5
6        Numero=2
7        MientrasQue Numero<=20
8            Escribir Numero
9            Numero = Numero + 2
10       FinMientras
11   Fin
```

Con el ciclo repita.

```
1    'Pseudocódigo pares de 1 a 20.
2
3    Inicio
4        Declaracion Numero
5
6        Numero=2
7        Repita
8            Escribir Numero
9            Numero = Numero + 2
10       HastaQue numero >20
11   Fin
```

Ejercicio.

Imprimir los números contenidos en el rango N y M, donde N y M son leídos por teclado. El valor de N debe ser menor que M.

Entrada: Los contenidos de las variables N y M.
Proceso: Validar que N sea menor que M. Si es así realizar el ciclo entre el rango N y M, de lo contrario mostrar un mensaje que indica que N no es menor que M.
Salida: Mostrar los números entre N y M.

Con el ciclo Para.

```
1   ' Pseudocódigo números en el intervalo n Y m.
2
3   Inicio
4       Declaracion N, M, Numero
5       N=Leer("ingrese el valor de N")
6       M=Leer("ingrese el valor de M")
7
8       Si N<M Entonces
9           Para Numero = N Hasta M Incremento 1
10              Escribir Numero
11          FinPara
12      Sino
13          Escribir "El valor de N debe ser menor que M"
14      FinSi
15  Fin
```

Con el ciclo mientras.

```
1   ' Pseudocódigo números en el intervalo n Y m.
2
3   Inicio
4       Declaracion N, M, Numero
5       N=CEnteroCorto(Leer("ingrese el valor de N"))
6       M=CEnteroCorto(Leer("ingrese el valor de M"))
7
8       Si N<M Entonces
9           Numero = N
10          MientrasQue Numero<=M
11              Escribir Numero
12              Numero = Numero + 1
13          FinMientras
14      Sino
15          Escribir "El valor de N debe ser menor que M"
16      FinSi
17  Fin
18
```

Con el ciclo repita.

```
1     ' Pseudocódigo números en el intervalo n Y m.
2
3     Inicio
4           Declaracion N, M, Numero
5           N=CEnteroCorto(Leer("ingrese el valor de N"))
6           M=CEnteroCorto(Leer("ingrese el valor de M"))
7
8           Si N<M Entonces
9                 Numero = N
10                Repita
11                      Escribir Numero
12                      Numero = Numero + 1
13                HastaQue Numero > M
14          Sino
15                Escribir "El valor de N debe ser menor que M"
16          FinSi
17    Fin
```

Ejercicio.

Calcular los cuadrados de los números entre 7 y 14.

Entrada: No existen datos de entrada.
Proceso: Generar los números entre 7 y 14 y calcular cada uno de sus cuadrados.
Salida: Mostrar los cuadrados.

Con el ciclo Para.

```
1     ' Pseudocódigo cuadrados
2
3     Inicio
4        Declaracion cuadrado, numero
5        Para numero = 7 Hasta 14 Incremento 1
6           cuadrado = numero ^ 2
7           Escribir "El cuadrado de " & numero & " es " & cuadrado
8        FinPara
9     Fin
```

Con el ciclo mientras.

```
1    ' Pseudocódigo cuadrados
2
3    Inicio
4       Declaracion cuadrado, numero
5
6       numero = 7
7       MientrasQue numero <= 14
8             cuadrado = numero ^ 2
9             Escribir "El cuadrado de " & numero & " es " & cuadrado
10            numero = numero + 1
11       FinMientras
12   Fin
```

Con el ciclo repita.

```
1    ' Pseudocódigo cuadrados
2
3    Inicio
4             Declaracion cuadrado, numero
5
6             numero = 7
7             Repita
8               cuadrado = numero ^ 2
9               Escribir "El cuadrado de " & numero & " es " & cuadrado
10              numero = numero + 1
11            HastaQue numero>14
12   Fin
```

Ejercicio.

Calcular el factorial de un numero dado por teclado.

Entrada: el numero que se desea calcular el factorial.
Proceso: Generar los números entre 1 y el numero dado y calcular el factorial.
Salida: Mostrar el factorial.

Con el ciclo Para.

```
1     ' Pseudocódigo factorial
2
3     Inicio
4           Declaracion n, factorial, numero
5
6           factorial = 1
7           n=Leer("un numero")
8
9           Para numero = 1 Hasta n Incremento 1
10                factorial = factorial * numero
11          FinPara
12
13          Escribir "el factorial es " & factorial
14    Fin
```

Con el ciclo mientras.

```
1     ' Pseudocódigo factorial
2
3     Inicio
4           Declaracion n, factorial, numero
5
6           factorial = 1
7           n=CEnteroLargo(Leer("un numero"))
8
9           numero = 1
10          MientrasQue numero <= n
11                factorial = factorial * numero
12                numero=numero+1
13          FinMientras
14
15          Escribir "el factorial es " & factorial
16    Fin
```

Con el ciclo repita.

```
1    ' Pseudocódigo factorial
2
3    Inicio
4            Declaracion n, factorial, numero
5
6            factorial = 1
7            n=CEnteroLargo(Leer("un numero"))
8
9            numero = 1
10           Repita
11                   factorial = factorial * numero
12                   numero=numero+1
13           HastaQue numero > n
14
15           Escribir "el factorial es " & factorial
16   Fin
```

7.3.2. Contadores.

La otra alternativa de uso de los ciclos es como contadores, esto se entiende como la tarea de llevar simplemente la cuenta de cuantas acciones u operaciones se van a realizar. La tarea de contar es en la gran mayoría de los casos, independiente de las acciones que se efectúen dentro del cuerpo del ciclo.

Observe los siguientes ejemplos para entender la anterior conceptualización.

Ejercicio.

Leer 5 números por teclado y mostrar los números leídos.

Entrada: Una serie de números, almacenados en cada iteración en una variable llamada **valor**.
Proceso: Realizar la lectura de un numero dentro del cuerpo del ciclo para y después de leído mostrar su valor.
Salida: Mostrar los números leídos.

Con el ciclo Para.

```
1   ' Pseudocódigo lectura de 5 números.
2
3   Inicio
4           Declaracion contador, valor
5
6           Para contador = 1 Hasta 5
7                   valor = Leer("ingrese un numero")
8                   Escribir "el numero leído es " & valor
9           FinPara
10  Fin
```

Con el ciclo mientras.

```
1   ' Pseudocódigo lectura de 5 números.
2
3   Inicio
4           Declaracion contador, valor
5
6           contador = 1
7           MientrasQue contador <= 5
8                   valor = Leer("ingrese un numero")
9                   Escribir "el numero leido es " & valor
10                  contador = contador + 1
11          FinMientras
12  Fin
```

Con el ciclo repita.

```
1   ' Pseudocódigo lectura de 5 números.
2
3   Inicio
4           Declaracion contador, valor
5
6           contador = 1
7           Repita
8                   valor = Leer("ingrese un numero")
9                   Escribir "el numero leido es " & valor
10                  contador = contador + 1
11          HastaQue contador > 5
12  Fin
```

Ejercicio.

Leer N números por teclado y mostrar la suma de los números leídos.

Entrada: El valor inicial **N** que indica la cantidad de números a leer. También el numero que se lee a la vez, se almacenara en la variable **Numero**.
Proceso: Realizar la lectura de un numero dentro del cuerpo del ciclo para, cada numero leído se ira almacenando acumulativamente en la variable
Salida: Mostrar la suma.

Con el ciclo Para.

```
1   'Pseudocódigo suma de n números.
2
3   Inicio
4           Declaracion n, suma, contador
5
6           suma=0
7           n=Leer("cantidad de números")
8
9           Para contador = 1 Hasta n
10                  numero = Leer("ingrese un numero")
11                  suma = CEnteroCorto(suma) + numero
12          FinPara
13
14          Escribir "la suma es " & suma
15  Fin
```

Con el ciclo Mientras.

```
1   'Pseudocódigo suma de n números.
2
3   Inicio
4           Declaracion n, suma, contador
5
6           suma=0
7           n=CEnteroCorto(Leer("cantidad de números"))
8
9           contador = 1
```

```
10          MientrasQue contador<=n
11                  numero = Leer("ingrese un numero")
12                  suma = CEnteroCorto(suma) + numero
13                  contador = contador +1
14          FinMientras
15
16          Escribir "la suma es " & suma
17  Fin
```

Con el ciclo repita

```
1   'Pseudocódigo suma de n números.
2
3   Inicio
4           Declaracion n, suma, contador
5
6           suma=0
7           n=CEnteroCorto(Leer("cantidad de números"))
8
9           contador = 1
10          Repita
11                  numero = Leer("ingrese un numero")
12                  suma = CEnteroCorto(suma) + numero
13                  contador = contador +1
14          HastaQue contador > n
15
16          Escribir "la suma es " & suma
17  Fin
18
```

Ejercicio.

Leer N números por teclado y mostrar cantidad de números pares e impares.

Entrada: El valor inicial **N** que indica la cantidad de números a leer. También el numero que se lee a la vez, se almacenara en la variable **Numero**.
Proceso: Realizar la lectura de un numero dentro del cuerpo del ciclo para, cada numero leído se ira almacenando acumulativamente en la variable
Salida: Mostrar la suma.

Con el ciclo Para.

```
1   'Pseudocódigo contador de números pares e impares
2
3   Inicio
4       Declaracion n, suma
5       Declaracion contadorPar, contadorImPar
6
7       suma=0
8       contadorPar=0
9       contadorImPar=0
10      n=Leer("cantidad de números")
11
12      Para contador = 1 Hasta n
13          numero = Leer("ingrese un numero")
14
15          Si numero mod 2 =0 Entonces
16              contadorPar = contadorPar + 1
17          Sino
18              contadorImpar = contadorImpar + 1
19          FinSi
20      FinPara
21
22      Escribir "la cantidad de los pares es " & contadorPar
23      Escribir "la cantidad de los impares es " & contadorImPar
24  Fin
```

Con el ciclo mientras

```
1   'Pseudocódigo contador de números pares e impares
2
3   Inicio
4       Declaracion n, suma
5       Declaracion contadorPar, contadorImPar
6
7       suma=0
8       contadorPar=0
9       contadorImPar=0
10      n=CEnteroCorto(Leer("cantidad de números"))
11
12      contador = 1
13      MientrasQue contador <= n
14          numero = Leer("ingrese un numero")
15
16          Si numero mod 2 =0 Entonces
17              contadorPar = contadorPar + 1
18          Sino
19              contadorImpar = contadorImpar + 1
20          FinSi
21          contador = contador + 1
22      FinMientras
23
24      Escribir "la cantidad de los pares es " & contadorPar
25      Escribir "la cantidad de los impares es " & contadorImPar
26  Fin
```

Con el ciclo mientras

```
1   'Pseudocódigo contador de números pares e impares
2
3   Inicio
4       Declaracion n, suma
5       Declaracion contadorPar, contadorImPar
6
7       suma=0
8       contadorPar=0
9       contadorImPar=0
10      n=CEnteroCorto(Leer("cantidad de números"))
```

```
11              contador = 1
12              Repita
13                      numero = Leer("ingrese un numero")
14
15                      Si numero mod 2 =0 Entonces
16                              contadorPar = contadorPar + 1
17                      Sino
18                              contadorImpar = contadorImpar + 1
19                      FinSi
20                      contador = contador + 1
21              HastaQue contador > n
22
23              Escribir "la cantidad de los pares es " & contadorPar
24              Escribir "la cantidad de los impares es " & contadorImPar
25
26      Fin
```

Ejercicio.

Leer N notas de un estudiante y calcular el promedio, la nota máxima y la mínima.

Entrada: El valor inicial **N** que indica la cantidad de notas que van a ingresar. Dentro de un ciclo se van a leer n cantidad de notas.
Proceso: Realizar la lectura de un n y luego la cantidad de notas, cada una de estas se ira sumando en una variable suma. Paralelamente se calcula la nota mínima con una condición y otra para la máxima. El promedio es la suma de la notas sobre la cantidad de las mismas.
Salida: Mostrar el promedio, la nota mínima y máxima.

Con el ciclo Para.

```
1     ‘ Pseudocòdigo promedio de notas
2
3     Inicio
4       Declaracion n, promedio, minima, maxima
5       Declaracion nota, contador, suma
6
7       minima=5
8       maxima=0
9       suma=0
10      n=Leer("ingrese la cantidad de notas")
11
12      Para contador=1 Hasta n
13            nota=CRealCorto(Leer("ingrese la nota numero: " & contador))
14            suma = suma + nota
15
16            Si minima > nota Entonces
17                        minima=nota
18            FinSi
19
20            Si maxima < nota Entonces
21                        maxima=nota
22            FinSi
23
24      FinPara
25
26      promedio = suma / n
27
28      Escribir "el promedio es:" & promedio
29      Escribir "la mínima es:" & minima & " la máxima es:" & maxima
30
31    Fin
```

Con el ciclo mientras.

```
1     ' Pseudocòdigo promedio de notas
2
3     Inicio
4       Declaracion n, promedio, minima, maxima
5       Declaracion nota, contador, suma
6
7       minima=5
8       maxima=0
9       suma=0
10      n=CEnteroCorto(Leer("ingrese la cantidad de notas"))
11
12      contador=1
13      MientrasQue contador<=n
14          nota=CRealCorto(Leer("ingrese la nota numero: " & contador))
15          suma = suma + nota
16
17          Si minima > nota Entonces
18             minima=nota
19          FinSi
20
21          Si maxima < nota Entonces
22             maxima=nota
23          FinSi
24
25          contador= contador +1
26
27       FinMientras
28
29       promedio = suma / n
30
31       Escribir "el promedio es:" & promedio
32       Escribir "la mínima es:" & minima & " la máxima es:" & maxima
33
34    Fin
```

Con el ciclo repita.

```
1     ' Pseudocòdigo promedio de notas
2
3     Inicio
4       Declaracion n, promedio, minima, maxima
5       Declaracion nota, contador, suma
6
7       minima=5
8       maxima=0
9       suma=0
10      n=CEnteroCorto(Leer("ingrese la cantidad de notas"))
11
12      contador=1
13      Repita
14          nota=CRealCorto(Leer("ingrese la nota numero: " & contador))
15          suma = suma + nota
16
17          Si minima > nota Entonces
18             minima=nota
19          FinSi
20
21          Si maxima < nota Entonces
22             maxima=nota
23          FinSi
24
25          contador= contador +1
26
27       HastaQue contador>n
28
29       promedio = suma / n
30
31       Escribir "el promedio es:" & promedio
32       Escribir "la mínima es:" & minima & " la máxima es:" & maxima
33
34    Fin
```

Ejercicio.

Leer N notas de M estudiantes y calcular el promedio.

Entrada: El valor inicial **N** que indica la cantidad de notas que van a ingresar.
El valor de M que es la cantidad de estudiantes.
Dentro de un ciclo se van a leer n cantidad de notas.
Proceso: Realizar la lectura de un n y luego la de m. Seguidamente la cantidad de notas, cada una de estas se ira sumando en una variable suma. El promedio es la suma de la notas sobre la cantidad de las mismas.
Salida: Mostrar el promedio por estudiante

Con el ciclo Para.

```
1   ' Pseudocòdigo promedio de notas de M estudiantes
2   Inicio
3
4   Declaracion n, m, promedio, contadorNotas
5   Declaracion nota, contadorEstudiantes, suma
6
7   m=CEnteroCorto(Leer("ingrese la cantidad de estudiantes"))
8
9   Para contadorEstudiantes=1 Hasta m
10    suma=0
11
12    n = CEnteroCorto(Leer("ingrese la cantidad de notas"))
13
14   Para contadorNotas=1 Hasta n
15    nota=CRealCorto(Leer("ingrese la nota numero: "&contadorNotas))
16    suma = suma + nota
17   FinPara
18
19    promedio = suma / n
20    Escribir "el promedio del estudiante es:" & promedio
21
22  FinPara
23
24  Fin
```

Con el ciclo mientras.

```
1   ' Pseudocòdigo promedio de notas de M estudiantes
2
3   Inicio
4
5     Declaracion n, m, promedio, contadorNotas
6     Declaracion nota, contadorEstudiantes, suma
7
8     m=CEnteroCorto(Leer("ingrese la cantidad de estudiantes"))
9
10    contadorEstudiantes=1
11    MientrasQue contadorEstudiantes<=m
12        suma=0
13
14          n = CEnteroCorto(Leer("ingrese la cantidad de notas"))
15
16          contadorNotas=1
17          MientrasQue contadorNotas <=n
18                  nota=CRealCorto(Leer("ingrese la nota numero: " &
19          contadorNotas))
20                  suma = suma + nota
21                  contadorNotas = contadorNotas + 1
22          FinMientras
23
24          promedio = suma / n
25
26          Escribir "el promedio del estudiante es:" & promedio
27
28          contadorEstudiantes = contadorEstudiantes+1
29
30    FinMientras
31
32  Fin
```

Con el ciclo mientras.

```
1    ' Pseudocòdigo promedio de notas de M estudiantes
2
3    Inicio
4
5      Declaracion n, m, promedio, contadorNotas
6      Declaracion nota, contadorEstudiantes, suma
7
8      m=CEnteroCorto(Leer("ingrese la cantidad de estudiantes"))
9
10     contadorEstudiantes=1
11     Repita
12           suma=0
13
14              n = CEnteroCorto(Leer("ingrese la cantidad de notas"))
15
16              contadorNotas=1
17              Repita
18              nota=CRealCorto(Leer("ingrese la nota numero: " &
19              contadorNotas))
20              suma = suma + nota
21                 contadorNotas = contadorNotas + 1
22              HastaQue contadorNotas >n
23
24              promedio = suma / n
25
26              Escribir "el promedio del estudiante es:" & promedio
27
28              contadorEstudiantes = contadorEstudiantes+1
29
30     HastaQue contadorEstudiantes>m
31
32   Fin
```

RESUMEN.

- Las estructuras repetitivas permiten reproducir una serie instrucciones un numero finito o infinito de veces, según el objetivo del algoritmo.

- El ciclo para ... finpara: Esta estructura repetitiva esta diseñada para cuando se conoce el valor inicial (en donde empieza el ciclo) y el final (en donde termina).

- El ciclo mientrasque ... finmientras: Esta estructura repetitiva esta diseñada generalmente para cuando no se conoce cuantas iteraciones o veces se debe repetir el ciclo, empleándose también cuando se conocen la cantidad de iteraciones como en el ciclo anterior.

- La estructura repetitiva MientrasQue se diseño para cuando no se conoce cuantas iteraciones o veces se debe repetir el ciclo, se emplea también cuando se conocen la cantidad de iteraciones como en el ciclo anterior. La diferencia con la anterior esta en que por lo menos una vez se hace una iteración, ya que la condición de salida se evalúa al final del bloque de acciones a ejecutar.

- Las estructuras repetitivas se pueden comportar como generadores de números, los algoritmos en los cuales cuando se definen los parámetros que indican donde inicia, finaliza e incrementa el ciclo y permiten con esos números o sus rangos efectuar tareas de impresión, calculo, entre otros.

- La otra alternativa de uso de los ciclos es como contadores, esto se entiende como la tarea de llevar simplemente la cuenta de cuantas acciones u operaciones se van a realizar. La tarea de contar es en la gran mayoría de los casos, independiente de las acciones que se efectúen dentro del cuerpo del ciclo.

EJERCICIOS DE AUTO EVALUACIÓN.

Desarrolle la gran mayoría de Pseudocódigos anteriores de estructuras repetitivas para que verifique su correcto funcionamiento y entendimiento.

EJERCICIOS PROPUESTOS.

Desarrolle los siguientes algoritmos

1. Leer una lista de 10 valores enteros. Calcular e informar:
   a) La suma de los valores positivos.
   b) El producto de los valores negativos.
   (Ignorar los valores nulos)

2. Ingresar 5 juegos de cuatro valores cada uno. Calcular y emitir el promedio de cada juego.

3. Ingresar N juegos de cuatro valores cada uno. Calcular y emitir el promedio de cada juego. El proceso finaliza al encontrarse un juego cuyo primer valor es 0.

4. Ingresar dos números enteros positivos y calcular el producto de los mismos por sumas sucesivas.

5. Leer una lista de números positivos que finaliza en 0 y emitir el valor mínimo de la lista.

6. Leer una lista de números enteros que finaliza en 0 y emitir el valor máximo de la lista.

7. Ídem 5, emitiendo además la ubicación del máximo dentro de la lista. (Suponer un único máximo).

8. Leer 4 juegos de N valores enteros cada uno, donde N se informa al comienzo de cada juego, y emitir el valor máximo de cada grupo. (Suponer un único máximo).

9. Dada una lista de valores numéricos positivos, finalizada en 0, indicar si esta ordenada en forma ascendente.

10. Una empresa nos informa para cada uno de sus 20 vendedores:
código de vendedor : 3 dígitos
importe de ventas del mes : real

Se desea emitir el importe máximo de ventas del mes y cuántos vendedores alcanzaron dicho importe.

11. En una Central Telefónica se procesan los llamados realizados en la siguiente forma:

Por cada llamada se ingresa:

código de llamada : 3 dígitos (0 al finalizar el proceso)
tipo de día : .1. hábil, .2. feriado
duración de la llamada : entero > 0.
Siendo los importes Primeros 3. Minuto Adicional
Días hábiles a 10 a 2
Feriados a 15 a 3

Se deberá emitir:

a) El importe a abonar por cada llamada (código - importe).
b) La cantidad de llamadas que superen los 3.
c) El % de llamados que superan los 3. (sobre el total de llamadas informadas).

12. Se leen 30 valores enteros (comprendidos entre 5 y 40), que representan la temperatura máxima de cada uno de los días de un mes. Se pide hallar e informar:

- La temperatura máxima del mes y el día que se produjo. (Se supone única)
- Cuántos días la temperatura supero los 25º C.
- El promedio de las temperaturas máximas del mes.

13. Se ingresan los resultados de la evaluación de un curso de Programación; por cada alumno se informa:

número de matricula : 4 dígitos (1-9999)
asistencia : 1, presente; o, ausente
calificación : 2 dígitos (0-10).

A partir de esta información se debe calcular e informar:

a) Cantidad y % de alumnos presentes.
b) Promedio de calificaciones de alumnos presentes.
c) % de alumnos aprobados (sobre el total de alumnos presentes).

d) Numero de matricula del alumno de mayor calificación. (Si hay varios

14. Dados tres números decir si son los lados de un triángulo rectángulo

15. Programar las siguientes series e iterar hasta obtener una diferencia en el paso de un .δ. ingresado por el usuario.

a) La serie del Coseno
b) La serie del Seno
c) La serie de ex

16. Implementar el método de la bisección. ¿Qué precondiciones debe imponérsele?

17. Implementar el método de la secante. ¿Qué precondiciones debe imponérsele?

17. Compare los métodos del ejercicio 16 y 17. ¿Cuál es mejor y por que?

18. Realizar un algoritmo que calcule los numero primos anteriores a un numero .x. dado

19. Decidir si un número ingresado por el usuario es primo.

20. Implementar la serie de Fibonacci.

21. Modifique el ejercicio 9 y realice la prueba pero sin sumarle uno a cada paso. ¿por que se obtienen esos resultados? Explique teóricamente. Además compare los resultados del ejercicio 9 con los explicados en clase.

22. Implemente el problema de .cambio de moneda.: dar cambio de .x. pesos en monedas a un cliente utilizando la menor cantidad de monedas posibles.

23. Analice el ejercicio anterior: ¿el algoritmo siempre da la menor cantidad de monedas? ¿En que casos no lo hace?

24. Escriba un algoritmo que siempre de el cambio de la moneda, sin importar el valor de las mismas.

25. Escribir un programa que calcule la suma de los n primeros números naturales. Razonar si se puede implementar con los dos tipos de bucles.

26. Escribir un programa que calcule la suma de los cuadrados de los n primeros números naturales: 1 + 22 + 32 +… + n2.

27. Escribir un programa que calcule la suma de los números enteros de n a m (m>n).

28. Implementar un programa que calcule el producto de dos números enteros (*n***m*) haciendo sólo sumas.

29. Diseñar un programa que calcule el cociente y resto de la división entera de dos números mediante restas y sumas.

30. Los términos de la serie de Fibonacci se calculan así:
a1 = 1
a2 = 2
an = an-1 + an-2

    Diseñar un programa que calcule el n-ésimo término de la serie de Fibonacci.

31. Escribir un programa que calcule el cuadrado de un número haciendo sólo sumas. Ayuda: el cuadrado de un número *n* es la suma de los *n* primeros números impares. Ejemplo: 32=1+3+5=9.

32. Escribir un programa que calcule *an*, enésima potencia de *a*.

33. Escribir un programa que calcule el factorial de un número natural entero positivo n: n!= n * (n-1) * (n-2) * … * 2 * 1
Observar el buen funcionamiento de la función para n=0 y para n<0.

34. Escribir un programa que convierta un número entero en otro número entero que será el primero pero con las cifras que lo forman escritas al revés. Ejemplo: convertirá el número entero 1842 en el 2481.

8

La prueba de Escritorio

Plan general.

8.1. Introducción.
8.2. Reglas para usar una prueba de escritorio.
Resumen.

OBJETIVOS

- Presentar unos componentes básicos para el desarrollo de algoritmos mas dinámicos.
- Desarrollar ejercicios detallados para explicación de las estructuras repetitivas.
- Fortalecer los conceptos de algoritmos.
- Presentar herramientas para afrontar problemas computacionales más complejos, donde se necesite la ejecución de un grupo de instrucciones una cantidad de veces.

8.1. INTRODUCCIÓN

La prueba de escritorio permite conocer el funcionamiento de un algoritmo de manera completa. La prueba de escritorio hace un seguimiento instrucción por instrucción sobre cada uno de los elementos dentro de un algoritmo. Es una herramienta de análisis y validación de algoritmos muy fácil de usar y de aplicar a cada uno de los algoritmos, e incluso a codificación sobre un lenguaje de programación.

8.2. REGLAS PARA USAR UNA PRUEBA DE ESCRITORIO.

1. Escribir por columnas separadas a una buena distancia entre si, cada una de las variables empleadas en un algoritmo. También se incluyen las constantes, ya que estos elementos hacen parte de un algoritmo.

2. Realizar una por una las instrucciones del algoritmo, empezando por el elemento **Inicio** y finalizando en **Fin**. Este seguimiento de las instrucciones necesariamente tiene que hacerse de manera ordenada y siguiendo el flujo u orden de aparición de una instrucción.

3. Cada vez que se calcula o asigna un valor a una variable, debe escribirse debajo del anterior valor y subrayar o tachar la anterior, con el propósito de conocer cual es el contenido actual de la variable.

Para poner en practica los anteriores conceptos se retomaran algunos de los ejemplos realizados en capítulos anteriores.

Ejercicio.

Diseñar un algoritmo en el cuál se lea un número introducido por el usuario, y mostrar el valor del número previamente dado.

Entrada: Un valor numérico dado por el usuario. Se almacenará en una variable llamada **Numero**.
Proceso: no se necesita efectuar ningún cálculo ni acción.
Salida: mostrar el valor de la variable **Numero** que contiene el valor dado por el usuario.

```
1   Inicio
2       Declaración Numero
3       Numero = Leer ("Ingrese un valor numérico")
4       Escribir "El valor numérico es:" & Numero
5   Fin
```

PASOS PARA SU REALIZACIÓN.

1. Tome una hoja de papel y preferiblemente un lápiz.

2. (Paso 1. en las reglas.) En el algoritmo anterior solo existe una variable que se llama **Numero**, escriba entonces el nombre de la variable.

3. (Paso 2. en las reglas.) El Pseudocódigo está numerado cada una de las instrucciones.

    - **Instrucción 1**: pertenece a la instrucción **Inicio** y la cual indica que el algoritmo se va a empezar.
    - **Instrucción 2**: Se esta declarando una variable en memoria RAM llamada **Numero**.

- **Instrucción 3**: se muestra el siguiente mensaje por pantalla: **Ingrese un valor numérico**. Este mensaje indica al usuario que debe ingresar un numero desde teclado. Cuando el usuario digita un valor numérico por teclado, es almacenado en la variable llamada **Numero**. Suponiendo que el valor desde teclado por ejemplo es **24**, escríbalo debajo de la palabra **numero** escrito en la hoja.

- **Instrucción 4**: la salida contiene la siguiente instrucción:

  "**El valor numérico es:", Numero** ahora observe el valor escrito en su hoja y donde se encuentra la variable **Numero** reemplácelo. La instrucción sería entonces:
  "**El valor numérico es:",24** lo anterior corresponde al mensaje que se visualizará por pantalla.

4. Ya sé a realizado una sencilla prueba de escritorio, de esta misma manera se hace para cualquier algoritmo sin importar su tamaño o cantidad de instrucciones. Véase el siguiente ejemplo.

Ejercicio.

Diseñar un algoritmo que sume el valor de 15 a cierta cantidad introducida desde el teclado.

Entrada:	Un número o valor introducido por teclado almacenado en la variable **Num1**.
Proceso:	Realizar la suma entre el valor contenido en la variable **Num1** y 15. El resultado se acumulará en una variable llamada **Suma**; así: **Suma ← Num1 + 15**.
Salida:	Enseñar el valor de la suma, contenido en la variable **Suma**.

```
1   Inicio
2
3       Declaración Num1
4
5       Num1=Leer ("Ingrese un número")
6       Suma = 0
7       Suma = Num1 + 15
8       Escribir "El valor de la suma es " & Suma
9   Fin
```

PASOS PARA SU REALIZACIÓN.

1. Tome nuevamente otra hoja o sobre la anterior y lápiz.

2. (Paso 1. en las reglas.) En este algoritmo existen 2 variables llamadas **num1** y **Suma**. Escríbalas por columnas en la hoja a una distancia de separación.

3. (Paso 2. en las reglas.) realice cada una de las instrucciones de la siguiente manera:

    - **Instrucción 1**: Inicio del algoritmo.

    - **Instrucción 5**: se muestra el siguiente mensaje por pantalla: **Ingrese un número**. Este mensaje indica al usuario que debe ingresar un numero desde teclado.

    - **Instrucción 5**: cuando el usuario digita un valor numérico por teclado, es almacenado en la variable llamada **Num1**. Nuevamente suponiendo que el valor desde teclado es **56**, escríbalo abajo de la palabra N**um1** escrito en la hoja.

    - **Instrucción 6:** corresponde a la asignación de la variable **suma** en cero (**0**). Escriba el número cero debajo de la palabra **Suma**. Actualmente su hoja debe verse similar a la siguiente figura:

Num1	**Suma**
56	**0**

    - **Instrucción 7**: en esta instrucción se efectúa una operación aritmética entre el contenido de la variable **Num1** mas el valor de **15.** Reemplace ahora el valor de la variable **Num1** por el escrito en la hoja. Siempre se debe sustituir el nombre de las variables por su contenido (todas las variables ubicadas a la derecha de la asignación o el igual), excepto la que se encuentra antes del igual (parte izquierda).

      La instrucción sería similar a la siguiente: **Suma = 56 + 15**. efectuando la operación **Suma** seria igual a **71**. escriba entonces este valor debajo del anterior, tachando el valor anterior. Su hoja nuevamente se vería como la siguiente figura:

Num1	Suma
56	~~0~~
	71

- **Instrucción 8**: se mostrara por pantalla un mensaje mostrando el valor de la variable **Suma**. Reemplace el valor de esta variable en el mensaje y obtendrá ahora un nuevo mensaje con el siguiente texto '**El valor de la suma es, 71'.**

- **Instrucción 9**: se finaliza el algoritmo con el objeto **Fin**.

4. En este ejemplo se emplearon mas variables y mas instrucciones en las que se incluía una inicialización en cero de la variable **suma**.

Ejercicio.

Calcular el cuadrado y la raíz cuadrada de un número introducido por el teclado.

Entrada: El número leído desde el teclado. Se almacenará en una variable llamada **Numero**
Proceso: Almacenar el cuadrado en la variable **Ncuadrado** y la raíz en la variable **Nraiz** del valor contenido en la variable **Numero** .
Salida: Mostrar el cuadrado del número.

```
1   Inicio
2
3       Declaración Nrariz, Numero, NCuadrado
4
5       Numero = Leer("Digite el número")
6       Ncuadrado = Numero^2
7       Nraiz = Raiz2(Numero)
8
9       Escribir "El cuadrado es:" & NCuadrado
10      Escribir "La raíz cuadrada es:" & Nraiz
11
12  Fin
```

PASOS PARA SU REALIZACIÓN.

Escriba las variables para este ejemplo (**Numero, Ncuadrado, Nraiz**)

Inicie al siguiente de la siguiente manera:

- **Instrucción 1**: inicio del algoritmo.
- **Instrucción 5**: se muestra un mensaje por pantalla solicitando un numero desde teclado.
- **Instrucción 5**: el valor obtenido de teclado se almacenará en la variable **Numero**, presuma entonces que el valor es **76**. Realice dicha asignación en la hoja.
- **Instrucción 6**: se esta calculando el cuadrado del numero (**76**) con la siguiente instrucción **Ncuadrado = Numero^2**. cambiando por el contenido de las variables, se tiene: **Ncuadrado = 76^2**. Efectuando la operación **Ncuadrado** vale **5776**. realice la anotación en la hoja.
- **Instrucción 7**: se calcula el valor de la raíz cuadrada de **Numero** que contiene el valor de **76**. ejecutando la operación **Nariz** es igual a **8,71779788708135**, nuevamente escriba el valor.
- **Instrucción 9, 10**: se muestran por pantalla los valores del cuadrado y la raíz cuadrada calculados.
- **Instrucción 12**: fin del algoritmo.

La hoja de la prueba de escritorio debe verse similar a la siguiente:

Numero	**Ncuadrado**	**Nraiz**
76	**5776**	**8,7177978**

Los ejemplos explicativos muestran la forma como se realiza una prueba de escritorio típica, para conocer el comportamiento y funcionamiento de un algoritmo.

Es una tarea sumamente sencilla que debe ser tenida en cuenta en el momento de comprobar si una algoritmo efectúa una tarea como se planeo o ideo. Practique realizando un par de ejemplos de cualquiera de los capítulos anteriores, hágalos muy analíticamente y tómese su tiempo también.

RESUMEN.

- La prueba de escritorio permite conocer el funcionamiento de un algoritmo de manera completa.

- La prueba de escritorio hace un seguimiento instrucción por instrucción sobre cada uno de los elementos dentro de un algoritmo.

- Es una herramienta de análisis y validación de algoritmos muy fácil de usar y de aplicar a cada uno de los algoritmos, e incluso a codificación sobre un lenguaje de programación.

EJERCICIOS DE AUTO EVALUACIÓN Y EJERCICIOS PROPUESTOS.

Realice un corto recorrido a lo largo de los capítulos anteriores y seleccione 5 ejercicios de cada uno, realíceles la prueba de escritorio y confronte los resultados con los mismos valores de entrada.

9

Arreglos

Plan general.

OBJETIVOS

- Presentar una nueva estructura de datos a partir de los tipos de datos convencionales.
- Fortalecer el manejo de algoritmos, en especial el manejo de la estructuras repetitivas y condicionales.
- Desarrollar una variada gama de ejercicios explicativos que afiancen el concepto de arreglo.

9.1. INTRODUCCIÓN

Dada la complejidad y características de algunos problemas, se proporcionan nuevos tipos de datos a partir de los existentes vistos en capítulos iniciales. Los problemas computacionales requieren de espacios en memoria más amplios para almacenar datos de gran similitud.

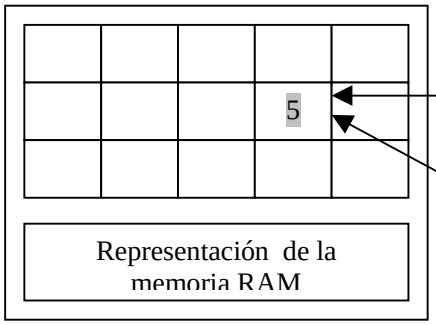
Los arreglos unidimensionales y bidimensionales, también llamados vectores y matrices respectivamente, proporcionan mayor capacidad en el almacenamiento de datos comparado con las variables usadas a lo largo del contenido.

9.2. ARREGLOS UNIDIMENSIONALES O VECTORES.

Retomando una explicación anterior cuando se estaban definiendo las variables, se explicaba que una variable es un espacio reservado en memoria que tiene un nombre característico y que tiene el espacio suficiente para almacenar un dato de cualquier tipo, como: números, cadenas, entre otros.

Para explicarlo de una manera gráfica observe el siguiente algoritmo.

```
1    'Algoritmo declaración de una variable.
2
3    Inicio
4            Declaracion Numero
5            Numero = 5
6            Escribir Numero
7    Fin
8
```

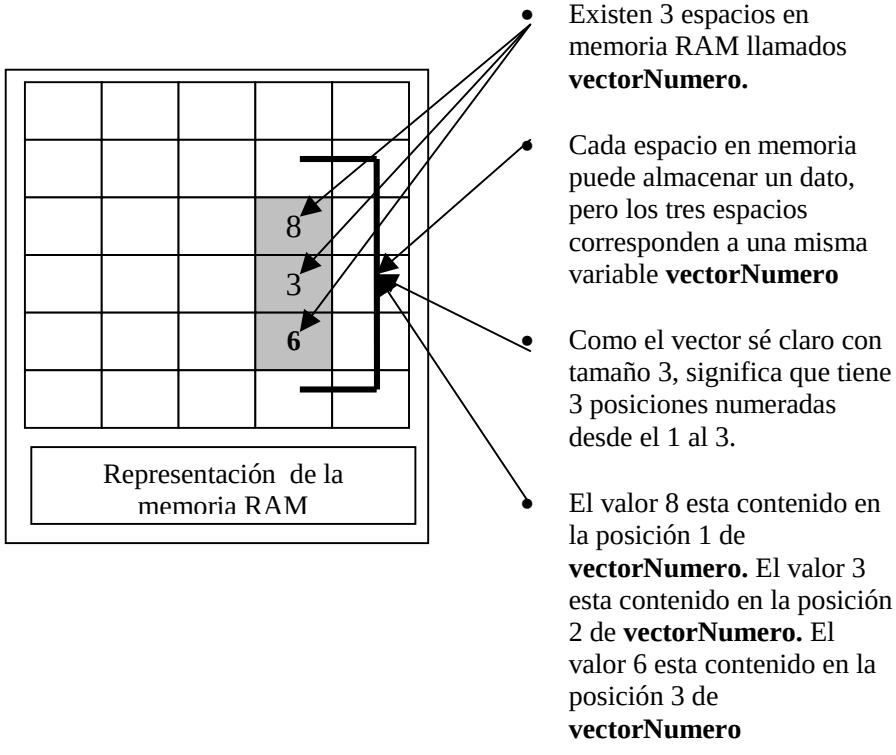


- Existe un espacio en memoria RAM llamado **Numero.**
- Que solo puede almacenar un dato para este caso es el valor 5.

Con los arreglos sucede algo especial, pueden almacenar mas datos pero en diferentes posiciones de memoria que atienden al nombre de la variable y a un índice o numero. Observe el siguiente ejemplo:

```
1    'Algoritmo Declaracion de un vector
2    Inicio
3            Declaracion vectorNumero(3)
4
5            vectorNumero(1)=8
6            vectorNumero(2)=3
7            vectorNumero(3)=6
8
9            Escribir vectorNumero(1) & " - " & vectorNumero(2) & " - "
10   & vectorNumero(3)
11   Fin
```

El resultado por pantalla es similar al siguiente:



La anterior explicación pertenece a un breve ejemplo acerca de los vectores o arreglos unidimensionales, a partir de aquí se comenzara a desarrollar la parte conceptual y algorítmica.

9.2.1. Reglas para nombrar vectores.

Estas reglas tienen muchas similitudes con las reglas al definir variables por que los vectores lo son también.

1. El nombre de una variable vector debe comenzar con una letra, seguida de otras letras, números o caracteres, seguido por un par de paréntesis **()** y en el interior un numero o variable con un dato de tipo numero entero. Como ejemplo:

   A(2), valor(6), num1(250),

   Siendo **pos** una variable con un valor de 8 entonces podría escribirse:

   Nombre(pos) vectorCiudades(pos+3)

2. El nombre de las variables vector nunca debe emplear caracteres especiales como:

   ¡,!, |, “, @, ·, #, $, %, /, (,), =, ‘, ¿, ?, +, -, *, {, }, [,], ^, entre otros.

   Es incorrecto nombrar:

   e-mail(2) cantidad_$(3) #1(24)

   Siendo **pos** una variable con un valor de 8 entonces:

   numero.1(pos) el%interes(pos) ∑Sumatoria(pos)

3. Cuando se nombre una variable vector, nunca emplee tildes, ni espacios, ni signos de puntuación. Por ejemplo:

   Usos incorrectos

   Código(4) Nombre Empleado(2) Cantidad.(pos)

   Usos correctos

   Codigo(4) NombreEmpleado(2) Cantidad(pos)

4. Los nombres de los vectores no pueden ser palabras reservadas.

9.2.2. Declaración de una variable de tipo vector.

El formato para la declaración de un vector es el siguiente:

```
Declaracion Nombre_vector(tamaño)
```

Por ejemplo:

```
Declaracion Numero(8)
```

El valor 8 representa el numero total de espacios en memoria que contendrá el vector llamado **Numero.**

Otra forma de declarar variables es también:

```
Declaración Nombre(6), Salario(12)
```

Donde se indica que ***Nombre*** y ***Salario*** son dos vectores independientes uno del otro, pero declaradas sobre la misma instrucción de ***Declaración*** y con tamaños diferentes o pueden ser iguales según se necesite.

9.2.3. Operaciones sobre vectores.

Sobre un vector se pueden desarrollar infinidades de acciones y representar enésimos sistemas, todo esta ligado a la creatividad y a la aplicación de los conceptos, pero todo se resume a estas 3 operaciones: declaración e inicialización, recorrido o desplazamiento y, asignación y obtención de valores.

Es de resaltar que como los vectores son variables que contienen posiciones que inician desde 1 hasta el tamaño que se da cuando se declara, es necesario el uso de las estructuras repetitivas para agilizar el proceso. En el siguiente punto se mostrara con mas detenimiento la importancia de ciclos para desplazarse dentro de un vector.

9.2.3.1. Declaración e inicialización.

En el numeral anterior se mostró la forma de cómo declarar un vector, inmediatamente de ser declarado es mejor por recomendación inicializar o dejar en cero en el caso de los vectores que almacenaran números, o en espacio vació los vectores de cadenas de caracteres.

El siguiente ejemplo hace la declaración de un vector de 5 posiciones y realiza la inicialización de cada posición.

```
1   ' Algoritmo Declaración e inicialización de un vector numérico
2   Inicio
3           ' así se declara un vector de 5 elementos
4           Declaracion vecNumero(5)
5
6           ' así se inicializa el valor de cero (0) a cada posición
7           vecNumero(1)=0
8           vecNumero(2)=0
9           vecNumero(3)=0
10          vecNumero(4)=0
11          vecNumero(5)=0
12  Fin
```

El programa que se muestra a continuación realiza la misma tarea que el anterior, pero ahora emplea estructuras repetitivas o ciclos. Como se conoce el valor inicial y el final, la estructura mas usada es el cicloPara. Como las posiciones tienen un comportamiento lineal, es decir, inician en la posicion 1 hasta n (la cantidad total de elementos) se emplean ciclos para hacer mas facil la labor de realizar operaciones sobre ellos.

```
1   ' Algoritmo Declaración e inicialización de un vector numérico
2   Inicio
3           ' así se declara un vector de 5 elementos
4           Declaracion vecNumero(5)
5
6           ' así se inicializa el valor de cero (0) a cada posición
7           ' haciendo uso de un ciclo
8           Para indice=1 Hasta 5
9                   vecNumero(indice)=0
10          FinPara
11  Fin
```

9.2.3.2. Recorrido.

Recorrer un vector es desplazarse por cada una de los espacios reservados de memoria que contiene la variable de tipo vector y obtener su valor.

```
1   ' Algoritmo recorrido
2    Inicio
3      ' así se declara un vector de 5 elementos
4      Declaración vecNumero(5)
5
6      ' así se inicializa O se asigna el valor de cero (0) a cada posición
7      Para índice =1 Hasta 5
8        vecNumero(indice)=0
9      FinPara
10
11     ' así se muestran por pantalla el contenido del
12     'vector posición por posición
13      Para índice = 1 Hasta 5
14            Escribir vecNumero(indice)
15      FinPara
16   Fin
```

```
1   ' Algoritmo recorrido
2    Inicio
3      Declaracion vecNumero(5)
4      Declaracion valor
5
6      ' así se inicializa O se asigna el valor de cero (0) a cada posición
7      Para índice =1 Hasta 5
8        vecNumero(indice)=0
9      FinPara
10
11     ' así se muestran por pantalla el contenido del
12     'vector posición por posición
13      Para índice = 1 Hasta 5
14            valor = vecNumero(indice)
15            escribir valor
16      FinPara
17   Fin
```

9.2.3.3. Asignación y obtención de valores.

```
1     ' Algoritmo asignación de valores e impresión
2     Inicio
3       ' así se declara un vector de 5 elementos
4       Declaración vecNumero(5)
5
6       ' así se inicializa O se asigna el valor de cero (0) a cada posición
7       ' haciendo uso de un ciclo
8       Para índice =1 Hasta 5
9         vecNumero(indice)=0
10      FinPara
11
12      ' así se asigna los datos leídos por teclado directamente al vector
13      Para indice = 1 Hasta 5
14        vecNumero(indice)=Leer("ingrese un valor”)
15      FinPara
16
17      ' así se muestran por pantalla el contenido del
18      'vector posición por posición
19       Para índice = 1 Hasta 5
20            Escribir vecNumero(indice)
21       FinPara
22    Fin
```

EJERCICIOS.

En esta sección se desarrollaran ejercicios de vectores con todas las estructuras repetitivas, vistas anteriormente.

Ejercicio.

Leer un vector de 10 elementos y calcular la suma de sus elementos.

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	Almacenar los números en el vector llamado **vecNumeros**, recorrer el vector e ir sumando sus valores en una variable denominada **suma**.
Salida:	Mostrar el valor de la suma.

El siguiente ejercicio se desarrollara con las tres estructuras repetitivas de forma separada, iniciando con el ciclo Para, luego con el MientrasQue y finalmente con la estructura Repita.

Ciclo Para.

```
1   ' Algoritmo lectura y suma de los elementos de un vector de 10
2   posiciones,
3
4   Inicio
5
6           'Declaracion del vector
7           Declaracion vecNumeros(10)
8
9           ' en este ciclo se leen los números desde teclado
1           Para indice = 1 Hasta 10
0                   vecNumeros(indice)=Leer("ingrese un numero")
11          FinPara
12
13          'este ciclo recorre el vector Y suma el valor de cada
14          'elemento
15          suma=0
16
17          Para indice =1 Hasta 10
18                  suma=suma + vecNumeros(indice)
19          FinPara
20
21          Escribir "la suma de los valores del vector es:" & suma
22  Fin
```

Ciclo Mientras

```
1   ' Algoritmo lectura Y suma de los elementos de un vector de 10
2   posiciones,
3
4   Inicio
5
6           'Declaracion del vector
7           Declaracion vecNumeros(10)
8
9           ' en este ciclo se leen los números desde teclado
10          indice=1
11
12          MientrasQue(indice<=10)
13                  vecNumeros(indice)=Leer("ingrese un numero")
14                  indice=indice+1
15          FinMientras
16
17          'este ciclo recorre el vector Y suma el valor de cada
18          'elemento
19          suma=0
20          indice=1
21
22          MientrasQue(indice<=10)
23                  suma=suma + vecNumeros(indice)
24                  indice=indice+1
25          FinMientras
26
27          Escribir "la suma de los valores del vector es:" & suma
28  Fin
```

Ciclo Repita

```
1   ' Algoritmo lectura Y suma de los elementos de un vector de 10
2   posiciones,
3
4   Inicio
5           'Declaracion del vector
6           Declaracion vecNumeros(10)
7
8           ' en este ciclo se leen los números desde teclado
9
10          indice=1
11          Repita
12                  vecNumeros(indice)=Leer("ingrese un numero")
13                  indice=indice+1
14          HastaQue(indice=11)
15
16          'este ciclo recorre el vector Y suma el valor de cada
17          'elemento
18          suma=0
19          indice=1
20
21          Repita
22                  suma=suma + vecNumeros(indice)
23                  indice=indice+1
24          HastaQue(indice=11)
25
26          Escribir "la suma de los valores del vector es:" & suma
27  Fin
```

Ejercicio.

Calcular independientemente la suma de los cubos y cuadrados de los elementos de un vector de 15 posiciones.

Entrada: Diez números que se ingresaran desde teclado.
Proceso: Almacenar los números en el vector llamado **vecNumeros**, recorrer el vector e ir sumando sus valores en una variable denominada **suma**.
Salida: Mostrar el valor de la suma.

Ciclo Repita

```
1   ' Algoritmo suma cubo y cuadrado de los elementos de un vector
2
3   Inicio
4           'declaracion del vector
5           Declaracion vecNumeros(15)
6
7           ' en este ciclo se leen los números desde teclado
8           indice=1
9           Repita
10              vecNumeros(indice)=Leer("ingrese un numero")
11              indice=indice+1
12          HastaQue(indice=16)
13
14          'este ciclo recorre el vector Y suma el valor de cada
15          'elemento
16          sumaCubo=0
17          sumaCuadrado=0
18          indice=1
19
20          Repita
21           sumaCubo=sumaCubo + vecNumeros(indice)^3
22           sumaCuadrado=sumaCuadrado+vecNumeros(indice)^2
23           indice=indice+1
24          HastaQue(indice=16)
25
26        Escribir "la suma de los cubos es:" & sumaCubo
27        Escribir "la suma de los cuadrados es:" & sumaCuadrado
28  Fin
```

Ciclo Para.

```
1     ' Algoritmo suma cubo Y cuadrado de los elementos de un vector
2
3     Inicio
4         'Declaracion del vector
5          Declaracion vecNumeros(15)
6
7         'en este ciclo se leen los números desde teclado Y a la vez se
8         'calculan los cudrados
9
1          sumaCubo=0
0          sumaCuadrado=0
11
12         Para indice=1 Hasta 15
13             vecNumeros(indice)=Leer("ingrese un numero")
14             sumaCubo=sumaCubo + vecNumeros(indice)^3
15             sumaCuadrado=sumaCuadrado + vecNumeros(indice)^2
16         FinPara
17
18         Escribir "la suma de los cubos es:" & sumaCubo
19         Escribir "la suma de los cuadrados es:" & sumaCuadrado
20    Fin
```

Ciclo mientras.

```
1     ' Algoritmo suma cubo Y cuadrado de los elementos de un vector
2
3     Inicio
4             'declaracion del vector
5             Declaracion vecNumeros(15)
6
7             ' en este ciclo se leen los números desde teclado
8             indice=1
9             MientrasQue indice<=15
10                vecNumeros(indice)=Leer("ingrese un numero")
11                indice=indice+1
12            FinMientras
13
```

```
14              'este ciclo recorre el vector Y suma el valor de cada
15              'elemento
16              sumaCubo=0
17              sumaCuadrado=0
18              indice=1
19
20              MientrasQue indice<=15
21               sumaCubo=sumaCubo + vecNumeros(indice)^3
22               sumaCuadrado=sumaCuadrado+vecNumeros(indice)^2
23               indice=indice+1
24              FinMientras
25
26          Escribir "la suma de los cubos es:" & sumaCubo
27          Escribir "la suma de los cuadrados es:" & sumaCuadrado
28    Fin
```

Ejercicio.

Dado un vector de 10 elementos, almacenar en otro la raíz cuadrada de cada elemento.

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	leer los números en el vector llamado **vecA**, recorrer el vector y a cada elemento calcularle la raiz cuadrada y guadarlo en cada posición del otro vector.
Salida:	Mostrar el valor de la suma.

Ciclo mientras.

```
1   ' Algoritmo raiz de un vector en otro
2
3   Inicio
4           'declaracion de los vectores
5           Declaracion vecA(10)
6           Declaracion vecB(10)
7
8           valorRaiz=0
9           ' este ciclo lee Y calcula el valor de la raiz
10          posicion=1
11
12          MientrasQue posicion<=10
13                  vecA(posicion)=Leer("ingrese un numero")
14
15                  valorRaiz= raiz2(vecA(posicion))
16                  vecB(posicion)=valorRaiz
17                  posicion=posicion+1
18          FinMientras
19
20          ' este ciclo imprime los resultados
21          posicion=1
22
23          MientrasQue posicion<=10
24              Escribir "vecA(" & posicion & ") = " &
25  vecA(posicion) & " - vecB(" & posicion & ") = " & vecB(posicion)
26              posicion=posicion+1
27          FinMientras
28
29  Fin
```

Ciclo mientras.

```
1   ' Algoritmo raiz de un vector en otro
2
3   Inicio
4           'declaracion de los vectores
5           Declaracion vecA(10)
6           Declaracion vecB(10)
7
8           valorRaiz=0
9           ' este ciclo lee Y calcula el valor de la raiz
10          posicion=1
11
12          Repita
13                  vecA(posicion)=Leer("ingrese un numero")
14
15                  valorRaiz= raiz2(vecA(posicion))
16                  vecB(posicion)=valorRaiz
17                  posicion=posicion+1
18          HastaQue posicion>10
19
20          ' este ciclo imprime los resultados
21          posicion=1
22
23          Repita
24                  Escribir "vecA(" & posicion & ") = " &
25  vecA(posicion) & " - vecB(" & posicion & ") = " & vecB(posicion)
26                  posicion=posicion+1
27          HastaQue posicion>10
28
29  Fin
```

Ciclo para.

```
1   ' Algoritmo raiz de un vector en otro
2
3   Inicio
4           'declaracion de los vectores
5           Declaracion vecA(10)
6           Declaracion vecB(10)
7
8           valorRaiz=0
9           ' este ciclo lee Y calcula el valor de la raiz
10          Para posicion=1 Hasta 10
11                  vecA(posicion)=Leer("ingrese un numero")
12
13                  valorRaiz= raiz2(vecA(posicion))
14                  vecB(posicion)=valorRaiz
15          FinPara
16
17          ' este ciclo imprime los resultados
18          Para posicion=1 Hasta 10
19                  Escribir "vecA(" & posicion & ") = " &
20                  vecA(posicion) & " - vecB(" & posicion & ") = "
21                  & vecB(posicion)
22          FinPara
23
24  Fin
```

Ejercicio.

Dados 2 vectores de 20 elementos cada uno calcular la resta y el producto. Para estas operaciones se emplearan 2 vectores adicionales.

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	leer los números en el vector llamado **vecA**, recorrer el vector y a cada elemento calcularle la raíz cuadrada y guardarlo en cada posición del otro vector.
Salida:	Mostrar el valor de la suma.

Combinación ciclo para y mientras.

```
1   ' Algoritmo producto Y diferencia de dos vectores
2
3   Inicio
4           'declaracion de los vectores
5           Declaracion vecA(20), vecB(20)
6           Declaracion vecProducto(20), vecResta(20)
7
8           posicion=1
9           MientrasQue(posicion<=3)
10              vecA(posicion)=Leer("ingrese un numero vector A")
11              vecB(posicion)=Leer("ingrese un numero vector B")
12
13              vecProducto(posicion)=vecA(posicion)*vecB(posicion)
14              vecResta(posicion)=vecA(posicion)-vecB(posicion)
15               posicion=posicion+1
16          FinMientras
17
18          ' este ciclo imprime los resultados
19
20           Para posicion=1 Hasta 3
21                  Escribir "vecProducto(" & posicion & ") = " &
22                  vecProducto(posicion) & " - vecResta(" &
23                  posicion & ") = " & vecResta(posicion)
24          FinPara
25  Fin
```

Combinación ciclo para y repita.

```
1   ' Algoritmo producto Y diferencia de dos vectores
2
3   Inicio
4           'declaracion de los vectores
5           Declaracion vecA(20), vecB(20)
6           Declaracion vecProducto(20), vecResta(20)
7
8           posicion=1
```

```
9               Repita
10                  vecA(posicion)=Leer("ingrese un numero vector A")
11
12                  vecB(posicion)=Leer("ingrese un numero vector B")
13
14                vecProducto(posicion)=vecA(posicion)*vecB(posicion)
15                  vecResta(posicion)=vecA(posicion)-vecB(posicion)
16                  posicion=posicion+1
17              HastaQue posicion>3
18
19              ' este ciclo imprime los resultados
20
21               Para posicion=1 Hasta 3
22                        Escribir "vecProducto(" & posicion & ") = " &
23    vecProducto(posicion) & " - vecResta(" & posicion & ") = " &
24    vecResta(posicion)
25              FinPara
26    Fin
```

Ejercicio.

Dado un vector de 5 elementos que representan las notas de un estudiante calcular el promedio de sus notas.

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	leer los números en el vector llamado **vecA**, recorrer el vector y a cada elemento calcularle la raiz cuadrada y guadarlo en cada posición del otro vector.
Salida:	Mostrar el valor de la suma.

```
1     ' Algoritmo producto Y diferencia de dos vectores
2
3     Inicio
4        'declaracion del vector
5        Declaracion vecNotas(5)
6
7         'inicialización de datos
8         suma=0
9         promedio=0
10
11        'lectura de datos
12         Para posicion=1 Hasta 5
13          vecNotas(posicion)=Leer("ingrese la nota numero"& posicion)
14           suma=suma+vecNotas(posicion)
15         FinPara
16
17        promedio=suma / 5
18
19         Escribir "el nota promedio es " & promedio
20
21    Fin
```

Ejercicio.

Dados 50 números por teclado almacenar en diferentes vectores lo siguiente:

- Los números pares.
- Los números impares
- Los números negativos.
- Y los menores de 79 y mayores de 17

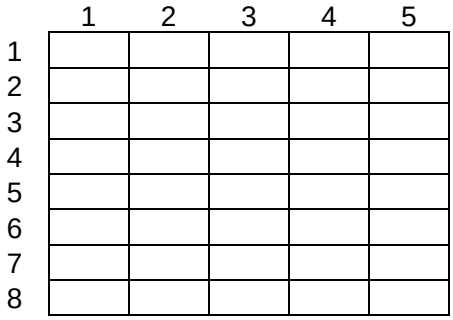
Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	leer los números en el vector llamado **vecA**, recorrer el vector y a cada elemento calcularle la raiz cuadrada y guadarlo en cada posición del otro vector.
Salida:	Mostrar el valor de la suma.

```
1   ' Algoritmo conjuntos de números con vectores
2
3   Inicio
4   Declaracion vecNumeros(10)
5   Declaracion vecPares(10), vecImpares(10)
6   Declaracion vecNegativos(10), vecRango(10)
7
8   indicePar=0
9   indiceImpar=0
10  indiceNegativo=0
11  indiceRango=0
12
13  Para indice=1 Hasta 10
14
15      vecNumeros(indice)=CEnteroCorto(Leer("ingrese un numero"))
16
17      Si (vecNumeros(indice) mod 2=0) entonces
18          indicePar=indicePar + 1
19          vecPares(indicePar)=vecNumeros(indice)
20      Sino
21          indiceImpar=indiceImpar + 1
22          vecImpares(indiceImpar)=vecNumeros(indice)
23      FinSi
24
25      Si (vecNumeros(indice) <=0) entonces
26          indiceNegativo=indiceNegativo + 1
27          vecNegativos(indiceNegativo)=vecNumeros(indice)
28      Sino
29
30          Si (vecNumeros(indice)>17 and vecNumeros(indice)<79)
            entonces
31                  indiceRango=indiceRango + 1
32                  vecRango(indiceRango)=vecNumeros(indice)
33          FinSi
34
35      FinSi
36
37  FinPara
38
```

```
39  ' impresión de los datos del vector de pares
40    resultado=""
41
42    Para indice=1 Hasta indicePar
43          Resultado = Resultado & "  " & vecPares(indice)
44     FinPara
45
46     Escribir "vector par :" & resultado
47
48    ' impresión de los datos del vector de impares
49     resultado=""
50
51     Para indice=1 Hasta indiceImpar
52          Resultado = Resultado & "  " & vecImpares(indice)
53     FinPara
54     Escribir "vector impar :" &resultado
55
56     ' impresión de los datos del vector de negativos
57     resultado=""
58     Para indice=1 Hasta indiceNegativo
59          Resultado = Resultado & "  " & vecNegativos(indice)
60  FinPara
61  Escribir "vector negativo :" & resultado
62
63   ' impresión de los datos del vector de rango
64  resultado=""
65  Para indice=1 Hasta indiceRango
66          Resultado = Resultado & "  " & vecRango(indice)
67  FinPara
68  Escribir "vector rango :" &resultado
69
    Fin
```

9.3. ARREGLOS BIDIMENSIONALES O MATRICES.

Al igual que los arreglos unidimensionales o vectores, las matrices son variables que contienen cierta cantidad de espacios en memoria definidos por un numero de filas y columnas. La forma que tienen las matrices es similar a la siguiente:



Este gráfico representa una matriz de 8 **filas** 5 **columnas**, es decir existen 40 espacios reservados en memoria que contendrá esta variable de tipo matriz. Al igual que los vectores, los índices o posiciones de las matrices tanto en las filas como en las columnas empiezan en 1 hasta el tamaño definido.

9.3.1. Reglas para nombrar matrices.

Estas reglas son semejantes a las de los vectores, la diferencia esta en la aparición del segundo índice o posición.

1. El nombre de una variable matriz debe comenzar con una letra, seguida de otras letras, números o caracteres, seguido por un par de paréntesis **()** y en el interior dos números enteros separados por coma (**,**) que representan el numero de la fila el primero y el numero de la columna el segundo. Como ejemplo:

A(2,3) valor(6,1) num1(250,5)
Siendo **fila y columna** variables con valores 5 y 2 respectivamente entonces podría escribirse:
Nombre(fila, columna) vectorCiudades(fila + 3, columna)

2. El nombre de las variables matriz nunca debe emplear caracteres especiales como:

¡,!, |, “, @, ·, #, $, %, /, (,), =, ‘, ¿, ?, +, -, *, {, }, [,], ^, entre otros.

Es incorrecto nombrar:

e-mail(2,1) cantidad_$(3,6) #1(24,20)

Siendo **fila y columna** variables con valores 5 y 2 respectivamente entonces podría escribirse:

numero.1(fila, columna) el%interes(fila, columna)

∑Sumatoria(fila, columna)

3. Cuando se nombre una matriz, nunca emplee tildes, ni espacios, ni signos de puntuación. Por ejemplo:

Usos incorrectos

Código(4, 3) Nombre Empleado(2,8)
Cantidad.(fila, columna)

Usos correctos

Codigo(4, 3) NombreEmpleado(2, 8)
Cantidad(fila, columna)

4. Los nombres de las matrices no pueden ser palabras reservadas.

9.3.2. Declaración de una variable de tipo matriz.

El formato para la declaración de una matriz es el siguiente:

```
Declaracion Nombre_matriz(numeroFilas, numeroColumnas)
```

Por ejemplo:

```
Declaracion Numero(8, 6)
```

El valor 8 representa el numero total filas y 6 el numero total de columnas, para un total de 48 espacios en memoria que contendrá la matriz llamada **Numero.**

Otra forma de declarar variables es también:

```
Declaracion Nombre(10, 5), Salario(12, 5)
```

Donde se indica que ***Nombre*** y ***Salario*** son dos matrices independientes una de la otra, pero declaradas sobre la misma instrucción de ***Declaración*** y con tamaños diferentes o pueden ser iguales según se necesite.

9.3.3. Operaciones sobre matrices.

Las operaciones que se desarrollan sobre matrices son totalmente iguales que en los vectores, y con en el manejo de posiciones se pueden implementar algoritmos aun más complejos y funcionales.

Cuando se trabajaron los vectores se hizo necesario el manejo de ciclos para recorrerlos, en este caso como se tienen 2 coordenadas o posiciones es aun más necesario el manejo de ciclos, uno para recorrer las filas y el otro las columnas de formas

9.3.3.1. Declaración ,inicialización y recorrido.

Como una matriz podría almacenar mayor cantidad de datos es indispensable inicializar la matriz para garantizar la calidad de los datos que se guardan allí.

El siguiente ejemplo hace la declaración de una matriz cuadrada de orden 2, es decir 2 filas por 2 columnas.

En este ejercicio se inicializa cada posición de la matriz de forma manual, es decir se defina cada posición con la constante de cero.

```
1   ' Algoritmo Declaración e inicialización de una matriz numérica
2   Inicio
3           ' así se declara una matriz de 2 por 2
4           Declaracion matNumero(2,2)
5
6           ' así se asigna el valor de cero (0) a cada posición
7           matNumero(1, 1)=0
8           matNumero(1, 2)=0
9           matNumero(2, 1)=0
10          matNumero(2, 2)=0
11
12  Fin
```

```
1   ' Algoritmo Declaración e inicialización de una matriz numérica
2   Inicio
3           ' así se declara una matriz de 2 por 2
4           Declaracion matNumero(2,2)
5
6           ' así se asigna el valor de cero (0) a cada posición
7           Para posFila =1 Hasta 2
8                   Para posColumna =1 Hasta 2
9                           matNumero(posFila, posColumna)=0
10                  FinPara
11          FinPara
12
13  Fin
```

9.3.3.2. Asignación y obtención de valores.

```
1   ' Algoritmo asignación de valores e impresión
2   Inicio
3           ' así se declara una matriz de 2 por 2
4           Declaracion matNumero(2,2)
5
6           ' así se asigna el valor de cero (0) a cada posición
7           Para posFila =1 Hasta 2
8                   Para posColumna =1 Hasta 2
9                           matNumero(posFila, posColumna)=0
10                  FinPara
11          FinPara
12
13          ' así se asignan los valores desde el teclado a cada posición
14          Para posFila =1 Hasta 2
15                  Para posColumna =1 Hasta 2
16                          matNumero(posFila,
17  posColumna)=Leer("ingrese un valor a la fila:" & posFila & "
18  columna:" & posColumna)
19                  FinPara
20          FinPara
21
22          ' así se muestran por pantalla el contenido del vector
23  posición por posición
24          Para posFila =1 Hasta 2
25                  Para posColumna =1 Hasta 2
26                          Escribir "el contenido de la fila:" &
27  posFila & " columna:" & posColumna & " es:" &
28  matNumero(posFila, posColumna)
29                  FinPara
30          FinPara
31
32  Fin
```

EJERCICIOS.

En esta sección se desarrollaran ejercicios de aplicación de matrices.

Ejercicio.

Leer una matriz de 3 filas por 4 columnas y calcular la suma de sus elementos.

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	Almacenar los números en el vector llamado **vecNumeros**, recorrer el vector e ir sumando sus valores en una variable denominada **suma**.
Salida:	Mostrar el valor de la suma.

```
1   ' Algoritmo suma de elementos de una matriz de 3 x 4
2
3   Inicio
4      Declaracion matNumero(3,4)
5
6     'Lectura de datos
7      Para numFila =1 Hasta 3
8            Para numColumna = 1 Hasta 4
9                              matNumero(numFila, numColumna) =
10                             CEnteroCorto(Leer("ingrese el valor
11                             mat(" & numFila & "," & numColumna
12                             & ")" ))
13           FinPara
14     FinPara
15
16     ' recorrido Y suma de los elementos
17     suma=0
18     Para numFila =1 Hasta 3
19           Para numColumna = 1 Hasta 4
20             suma = suma + matNumero(numFila, numColumna)
21           FinPara
22     FinPara
23     Escribir "la suma de los elementos es:" & suma
24  Fin
```

Ejercicio.

Leer una matriz cuadrada de orden 5 y almacenar en otras matrices el cubo y el cuadrado

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	Almacenar los números en el vector llamado **vecNumeros**, recorrer el vector e ir sumando sus valores en una variable denominada **suma**.
Salida:	Mostrar el valor de la suma.

```
1   ' Algoritmo cuadrado, cubo de una matriz cuadrada de orden 5
2   Inicio
3     Declaracion matNumero(5,5), matCubo(5,5), matCuadrado(5,5)
4
5     'Lectura de datos
6     Para numFila =1 Hasta 5
7           Para numColumna = 1 Hasta 5
8                 matNumero(numFila, numColumna) =
9   CEnteroCorto(Leer("ingrese el valor mat(" & numFila & "," &
10  numColumna & ")" ))
11          FinPara
12    FinPara
13
14    ' recorrido Y suma de los elementos
15    Para numFila =1 Hasta 5
16          Para numColumna = 1 Hasta 5
17             matCubo(numFila, numColumna) =
18  matNumero(numFila, numColumna) ^ 3
19
20  matCuadrado(numFila,numColumna)=matNumero(numFila,numC
21  olumna)^2
22          FinPara
23    Fin
24     'Impresión de la matriz cubo
25     resultado=""
26     Para numFila =1 Hasta 5
27          resultado = resultado & vbnewLine
28          Para numColumna = 1 Hasta 5
29               resultado = resultado & " - " & matCubo(numFila,
30  numColumna)
```

```
31
32              FinPara
33          FinPara
34
35          Escribir "matriz cubo" & resultado
36
37          ' Impresión de la matriz cuadrado
38          resultado=""
39          Para numFila =1 Hasta 5
40              resultado = resultado & vbnewLine
41              Para numColumna = 1 Hasta 5
42                  resultado = resultado & " - " & matCuadrado(numFila,
43  numColumna)
44              FinPara
45          FinPara
46          Escribir "matriz cuadrado" & resultado
47  Fin
```

Ejercicio.

Leer una matriz de 4 filas por 5 columnas, almacenar en una nueva la raiz cuadrada de sus elementos

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	Almacenar los números en el vector llamado **vecNumeros**, recorrer el vector e ir sumando sus valores en una variable denominada **suma**.
Salida:	Mostrar el valor de la suma.

```
1          ' Algoritmo raiz cuadrada matriz 4 x 5
2   Inicio
3     Declaracion matNumero(4,5)
4     Declaracion matRaiz(4,5)
5
6     ' Lectura de datos
7     Para numFila =1 Hasta 4
8           Para numColumna = 1 Hasta 5
9                 matNumero(numFila, numColumna) =
10  CEnteroCorto(Leer("ingrese el valor mat(" & numFila & "," &
11  numColumna & ")" ))
12          FinPara
13    FinPara
14
15    ' recorrido Y suma de los elementos
16    Para numFila =1 Hasta 4
17      Para numColumna = 1 Hasta 5
18            matRaiz(numFila, numColumna) =
19  raiz2(matNumero(numFila, numColumna))
20      FinPara
21    FinPara
22
23    ' Impresión de la matriz raiz
24    resultado=""
25    Para numFila =1 Hasta 4
26          resultado = resultado & vbnewLine
27          Para numColumna = 1 Hasta 5
28                resultado = resultado & " - " & matRaiz(numFila,
29  numColumna)
30          FinPara
31    FinPara
32    Escribir "matriz raiz" & resultado
33  Fin
34
```

Ejercicio.

Dada una matriz de 10 x 5 que representa las notas de los estudiantes de grupo, donde las filas corresponden a los estudiantes y las columnas a las materias, calcular en un vector las notas definitivas por estudiante y mostrarlas.

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	Almacenar los números en el vector llamado **vecNumeros**, recorrer el vector e ir sumando sus valores en una variable denominada **suma**.
Salida:	Mostrar el valor de la suma.

```
1               ' Algoritmo notas estudiantes
2       Inicio
3          Declaracion matNotas(10,5)
4          Declaracion vecDefinitiva(10)
5
6          ' Lectura de datos
7          Para numFila =1 Hasta 10
8                  Para numColumna = 1 Hasta 5
9                          matNotas(numFila, numColumna) =
10      CRealCorto(Leer("ingrese el valor mat(" & numFila & "," &
11      numColumna & ")" ))
12                 FinPara
13         FinPara
14
15         ' recorrido Y suma de los elementos
16         Para numFila =1 Hasta 10
17                 suma=0
18                 promedio=0
19                 Para numColumna = 1 Hasta 5
20                         suma = suma + matNotas(numFila, numColumna)
21                 FinPara
22                 promedio=suma / 5
23                 vecDefinitivas(numFila)=promedio
24         FinPara
25
```

```
26
27      ' Impresión del vector
28      resultado=""
29      Para numFila =1 Hasta 10
30              resultado = resultado & vecDefinitivas(numFila) &
31  vbnewLine
32      FinPara
33
34     Escribir "vector resultado" & resultado
35  Fin
```

Ejercicio.

Dados 50 números por teclado almacenar en diferentes vectores lo siguiente:

- Los números pares.
- Los números impares
- Los números negativos.
- Y los menores de 79 y mayores de 17

Entrada:	Diez números que se ingresaran desde teclado.
Proceso:	leer los números en el vector llamado **vecA**, recorrer el vector y a cada elemento calcularle la raiz cuadrada y guadarlo en cada posición del otro vector.
Salida:	Mostrar el valor de la suma.

```
1   ' Algoritmo conjuntos de números con vectores Y matriz de datos
2
3   Inicio
4
5           Declaracion matNumeros(9, 4)
6           Declaracion vecPares(36), vecImpares(36)
7           Declaracion vecNegativos(36), vecRango(36)
8
9           indicePar=0
10          indiceImpar=0
11          indiceNegativo=0
12          indiceRango=0
13
14          Para fila=1 Hasta 9
15            Para columna=1 Hasta 4
16                matNumeros(fila,
17  columna)=CEnteroCorto(Leer("ingrese un numero"))
18
19                Si (matNumeros(fila, columna) mod 2=0) entonces
20                        indicePar=indicePar + 1
21                        vecPares(indicePar)=vecNumeros(indice)
22                Sino
23                        indiceImpar=indiceImpar + 1
24
25          vecImpares(indiceImpar)=vecNumeros(indice)
26                FinSi
27
28                  Si (matNumeros(fila, columna) <=0) entonces
29                        indiceNegativo=indiceNegativo + 1
30
31          vecNegativos(indiceNegativo)=vecNumeros(indice)
32                 Sino
33                        Si (matNumeros(fila, columna) >17 and
34  matNumeros(fila, columna)<79 ) entonces
35                                indiceRango=indiceRango + 1
36
37          vecRango(indiceRango)=vecNumeros(indice)
38                        FinSi
39                 FinSi
40              FinPara
```

```
41
42              ' impresión de los datos del vector de pares
43              resultado=""
44              Para indice=1 Hasta indicePar
45                      Resultado = Resultado & "  " & vecPares(indice)
46              FinPara
47              Escribir "vector par :" & resultado
48
49              ' impresión de los datos del vector de impares
50              resultado=""
51              Para indice=1 Hasta indiceImpar
52                      Resultado = Resultado & "  " &
53      vecImpares(indice)
54              FinPara
55              Escribir "vector impar :" &resultado
56
57              ' impresión de los datos del vector de negativos
58              resultado=""
59              Para indice=1 Hasta indiceNegativo
60                      Resultado = Resultado & "  " &
61      vecNegativos(indice)
62              FinPara
63              Escribir "vector negativo :" & resultado
64
65              ' impresión de los datos del vector de rango
66              resultado=""
67              Para indice=1 Hasta indiceRango
68                      Resultado = Resultado & "  " & vecRango(indice)
69              FinPara
70              Escribir "vector rango :" &resultado
70
72      Fin
```

9.4. MÉTODOS DE ORDENAMIENTO.

Los vectores y matrices permiten almacenar datos para ser manipulados mas fácilmente, pero existen circunstancias en las cuales es necesario mantener los datos en un orden para su posterior uso. Un directorio telefónico podría considerarse una gran matriz por ejemplo, donde se almacenen los datos del dueño de la línea, numero telefónico y dirección de la residencia.

	1	2	3	4
1	Pérez	Carlos	3125545	cra 35 # 45
2	Medina	Kelly	3127441	Cll 89 a 14
3	Orrego	Viky	3181241	cra12 # 10-12
4				
,,,				
N	Lavoe	Hector	6512423	Cll 11 a 143
n +1				
n+ 2				

Un tipo de orden para este ejemplo es alfabético por el apellido de la A – Z. O que tal que se quisiera ordenar por numero telefónico para conocer cuantos líneas telefónicas pertenecen a un sector, se tendría una cierta cantidad de combinaciones para este caso.

Partiendo de lo anterior se van a conocer los métodos de ordenamiento aplicables a cualquier tipo de arreglo.

9.4.1. Método burbuja.

Este es el algoritmo más sencillo probablemente. Ideal para empezar, consiste en comparar repetidamente a través del vector a la matriz, comparando elementos adyacentes de dos en dos. Si un elemento es mayor que el que está en la siguiente posición se intercambian.

Algoritmo base del método burbuja.

```
1     para i=1 hasta  TAM1
2           Para j=0 hasta (TAM – 1)
3                 si (Vec[j] > Vec [j+1])
4                       temp = Vec [j]
5                       Vec [j] = Vec [j+1]
6                       Vec [j+1] = temp
7                 Finsi
8           FinPara
9     FinPara
10
```

```
1     ' Algoritmo ordenamiento burbuja.
2
3     Inicio
4           Declaracion vecNumero(10)
5
6           Para i=1 Hasta 10
7           vecNumero(i)=CEnteroCorto(Leer("un numero"))
8           FinPara
9
10          Para i=1 Hasta 9
11                Para j=i+1 Hasta 10
12                  Si (vecNumero(i) > vecNumero(j)) entonces
13                              temp=vecNumero(j)
14                              vecNumero(j)=vecNumero(i)
15                              vecNumero(i)=temp
16                    FinSi
17                FinPara
18          FinPara
19
20          Para i=1 Hasta 10
21                Escribir vecNumero(i)
22          FinPara
23    Fin
```

9.4.2. Método por selección.

Este algoritmo también es sencillo. Consiste en lo siguiente:

- Buscar el elemento más pequeño de la lista.
- Se intercambia con el elemento ubicado en la primera posición de la lista.
- Buscar el segundo elemento más pequeño de la lista.
- Se intercambia con el elemento que ocupa la segunda posición en la lista.
- Se Repite este proceso hasta que se haya ordenado toda la lista.

Algoritmo base del método por selección.

```
1       Para i=0 hasta (TAM – 1)
2               pos_men = Menor(vec, TAM, i);
3               temp = vec [i];
4               vec [i] = vec [pos_men];
5               vec [pos_men] = temp;
6       FinPara
7
```

9.4.3. Método por inserción.

Este algoritmo también es bastante sencillo. ¿Ha jugado cartas?. ¿Cómo las va ordenando cuando las recibe? Se hace de la siguiente manera: se toma la primera y la coloco en la mano. Luego se toma la segunda y se compara con la que se tiene: si es mayor, se pone a la derecha, y si es menor a la izquierda. Después se toma la tercera y se compara con las que se tienen en la mano, desplazándola hasta que quede en su posición final. Se continúa haciendo esto, insertando cada carta en la posición que le corresponde, hasta que tienen todas en orden.

Para simular esto en un programa se necesita tener en cuenta algo: no se puede desplazar los elementos así como así o se perderá un elemento. Lo que se hace es guardar una copia del elemento actual (que sería como la carta que se toma) y desplazar todos los elementos mayores hacia la derecha. Luego se copia el elemento guardado en la posición del último elemento que se desplazó.

Algoritmo base del Método por inserción.

```
1    Para i=1 hasta TAM
2        temp = vec[i]
3        j = i - 1
4        MientrasQue((vec[j]>temp)And (j>= 0) )
5            vec[j+1] = vec[j]
6            j=j-1
7        FinMientras
8        vec[j+1] = temp
9    FinPara
```

RESUMEN.

- Los arreglos unidimensionales y bidimensionales, también llamados vectores y matrices respectivamente, proporcionan mayor capacidad en el almacenamiento de datos comparado con las variables comúnmente usadas.

- Un vector es una cantidad de espacios en memoria que se acceden a través del nombre del vector y la posición.

- Las matrices son variables que contienen cierta cantidad de espacios en memoria definidos por un numero de filas y columnas.

- Los vectores y las matrices permiten almacenar datos para ser manipulados mas fácilmente, pero existen circunstancias en las cuales es necesario mantener los datos en un orden para su posterior uso.

- Método burbuja. Este es el algoritmo más sencillo probablemente. Ideal para empezar, consiste en comparar repetidamente a través del vector a la matriz, comparando elementos adyacentes de dos en dos.

- Método por inserción. Este algoritmo también es bastante sencillo. Por que los números se ordenan como un juego de cartas

- Método por selección. Este algoritmo también es sencillo. Consiste en lo siguiente:

  - Buscar el elemento más pequeño de la lista.
  - Se intercambia con el elemento ubicado en la primera posición de la lista.
  - Buscar el segundo elemento más pequeño de la lista.
  - Se intercambia con el elemento que ocupa la segunda posición en la lista.
  - Se Repite este proceso hasta que se haya ordenado toda la lista.

EJERCICIOS DE AUTO EVALUACIÓN.

Desarrolle la gran mayoría de Pseudocódigos anteriores de vectores y matrices e implemente los métodos de ordenamiento en un algoritmos completos para ver su funcionamiento con datos reales.

EJERCICIOS PROPUESTOS.

Desarrolle los siguientes algoritmos

1. Realizar un programa que lea una serie de números enteros con valores comprendidos entre 0 y 1000, hasta que nos introduzcan un número entero que no esté comprendido entre esos valores. El programa debe escribir después, los valores entre 0 y 1000 que el usuario había introducido. Utilizar un vector.

2. Realizar un programa que lea una serie de números enteros con valores comprendidos entre 0 y 1000 y entre -1000 y -1, hasta que nos introduzcan un número entero que no esté comprendido entre esos valores. El programa debe escribir después, los valores entre -1000 y 1000 que el usuario había introducido.

3. Lo mismo que el Problema 2 pero indicando además, cuantas veces el usuario dijo cada valor (frecuencia).

4. Leer n números enteros por teclado, adicionalmente leer dos números. Determinar si los dos números forman parte de la secuencia de números inicial leídas. La secuencia de números finaliza en -1.

5. Conocida una secuencia de números enteros positivos finalizada en -1 (fin de secuencia), desarrolle un programa que determine:

- ¿Cuántos de esos números son pares?
- ¿Cuál es el valor del número máximo?
- ¿Cuál es el valor del número mínimo?

6. Escribir un programa que lea diez números, los guarde en un vector y a continuación los imprima en orden inverso al de su entrada.

7. Escribir un programa que lea tres números y los guarde en un vector. A continuación los ordenará y guardará los valores ordenados en otro vector. Finalmente sacará ambas listas de números por la pantalla.

8. Repetir el ejercicio anterior con un número cualquiera de valores.

10

Funciones

Plan general.

OBJETIVOS

- Presentar una nueva estructura de datos a partir de los tipos de datos convencionales.
- Fortalecer el manejo de algoritmos, en especial el manejo de las estructuras repetitivas y condicionales.
- Desarrollar una variada gama de ejercicios explicativos que afiancen el concepto de arreglo.

10.1. INTRODUCCIÓN

En él capitulo cinco se conoció una serie de elementos fijos por un lenguaje de programación para desarrollar tareas especificas como: el calculo de la raíz cuadrada, el seno, el coseno, la fecha del sistema, entre otras, denominadas funciones establecidas. Estas permiten acelerar el proceso de desarrollo de programas de computo, siendo usadas tantas veces como sea necesario. En este capitulo se va a tratar el desarrollo de otras funciones definidas por el programador, adicionales y complementarias a las existentes.

10.2. DEFINICIÓN.

Una función es un programa que realiza una tarea especifica, por ejemplo la función que calcula la raíz cuadrada de un numero, solo obtiene valores de raíces cuadradas, no puede resolver otro calculo matemático. Las funciones son especializadas, por tanto solo hacen una sola tarea. Una función se considera también como un subprograma que puede ser llamado por otros subprogramas o desde la función principal según se necesita.

Ejercicio.

Calcular la raíz cuadrada de un numero dado por teclado.

Entrada:	Un numero que se almacenara en una variable llamada numero.
Proceso:	Empleando la función **Raiz2** se calcula el valor de la variable numero y se almacenara en la variable resultado.
Salida:	Mostrar el valor de la variable resultado.

```
1   ' Algoritmo calculo de la raíz cuadrada de un numero dado por
2   teclado
3
4   Inicio
5           Declaración resultado,numero
6
7           numero=Leer("ingrese un numero")
8           resultado=Raiz2(numero)
9           Escribir resultado
10  Fin
```

Todos los algoritmos desarrollados desde el principio están compuestos por funciones, de manera mecánica se han estado usando para labores comunes como mostrar datos o mensajes con la función **Escribir,** o para adquirir datos por teclado con la función **Leer.**

Cuando se quiere mostrar un mensaje o el contenido de una o más variables por pantalla se emplea la función **Escribir**. En este caso el mensaje que se quiere mostrar se conoce como **argumento(s) de entrada** a la función. Estos se ubican después del nombre de la función.

```
1   ' Algoritmo mostrar
2
3   Inicio
4           Escribir "Hola, esta es una función"
5   Fin
```

La estructura de la función **Escribir** es la siguiente:

Escribir *mensaje(s) o variable(s)*

Para obtener datos del teclado se usa la función **Leer**. Si se observa con detenimiento posee tanto argumentos de entrada como de salida.

```
1    ' Algoritmo lectura
2
3    Inicio
4            Numero = Leer("Ingrese un valor")
5    Fin
```

La estructura de esta función es:

Nombre_Variable_Recibe = **Leer(** *mensaje(s) o variable(s)* **)**

Donde *Nombre_Variable_Recibe* corresponde a la variable que recibe el valor que devuelve la función **Leer**, téngase en cuenta que cuando el usuario digita un valor, ese valor queda almacenado en un espacio de memoria denominada variable. Y *mensaje(s) o variable(s)* es el argumento de entrada de la función donde se especifica el mensaje que el usuario va ver cuando se le soliciten datos.

En los casos anteriores surge lo siguiente: ¿Por qué existen funciones que retornan valores y otras no? Para una mayor compresión se definirán tipos de funciones, proporcionando varios modelos según la tarea que se desea realizar.

10.3. TIPOS DE FUNCIONES.

La siguiente clasificación permite diseñar funciones según las necesidades y requerimientos del algoritmo que se va a desarrollar, además va permitir comprender todo el uso que se ha dado.

Dentro del ejercicio de muestra se desarrollo una función llamada **Saludo**, pero dentro del algoritmo se llamo Procedimiento, esto significa que las funciones que solo ejecutan sus instrucciones y no devuelven valores se les conoce también como **Procedimientos** y se va llamar **Función** a los subprogramas se devuelven un valor. Esta explicación se va ampliar a continuación.

En el ejemplo anterior se desarrollo una función de este tipo, que no recibe parámetros y como no retorna parámetros se conoce como **Procedimiento,** generalmente son las menos usadas. Su estructura es la siguiente:

```
Procedimiento nombreProcedimiento ()
        Acción 1
        Acción 2
        ...
        Acción n
FinProcedimiento
```

Tanto **Procedimiento** como **FinProcedimiento** son palabras reservadas por tanto no pueden existir variables o funciones con estos nombres. Dentro de cualquier tipo de procedimiento o función se pueden leer y mostrar datos, declarar variables teniendo en cuenta que son variables locales no existen en otra parte del programa.

Nota.

Cuando se finalice el recorrido por los tipos de funciones se va explicar con mas detenimiento el ámbito de las variables para tener bien claro este concepto.

Para ser llamado un procedimiento se hace a través de su nombre, tanto en la función principal así:

```
Inicio
        nombreProcedimiento
Fin
```

Como en otro procedimiento o función:

```
Procedimiento Suma()
        nombreProcedimiento
FinProcedimiento
```

```
Funcion Producto()
        NombreProcedimiento
        Producto=10
FinProcedimiento
```

Ejercicio.

Desarrollar un algoritmo que muestre el mensaje “Hola mundo” 5 veces.

Entrada:	no existen datos de entrada.
Proceso:	Mostrar por pantalla cinco veces hola mundo.
Salida:	**Hola mundo 5 veces**.

Forma 1.

```
1    ' Algoritmo saludo cinco veces
2
3    Inicio
4            Escribir "hola mundo"
5            Escribir "hola mundo"
6            Escribir "hola mundo"
7            Escribir "hola mundo"
8            Escribir "hola mundo"
9    Fin
```

Este algoritmo realiza la tarea pedida, es labor del programador optimizarla, la idea es utilizar la menor cantidad de instrucciones para lograr el objetivo.

Forma 2.

```
1    ' Algoritmo saludo cinco veces con ciclos
2
3    Inicio
4            Para contador=1 Hasta 5
5                    Escribir "hola mundo"
6            FinPara
7    Fin
```

Este algoritmo emplea la menor cantidad de instrucciones, usando una estructura repetitiva y mostrando el mensaje y él numero de veces solicitado, si se quisiera mas veces solo se cambiaría el número 5 por el deseado y ya, mientras que con la anterior habría que copiar y pegar las instrucciones restantes para él numero total.

Para hacer un primer acercamiento a los procedimientos obsérvese el siguiente programa que imprime una sola vez el mensaje deseado.

```
1    'Algoritmo saludo
2
3    Inicio
4            Escribir "Hola Mundo"
5    Fin
```

La idea es que este programa sea llamado por otro cinco veces, como se hizo en la forma 1 o solo llamado una vez dentro de un ciclo como la forma 2.

Para ser usado por otro programa debe tenerse en cuenta lo siguiente:

- En un algoritmo solo puede haber un solo **Inicio** y **Fin**, no pueden existir mas de uno.
- Cuando se ha creado un programa especializado se debe colocarse un nombre representativo[4] y las palabras reservadas **Inicio** y **Fin** cambian por **Procedimiento** y **FinProcedimiento** respectivamente.

Como este programa lo que sabe hacer es dar un saludo, su nombre va a hacer **Saludo** y teniendo lo anterior va quedar así:

```
1    'Algoritmo saludo
2
3    Inicio ──────────────►
4       Escribir "Hola Mundo"
5    Fin ─────────────────►
```

```
1    'Algoritmo saludo
2
3    Procedimiento Saludo()
4       Escribir "Hola Mundo"
5    FinProcedimiento
```

Obsérvese que se cambio la palabra **Inicio** por **Procedimiento** seguido del nombre representativo que se dio **Saludo** para este caso y acompañado por paréntesis que abren y cierran. La parte de desarrollo del algoritmo se dejo intacta y al final se reemplazo la palabra **Fin** por **FinProcedimiento.** Para poder ejecutar el algoritmo se debe

[4] Representativo se refiere a que va hacer o cual es su tarea en especial. También corresponde al uso de nombres mnemotécnicos.

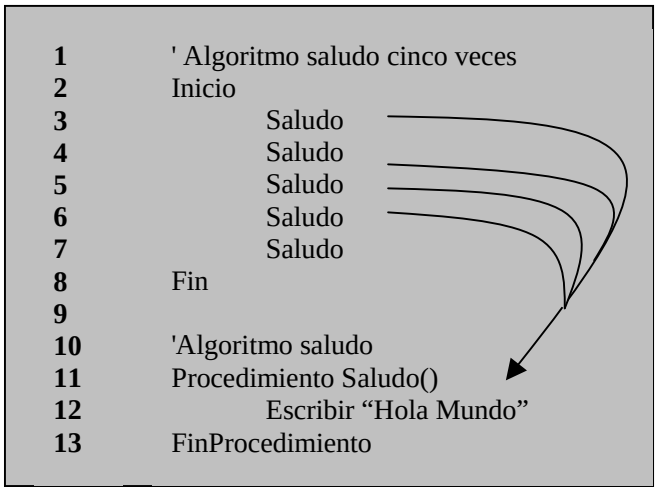
implementar el programa principal, es decir el que se encuentra contenido entre **inicio** y **fin**. Este se colocara en la parte superior de la función que se acabo de desarrollar. El resultado será entonces:

```
1   'Algoritmo llamado a funciones
2   Inicio
3           Saludo
4   Fin
5
6   ' Algoritmo saludo
7   Procedimiento Saludo()
8           Escribir "Hola Mundo"
9   FinProcedimiento
```

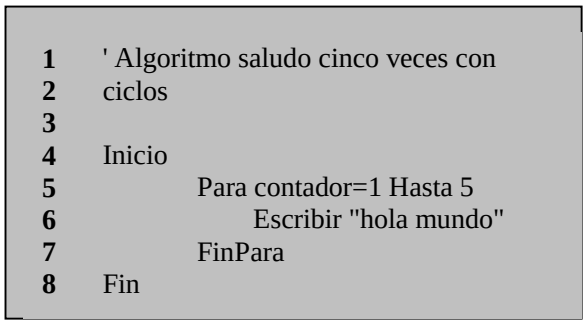
Retomando de nuevo los algoritmos de la formas 1 y 2, se tiene:

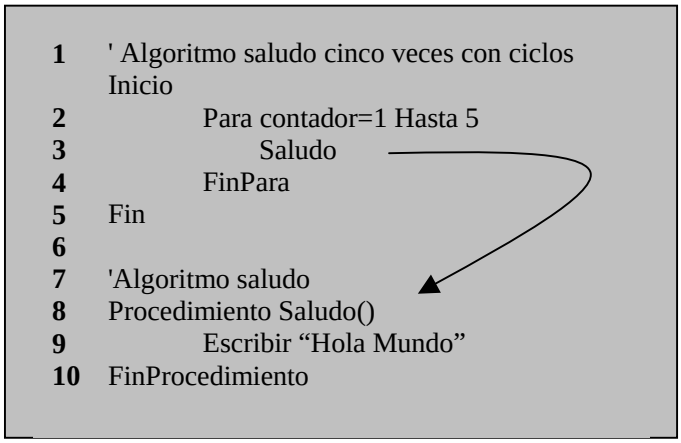
Forma 1.

```
1       ' Algoritmo saludo cinco veces
2
3       Inicio
4         Escribir "hola mundo"
5         Escribir "hola mundo"
6         Escribir "hola mundo"
7         Escribir "hola mundo"
8         Escribir "hola mundo"
9       Fin
```


1 ' Algoritmo saludo cinco veces
2 Inicio
3 Saludo
4 Saludo
5 Saludo
6 Saludo
7 Saludo
8 Fin
9
10 'Algoritmo saludo
11 Procedimiento Saludo()
12 Escribir “Hola Mundo”
13 FinProcedimiento

Forma 2.


1 ' Algoritmo saludo cinco veces con
2 ciclos
3
4 Inicio
5 Para contador=1 Hasta 5
6 Escribir "hola mundo"
7 FinPara
8 Fin


1 ' Algoritmo saludo cinco veces con ciclos
Inicio
2 Para contador=1 Hasta 5
3 Saludo
4 FinPara
5 Fin
6
7 'Algoritmo saludo
8 Procedimiento Saludo()
9 Escribir “Hola Mundo”
10 FinProcedimiento

Para los anteriores algoritmos cuando se llama a un procedimiento se hace por medio de su nombre, para este ejemplo se llama **Saludo**. Cuando se le llama, inmediatamente se salta a ella como lo indican las flechas para ambos casos, ejecutándose o desarrollándose las instrucciones contenidas entre **Procedimiento** y **FinProcedimiento** que para este caso imprime el mensaje **Hola Mundo**. Cuando se termina la ejecución dentro de la función se regresa inmediatamente una línea después de la que la llamo.

Ejercicio.

Desarrollar un procedimiento que pida un numero por teclado y muestre el valor dado.

Entrada: un dato que se almacenara en la variable valor.
Proceso: ninguno
Salida: mostrar el contenido de la variable valor.

Algoritmo tradicional.

```
1   ' Algoritmo pedir dato
2
3   Inicio
4           valor=0
5           valor=Leer("ingrese un numero")
6           Escribir "el numero que ingreso es:" & valor
7   Fin
```

Realizando la transformación a un procedimiento que se llamará **PedirDato** se obtiene:

```
' Algoritmo pedir dato

Inicio
        valor=0
        valor=Leer("ingrese un numero")
        Escribir "el numero que ingreso es:" & valor
Fin
```

```
' Procedimiento pedir dato

Procedimiento PedirDato()
        valor=0
        valor=Leer("ingrese un numero")
        Escribir "el numero que ingreso es:" & valor
FinProcedimiento
```

Dejando el algoritmo completo con la función principal y el procedimiento queda entonces:

```
1   'Algoritmo llamado a un procedimiento
2   Inicio
3           PedirDato
4   Fin
5
6   'Procedimiento pedir dato
7
8   Procedimiento PedirDato()
9           valor=0
10          valor=Leer("ingrese un numero")
11          Escribir "el numero que ingreso es:" & valor
12  FinProcedimiento
```

Ejercicio.

Desarrollar un algoritmo que calcule el cubo de un numero, empleando procedimientos.

Entrada: un valor numérico que se almacenara en la variable numero,
Proceso: calcular el cubo
Salida: mostrar el valor del cubo

```
1     'Algoritmo calculo del cubo
2
3     Inicio
4             CalcularCubo
5     Fin
6
7     'Procedimiento que calcula el cubo de un numero
8
9     Procedimiento CalcularCubo()
10            numero=0
11            numero=Leer("ingrese un valor")
12            cubo=numero^3
13            Escribir "el valor del cubo de " & numero & " es " & cubo
14    FinProcedimiento
```

Ejercicio.

Desarrollar un algoritmo que pida un numero positivo por teclado e imprima los valores entre ese numero y 1.

Entrada: un numero positivo.
Proceso: generar los números entre 1 y el numero positivo.
Salida: los números entre 1 y el numero positivo.

```
1     'Algoritmo listado números descendentes
2
3     Inicio
4             ListaNumeros
5     Fin
6
7     'Procedimiento que imprime lo valores entre un numero Y 1
8
9     Procedimiento ListaNumeros()
10            valor=0
11            valor=Leer("ingrese un numero")
12
```

```
13              Si valor > 0 Entonces
14                      Para numero=valor Hasta 1 Incremento -1
15                              Escribir numero
16                      FinPara
17              Sino
18                      Escribir "el numero ingresado es negativo"
19              FinSi
20      FinProcedimiento
```

Ejercicio.

Desarrollar un algoritmo que calcule el factorial de un numero dado por teclado.

Entrada: un numero entero
Proceso: calcular el factorial
Salida: el valor del factorial

```
1       Inicio
2               Factorial
3       Fin
4
5       Procedimiento Factorial()
6               numero=Leer("ingrese el numero")
7               fac=1
8               Para cont=1 Hasta numero
9                       fac=fac*cont
10              FinPara
11              Escribir "el factorial de " & numero & " es " & fac
12      FinProcedimiento
```

RESUMEN.

- Una función es un programa que realiza una tarea especifica.
- Las funciones son especializadas, por tanto solo hacen una sola tarea.
- Una función se considera también como un subprograma que puede ser llamado por otros subprogramas o desde la función principal según se necesita.
- Las funciones que solo ejecutan sus instrucciones y no devuelven valores se les conoce también como Procedimientos y se va llamar Función a los subprogramas se devuelven un valor.

EJERCICIOS DE AUTO EVALUACIÓN.

Desarrolle la gran mayoría de Pseudocódigos anteriores de funciones para ver su funcionamiento con datos reales. Seleccione de cada capitulo 5 ejercicios y desarróllelos usando funciones y procedimientos. Debe realizar el cambio teniendo muy claro que programa original y los conceptos de procedimientos y funciones.

EJERCICIOS PROPUESTOS.

Desarrolle los siguientes algoritmos en funciones y procedimientos.

1. Diseñar una función que calcule la media de tres números leídos por teclado.
2. Diseñar una función factorial.
3. Diseñar una función para calcular el mínimo común divisor de 4 números dados por teclado.
4. Diseñar varias funciones que encuentren: el mayor, el menor de 3 números dados por teclado.
5. Diseñar una función que calcule X^n.
6. desarrolle un grupo de funciones para calcular el cuadrado, el cubo, la raíz cuadrada. Dado un valor numérico por teclado.

Anexos.

TABLA DE CARACTERES ASCII.[5]

American Standard Code for Information Interchange De Wikipedia, la enciclopedia libre (Redirigido desde ASCII)
...
Wikipedia:Artículos destacados
Hay 95 caracteres ASCII imprimibles, numerados del 32 al 126.
Hay 95 caracteres ASCII imprimibles, numerados del 32 al 126.

El código ASCII (acrónimo inglés de American Standard Code for Information Interchange —Código Estadounidense Estándar para el Intercambio de Información), pronunciado generalmente [áski], es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales. Fue creado en 1963 por el Comité Estadounidense de Estándares (ASA, conocido desde 1969 como el Instituto Estadounidense de Estándares Nacionales, o ANSI) como una refundición o evolución de los conjuntos de códigos utilizados entonces en telegrafía. Más tarde, en 1967, se incluyeron las minúsculas, y se redefinieron algunos códigos de control para formar el código conocido como US-ASCII.

El código ASCII utiliza 7 bits para representar los caracteres, aunque inicialmente empleaba un bit adicional (bit de paridad) que se usaba para detectar errores en la transmisión. A menudo se llama incorrectamente ASCII a otros códigos de caracteres de 8 bits, como el estándar ISO-8859-1 que es una extensión que utiliza 8 bits para proporcionar caracteres adicionales usados en idiomas distintos al inglés, como el español.

ASCII fue publicado como estándar por primera vez en 1967 y fue actualizado por última vez en 1986. En la actualidad define códigos para 33 caracteres no imprimibles, de los cuales la mayoría son caracteres de control obsoletos que tienen efecto sobre como se procesa el texto, más otros 95 caracteres imprimibles que les siguen en la numeración (empezando por el carácter espacio).

Casi todos los sistemas informáticos actuales utilizan el código ASCII o una extensión compatible para representar textos y para el control de dispositivos que manejan texto.

[5] Tomado de: http://es.wikipedia.org/wiki/ASCII. Empleando los derechos de Wikipedia.

Códigos ASCII (0-127).

Carácteres no imprimibles				Carácteres imprimibles											
Nombre	Dec	Hex	Car.	Dec	Hex	Car.	Dec	Hex	Car.	Dec	Hex	Car.
Nulo	0	00	NUL	32	20	Espacio	64	40	@	96	60	`
Inicio de cabecera	1	01	SOH	33	21	!	65	41	A	97	61	a
Inicio de texto	2	02	STX	34	22	"	66	42	B	98	62	b
Fin de texto	3	03	ETX	35	23	#	67	43	C	99	63	c
Fin de transmisión	4	04	EOT	36	24	$	68	44	D	100	64	d
enquiry	5	05	ENQ	37	25	%	69	45	E	101	65	e
acknowledge	6	06	ACK	38	26	&	70	46	F	102	66	f
Campanilla (beep)	7	07	BEL	39	27	'	71	47	G	103	67	g
backspace	8	08	BS	40	28	(	72	48	H	104	68	h
Tabulador horizontal	9	09	HT	41	29	)	73	49	I	105	69	i
Salto de línea	10	0A	LF	42	2A	*	74	4A	J	106	6A	j
Tabulador vertical	11	0B	VT	43	2B	+	75	4B	K	107	6B	k
Salto de página	12	0C	FF	44	2C	,	76	4C	L	108	6C	l
Retorno de carro	13	0D	CR	45	2D	-	77	4D	M	109	6D	m
Shift fuera	14	0E	SO	46	2E	.	78	4E	N	110	6E	n
Shift dentro	15	0F	SI	47	2F	/	79	4F	O	111	6F	o
Escape línea de datos	16	10	DLE	48	30	0	80	50	P	112	70	p
Control dispositivo 1	17	11	DC1	49	31	1	81	51	Q	113	71	q
Control dispositivo 2	18	12	DC2	50	32	2	82	52	R	114	72	r
Control dispositivo 3	19	13	DC3	51	33	3	83	53	S	115	73	s
Control dispositivo 4	20	14	DC4	52	34	4	84	54	T	116	74	t
neg acknowledge	21	15	NAK	53	35	5	85	55	U	117	75	u
Sincronismo	22	16	SYN	54	36	6	86	56	V	118	76	v
Fin bloque transmitido	23	17	ETB	55	37	7	87	57	W	119	77	w
Cancelar	24	18	CAN	56	38	8	88	58	X	120	78	x
Fin medio	25	19	EM	57	39	9	89	59	Y	121	79	y
Sustituto	26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
Escape	27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
Separador archivos	28	1C	FS	60	3C	<	92	5C	\	124	7C	|
Separador grupos	29	1D	GS	61	3D	=	93	5D	]	125	7D	}
Separador registros	30	1E	RS	62	3E	>	94	5E	^	126	7E	~
Separador unidades	31	1F	US	63	3F	?	95	5F	_	127	7F	DEL

EL ANALIZADOR DE ALGORITMOS.

Es una herramienta basica para la creacion, edicion, correcion y ejecucion de algoritmos. Es totalmente compatible con la sintaxis y estructuras vistas a lo largo de este libro.

El software se creo como apoyo a la labor academica que desempeña el autor. Es una herramienta que se puede distribuir de manera gratiuta, la unica recomendacion es que lo usen y saquen el maxomo prvecho.

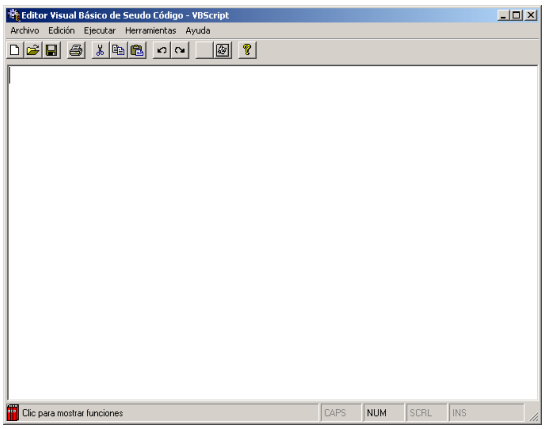
Instalacion del analizador de algoritmos.

La version de prueba que contiene el cd, esta diseñada la para instalacion sobre la plataforma Windows, sobre el sistema operativo Windows Xp de Microsoft. Dentro cd CD, hay una carpeta que se llama analizador de algoritmos, de doble click sobre el siguiente icono:

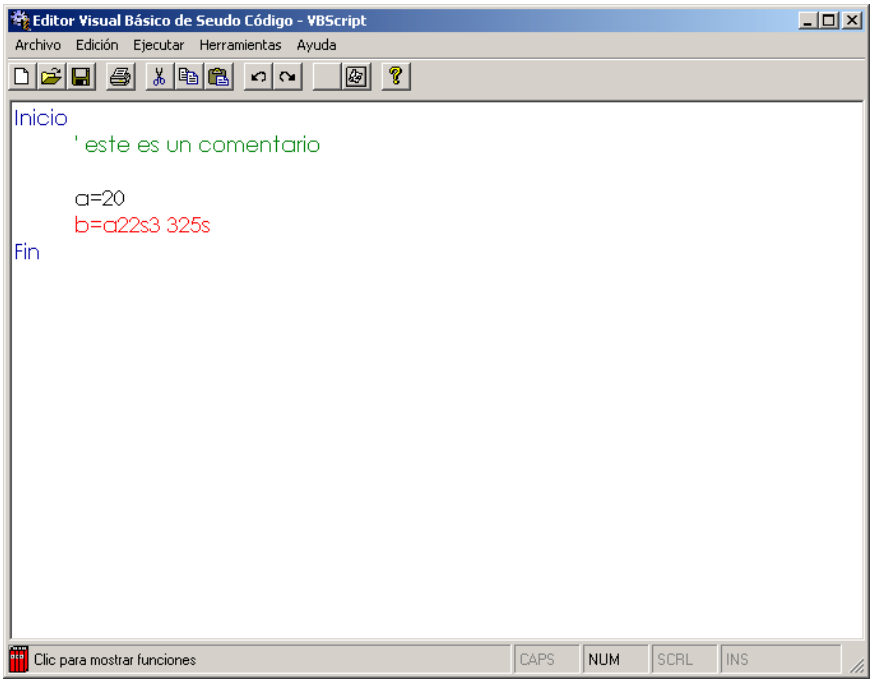
Este programa iniciarà la instalacion sobre el disco duro del ordenador. Por recomendacion hagala la instalacion estandar es decir, de click sobre las opciones por defecto.

Entorno de trabajo.

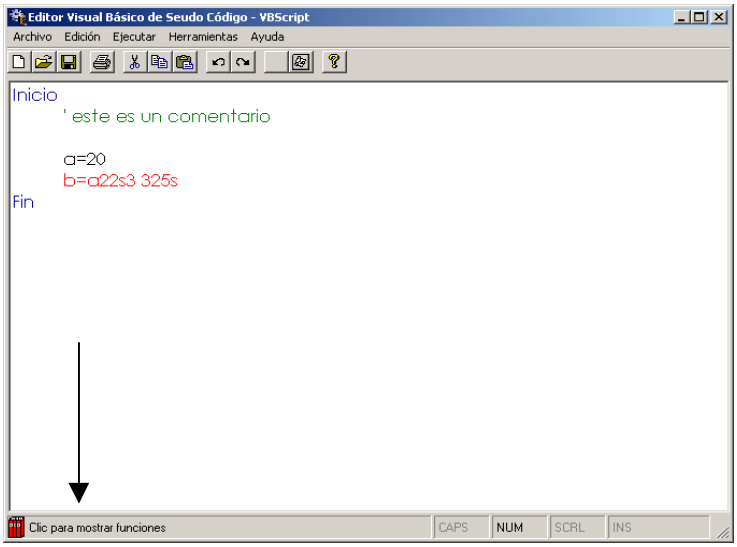
Una vez instalada la aplicación puede acceder a ella en:
Inicio / Programas / Proanalizador.

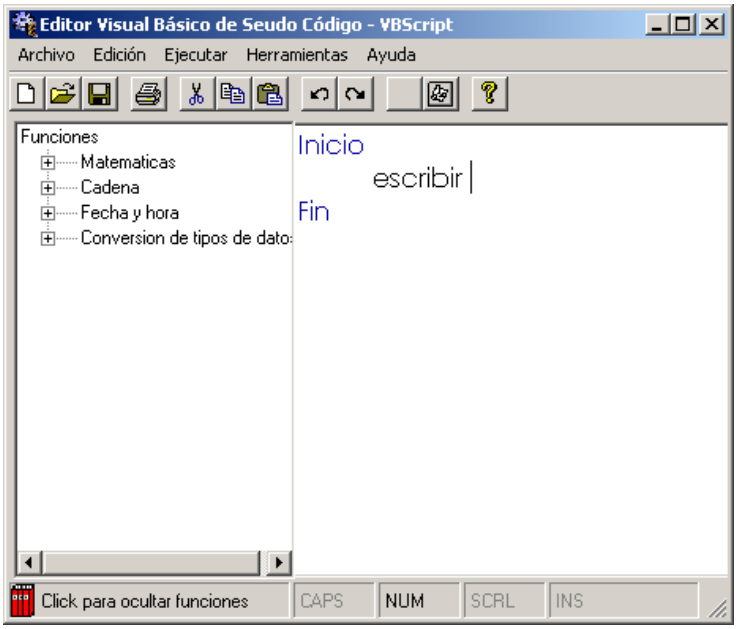


Como ayuda al programador, las palabras reservadas se colorean en azul, los errores en rojo, las instruciones de operaciones en negro y los comentario en verde como se muestra a continuacion con el siguiente algoritmo.



Para utilizar las funciones del sistema de click en el siguiente icono, de forma inmediata se muestra en esquema de arbol el listado de todas las funciones para reutilizar.





Para ejecutar la aplicación una vez finalizada de click en el menu **ejecutar, iniciar**.

INDICE.

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z