UNIVERSIDAD CONTINENTAL

3i INNOVADOR INTERACTIVO INTEGRAL

COMPUTACIÓN

Datos de catalogación bibliográfica

Computación. Manual Autoformativo /
Carlos Calderon Sedano–Huancayo:
Universidad Continental. Modalidad Virtual ; 2016.–162 p.

Datos de catalogación del CENDOC UC

Computación. Manual Autoformativo
Carlos Calderon Sedano

Primera edición
Huancayo, mayo de 2016

Av. San Carlos 1795, Huancayo-Perú
Teléfono: (51 64) 481-430 anexo 7361
Correo electrónico: recursosucvirtual@continental.edu.pe
http://www.continental.edu.pe/

Versión e-book
Disponible en http://repositorio.continental.edu.pe/
ISBN electrónico N.° 978-612-4196-

Dirección: Emma Barrios Ipenza
Edición: Eliana Gallardo Echenique
Asistente de edición: Andrid Poma Acevedo
Asesora didáctica: Luisa Aquije de Lozano
Corrección de textos: Corina Delgado Morales
Diseño y diagramación: Francisco Rosales Guerra

ÍNDICE

INTRODUCCIÓN

La presente asignatura se desarrolla en la modalidad virtual y el presente manual autoformativo es un material didáctico importante para su aprendizaje.

Esta asignatura tiene como finalidad proporcionar al estudiante, los conocimientos necesarios para desarrollar en él la capacidad de identificar, formular y resolver problemas de ingeniería haciendo uso de algoritmos y lenguajes de programación.

El presente material para la asignatura de Computación, consta de cuatro unidades: Unidad I: Introducción a las Ciencias de la Computación en el cual se desarrolla La evolución de la computación, funcionamiento del computador, hardware y software. Unidad II: Algoritmos donde se explica los conceptos de algoritmos y se muestran la forma de su representación a través de los diagramas de Flujo. Unidad III: Introducción a la programación en la cual se muestra las etapas en la construcción de un programa. Unidad IV: Arreglos en la cual se desarrollan los algoritmos y programas para vectores y matrices.

Es recomendable que el estudiante desarrolle una permanente lectura de estudio, así como la investigación en otros textos y vía internet. El contenido del material se complementará con las lecciones por videoclase. Agradecemos a quienes con sus aportes y sugerencias han contribuido a mejorar la presente edición, que sólo tiene el valor de una introducción al conocimiento de los conceptos de la computación, algoritmos y programas.

DIAGRAMA DE PRESENTACIÓN DE LA ASIGNATURA

RESULTADOS DEL APRENDIZAJE:

Al finalizar la asignatura, los estudiantes serán capaces de reconocer los conocimientos básicos de la computación, diseñar soluciones estructuradas aplicando técnicas algorítmicas y estructuras de datos, con la ayuda de software de computadora independientemente del lenguaje de programación para la resolución de problemas de su entorno personal y laboral.

UNIDADES DIDÁCTICAS:

UNIDAD I	UNIDAD II	UNIDAD III	UNIDAD IV
INTRODUCCIÓN A LAS CIENCIAS DE LA COMPUTACIÓN	FUNDAMENTOS DE LOS ALGORITMOS	INTRODUCCIÓN A LA PROGRAMACIÓN	FUNDAMENTOS DE ARREGLOS UNIDIMENSIONALES Y BIDIMENSIONALES

TIEMPO MINIMO DE ESTUDIO:

UNIDAD I	UNIDAD II	UNIDAD III	UNIDAD IV
1era. Semana y 2da. Semana 24 horas	3era. Semana y 4ta. Semana 24 horas	5ta. Semana y 6ta. Semana 24 horas	7ma. Semana y 8va. Semana 24 horas

UNIDAD I

"INTRODUCCIÓN A LAS CIENCIAS DE LA COMPUTACIÓN"

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD

- Reconoce y distingue el propósito de las ciencias de la computación.
- Explica y esquematiza las partes físicas de un computador señalando las funciones de cada unida de ellas.

CONTENIDOS	ACTIVIDADES FORMATIVAS (HABILIDADES Y ACTITUDES)	SISTEMA DE EVALUACIÓN (TÉCNICAS Y CRITERIOS)
TEMA Nº 1: Introducción a las ciencias de la computación. 1 Evolución de la computación. 2 La computación en la actualidad. 3 Computación e interconexión. **TEMA Nº 2: Representación de datos.** 1 Funcionamiento de un computador 2 Unidades de representación. **TEMA Nº 3: Hardware y Software** **Hardware:** 1 Unidad Central de Procesamiento. 2 La memoria de la computadora.. 3 Buses, puertos y periféricos. **Software:** 1 El software como lenguaje de las computadoras. 2 Aplicaciones como herramientas para el usuario. **TEMA Nº 4: Sistema Operativo.** 1 La conexión entre el hardware y el software. 2 Interfaz de usuario Hombre-Máquina.	• **Prepara una Línea de Tiempo sobre la evolución de la computación con datos obtenidos de la lectura analítica de los subtemas 1,2 y 3.** • **Elabora un organizador Visual con las funciones básicas de un computador y sus unidades de representación. Y realiza la conversión de 5 numeros(0 al 100) a base binaria según el cuadro Nº 3** • **Elabora un cuadro de doble entrada señalando las partes constitutivas de un computador y explicando las funciones de cada una.** • **Participa en un foro de debate sobre la importancia del software de base y reforzamiento de todos los temas.**	**Procedimientos e indicadores de evaluación permanente** • Entrega puntual de trabajos realizados. • Calidad, coherencia y pertinencia de contenidos desarrollados. • Prueba teórico-práctica, individual. • Actividades desarrolladas en sesiones tutorizadas **Criterios de evaluación para el cuadro de doble entrada: partes físicas del computador y sus funciones**. • **Variables** • **Descripción del tema** • **Orden y diseño** • **Comparación** • **Presentación del cuadro de doble entrada**

RECURSOS:

Vídeos o imagenes:

Tema Nº 1:
COMO ELABORAR UNA LÍNEA DE TIEMPO
http://laculturainca-cusi.blogspot.com/2010_11_01_archive.html

Tema Nº 2:
TIPOS DE ORGANIZADORES:
https://www.youtube.com/watch?v=mvo6KhMN5sI

Tema Nº 3
COMO HACER UN CUADRO DE DOBLE ENTRADA
https://www.youtube.com/watch?v=nU8tScHyoYs

Lectura complementaria:

Lectura Seleccionada Nº 1
Charles Babbage, Lady Lovelace y la madre de todas las computadoras
Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

Lectura Seleccionada Nº 2
Linus Torvalds y el software que no es propiedad de nadie
Autor: Beekman,, George., Pacheco, Rosbinda.,Tábora Alez.
Páginas:

INSTRUMENTO DE EVALUACIÓN	Prueba Objetiva. Prueba de Desarrollo
BIBLIOGRAFÍA (BÁSICA Y COMPLEMENTARIA)	**BÁSICA** SWADE Doron y BABBAGE Charles. The Difference Engine: the Quest to Build the First Computer. New York: Viking Press, 2001. **COMPLEMENTARIA** FREIBERGER, Paul y SWAINE Michael. *The Making of the Personal Computer*. Second Edition, CA: McGraw-Hill, 1999.
RECURSOS EDUCATIVOS DIGITALES	BUSTAMANTE VELEZ, John Fabio. *Software y Hardware. 2011.* Disponible en web http://www.informatica-hoy.com.ar/aprender-informatica/que-es-hardware-y-software.php

TEMA N° 1: INTRODUCCIÓN A LAS CIENCIAS DE LA COMPUTACIÓN.

Hoy en día es casi imposible pensar en un mundo sin computadoras, sin software que nos ayude a realizar nuestras tareas cotidianas, laborales y recreativas. En la presente unidad conoceremos la evolución de la computación desde sus orígenes hasta la actualidad. Para ello es importante conocer el desarrollo de la computación y de las tecnologías relacionadas que han permitido realizar muchos tareas como: la elaboración de diversos tipos de documentos, el envío y la recepción de correo electrónico la creación de dibujos digitales, la edición de audio y la impresión de libros, entre muchos otros procesos.

La tecnología utilizada en computación es de tipo microelectrónica con componentes físicos: Hardware (procesador, memoria, etc) y Lógicos: Software (sistema operativo y programas).

1. EVOLUCIÓN DE LA COMPUTACIÓN1

Aunque las computadoras llevan con nosotros desde hace medio siglo, las raíces de estos dispositivos se extienden mucho más allá de cuando Charles Babbage concibió la Máquina analítica en 1823.

Estas extraordinarias máquinas están construidas sobre siglos de esfuerzo intelectual.

1.1 Antes de las computadoras

Las computadoras nacieron por la necesidad humana de cuantificar. Antes, a los seres humanos les bastaba con contar con los dedos, las piedras o cualquier otro objeto cotidiano.

A la vez que las culturas iban haciéndose más complejas, necesitaron herramientas para contar. El ábaco (un tipo de herramienta para contar y calcular usado por los babilonios, los chinos y otras culturas hace miles de años) y el sistema numérico indo-arábigo son ejemplos de métodos de cálculo antiguos que han afectado de forma significativa a la raza humana (imagine cómo sería intentar llevar cualquier negocio sin un sistema numérico que permitiera suma y restar de forma sencilla).

La Máquina analítica tuvo un impacto reducido hasta un siglo después de su invención, cuando sirvió como punto de partida de la primera computadora programable real. Virtualmente, cualquier computadora actual sigue el esquema ideado por Babbage y Lady Lovelace.

1.2 La máquina de procesamiento de información

Al igual que la Máquina analítica, la computadora es un dispositivo que cambia la información de un formato a otro. Todas ellas toman información de entrada y generan información de salida. Ya que la información puede tomar muy distintas formas, la computadora se convierte en una herramienta increíblemente versátil capaz de procesar los impuestos y guiar los misiles que dichos impuestos compran.

Para el cálculo de esos impuestos, la entrada de la computadora podría contener los salarios, otro tipo de pagos, las deducciones, las exenciones y las tablas de retenciones, mientras que la salida mostraría un número indicativo de las declaraciones que son a pagar y las que son a devolver. Si la computadora está encargada de lanzar un misil, la entrada, por ejemplo, podría ser las señales procedentes del satélite que indicarían el blanco a alcanzar,

1 Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación*. México: Pearson Educación.

y la salida podría ser las señales eléctricas que controlan la ruta del misil. Y lo que resulta más sorprendente aún es que la misma computadora puede utilizarse para ambos propósitos. ¿Cómo es posible que una máquina sea tan versátil?

La flexibilidad de la computadora no se encuentra en el hardware (la parte física de un sistema computarizado). El secreto está en el software, también llamados programas, que son las instrucciones que le dicen al hardware cómo transformar el dato de entrada (la información en un formato que pueda leer) en la salida adecuada. Ya esté efectuando una sencilla operación o llevando a cabo una compleja animación, siempre existirá un programa software controlando el proceso de principio a fin.

En efecto, el cambio de estos programas puede hacer variar la utilidad de la computadora. Como es posible programarla para llevar a cabo diferentes tareas, la computadora típica moderna es una herramienta de propósito general, y no un dispositivo especializado de un único uso.

1.3 Las primeras computadoras reales

Aunque Lady Lovelace predijo que la Máquina analítica podría llegar a componer música algún día, los científicos y matemáticos que diseñaron y construyeron las primeras computadoras hace un siglo tenían un objetivo más modesto: crear máquinas capaces de efectuar cálculos repetitivos. Aun así, sus historias no están exentas de drama e ironía. He aquí algunas de ellas:

- En 1939, un joven ingeniero alemán llamado Konrad Zuse completó la primera computadora digital programable de propósito general. «Era tan vago a la hora de realizar cálculos que inventé la computadora», dijo Zuse. En 1941, Zuse y un amigo solicitaron al gobierno alemán fondos para construir una computadora electrónica más rápida que ayudara a descifrar los códigos enemigos durante la Segunda Guerra Mundial. El ejército nazi desechó el proyecto confiando en que su aviación sería capaz de ganar la guerra rápidamente sin la ayuda de sofisticados dispositivos de cálculo.

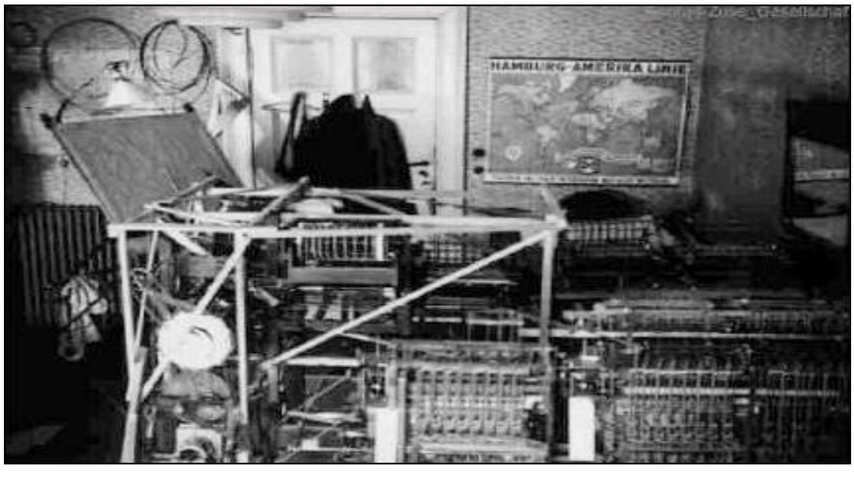


Figura N° 1 : Primera computadora digital programable de propósito general
Fuente http://www.kerryr.net/pioneers/gallery/ns_zuse5.htm

- Casi al mismo tiempo, y en el más alto de los secretos, el gobierno británico formó un equipo de matemáticos e ingenieros para que desarrollaran un modo de descifrar los códigos secretos del ejército nazi. En 1943, el equipo, dirigido por el matemático Alan Turing, completó Colossus, considerada por muchos como la primera computadora digital electrónica. Este dispositivo de propósito específico logró descifrar fácilmente esos códigos militares, lo que permitió a la inteligencia militar británica «escuchar» hasta el más insignificante mensaje alemán.

- En 1939, el profesor John Atanasoff de la Iowa State University, buscando una herramienta que ayudara a sus alumnos a resolver ecuaciones diferenciales, desarrolló lo que puede considerarse como la primera computadora digital electrónica, la ABC (Computadora Atanasoff–Berry, Atanasoff–Berry Computer). Su universidad se olvidó de patentar la máquina, y Atanasoff nunca intentó convertir su idea en un producto operativo. La empresa IBM (International Business Machines) respondió a su consulta diciéndole «IBM nunca estará interesada en una máquina de computación electrónica».

- El profesor Howard Aiken, de la Universidad de Harvard, tuvo más éxito en la financiación de la calculadora automática de propósito general que estaba desarrollando. Gracias a un millón de dólares donados por IBM, completó la Mark I en 1944. Este monstruo de unos 15 metros de largo por 2,5 de alto utilizaba ruidosos relés electromecánicos para calcular cinco o seis veces más rápido que una persona, aunque era mucho más lenta que cualquiera de las calculadoras de bolsillo que pueden encontrarse hoy día en las tiendas por 5 dólares.

- Tras consultar con Atanasoff y estudiar la ABC, John Mauchly se alió con J. Presper Eckert para ayudar al ejército americano de la II Guerra Mundial construyendo una máquina capaz de calcular tablas de trayectorias para las nuevas armas. La máquina fue la ENIAC (Electronic Numerical Integrator and Computer), un «artilugio » de 30 toneladas y 18,000 válvulas de vacío que se estropeaba, de media, una vez cada siete minutos. Cuando estaba en funcionamiento, podía calcular 500 veces más rápido que las calculadoras electromecánicas existentes (más o menos, como nuestras calculadoras actuales). No estuvo terminada hasta dos meses después del final de la guerra, aunque sí convenció a sus creadores de que las computadoras a gran escala podían tener interés comercial. Tras la guerra, Mauchly y Eckert crearon una compañía privada llamada Sperry y crearon la UNIVAC I, la primera computadora comercial de propósito general. UNIVAC I entró en funcionamiento para la Oficina del Censo de los Estados Unidos en 1951.

1.4 Evolución y aceleración

El hardware de las computadoras ha evolucionado rápidamente desde sus primeros días con nuevas tecnologías que se han ido reemplazando cada pocos años. Las primeras computadoras eran grandes, caras y muy «complicadas». Sólo una gran institución como un banco importante o el Gobierno de los Estados Unidos podían permitirse una computadora, por no mencionar el centro de computación climatizado y la plantilla de técnicos que eran necesarios para programarla y mantenerla en funcionamiento. Pero con todos sus fallos, las computadoras se convirtieron rápidamente en herramientas indispensables para científicos, ingenieros y otros profesionales. El transistor, inventado en 1948, podía realizar las mismas tareas que las válvulas de vacío que se empleaban en las primeras computadoras transfiriendo electricidad a través de una fina resistencia. Los transistores fueron usados por primera vez en computadoras en 1956.

Dichas computadoras eran radicalmente más pequeñas, fiables y baratas que las basadas en válvulas. Gracias a las mejoras en el software que se produjeron casi al mismo tiempo, estas máquinas eran también mucho más sencillas y rápidas de programar y usar. Como resultado, las computadoras se empezaron a utilizar ampliamente en empresas y para estudios científicos y de ingeniería. Pero el programa espacial americano precisaba de máquinas que fueran aun más potentes y pequeñas que las basadas en transistores, lo que obligó a los investigadores a desarrollar una tecnología que les permitiera empaquetar cientos de estos transistores en un único circuito integrado dentro un delgado chip de silicio. Hacia mediados de los 60, las computadoras basadas en transistores fueron sustituidas por otras más pequeñas y potentes construidas alrededor de los nuevos circuitos integrados. Estos componentes reemplazaron rápidamente a los transistores por las mismas razones que éstos, anteriormente, habían sustituido a las válvulas de vacío:

- Fiabilidad. Las máquinas construidas con circuitos integrados eran menos propensas a los fallos que sus predecesoras, ya que los chips podían ser verificados rigurosamente antes de su instalación.

- Tamaño. Un solo chip podía sustituir a una placa con cientos o miles de transistores, lo que permitía una reducción considerable del tamaño de las máquinas.

- Velocidad. Como la electricidad tenía que recorrer menores distancias, estas máquinas eran considerablemente más veloces que sus predecesoras.
- Eficiencia. Ya que los chips eran tan pequeños, necesitaban menos energía eléctrica. Como resultado de ello, generaban menos calor.
- Coste. Las técnicas de producción en masa hicieron posible la fabricación de chips baratos.

Desde su inicio, todos los avances en la tecnología de las computadoras han presentado ventajas similares sobre aquélla a la que sustituía El implacable progreso de esta industria está mostrado en la ley de Moore. En 1965, Gordon Moore, el presidente del fabricante de chips Intel, predijo que la potencia de un chip de silicio del mismo precio podría doblarse cada 18 meses durante al menos dos décadas. En la actualidad, tres décadas más tarde, su predicción se ha mostrado totalmente acertada. En resumen, los tres dispositivos que definen las tres primeras generaciones de computadoras son las válvulas de vacío, que albergaban unos pocos conmutadores en un espacio similar al de una bombilla, el transistor, que permitía a los ingenieros incluir la misma circuitería en un paquete semiconductor que era pequeño, más frío y mucho más fiable, y los chips de silicio, cuyos primeros ejemplares incluían varios transistores en una «manchita» mucho más pequeña que un solo transistor.

1.5 La revolución de las microcomputadoras

La invención de las válvulas de vacío, los transistores y los chips de silicio han tenido un tremendo impacto en nuestra sociedad. Pero el impacto de cualquiera de ellos no puede compararse con el que tuvo la invención del primer microprocesador en 1971: el componente crítico de una computadora doméstica completa contenido en un delgado chip de silicio. El desarrollo del microprocesador por parte de los ingenieros de Intel provocó cambios radicales e inmediatos en el aspecto, potencia y disponibilidad de las computadoras. Actualmente, un sólo chip del tamaño de una uña puede contener el equivalente a millones de transistores. Los costes de investigación y desarrollo del primer microprocesador fueron astronómicos. Pero una vez que las líneas de ensamblaje estuvieron en funcionamiento, las computadoras con chips de silicio pudieron ser fabricadas en masa a unos costos muy inferiores.

Las materias primas eran verdaderamente baratas; el silicio, ingrediente principal de la arena de la playa, es el segundo elemento más común (tras el oxígeno) en la superficie de la Tierra. Las compañías de los Estados Unidos inundaron rápidamente el mercado con relojes y calculadoras de bolsillo construidas alrededor de los baratos microprocesadores. El efecto económico fue inmediato: de la noche a la mañana, las calculadoras mecánicas y las reglas de cálculo quedaron obsoletas, los aficionados a la electrónica se convirtieron en saludables empresarios, y el área de San José en California se ganó el apodo de Silicon Valley cuando docenas de empresas fabricantes de microprocesadores se afincaron y crecieron ahí. La revolución de las microcomputadoras comenzó a finales de los 70 cuando compañías como Apple, Commodore y Tandy presentaron computadoras de bajo coste y del tamaño de una máquina de escribir tan potentes como los antiguos «armarios».

Las PC (Computadoras personales), nombre con el que se conocen a las microcomputadoras, son en la actualidad elementos comunes en oficinas, empresas, domicilios particulares, escuelas, etc. Debido al cumplimiento de la ley de Moore por parte de los fabricantes de chips, las microcomputadoras han ido ganando velocidad y potencia durante las dos últimas décadas. Al mismo tiempo, las PC han empezado a desempeñar tareas que, hasta el momento, estaban restringidas a grandes computadoras, y cada año la gente encuentra nuevas e innovadoras formas de aprovechar estos pequeños y versátiles «caballos de labor». Con el incremento de las PC, la era de la computación institucional llegó a su fin. En verdad, las computadoras pequeñas han tenido un impacto mucho mayor en la sociedad que sus predecesores del tamaño de armarios. Sin embargo, las computadoras de escritorio aún no han podido sustituir por completo a las grandes computadoras, las cuales también han evolucionado.

2. LAS COMPUTADORAS EN LA ACTUALIDAD

En la actualidad, la gente trabaja con mainframes (máquinas del tamaño de una gran sala), supercomputadoras, estaciones de trabajo, portátiles, computadoras de bolsillo, computadoras incrustadas. Aunque todas ellas están basadas en la misma tecnología, todas estas máquinas tienen sustanciales diferencias.

3. COMPUTACIÓN E INTERCONEXIÓN

La invención del tiempo compartido en los años 60 permitió que múltiples usuarios se conectaran a un único mainframe central mediante terminales individuales. Cuando las computadoras personales comenzaron a sustituir a estos últimos, muchos usuarios se dieron cuenta que tenían toda la potencia de computación que necesitaban en sus escritorios. A pesar de ello, también encontraron que enlazar algunas de estas computadoras en una LAN (Red de área local, Local Area Network), o red para abreviar, ofrecía muchas ventajas. Cuando las máquinas se agrupaban, podían compartir recursos como dispositivos de almacenamiento, impresoras e, incluso, capacidad de procesamiento. Mediante una red, una única impresora de alta velocidad podía dar servicio a toda una oficina.

Como premio añadido, la gente podía usar las computadoras para enviar y recibir mensajes electrónicos a través de las redes. Las ventajas de la comunicación electrónica y la compartición de recursos se vieron multiplicada cuando las redes más pequeñas se unieron en otras de mayor tamaño. La aparición de la tecnología de telecomunicación permitió que las WAN (Red de área amplia, Wide Area Network) no respetaran ni continentes ni océanos. Una computadora remota podía conectarse con una red a través de las líneas telefónicas estándar usando un módem (un dispositivo electrónico que podía convertir los datos de la computadora en señales compatibles con el sistema telefónico). Los bancos, las agencias gubernamentales y otras instituciones separadas geográficamente comenzaron a construir sistemas de procesamiento de información para beneficiarse de la tecnología de red de larga distancia. Pero fuera de este tipo de organizaciones, la red era algo «vedado» para el usuario normal.

La gente veía las computadoras como elementos para realizar cálculos, almacenar datos e imprimir documentos, y no como una herramienta de comunicación. Hasta finales de los años 90, la mayoría de las PC eran dispositivos aislados, islas de información. Sin embargo, había excepciones: un grupo de científicos e ingenieros informáticos, financiados por el Gobierno de los Estados Unidos, construyó una red experimental llamada ARPANET en 1969. Esta red fue la semilla que, más adelante, dio vida a Internet: el grupo global de redes que transformó radicalmente el modo en el que los usuarios emplearían sus computadoras.

VIDEOS

Video 1: Las máquinas automáticas y Charles Babbage.

Datos del Video seleccionado

Título o Tema: Charles Babbage, Konrad Zuse y La Computadora.
URL: https://youtu.be/tcG1RjSJjrw?t=2m12s
Duración: 3 min 53 s.
Autor(a): Target Film GmbH, Uwe von Schumann, Jürgen A. Knoll.
Año: 2000.
Licencia: YouTube estándar.

Video 2: Konrad Zuse.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Charles Babbage, Konrad Zuse y La Computadora.
URL: https://youtu.be/tcG1RjSJjrw?t=32s
Duración: 6 min 42 s.
Autor(a): Target Film GmbH, Uwe von Schumann, Jürgen A. Knoll.
Año: 2000.
Licencia: YouTube estándar.

Video 3: El primer transistor.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: El primer transistor.
URL: https://youtu.be/9MMaAgtnb6E?t=15s
Duración: 6 min 37 s.
Autor(a): History.
Año: 2000.
Licencia: YouTube estándar.

Video 4: La evolución de la computadora.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: El ordenador.
URL: https://youtu.be/dIwA-txArgg?t=19m57s
Duración: 11 min 1 s.
Autor(a): History.
Año: 2000.
Reseña: Este vídeo nos muestra la evolución de las computadoras.
Licencia: YouTube estándar.

Video 5: Tim Berners Lee y la World Wide Web.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Tim Berners Lee y el WWW.
URL: https://youtu.be/5PGsqRG9Zdk?t=12s
Duración: 11 min 42 s.
Autor(a): Südwestrundfunk SWR, Inter / Aktion GmbH.
Año: 2005.
Licencia: YouTube estándar.

ACTIVIDAD FORMATIVA Nº 1

Prepara una Línea de Tiempo sobre la evolución de la computación con datos obtenidos de la lectura analítica de los subtemas 1,2 y 3.

Instrucciones:

1. Lee y analiza los contenidos de los subtemas señalados
2. Extrae las ideas fundamentales
3. Organiza datos sobre fechas y hechos, casos o sucesos importantes
4. Observa ejemplos: ¿Cómo elaborar una línea de tiempo?

   http://laculturainca-cusi.blogspot.com/2010_11_01_archive.html
5. Diseña de manera creativa y organizada su propia línea de tiempo
6. La envía al aula virtual.

TEMA N° 2: REPRESENTACIÓN DE DATOS.

Las computadoras procesan exclusivamente señales electrónicas binarias (cero y uno). Dar una instrucción a una computadora supone en realidad enviar series de unos y ceros espaciadas en el tiempo de una forma determinada. En el presente tema comprenderemos cómo una computadora trabaja con señales binarias y puede realizar todas las maravillas de las que somos testigos utilizando únicamente al cero y al uno.

1. FUNCIONAMIENTO DE UN COMPUTADOR[2]

La simple verdad es que las computadoras realizan sólo **cuatro funciones básicas:**

- Recibir una entrada. Aceptan información del mundo exterior.
- Procesar información. Realizan operaciones aritméticas o lógicas (toma de decisiones) sobre la información.
- Producir una salida. Comunican información al mundo exterior.
- Almacenar información. Desplazan y mueven información en la memoria.

2. UNIDADES DE REPRESENTACIÓN

El término información es difícil de definir, porque tiene muchos significados. De acuerdo con una definición popular, la información es comunicación que tiene valor porque informa. Esta distinción puede ser útil para tratar con datos de la televisión, revistas, computadoras y otras fuentes. Pero no siempre está claro, y no es absoluto. Como educador y autor, Richard Saul Wurman señala: «Todo el mundo necesita una medida personal con la que definir la información. Lo que significa información para una persona pueden ser datos para otra. Si no tiene sentido para usted, no cuenta». En el extremo opuesto, una teoría de comunicación define la información como cualquier cosa que puede ser comunicada, tenga valor o no. Según esta definición, la información viene en muchas formas. Las palabras, números e imágenes de estas páginas son símbolos que representan información.

Si subraya o destaca esta frase, está añadiendo nueva información a la página. Incluso los sonidos y las imágenes que emanan de un anuncio de televisión están envueltos en información, aunque sea discutible si la mayoría de esa información es útil. Algunas personas intentan aplicar estrictamente la primera definición a las computadoras, sosteniendo que éstas convierten los datos brutos, que no tienen valor en su forma actual, en información que es valiosa. Este método enfatiza el rol de la computadora como una máquina comercial de proceso de datos. Pero en nuestro mundo moderno interconectado, la salida de una computadora es a menudo la entrada de otra. Si una computadora recibe un mensaje de otra, ¿el mensaje son datos sin valor o es información valiosa? ¿Y qué medida personal de valor se aplica? Para nuestros propósitos, describir la mecánica de las computadoras, nos inclinamos hacia el segundo método, más subjetivo, y a utilizar los términos datos e información de forma más o menos intercambiable.

2.1. Fundamentos de los bits

De un modo u otro, en el mundo de las computadoras, la información es digital: esto significa que está hecha de unidades contables, separadas (dígitos) de modo que puede subdividirse. En muchas situaciones, la gente necesita reducir la información a unidades más simples para usarla con eficacia. Por ejemplo, un niño que intente pro-

2 Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación.* México: Pearson Educación.

nunciar una palabra no familiar puede pronunciar cada letra o silabear individualmente antes de decir la palabra entera. Una computadora no entiende palabras, números, imágenes, notas musicales o letras del alfabeto. Igual que un joven lector, una computadora no puede procesar información sin dividirla en unidades más pequeñas. De hecho, sólo pueden digerir la información que se ha dividido en bits. Un bit, o dígito binario, es la unidad más pequeña de información que puede procesar una computadora. Un bit puede tener uno de dos valores, 0 o 1.

También puede pensar en esos valores como sí y no, encendido y apagado, blanco y negro o alto y bajo. Si piensa en las interioridades de una computadora como una colección de microscópicos conmutadores on/off, es fácil entender por qué procesan la información bit a bit. Cada conmutador almacena una pequeña cantidad de información: una señal para encender una luz, por ejemplo, o la respuesta a una pregunta del tipo sí/no. (En los circuitos integrados modernos, las cargas eléctricas altas y bajas representan bits, pero estos circuitos funcionan lo mismo que si realmente estuvieran hechos de pequeños conmutadores). ¿Recuerda la famosa cabalgada a media noche de Paul Revere para avisar a los colonos americanos de la invasión británica? Sus compañeros de conspiración utilizaron un par de linternas para comunicar una opción entre dos mensajes, «Una si es por tierra, dos si es por mar», es decir, una opción binaria.

Es teóricamente posible enviar un mensaje como éste sólo con una linterna. Pero «una por tierra, cero por mar» no hubiera funcionado muy bien a no ser que hubiera algún modo de saber exactamente cuándo se enviaría el mensaje. Con dos linternas, la primera linterna podría decir «aquí está el mensaje» cuando se encendiera. La segunda linterna comunicaría la validez crítica de la información del bit. Si los revolucionarios hubieran querido enviar un mensaje más complejo, hubieran usado más linternas («¡Tres si vienen en metro!»). De forma muy parecida, una computadora puede procesar fragmentos más grandes de información tratando grupos de bits como unidades lógicas.

Por ejemplo, una colección de 8 bits, llamada byte, puede representar 256 mensajes diferentes (256 = 28). Si piensa en cada bit como una luz que puede estar encendida o apagada, puede hacer que diferentes combinaciones de luces representen mensajes distintos. (Los informáticos hablan generalmente en términos de 0 y 1, en lugar de on y off, pero el concepto es el mismo.) La computadora tiene una ventaja sobre Paul Revere, puesto que no sólo ve el número de luces encendidas, sino también su orden, ya que 01 (off-on) es diferente de 10 (on-off).

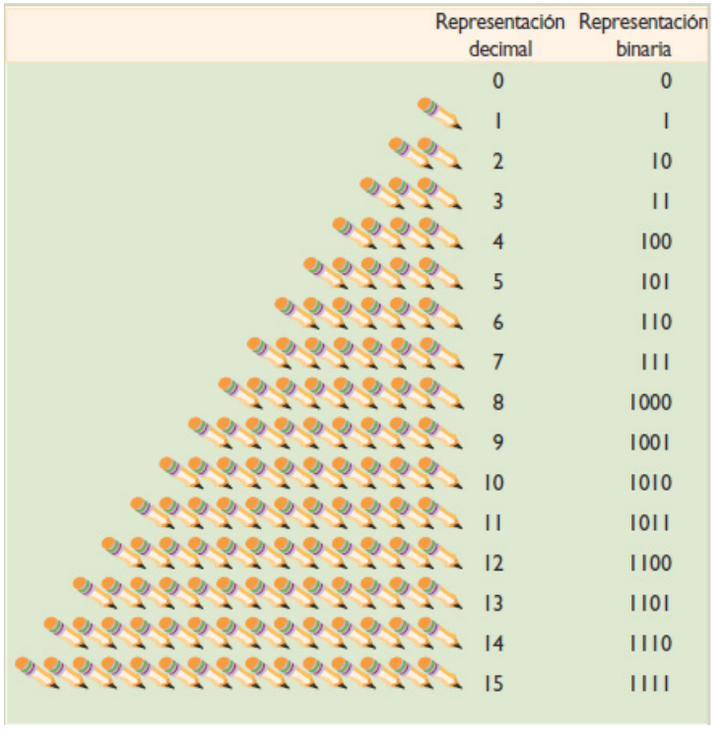
Dada la información (instrucciones de programas, imágenes, texto, sonidos o valores matemáticos) es representada por patrones de conmutadores microscópicos. En la mayoría de los casos, estos grupos de conmutadores representan números o códigos numéricos. El conmutador de fabricación más fácil es el que conmuta entre on y off: tiene sólo dos posiciones, on y off, como un conmutador de luz ordinario. Es la clase de conmutador utilizado en las computadoras modernas. La aritmética binaria sigue las mismas reglas que la aritmética decimal ordinaria. Pero con sólo dos dígitos disponibles para cada posición, tiene que tomar y llevar (manipular dígitos en otras posiciones) más a menudo. Incluso la adición de 1 y 1 da como resultado un número de dos dígitos. La multiplicación, la división, los números negativos y las fracciones también pueden representarse en código binario, pero la mayoría de la gente lo encuentra confuso y complicado comparado con el sistema decimal usado comúnmente.

1. En el sistema numérico decimal, la posición de un dígito es importante: en el número 7.357, el 7 de la izquierda representa siete mil, y el otro representa 7 unidades. El uso de conmutadores para representar números sería fácil de entender si los conmutadores tuvieran 10 posiciones (0 a 9).

2. En el sistema binario, los valores posicionales son potencias de 2, no de 10. Empiezan por 1 (el lugar de la unidad) y doblan de valor por cada lugar adicional. Cada conmutador representa un bit, y el conjunto de ocho conmutadores es un byte.

3. Un byte (8 bits) puede representar cualquier número entre 0 y 255. Si todos los conmutadores están apagados, el valor representado es 0; si los ocho conmutadores están encendidos, el valor es 255 (1 + 2 + 4 + 8 + 16 + 32 + 64 + 128).

4 Los números mayores que 255 se representan utilizando bytes múltiples, llamados palabras. Por ejemplo, una palabra de 2 bytes puede representar números de 0 a 65.535.



Cuadro Nº 1 : Representación Binaria de los números
Fuente: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación*

2.2 Bits, bytes y palabras que zumban

Intentar aprender de computadoras examinando su funcionamiento a nivel de los bits es como intentar aprender sobre la apariencia de la gente o sus actos estudiando las células humanas; hay allí mucha información, pero no es el modo más eficaz de hallar lo que necesita saber. Afortunadamente, las personas pueden utilizar las computadoras sin pensar en los bits. Sin embargo, alguna terminología relacionada con los bits viene en el funcionamiento cotidiano de la computadora. La mayoría de los usuarios necesitan tener al menos un conocimiento básico de los siguientes términos para cuantificar los datos:

- Byte: grupo lógico de 8 bits. Si trabaja sobre todo con palabras, puede considerar un byte como un carácter del texto codificado en ASCII.
- KB (kilobyte o K): unos 1.000 bytes de información. Por ejemplo, se necesitan unos 5 K de almacenamiento para contener 5.000 caracteres de texto ASCII. (Técnicamente, 1 K es 1.024 bytes, porque 1.024 es 210, lo que simplifica el cálculo para las computadoras basadas en el sistema binario. Para aquellos de nosotros que no pensamos en binario, 1.000 se aproxima lo bastante.)
- MB (megabyte o mega): aproximadamente 1.000 KB, o 1 millón de bytes.
- GB (gigabyte o giga): aproximadamente 1.000 MB.
- TB (terabyte): aproximadamente 1 millón de MB o 1 billón de bytes. Esta masiva unidad de medida se aplica a los mayores dispositivos de almacenamiento disponibles actualmente.

- PB (petabyte): este valor astronómico es el equivalente a 1.024 terabytes, o 1.000 billones de bytes. Aunque es improbable que nadie sea capaz de almacenar por ahora 1PB de datos en su PC doméstico, vamos definitivamente en esa dirección.

Las abreviaturas K, MB, GB y PB describen la capacidad de los componentes de almacenamiento y de memoria. Podría, por ejemplo, describir una computadora diciendo que tiene 512 MB de memoria (RAM) y un disco duro diciendo que tiene 120 GB de capacidad de almacenamiento. Los mismos términos se utilizan para cuantificar los tamaños de los archivos. Un archivo es un conjunto organizado de información, tal como un trabajo trimestral o un conjunto de nombres y direcciones, almacenado en un formato legible por la computadora.

Por ejemplo, el texto de este capítulo está almacenado en un archivo que ocupa unos 132 KB de espacio en un disco duro. Para añadir más leña al fuego de la confusión, a menudo la gente mide la velocidad de transferencia de datos o el tamaño de la memoria en megabits (Mb) en lugar de hacerlo en megabytes (MB). Un megabit, como puede suponer, es aproximadamente 1.000 bits (un octavo del tamaño de un megabyte). Cuando hablamos de bits y de bytes, un pequeño detalle como el de las mayúsculas puede suponer una diferencia significativa.

Carácter	Código binario ASCII
A	0 1 0 0 0 0 0 1
B	0 1 0 0 0 0 1 0
C	0 1 0 0 0 0 1 1
D	0 1 0 0 0 1 0 0
E	0 1 0 0 0 1 0 1
F	0 1 0 0 0 1 1 0
G	0 1 0 0 0 1 1 1
H	0 1 0 0 1 0 0 0
I	0 1 0 0 1 0 0 1
J	0 1 0 0 1 0 1 0
K	0 1 0 0 1 0 1 1
L	0 1 0 0 1 1 0 0
M	0 1 0 0 1 1 0 1
N	0 1 0 0 1 1 1 0
O	0 1 0 0 1 1 1 1
P	0 1 0 1 0 0 0 0
Q	0 1 0 1 0 0 0 1
R	0 1 0 1 0 0 1 0
S	0 1 0 1 0 0 1 1
T	0 1 0 1 0 1 0 0
U	0 1 0 1 0 1 0 1
V	0 1 0 1 0 1 1 0
W	0 1 0 1 0 1 1 1
X	0 1 0 1 1 0 0 0
Y	0 1 0 1 1 0 0 1
Z	0 1 0 1 1 0 1 0
0	0 0 1 1 0 0 0 0
1	0 0 1 1 0 0 0 1
2	0 0 1 1 0 0 1 0
3	0 0 1 1 0 0 1 1
4	0 0 1 1 0 1 0 0
5	0 0 1 1 0 1 0 1
6	0 0 1 1 0 1 1 0
7	0 0 1 1 0 1 1 1
8	0 0 1 1 1 0 0 0
9	0 0 1 1 1 0 0 1

Figura Nº 2: Código binaria de los caracteres
Fuente: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación*

LECTURA SELECCIONADA Nº 1:

CHARLES BABBAGE, LADY LOVELACE Y LA MADRE DE TODAS LAS COMPUTADORAS

Autor: Beekman,, George., Pacheco, Rosbinda.,Tábora Alez.

La Máquina analítica Lady Lovelace, considerada como la madre de todas las computadoras, fue concebida por Charles Babbage, un profesor de matemáticas del siglo XIX de la universidad de Cambridge. Babbage era un excéntrico genio conocido por el público por su aversión a los músicos callejeros y los esfuerzos que hizo por ilegalizarlos. Pero este personaje era algo más que un irascible excéntrico; entre sus muchos inventos se cuentan la llave maestra, el cuentakilómetros y... la computadora.

La visión de la computadora de Babbage surgió debido a la frustración que sentía durante el tedioso, y con frecuencia erróneo, proceso de creación de tablas matemáticas. En 1823 recibió una subvención del gobierno británico para desarrollar una «máquina distinta», un dispositivo mecánico para llevar a cabo sumas numéricas continuas. Dos décadas antes, Joseph-Marie Charles Jacquard, un fabricante textil francés, había desarrollado un telar que era capaz de reproducir automáticamente patrones de hilado mediante la lectura de información codificada en tarjetas de papel rígido punteado. Una vez estudiado el telar programable de Jacquard, Babbage abandonó esa idea y atacó un plan mucho más ambicioso: una **Máquina analítica** que pudiera ser programada con el mismo tipo de tarjetas y que fuera capaz de llevar a cabo cálculos con 20 dígitos de precisión. El diseño de Babbage incluía los cuatro componentes básicos que se pueden encontrar en las computadoras actuales: entrada, salida, procesamiento y almacenamiento.

Augusta Ada King (1815-1852), Condesa de Lovelace (llamada a veces de forma errónea «Ada Lovelace») e hija del poeta Lord Byron, visitó a Babbage y su Máquina analítica. Ada se carteaba frecuentemente con él, y con frecuencia se suele decir que fue la primera programadora de computadoras porque escribió un plan para usar la Máquina analítica de forma que permitiera calcular los números de la secuencia de Bernoulli. Pero, probablemente, **programadora** es el término más erróneo para describir su contribución actual. Fue mucho más la intérprete y promotora del visionario trabajo de Babbage.

Babbage estaba obsesionado con completar la Máquina analítica. Eventualmente, el gobierno retiró el soporte económico, ya que no existía una demanda pública lo suficientemente importante como para justificar este importante costo. La tecnología del momento no era suficiente como para llevar a cabo sus ideas. El mundo no estaba preparado para las computadoras, y no lo estaría por otros 100 años.

VIDEOS

Video 6: Los números binarios.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: [Programa] Alterados por Pi: Números binarios.
URL: https://youtu.be/iJkXq9kmQnc?t=26s
Duración: 8 min 24 s.
Autor(a): Canal Encuentro (Argentina).
Expositor(a): Adrián Paenza.
Año: 2008.
Licencia: YouTube estándar.

Video 7: Tipos de Organizadores.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Tipos de organizadores.
URL: https://youtu.be/mvo6KhMN5sI?t=9s
Duración: 2 min 29s.
Autor(a): D.R.
Año: 2011.
Licencia: YouTube estándar.

ACTIVIDAD FORMATIVA Nº 2

Elabora un organizador Visual con las funciones básicas de un computador y sus unidades de representación. Realiza la conversión de 5 números (0 al 100) a base binaria según el cuadro Nº 3

INSTRUCCIONES:

1. Lee y analiza los contenidos del tema N° 2
2. Extrae las funciones básicas del funcionamiento de un computador
3. Identifica las Unidades de representación y extrae la función que cumplen en forma muy resumida,
4. Con los datos obtenidos, diseña un organizador Visual, aplicando su creatividad e imaginación para organizar todos los datos obtenidos dentro del cuadro.
5. Complementa la información observando el video: Tipos de Organizadores https://www.youtube.com/watch?v=mvo6KhMN5sI
6. Realiza la conversión de 5 números (0 al 100) a base binaria según el cuadro N° 3
7. Envía su trabajo al aula virtual

TEMA Nº 3: HARDWARE Y SOFTWARE

Probablemente nos hemos hecho esta pregunta ¿Qué partes físicas forman una computadora? Y podemos ser aún más curiosos y pretender conocer el rol que cumple cada parte dentro de computadora. Revisaremos y explicaremos las interrogantes en el desarrollo de este tema y podremos reconocer que no sólo es importante lo físico (hardware) sino que es vital el software que hace posible el real funcionamiento de las computadoras.

El hardware y el software son muy importantes para el funcionamiento de una computadora se necesita a ambos, uno depende del otro, uno sólo no podría realizar las tareas designadas.

1. UNIDAD CENTRAL DE PROCESAMIENTO[3]

La CPU, a menudo llamada sólo procesador, realiza las transformaciones de entrada en salida. Cada computadora tiene al menos una CPU para interpretar y ejecutar las instrucciones de cada programa, para hacer las manipulaciones aritméticas y lógicas de datos, y para comunicarse con las otras partes del sistema indirectamente a través de la memoria. Un microprocesador moderno, o CPU, es un conjunto extraordinariamente complejo de circuitos electrónicos. En una computadora de escritorio, la CPU está junto con otros chips y componentes electrónicos en un panel de circuitos. El panel de circuitos que contiene la CPU se llama placa madre o placa base. Actualmente todavía hay en uso muchas clases diferentes de CPU; cuando elige una computadora, el tipo de CPU es una parte importante de la decisión. Aunque hay muchas variaciones de diseño entre ellas, sólo dos factores son importantes para un usuario ocasional: la compatibilidad y el rendimiento.

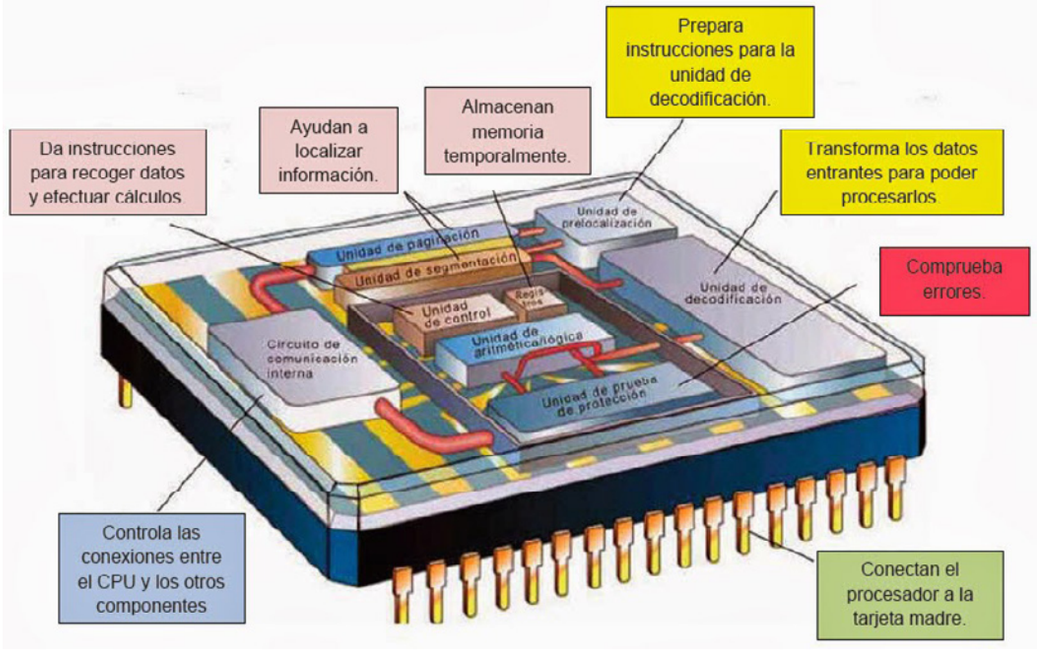


Figura Nº 3: Unidad Central de porcesamiento
Fuente: http://puntoescolar.blogspot.com/2014/03/infokids-2-ficha-2-la-unidad-central-de.html

3 Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación*. México: Pearson Educación.

2. LA MEMORIA DE LA COMPUTADORA

La principal tarea de la CPU es seguir las instrucciones codificadas en los programas. Pero igual que Alicia en el país de las maravillas, la CPU sólo puede manipular una instrucción y unos pocos datos cada vez. La computadora necesita un lugar donde almacenar el resto del programa y los datos hasta que el procesador esté listo. Para eso está la RAM. La RAM (random access memory, memoria de acceso aleatorio) es el tipo más común de almacenamiento primario, o de memoria. Los chips de la RAM contienen circuitos que almacenan temporalmente las instrucciones y los datos del programa.

La computadora divide cada chip de la RAM en muchas ubicaciones de memoria del mismo tamaño. Las ubicaciones de memoria, como las casas, tienen direcciones únicas para que la computadora pueda indicarles aparte cuándo ha de guardar o recuperar la información. Puede almacenar un fragmento de información en cualquier ubicación RAM (puede elegir una al azar) y la computadora puede, si se le indica, recuperarla rápidamente. De aquí el nombre de memoria de acceso aleatorio. La información almacenada en la RAM no es más que un patrón de corriente eléctrica fluyendo a través de circuitos microscópicos en chips de silicio. Esto significa que cuando la energía sale de la computadora se olvida instantáneamente de todo lo que se recordaba en la RAM.

La RAM se llama a veces memoria volátil, porque la información almacenada allí no se mantiene permanentemente. Esto podría ser un serio problema si la computadora no tuviera otro tipo de memoria donde almacenar la información que no se quiere perder. Esta memoria no volátil se llama ROM (read-onlymemory, memoria de sólo lectura) porque la computadora sólo puede leer la información almacenada en ella; nunca puede escribir ninguna información nueva. Todas las computadoras modernas incluyen ROM que contiene instrucciones de arranque y otra información crítica. La información de la ROM fue grabada en ella cuando se fabricó el chip, así que está disponible siempre que la computadora está funcionando, pero no puede cambiarse salvo reemplazando el chip de la ROM. Hay otros tipos de memoria disponibles; la mayoría se utiliza raramente fuera de los laboratorios de ingeniería. Hay dos excepciones notables:

- La CMOS (complementary metal oxide semiconductor, semiconductor complementario de óxido de metal) es una clase especial de RAM de baja energía que puede almacenar pequeñas cantidades de datos durante largos periodos de tiempo con la energía de la batería. La CMOS RAM almacena la fecha, la hora y el calendario de una PC. (La CMOS RAM se llama parameter RAM [PRAM] en las Macintosh.)

- Los chips de memoria Flash, como los chips de la RAM, pueden escribirse y borrarse rápida y repetidamente. Pero a diferencia de la RAM, la memoria flash no es volátil; puede mantener sus contenidos sin flujo de electricidad. Las cámaras digitales, los teléfonos móviles, las computadoras portátiles, las de bolsillo, las PDA y otros dispositivos digitales utilizan memoria flash para almacenar los datos que necesitan cambiarse de vez en cuando. Los grabadores de datos del vuelo también la utilizan. La memoria flash es aún demasiado cara para sustituir a la RAM y otros medios comunes de almacenamiento, pero puede que en el futuro sustituya a los discos duros, así como a los chips de memoria.

Recuperar los datos de la memoria lleva algún tiempo al procesador, pero no mucho. El tiempo de acceso de la mayoría de las memorias se mide en nanosegundos (ns), milmillonésimas de segundo. Compare esta cifra con el tiempo de acceso del disco duro, que se mide en milisegundos (ms), milésimas de segundo. La velocidad de la memoria (el tiempo de acceso) es otro factor que afecta a la velocidad global de la computadora.

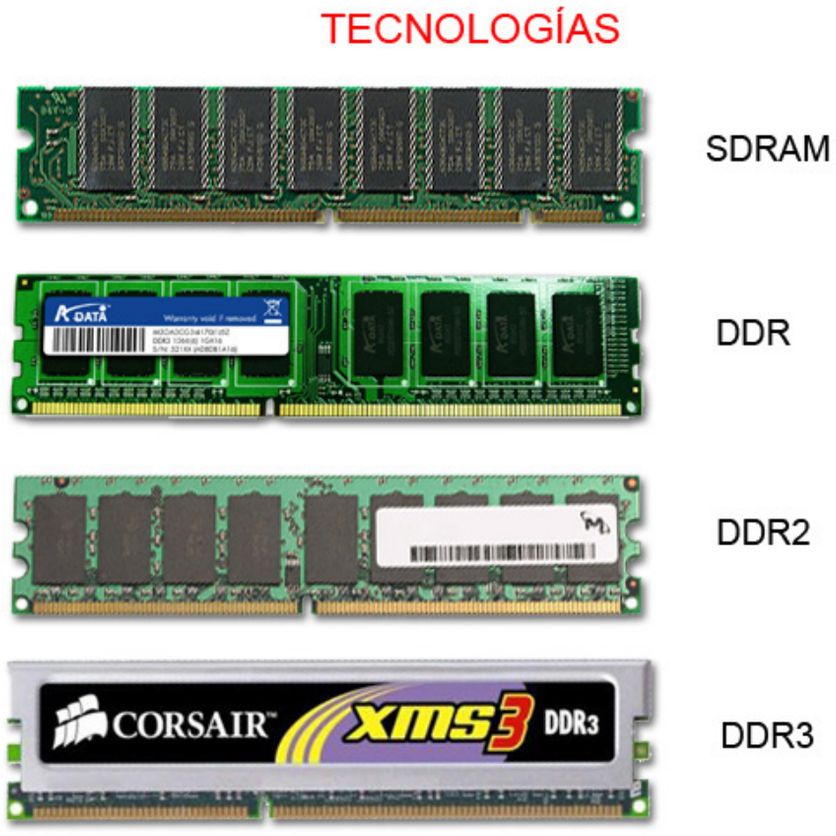


Figura Nº 4 Memoria del Computador
Fuente: http://blog.phonehouse.es/2009/12/01/informatica-para-todos-la-memoria-ram/

2. BUSES, PUERTOS Y PERIFÉRICOS

En una computadora portátil, la CPU, los chips de memoria y otros componentes clave están integrados en la placa madre. La información viaja entre los componentes de la placa madre a través de grupos de cables llamados buses del sistema, o sólo buses. Los buses tienen generalmente 32 o 64 cables, o rutas de datos; un bus de 32 cables se llama bus de 32 bits, porque puede transmitir 32 bits de información a la vez, el doble que un antiguo bus de 16 bits. Igual que las autopistas de muchos carriles permiten a grandes masas de automóviles desplazarse a mayor velocidad que las carreteras de un solo carril, los buses más amplios pueden transmitir la información más deprisa que los buses más estrechos. Las computadoras nuevas, más potentes, tienen buses más amplios para que puedan procesar la información con más rapidez.

Los buses conectan con los dispositivos de almacenamiento situados en las bahías (áreas abiertas en la caja del sistema para discos duros y otros dispositivos). Los buses también pueden conectarse a las ranuras de expansión (a veces llamadas sólo ranuras o slots) dentro de la computadora. Los usuarios pueden personalizar sus computadoras insertando paneles de circuitos con propósitos especiales (llamados tarjetas de expansión, o sólo tarjetas) en esas ranuras. Los buses también se conectan a buses externos y a puertos (conectores en el exterior del chasis de la computadora). La parte posterior de una computadora tiene generalmente varios tipos de puertos para cubrir distintas necesidades.

Algunos de esos puertos (donde puede conectar el teclado y el ratón, por ejemplo) están conectados directamente a la placa del sistema. Otros, como el puerto del monitor, están disponibles generalmente mediante una tarjeta de expansión. De hecho, muchas tarjetas de expansión hacen poco más que proporcionar los puertos

convenientes para conectar tipos particulares de periféricos. Otro ejemplo son las portátiles, en las que podemos encontrar generalmente una o dos ranuras de tarjeta PC para añadir tarjetas del tamaño de una tarjeta de crédito.

En las computadoras portátiles, en las que el tamaño es crítico, la mayoría de los puertos comunes van directamente a la tarjeta del sistema. Como las computadoras portátiles no tienen espacio para las tarjetas de tamaño grande, muchas tienen ranuras para tarjetas PC (tarjetas del tamaño de una tarjeta de crédito que contienen memoria, periféricos en miniatura y puertos adicionales). (Cuando salieron por primera vez estas tarjetas, se conocían por el nombre de tarjetas PCMCIA. Un escritor sugirió humorísticamente que eran las siglas de People Can't Memorize Computer Industry Acronyms [la gente no puede entender los acrónimos de la industria informática], aunque el desafortunado acrónimo significa realmente Personal Computer Memory Card International Association. Por suerte, el nombre se acortó por el más simple de tarjeta PC.) Las ranuras y los puertos facilitan la adición de dispositivos externos, llamados periféricos, al sistema de la computadora, para que la CPU pueda comunicarse con el mundo exterior y almacenar información para su uso posterior. Sin periféricos, la CPU y la memoria juntas son como un cerebro sin cuerpo.

Algunos periféricos, como teclados e impresoras, sirven como vínculos de comunicación entre las personas y las computadoras. Otros periféricos vinculan a la computadora con otras máquinas. Otros proporcionan medios de almacenamiento a largo plazo.

3. EL SOFTWARE COMO LENGUAJE DE LAS COMPUTADORAS

Por desgracia, la computadora reconoce sólo ceros y unos. Un gran abismo separa a la persona que tiene una colección de vagos problemas del árido y rígidamente delimitado mundo de las computadoras. ¿Cómo pueden los humanos traspasar esta grieta y poder comunicarse con la computadora? Éste es el punto en el que el software entra en acción. El software permite que las personas puedan contarle a la computadora cierto tipo de problemas y que ésta a su vez les ofrezca algún tipo de solución a los mismos.

El software actual no se ha materializado de la nada; ha evolucionado a partir de las placas base y otros tipos de dispositivos hardware que fueron usados para programar las primeras computadoras como la ENIAC. El matemático John von Neumann, que trabajó con los creadores de la ENIAC, J. Presper Eckert y John Mauchly, escribió un informe en 1945 en el que sugería que las instrucciones de un programa podían almacenarse en la memoria junto con los datos. Cada computadora creada desde entonces se ha basado en el concepto de programa almacenado descrito en dicho informe.

La idea estableció la industria del software. En lugar de jugar con interruptores o de parchear cables, los programadores de hoy en día escriben programas (un conjunto de instrucciones informáticas diseñadas para resolver problemas) y las introducen en la memoria de la computadora mediante teclados o cualquier otro dispositivo de entrada. Estos programas constituyen el software de la máquina. Debido a que está almacenado en memoria, una computadora puede cambiar de una tarea a otra y volver después a la primera sin necesidad de modificar el hardware.

Por ejemplo, la computadora que sirve como procesador de textos para escribir este libro puede, con un simple clic de ratón, convertirse en un cliente de correo electrónico, un navegador Web, una hoja de cálculo, una estación de trabajo para la edición de vídeo, un instrumento musical o una máquina de juegos. ¿Qué es el software y cómo puede cambiar un amasijo de circuitos en un camaleón electrónico? Se ofrecen algunas respuestas generales a esta pregunta mediante detalles de las tres categorías principales de software:

- Compiladores y otros programas de traducción, los cuales permiten que los programadores creen otro software.
- Aplicaciones, que sirven como herramientas productivas para ayudar a los usuarios a resolver sus problemas.

- Software de sistema, que es el encargado de coordinar las operaciones de hardware y que se encuentra en la trastienda que raramente ve el usuario de un sistema informático

4. APLICACIONES COMO HERRAMIENTAS PARA EL USUARIO.

Las aplicaciones de software permiten a los usuarios controlar sus computadoras sin pensar del mismo modo que los programadores. Vamos a centrar ahora nuestra atención en ellas.

¿Por qué usamos aplicaciones?

Podría sonar raro que alguien pagase una cantidad de dinero por un producto que no tiene garantía y que incluye docenas de restricciones legales relacionadas con su uso. De hecho, el rápido crecimiento de la industria de software ha producido una gran cantidad de programas que han vendido millones de copias. ¿Por qué tanta gente compra y usa este tipo de programas? Desde luego, la respuesta varía de una persona a otra y de un producto a otro. **Pero, en general, los programas de más éxito comparten dos importantes rasgos:**

- Están construidos alrededor de metáforas visuales de herramientas del mundo real. Un programa de dibujo convierte la pantalla en una hoja de papel y una colección de herramientas de dibujo. Las hojas de cálculo reúnen las cuentas de un libro de contabilidad. El software de edición de vídeo sitúa en pantalla los familiares controles de un vídeo. Pero si estos programas solamente copiaran a sus equivalentes en la vida real, la gente no se habría decidido a utilizarlos.

- Expanden, de alguna forma, las capacidades humanas. Los programas más populares permiten que las personas realicen cosas que no pueden llevarse a cabo de una manera sencilla o, en todo caso, con herramientas convencionales. Un artista que use un programa de gráficos puede fácilmente cambiar el color del pelo de una imagen y volver atrás si el resultado no es el adecuado. Las hojas de cálculo permiten que los directivos calculen los ingresos futuros basándose en las mejores proyecciones para, a continuación, recalcular instantáneamente todos los datos con unos valores diferentes. Y las posibilidades abiertas gracias al software de edición de vídeo van más allá de nuestra imaginación. Cualquier tipo de aplicación que impulse las capacidades humanas es la fuerza impulsora que se esconde tras la revolución informática.

PRUEBA DE DESARROLLO Nº1

Elabora un cuadro de doble entrada comparativo, señalando las partes constitutivas de un computador (Hardware y software) y explica las funciones que cumple cada una acompañando el esquema de cada parte.

Instrucciones:

- Lee y analiza, todos los contenidos de tema N° 3
- Identifica las partes del hardware y extrae máximo en 2 líneas la idea principal que define la función de cada una de ellas
- Identifica las partes del Software y resume en 2 líneas las funciones de cada una de ellas.
- Busca en internet, el esquema de cada una de las partes del computador para complementar su cuadro comparativo.
- Diseña, un cuadro de doble entrada comparativo en forma creativa para localizar las partes y funciones de cada una, acompañándolas de un esquema de la parte que corresponde.
- Complementa la información observando el video: COMO HACER UN CUADRO DE DOBLE ENTRADA https://www.youtube.com/watch?v=nU8tScHyoYs
- Considera darle a tu producto final, coherencia, pertinencia, organización y presentación motivadora. Envíalo al aula Virtual.

RUBRICA DE EVALUACIÓN PARA UN CUADRO DE DOBLE ENTRADA

Nombre del estudiante: _____

Sección: _____ Fecha: _____

INDICADORES CRITERIOS	4 EXCELENTE	3 BUENO	2 REGULAR	1 INSUFICIENTE	TOTAL
Variables	Las variables, temas o características son claramente identificados.	Descripción clara de algunas variables o temas pero con detalles poco específicos	Descripción ambigua del cuadro con detalles poco claros	Descripción incorrecta del cuadro, sin detalles significativos	
Descripción sobre el tema	Descripciones muy detalladas entre la unión de ambas variables y los valores que pueden tener. Acompaña esquemas	Descripciones poco detalladas entre la unión de ambas variables y los valores que puedan tener entre las mismas, acompaña esquemas	Descripciones incorrectas de algunos uniones de ambas variables y los valores que pueden tener entre ellas sin esquemas	Falta de detalle entre las descripciones en la unión de ambas variables y los valores que pueden tener entre ellas	
Orden y diseño	Cuadro horizontal con un aspecto en la parte superior horizontal y otra en la parte lateral izquierda, con denominación correcta	Cuadro horizontal con un aspecto en la parte superior horizontal y otra en la parte lateral izquierda, pero con error al nombrar los aspectos	Un aspecto en la parte superior o inferior y otro en una de las dos partes laterales.	Falta de orden en los aspectos a correlacionar.	
Comparaciones	Las cuadrículas que marcan la intersección correcta de las variables, y comparación clara y correcta entre ambas variables	Las cuadrículas marcan la intersección correcta entre las variables, pero la comparación de algunas variables no son muy claras.	La cuadrícula marca las intersecciones de las correcta de las variables pero algunas comparaciones son incorrectas	La cuadrícula no marca claramente las divisiones y la comparación entre ambas variables no es clara.	
Presentación del cuadro de doble entrada	Presentación fue hecha en tiempo establecido. Se entregó en formato organizado motivador y en digital.	La presentación fue hecha con 24 horas de retraso pero en formato organizado motivador y en digital.	Presentación fue hecha con 48 horas de retraso pero en formato organizado y en digital.	La presentación fuera del tiempo adicional sin formato establecido.	
CALIFICACIÓN DE LA ACTIVIDAD					

TEMA Nº 4: SISTEMA OPERATIVO.

¿Para qué nos sirve Microsoft Windows? Cuando utilizamos una computadora y necesitamos interactuar con el hardware es precisamente el Sistema Operativo de dicha computadora la que nos facilita esta interacción, además Windows, Unix, Linux, DOS, Mac OS, etc. Controlan las asignaciones de memoria, ordenan las solicitudes al sistema, controlan los dispositivos de entrada y salida, facilitan la conexión a redes y el manejo de archivos.

Los Sistemas operativos encargados de la gestión eficiente de los recursos del equipo se caracterizan en proporcionar comodidad al usuario al momento de utilizar los equipos.

Un sistema operativo es el encargado de brindar al usuario una forma amigable y sencilla de operar, interpretar, codificar y emitir las ordenes al procesador central para que este realice las tareas necesarias y específicas para completar una acción.

1. LA CONEXIÓN ENTRE EL HARDWARE Y EL SOFTWARE[4]

Ya sea para escribir un documento o un programa, usted no debe preocuparse de los pequeños detalles como la parte de la memoria de la computadora ocupada por ese documento, los segmentos del procesador de textos que se encuentran actualmente en la memoria o las instrucciones de salida enviadas por la computadora a la impresora.

El software de sistema, un tipo de software que incluye el sistema operativo y los programas de utilidad, es el encargado de gestionar estos detalles y otros muchos sin que usted se entere de ello.

¿Qué hace un sistema operativo?

Virtualmente, cada computadora actual, ya sea una supercomputadora de tiempo compartido o un portátil, depende de un SO (sistema operativo) que mantenga el hardware funcionando de forma eficiente y facilite el proceso de comunicación con él.

El sistema operativo está ejecutándose continuamente desde el mismo momento en el que se enciende la computadora, proporcionando una capa de aislamiento entre usted y los bits y bytes que componen el mundo del hardware de la computadora. Ya que se encuentra entre el software y el hardware, la compatibilidad de las aplicaciones suele venir determinada por el sistema operativo así como por el hardware.

El sistema operativo, como su propio nombre indica, es un sistema de programas que llevan a cabo una serie de operaciones técnicas, desde la comunicación básica con los periféricos a complejas tareas de comunicación y seguridad dentro de una red.

2. INTERFAZ DE USUARIO HOMBRE-MÁQUINA

Los usuarios de las primeras computadoras tenían que gastar grandes cantidades de tiempo escribiendo y depurando instrucciones en lenguaje máquina. Más adelante empezaron a utilizar lenguajes de programación que eran más sencillos de entender aunque aún seguían siendo complicados en el ámbito técnico.

4 Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación*. México: Pearson Educación.

En la actualidad, los usuarios emplean la mayoría de su tiempo en el trabajo con aplicaciones ya programadas, como los procesadores de texto, que simulan y aumentan las posibilidades de herramientas reales. A medida que el software evoluciona, también lo hace la interfaz de usuario (el aspecto y el comportamiento de una computadora desde el punto de vista de un humano).

Administración de ficheros: ¿dónde están mis cosas?

Un sistema operativo actúa como una capa intermedia entre el usuario y los datos contenidos en la computadora. Windows y Mac OS emplean una interfaz de usuario que representa los datos de la computadora como ficheros almacenados en carpetas que se encuentran en un escritorio virtual. Al igual que ocurre con un escritorio real, estos archivos pueden estar esparcidos por todo el sistema, haciendo difícil su administración. Una solución a este problema es organizar los ficheros de datos de un modo lógico. Para ello, tanto Windows como Mac soportan el concepto de carpetas de sistema con nombres autoexplicativos.

De igual forma, las fotografías digitales podrían encontrarse en Mis imágenes (Imágenes) y los archivos de música digital en Mi música (música). Estas carpetas son específicas de cada usuario, por lo que si varios de ellos acceden a la misma PC tendrán sus datos perfectamente localizados. Los sistemas operativos actuales incluyen herramientas de búsqueda que pueden ayudar a encontrar ficheros almacenados en cualquier parte, En Windows, es posible buscar por nombre de fichero, aunque también puede hacerse por palabras o frases contenidas en los mismos. De este modo, si no recuerda el nombre que le dio a un fichero al guardarlo (cosa bastante común) siempre puede utilizar alguna frase contenida en el mismo para localizarlo. En Mac OS X, se puede localizar información almacenada en su disco duro mediante la herramienta integrada Buscar, que es similar a la de Windows. Mac OS X también incluye otra herramienta especial llamada Sherlock para buscar información online.

Los comandos de búsqueda están diseñados para contestar a una pregunta que cualquier usuario de una computadora se ha hecho alguna vez: ¿dónde están mis cosas? Windows y Mac fueron diseñados cuando los discos duros de gran capacidad de los que disponemos en la actualidad sólo eran una quimera, y el espacio de un disquete de antaño parecía no acabarse nunca. A medida que nuestras máquinas crecen para almacenar más datos y de mayor importancia, Apple y Microsoft están desarrollando nuevas técnicas para ayudarnos a mantener nuestros ficheros de datos. En los últimos años, Apple ha enfatizado el papel de Macintosh como medio digital con aplicaciones como iTunes, iMovie e iPhoto. Pero la proliferación de archivos digitales en los discos duros de cualquiera de estas máquinas está haciendo que la GUI de ventanas y carpetas sea menos efectiva a la hora de localizar una canción, una fotografía o una película. Apple ha incluido una interfaz basada en vistas en muchas de sus aplicaciones digitales; esta novedosa interfaz es un modo sencillo de organizar y localizar sus ficheros. Por ejemplo, iTunes soporta la noción de lista de reproducción que permite dividir una librería completa de canciones en bloques más digeribles.

Algunas de estas listas las genera directamente el usuario. Otras son creadas automáticamente por el programa, y entre ellas se incluyen cosas como Top 25 Most Played y 60's Music. Apple iTunes también soporta listas de reproducción inteligentes, las cuales pueden rellenarse automáticamente con canciones basadas en criterios de bases de datos. Por ejemplo, podría puntuar las canciones de su biblioteca de música y después crear una lista de reproducción inteligente que mostrase sólo aquéllas que hayan recibido la mayor puntuación.

Este tipo de listas es una relación viva de sus canciones favoritas, ya que cualquier cambio en la puntuación de estas canciones modificará el contenido de la lista. La siguiente versión de Windows incluirá características de bases de datos en el sistema de ficheros que permitirá localizar de forma fácil información almacenada en cualquier parte de su PC. Al igual que las listas inteligentes del Apple iTunes, esta característica ayudará a los usuarios a localizar sus datos de un modo rápido y fácil, manteniéndolos protegidos del sistema subyacente. Y según nos vayamos desplazando hacia entornos de computación distribuidos, en donde los datos pueden encontrarse en diferentes sistemas como una red o Internet, estas tecnologías serán cada vez más valiosas.

LECTURA SELECCIONADA Nº. 2:

LINUS TORVALDS Y EL SOFTWARE QUE NO ES PROPIEDAD DE NADIE

Autor: Beekman,, George., Pacheco, Rosbinda.,Tábora Alez

Cuando Linus Torvalds compró su primera PC en 1991, nunca pensó que iba a convertirse en un arma fundamental en la guerra de liberación del software. Sólo quería evitar tener que esperar para conseguir una terminal que le permitiera conectar con el *mainframe* de su universidad. Torvalds, un estudiante de 21 años de la universidad de elsinki en Finlandia, había eludido comprar una PC porque no le gustaba «ni su pésima arquitectura ni su pésimo sistema operativo MS-DOS». Pero Torvalds había estudiado sistemas operativos, y decidió construirse uno propio. Basó su trabajo en Minix, una versión de libro de texto a baja escala del sistema operativo UNIX diseñado para funcionar en una PC. Poco a poco, fue uniendo las piezas de un *kernel*, la parte del sistema operativo en el que se llevan a cabo realmente las tareas de procesamiento y de control.

Cuando comentó su proyecto en un foro de discusión de Internet, un miembro del mismo ofreció su espacio para colocarlo en un servidor de una universidad. Otros lo copiaron, se ocuparon ligeramente de él y le devolvieron los cambios a Torvalds. El trabajo en comunidad se convirtió eventualmente en lo que se conoce hoy en día como **Linux** (pronunciado por su creador como «Linn-uks»). En un par de años, fue lo suficientemente bueno como para distribuirse como producto.

En lugar de registrar y vender Linux, Torvalds hizo que fuera totalmente libre bajo la GPL (Licencia pública general, *General Public License*) desarrollada por la *Free Software Foundation*. Según la GPL, cualquiera puede obtener, modificar e incluso vender Linux, siempre que el código fuente (las instrucciones del programa) permanezca disponible de forma gratuita para el resto de personas que deseen mejorarlo. Linux es el mejor ejemplo de lo que se conoce como **software de código abierto**, y en la actualidad es la punta de lanza del popular movimiento a favor de este tipo de software.

Miles de programadores de todo el mundo han trabajado con Linux, con Torvalds en el centro de la actividad. Algunos lo hacen porque creen que debería haber alternativas a los caros productos comerciales; otros porque pueden optimizar el software; y otros, simplemente, porque es divertido. Como resultado de todos estos esfuerzos, Linux se ha convertido en un potente y versátil producto con millones de usuarios.

En la actualidad, Linux impulsa servidores Web, estaciones de trabajo de filmación y animación, supercomputadoras científicas, un puñado de computadoras de bolsillo, algunas PC de propósito general e incluso electrodomésticos inteligentes con acceso a Internet (por ejemplo, frigoríficos). Linux es especialmente popular en los lugares en los que se utiliza la informática con un bajo presupuesto, particularmente en países del Tercer Mundo.

El éxito de Linux ha inspirado a Apple, Sun, Hewlett-Packard y otras empresas de software a liberalizar productos de código abierto. Incluso la todopoderosa Microsoft está prestando atención a medida que este sistema operativo está creciendo en popularidad, y ha respondido con una estrategia de código pseudo-abierto que incluye productos que compiten directamente con Linux.

En la actualidad, Torvalds es un héroe para la gente de Internet. Las páginas Web rinden homenaje a su persona, a su creación y a Tux, el pingüino que se ha convertido en la mascota de Linux. En 1996, completó su titulación en informática y se puso a trabajar en Transmeta Corp, una compañía de diseño de chips ubicada en Silicon Valley. Sin embargo, aun gasta horas y horas de su tiempo conectado con las legiones Linux, mejorando un sistema operativo que pertenece a todo el mundo, y a nadie.

ACTIVIDAD FORMATIVA Nº 3

Participa en un Foro Debate sobre la importancia del software de base y reforzamiento de todos los temas desarrollados en la Unidad

Instrucciones:

- Lee y analiza, el tema N" 4 y extrae las ideas fundamentales
- Organiza los datos extraídos y determina la importancia del Software de base
- Consulta páginas Web para observa las partes de un computador de buena procedencia acerca del tema en referencia.
- Participa en el Foro debate, opinando sobre el tema
- Analiza críticamente los planteamientos hechos por sus compañeros
- Elabora conclusiones finales.

PRUEBA OBJETIVA

INSTRUCCIONES:

Lea cuidadosamente cada enunciado y responda según se requiera (Remarque/escriba con color azul su respuesta).

1. La flexibilidad de la computadora no se encuentra en el _____, se encuentra en el _____: **(1 punto)**

   a. Hardware – Software.

   b. Hardware - Memorias.

   c. Software - CPU.

   d. Software – Hardware

   e. Software – Monitor.

2. Red creada en 1969 que fue la semilla de Internet: **(1 punto)**

   a. ARPANET.

   b. HDMI.

   c. CPU.

   d. WIFI.

   e. ARPAWIFI.

3. No es una función de las computadoras: **(1 punto)**

   a. Conocer el futuro.

   b. Recibir una entrada.

   c. Procesar información.

   d. Producir una salida.

   e. Almacenar información.

4. Es la unidad más pequeña de información que puede procesar una computadora: **(1 punto)**

   a. Bit.

   b. Byte.

   c. MB.

   d. GB.

   e. TB.

5. Cantidad de bits que existen en 01 byte: **(1 punto)**

   a. 8 bits.

   b. 16 bits.

   c. 32 bits.

   d. 64 bits.

   e. 12 bits.

6. Realizar la conversión de los siguientes números a Base 2 (debe mostrar el procedimiento)

7. Explique brevemente el significado de:

Número 100 **(2 puntos)**	Número 95 **(2 puntos)**
Número 64 **(2 puntos)**	Número 89 **(2 puntos)**

-

Sistema Operativo **(2 puntos)**

- Aplicación de software **(2 puntos)**
- Placa Madre **(2 puntos)**
- Memoria RAM **(2 puntos)**
- Unidad Central de Procesamiento **(2 puntos)**

GLOSARIO DE LA UNIDAD I

B

BIT
Unidad de medida de información que equivale a la selección entre dos alternativas que tienen el mismo grado de probabilidad.

BYTE
Unidad de información utilizada como un múltiplo del bit.

C

CPU
Unidad central de procesamiento

AUTOEVALUACION Nº 1

INSTRUCCIONES: Lea cuidadosamente cada enunciado y responda según se requiera (Remarque/escriba con color azul su respuesta).

1. El _____, inventado en 1948 podía realizar la misma tarea que las válvulas de vacío.

   a. Transistores.

   b. Válvulas de vacío.

   c. CPU.

   d. Bus de datos.

   e. Chip.

2. Si en un byte los 8 conmutadores están encendidos el valor que representan es:

   a. 255

   b. 128

   c. 32

   d. 16

   e. 8

3. Los bytes múltiples son utilizados para representar números mayores a :

   a. 255

   b. 128

   c. 32

   d. 16

   e. 8

4. Un Petabyte (PB) representa a :

   a. 1024 terabytes.

   b. 1024 Gigabytes

   c. 1024 Megabytes.

   d. 1024 bytes

   e. 1024 bits.

5. Interpreta y ejecuta las instrucciones de lso programas para hacer las manipu-laciones aritméticas y lógi-cas de datos y para comunicarse con las otras partes del sistema indirectamente a través de la memoria:

   a. CPU (Unidad Central de Procesamiento)

   b. Memoria de la computadora

   c. Bus de datos.

   d. Memoria RAM

   e. Monitor.

6. Lugar en el cual se almacenan los datos y el resto del programa hasta que el procesador esté listo.

   a. Memoria de la computadora

   b. CPU (Unidad Central de Procesamiento)

   c. Bus de datos.

   d. Memoria RAM.

   e. Monitor.

7. La información viaja entre los componentes de la Placa Madre a través de gru-pos de cables llamados.

   a. Memoria de la computadora

   b. CPU (Unidad Central de Procesamiento)

   c. Bus de datos.

   d. Memoria RAM.

   e. Monitor.

8. Es un software que nos permite crear nuevo software.

   a. Lenguajes de programación.

   b. Sistema Operativo

   c. Sql Server.

   d. Microsoft Word.

   e. Microsoft Power Point.

9. Es un software del tipo Sistema Operativo.

   a. Microsoft Windows

   b. Microsoft Office

c. Sql Server.

d. Microsoft Word.

e. Microsoft Power Point.

10. Actúa como una capa intermedia entre el usuario y los datos contenidos en la computadora.

a. Sistema Operativo.

b. Prolog.

c. Sql Server.

d. Microsoft Word.

e. Microsoft Power Point.

BIBLIOGRAFÍA DE LA UNIDAD I

- Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación.* México: Pearson Educación.
- Joyanes, Luis. (2008). Fundamentos de Programación. Madrid: Mc Graw Hill.

UNIDAD II

ALGORITMOS

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD

CONTENIDOS	ACTIVIDADES FORMATIVAS (HABILIDADES Y ACTITUDES)	SISTEMA DE EVALUACIÓN (TÉCNICAS Y CRITERIOS)
TEMA Nº 1: Algoritmos. 1 Definición de algoritmo 2 Diseño del algoritmo 3 Flujogramas y Diagramas **Tema Nº 2: Estructura** 1 Estructura Secuencial, Selectiva y Repetitiva	• Explica algoritmos usando el entorno empresarial y **elabora flujograma para representar un algoritmo.** • **Elabora un flujogramas que represente a un algoritmo utilizando estructuras secuencial, selectivas, repetitivas.**	**Procedimientos e indicadores de evaluación permanente** • Entrega puntual de trabajos realizados. • Calidad, coherencia y pertinencia de contenidos desarrollados. • Prueba teórico-práctica, individual. • Actividades desarrolladas en sesiones tutorizadas **Criterios de evaluación para flujogramas:** • **Variables** • **Entradas** • **Proceso** • **Salida** • **Habilidad de comunicación**

RECURSOS:

Vídeos o imágenes:

Tema N° 1 :
Introducción a los Algoritmos
https://www.youtube.com/watch?v=PWgVXuQKrdo

Tema N° 2
Estructura selectiva simple
https://www.youtube.com/watch?v=T9sg17TVgo4
Estructura repetitiva (While)
https://www.youtube.com/watch?v=ERTzfGoCXds
Estructura repetitiva (Do… while)
https://www.youtube.com/watch?v=REQkHnZTS18

Lectura complementaria:

Lectura Seleccionada Nº 1
Espacios virtuales compartidos
Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez

Lectura Seleccionada Nº 2
Inteligencia integrada y computación omnipresente
Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

INSTRUMENTO DE EVALUACIÓN	• Prueba Objetiva. • Prueba de Desarrollo
BIBLIOGRAFÍA (BÁSICA Y COMPLEMENTARIA)	**BÁSICA** JOYANES AGUILAR, Luis. *Fundamentos de programación. Algoritmos, estructuras de datos y objetos*, Tercera Edición. Madrid: Editorial McGraw-Hill, 2003. **COMPLEMENTARIA** PRIETO, A., LLORIS, A. y TORRES, *Introducción a la Informática*, Tercera Edición, Madrid: Editorial McGraw-Hill, 2005.
RECURSOS EDUCATIVOS DIGITALES	LÓPEZ GARCÍA, Juan Carlos, "Algoritmos y Programación", Año 2009 [ref. de 09 de noviembre de 2009].Disponible en Web:<http://www.eduteka.org/GuiaAlgoritmos.php>

TEMA N° 1
ALGORITMOS.

Sabemos que las computadoras pueden resolver varios problemas mediante la ejecución de pasos previamente determinados, pero ¿Cómo podemos hacer que un computador siga (ejecute) estos pasos?, no sólo basta con tener claro el problema sino que debemos realizar los pasos requeridos para su solución (Algoritmo) y estos pasos deben ser traducidos a un lenguaje de programación para su ejecución por la computadora y así tener el ciclo completo para dar solución automatizada a un problema.

La computadora es una máquina que por sí sola no puede hacer nada, necesita ser programada, es decir, introducirle instrucciones u órdenes que le digan lo que tiene que hacer y esa es la función del algoritmo que nos ayudan a poder expresar y diseñar la secuencia lógica que debe seguir la computadora para dar solución al problema planteado.

1. DEFINICIÓN DE ALGORITMO[5]

Un algoritmo es un método para resolver un problema. Aunque la popularidad del término ha llegado con el advenimiento de la era informática, algoritmo proviene de *Mohammed AlKhoWarizmi*, matemático persa que vivió durante el siglo IX y alcanzó gran reputación por el enunciado de las reglas paso a paso para sumar, restar, multiplicar y dividir números decimales; la traducción al latín del apellido en la palabra *algorismus* derivó posteriormente en algoritmo.

El Profesor Niklaus Wirth tituló en uno de sus más famosos libros, Algoritmos + Estructura de datos = Programas, significándolos que sólo se puede llegar a realizar un buen programa con el diseño de un algoritmo y una correcta estructura de datos.

La resolución de un problema exige el diseño de un algoritmo que resuelva el problema propuesto

Figura Nº 5 Resolución de un Problema

Los pasos para la resolución de un problema son:

- Diseño de un algoritmo, que describe la secuencia ordenada de pasos que conducen a la solución de un problema.
- Expresar un algoritmo como un programa en un lenguaje de programación adecuado.
- Ejecución y validación del programa por la computadora.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como de la computadora que los ejecuta.

Dada la importancia del algoritmo en la ciencia de la computación, un aspecto muy importante será el diseño de algoritmos. El diseño de la mayoría de los algoritmos requiere creatividad y conocimientos profundos de la técnica de la programación. En esencia, la solución de un problema se puede expresar mediante un algoritmo.

5 Joyanes, Luis. (2008). Fundamentos de Programación. Madrid: Mc Graw Hill.

Las características fundamentales que debe cumplir un algoritmo es:

- Debe ser preciso e indicar el orden de realización de cada paso.
- Debe estar bien definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Debe ser finito. Si se sigue un algoritmo se debe terminar en algún momento, o sea, debe tener un número finito de pasos

La definición de un algoritmo debe describir tres partes: Entrada, Proceso y salida.

Por ejemplo en un algoritmo de receta de cocina se tendrá:

Entrada: Ingredientes y utensilios empleados.

Proceso: Elaboración de la receta en la cocina.

Salida: Terminación del plato (Ejemplo ceviche).

La computadora se encuentra siempre ejecutando un algoritmo. Por lo general, estos algoritmos, escritos para que los entienda una máquina, terminan siendo vagos y confusos para la mayoría de quienes no han estudiado programación. Una máquina no puede entender "escribe Hola Mundo" porque no sabe lo que es "escribe" ni lo que es una letra o un espacio, ni lo que es una pantalla. En cambio, puede entender:

"mov eax, 0x23afb31" (escribir en el registro eax el número 0x23afb31)

Aunque nosotros no. La computadora es solo un circuito electrónico, no funciona a base de magia ni nada por el estilo. Debido a lo difícil que es escribir en lenguaje máquina, e incluso en ensamblador, se crearon diferentes lenguajes de programación, más o menos parecidos al inglés actual que seguirán los pasos descritos por un algoritmo.

Además, a la hora de estudiar la calidad de un algoritmo, es deseable que los algoritmos presenten también otra serie de características como son:

- **Validez.** El algoritmo construido hace exactamente lo que se pretende hacer.
- **Eficiencia.** El algoritmo debe dar una solución en un tiempo razonable. Por ejemplo, para sumar 20 a un número dado podemos dar un algoritmo que sume uno veinte veces, pero esto no es muy eficiente. Sería mejor dar un algoritmo que lo haga de un modo más directo.
- **Optimización.** Se trata de dar respuesta a la cuestión de si el algoritmo diseñado para resolver el problema es el mejor. En este sentido y como norma general, será conveniente tener en cuenta que suele ser mejor un algoritmo sencillo que no uno complejo, siempre que el primero no sea extremadamente ineficiente.

Para encontrar una solución computacional a un problema dado requiere el máximo de creatividad por parte de la persona que pretende encontrar una solución. El primer objetivo que nos debemos plantear es obtener una correcta comprensión de la naturaleza del problema. El análisis del problema exige una primera lectura del problema a fin de obtener una idea general de lo que se solicita. Una segunda lectura deberá servir para responder a las preguntas:

1) ¿Qué información debe proporcionar la resolución del problema?

2) ¿Qué datos se necesitan para resolver el problema?

La respuesta a la primera pregunta indicará los resultados deseados o salida del programa. La respuesta a la segunda pregunta indicará qué datos se deben proporcionar o las entradas del problema.

Veamos algunos ejemplos de algoritmos que los vamos a redactar en nuestro lenguaje, es decir, en castellano y con nuestras propias palabras, cuidando de no utilizar palabras o expresiones ambiguas que lejos de dar solución nos confundan más. A esta forma de expresar los algoritmos las llamaremos Pseudocódigo.

Ejemplo 01:

Un cliente ejecuta un pedido a una fábrica. La fábrica examina en su banco de datos la ficha del cliente, si el cliente es solvente entonces la empresa acepta el pedido; en caso contrario, rechazará el pedido. Se pide realizar el algoritmo correspondiente.

Solución:

1. Inicio.

2. Leer el pedido.

3. Examinar la ficha del cliente.

4. Si el cliente es solvente, aceptar pedido;

En caso contrario, rechazar pedido.

5. Fin.

Ejemplo 02:

Realizar un algoritmo que nos ayude a determinar el mayor de dos números

Solución:

1. Inicio.

2. Leer el 1er y 2do números (ambos números distintos).

3. Si el 1er número es mayor que el segundo, Número mayor es el 1er número.

En caso contrario, Número mayor es el 2do número.

4. Fin.

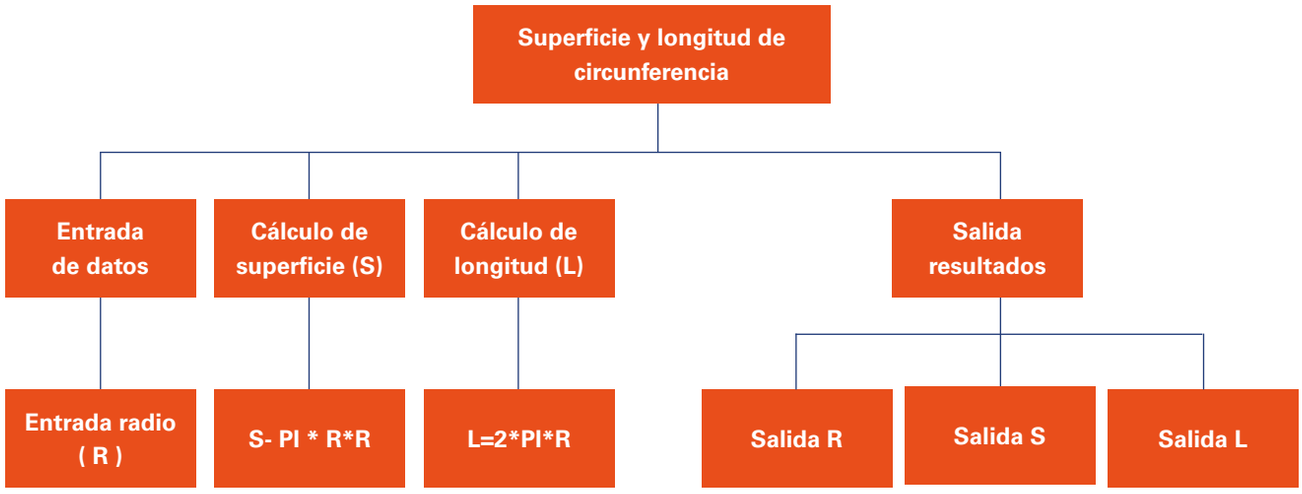
2. DISEÑO DEL ALGORITMO

Una computadora no tiene capacidad para solucionar problemas más que cuando se le proporcionan los sucesivos pasos a realizar. Estos pasos sucesivos que indican las instrucciones a ejecutar por la máquina constituyen, como ya conocemos, el algoritmo.

La información proporcionada al algoritmo constituye su entrada y la información producida por el algoritmo constituye su salida.

Los problemas complejos se pueden resolver más eficazmente con la computadora cuando se rompen en subproblemas que sean más fáciles de solucionar que el original. Es el método de divide y vencerás, y que consiste en dividir un problema complejo en otros más simples. Así, el problema de encontrar la superficie y la longitud de un círculo se puede dividir en tres problemas más simples o subproblemas.

Figura N° 6 Problema y Subproblemas



La descomposición del problema original en subproblemas más simples y a continuación la división de estos subproblemas en otros más simples que y a continuación la división de estos subproblemas en otros más simples que pueden ser implementados para su solución en la computadora se denominan diseño descendente (top-down design). Normalmente, los pasos diseñados en el primer esboza del algoritmo son incompletos e indicarán sólo unos pocos pasos (12 aprox). Tras esta primera descripción éstos se amplían en una descripción más detallada con más pasos específicos. Este proceso se denomina refinamiento del algoritmo (stepwise refinement).

El problema del cálculo de la circunferencia y superficie de un círculo se puede descomponer en subproblemas más simples:

- Leer datos de entrada
- Calcular superficie y longitud de circunferencia
- Escribir resultados

SUBPROBLEMA	REFINAMIENTO
Leer radio	Leer radio
Calcular superficie	Superficie = 3.1416 * radio * radio
Calcular circunferencia	Circunferencia = 2 * 3.1416 * radio
Escribir resultados	Escribir radio, circunferencia, superficie

Las ventajas más importantes del diseño descendente son:

- El problema se comprende más fácilmente al dividirse en partes más simples denominadas módulos.
- Las modificaciones en los módulos son más fáciles.
- La comprobación del problema se puede verificar fácilmente.

Entonces, es importante para construir un algoritmo tener presente que debemos empezar a subdividir el pro-

blema en subproblemas más simples y así sucesivamente, de esta manera, será más fácil encontrar la solución de estos subproblemas y al resolver los subproblemas estaremos resolviendo el problema.

No se olviden de la estrategia: "Divide y vencerás"

Una vez que se ha descrito el algoritmo utilizando una herramienta adecuada, es necesario comprobar que realiza las tareas para las que fue diseñado y produce los resultados correctos y esperados a partir de la información de entrada. Este proceso se conoce como prueba del algoritmo y consiste básicamente en recorrer todos los caminos posibles del algoritmo comprobando en cada caso que se obtienen los resultados esperados. Para lo cual realizaremos una ejecución manual del algoritmo con datos significativos que abarquen todo el posible rango de valores y comprobaremos que la salida coincide con la esperada en cada caso. La aparición de errores puede conducir a tener que rediseñar determinadas partes del algoritmo que no funcionaban bien y a aplicar de nuevo el proceso de localización de errores, definiendo nuevos casos de prueba y recorriendo de nuevo el algoritmo con dichos datos.

Ahora es importante representar al algoritmo mediante una herramienta: Pseudocódigo, Diagrama de Flujo o Diagrama N-S.

Estas herramientas nos ayudarán a crear los algoritmos con una notación estándar que pueda ser también comprendido por otras personas o desarrolladores y de esta forma nuestros algoritmos puedan ser utilizados por programadores para crear utilizando un lenguaje de programación apropiado los programas que serán entendidos por las computadoras.

3. FLUJOGRAMAS Y DIAGRAMA

3.1 Diagramas de Flujo

Un diagrama de Flujo (Flowchart) es una de las técnicas de representación de algoritmos más antigua y a la vez más utilizada, Un diagrama de flujo es un diagrama que utiliza los símbolos estándar.

Un diagrama de flujo es una representación gráfica de un proceso. Cada paso del proceso es representado por un símbolo diferente que contiene una breve descripción de la etapa de proceso. Los símbolos gráficos del flujo del proceso están unidos entre sí con flechas que indican la dirección de flujo del proceso.

El diagrama de flujo ofrece una descripción visual de las actividades implicadas en un proceso mostrando la relación secuencial ente ellas, facilitando la rápida comprensión de cada actividad y su relación con las demás.

Un diagrama de flujo muestra la lógica del algoritmo, acentuando los pasos individuales y sus interconexiones.

Un diagrama de flujo debe reflejar:

El comienzo del programa.

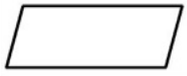
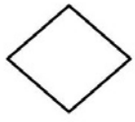
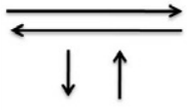
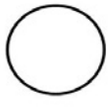
Las operaciones que el programa realiza.

El orden en que se realizan.

El final del programa.

Los símbolos utilizados han sido normalizados por las organizaciones ANSI (American National Standard Institute) y por ISO (International Standard Organization) y son los siguientes:

Figura Nº 7 Símbolos de Diagrama de Flujo

SÍMBOLO	NOMBRE	ACCIÓN
	Terminal	Representa el inicio o el fin del diagrama de flujo.
	Entrada y salida	Representa los datos de entrada y los de salida.
	Decisión	Representa las comparaciones de dos o mas valores, tiene dos salidas de información falso o verdadero
	Proceso	Indica todas las acciones o cálculos que se ejecutaran con los datos de entrada u otros obtenidos.
	Líneas de flujo de información	Indican el sentido de la información obtenida y su uso posterior en algún proceso subsiguiente.
	Conector	Este símbolo permite identificar la continuación de la información si el diagrama es muy extenso.

Fuente: http://3.bp.blogspot.com/-UZvOJ8Dbxw/DIAGRAMA+DE+FLUJO.jpg

Los beneficios de utilizar los diagramas de flujo:

- Facilita la obtención de una visión transparente del proceso, mejorando su comprensión. El conjunto de actividades, relaciones e incidencias de un proceso no es fácilmente discernible a priori. La diagramación hace posible aprender ese conjunto e ir más allá, centrándose en aspectos específicos del mismo, apreciando las interrelaciones que forman parte del proceso así como las que se dan con otros procesos y subprocesos.
- Permiten definir los límites de un proceso. A veces estos límites no son tan evidentes, no estando definidos los distintos proveedores y clientes (internos y externos) involucrados.
- El diagrama de flujo facilita la identificación de los clientes, es más sencillo determinar sus necesidades y ajustar el proceso hacia la satisfacción de sus necesidades y expectativas.
- Estimula el pensamiento analítico en el momento de estudiar un proceso, haciendo más factible generar alternativas útiles.
- Proporciona un método de comunicación más eficaz, al introducir un lenguaje común, si bien es cierto que para ello se hace preciso la capacitación de aquellas personas que entrarán en contacto con la diagramación.
- Un diagrama de flujo ayuda a establecer el valor agregado de cada una de las actividades que componen el proceso.
- Igualmente, constituye una excelente referencia para establecer mecanismos de control y medición de los procesos, así como de los objetivos concretos para las distintas operaciones llevadas a cabo.

- Facilita el estudio y aplicación de acciones que redunden en la mejora de las variables *tiempo* y *costes de actividad* e incidir en la mejora de la eficacia y la eficiencia.
- Constituyen el punto de comienzo indispensable para acciones de mejora o reingeniería.

Veamos algunos ejemplos de la construcción de algoritmos utilizando los Diagramas de flujo.

Ejemplo 01:

Escribir un algoritmo pueda determinar la suma y la multiplicación de dos números ingresados por el usuario.

Solución:

Pseudocódigo:

1. Inicio.
2. Definir Variables Num01, Num02, Suma, Mult.
3. Leer número 01 y número 02(Num01, Num02).
4. Calcular Suma = Num01 + Num02
5. Calcular Mult = Num01 * Num02
6. Mostrar Suma, Mult.
7. Fin.

Diagrama de Flujo

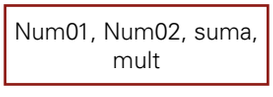
Primero realizaremos la representación gráfica paso por paso de acuerdo al algoritmo en pseudocódigo para entender la representación de cada figura.

1. **Inicio.**

Figura N° 8: Representación: Inicio

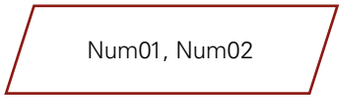
2. **Definir Variables** Num01, Num02, Suma, Mult.

Figura N° 9: Representación: Definición de Variables



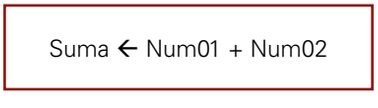
3. **Leer número** 01 y número 02(num01, num02).

Figura N° 10: Representación: Ingreso de datos



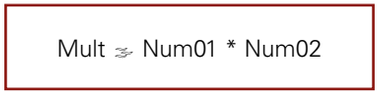
4. **Calcular suma** = num01 + num02

Figura N° 11: Representación: Proceso Suma



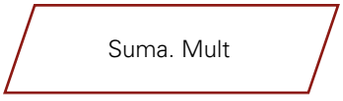
5. **Calcular mult** = num01 * num02

Figura N° 12: Representación: Proceso Multiplicación



6. **Mostrar suma**, mult.

Figura N° 13 Representación: Salida de datos

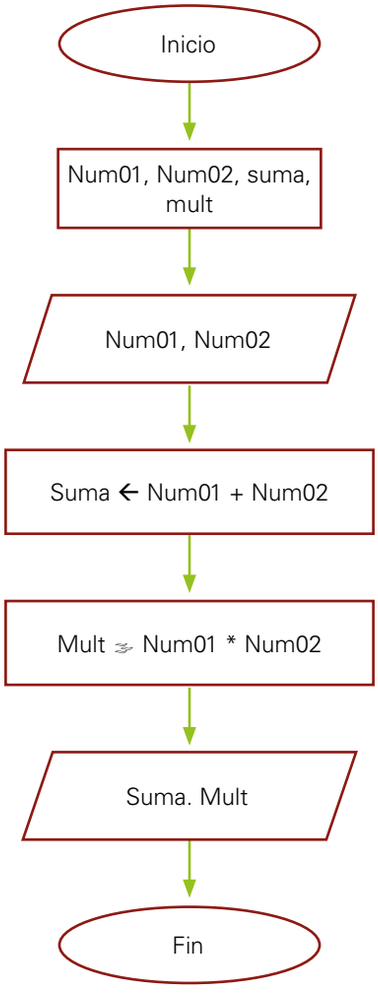


7. **Fin.**

Figura N° 14: Representación: Fin

Ahora la forma correcta de representar el algoritmo en diagrama de flujo es en un solo gráfico, como se muestra a continuación

Figura N° 15: Diagrama de Flujo Ejemplo 01



Fuente: Carlos Calderon Sadano

Como podemos apreciar en la figura 15 Un Diagrama de flujo tiene un inicio y un final y describe la secuencia lógica de pasos para solucionar el problema planteado.

Ejemplo 02:

Escribir un algoritmo pueda determinar el mayor de dos números ingresados por el usuario.

Solución:

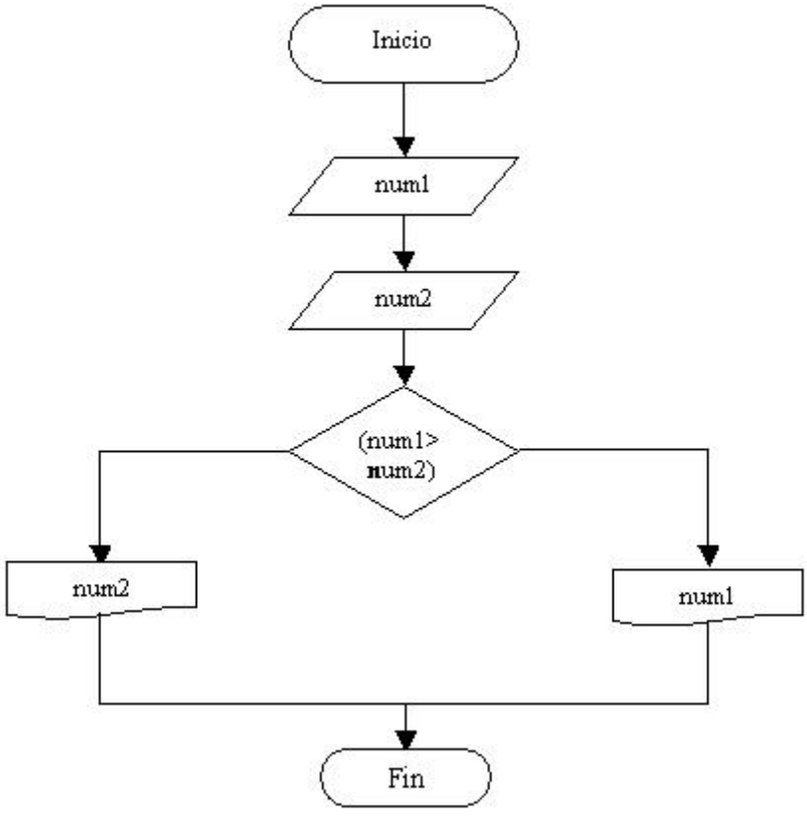
Pseudocódigo:

1. Inicio.
2. Definir Variables Num01, Num02, Suma, Mult.
3. Leer número 01 y número 02(Num01, Num02).
4. Si Num01> Num02

El mayor es Num01

5. Sino el mayor es Num02t.
6. Fin.

Figura N° 16: Diagrama de Flujo Ejemplo 02



Fuente: Luis Joyanes: "Fundamentos de Programación"

En la figura 16 se muestra el símbolo de una decisión para conocer si Num01 es mayor que Num02 y vemos que tenemos dos caminos si la respuesta es verdadera (SI) entonces el número mayor es Num01, pero si es falsa (NO) el número mayor será Num02.

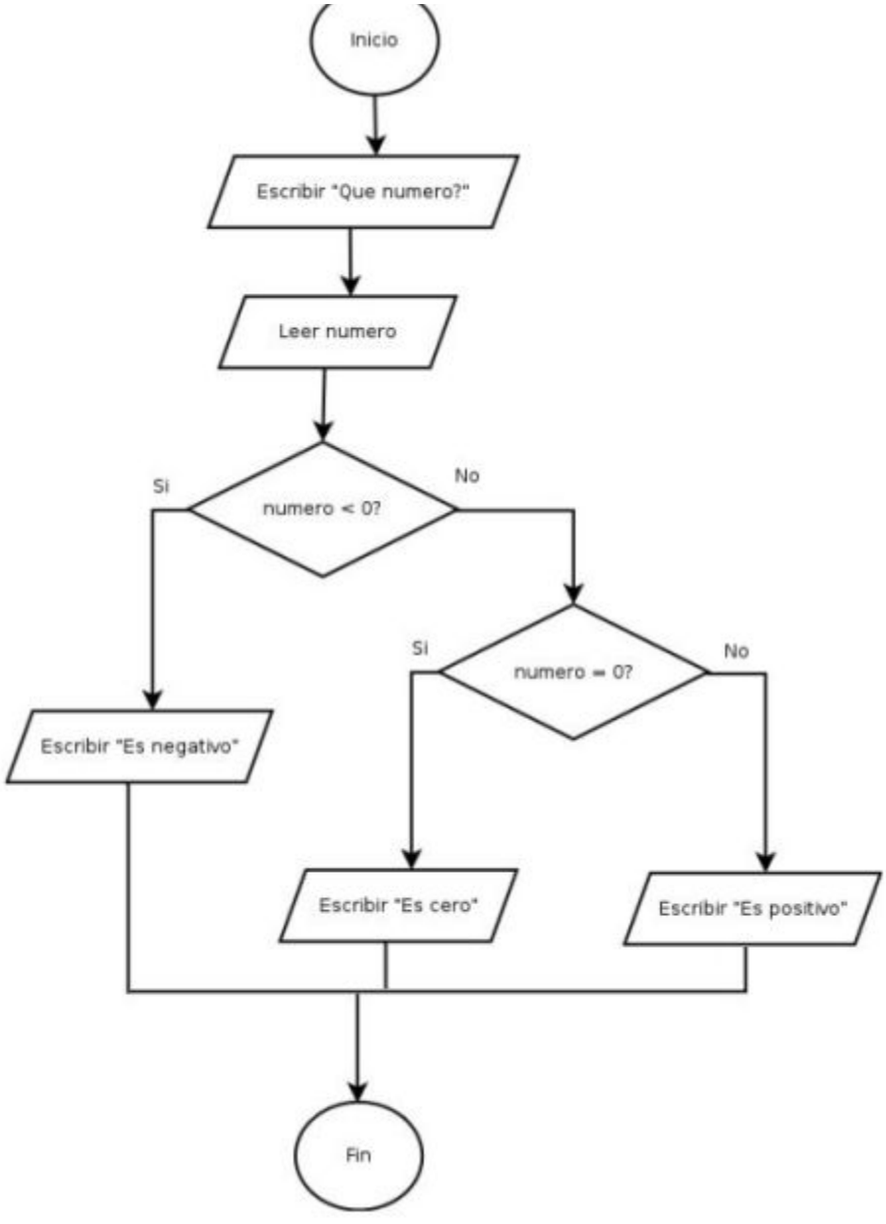
Entonces cuando nos enfrentamos a una decisión siempre existirán dos alternativas (caminos) uno para cuando sea verdad y otro para cuando sea falso.

Ejemplo 03:

Escribir un algoritmo determine si un número es negativo o positivo.

Solución:

Figura N° 17: Diagrama de Flujo Ejemplo 03



Fuente: Luis Joyanes: "Fundamentos de Programación"

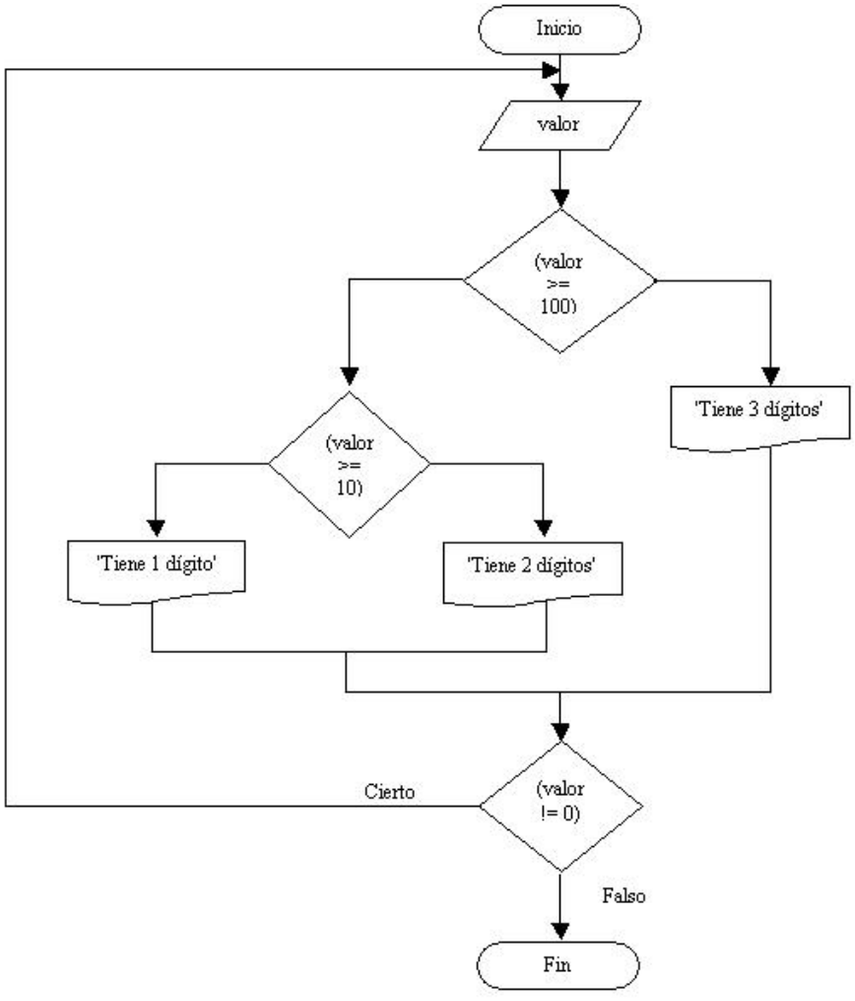
Pueden existir decisiones anidadas como el ejemplo de la figura 17.

Ejemplo 04:

Escribir un algoritmo determine si un número determina la cantidad de cifras que tiene dicho número se debe trabajar con valores menores a 999 y además el algoritmo debe terminar cuando se ingresa el número cero.

Solución:

Figura N° 18: Diagrama de Flujo Ejemplo 04



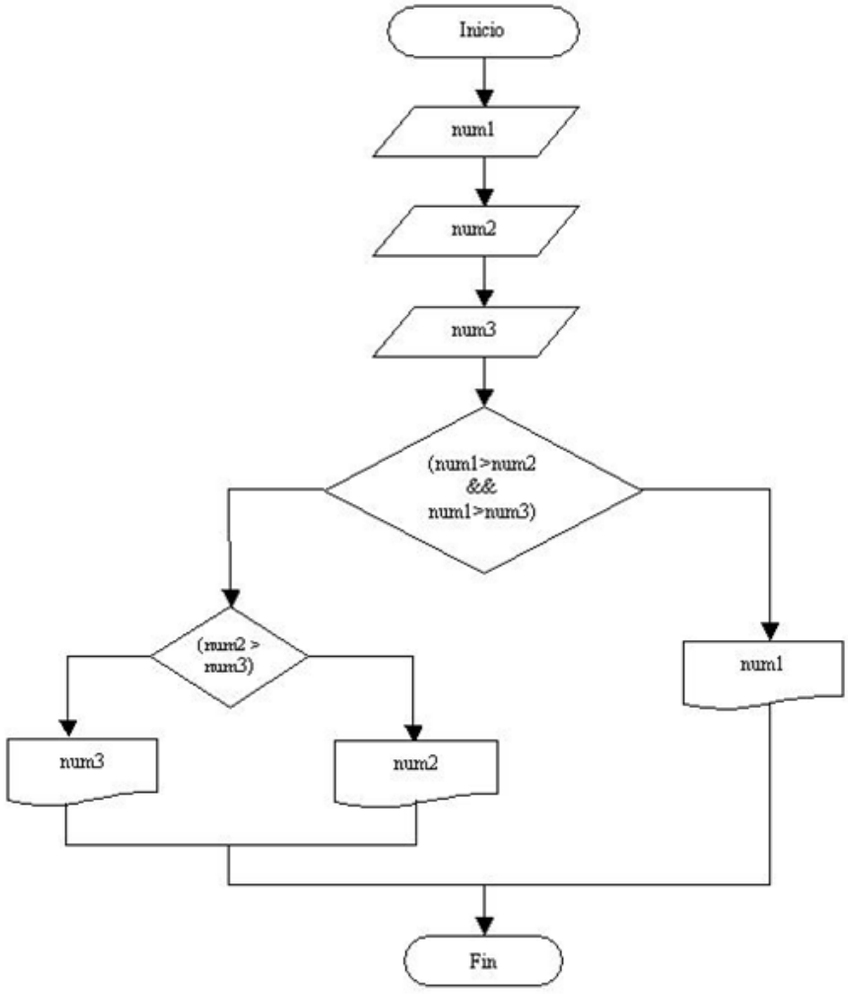
Fuente: Luis Joyanes: "Fundamentos de Programación"

Ejemplo 05:

Escribir un algoritmo que permita el ingreso de 3 números y determine cuál es el número mayor

Solución:

Figura N° 19: Diagrama de Flujo Ejemplo 05



Fuente: Luis Joyanes: "Fundamentos de Programación"

LECTURA SELECCIONADA N°. 1:

ESPACIOS VIRTUALES COMPARTIDOS

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez

Es probable que la multimedia del futuro se extienda más allá de la pantalla plana, creando experiencias que desafíen nuestra noción de la realidad.

REALIDAD VIRTUAL Los investigadores vienen experimentando con los mundos virtuales desde los años sesenta: mundos generados por computadora que crean la ilusión de sumergirse en ellos. Normalmente, los mundos virtuales implican un hardware especial: para la entrada, un guante o traje corporal equipado con sensores de movimiento, y para la salida, una pantalla montada en la cabeza, un casco con pantallas del tamaño de los ojos cuyo contenido cambia con el movimiento del casco. Este equipamiento, acoplado al software adecuado, permite al usuario explorar un mundo artificial de datos como si se encontrara en un espacio físico tridimensional. La realidad virtual combina los mundos virtuales con las redes, colocando a varios participantes en un espacio virtual. Las personas ven representaciones de los demás, en ocasiones denominadas avatares. La mayoría de ellos son personajes de dibujos animados, pero cuentan con un sentido de presencia y emoción.

TELE-INMERSIÓN Jaron Lanier, que acuñó el término de realidad virtual, es ahora el principal científico del National Tele-Immersion Initiative. La tele-inmersión utiliza varias cámaras y redes de alta velocidad para crear un entorno en el que varios usuarios remotos pueden interactuar entre sí y con los objetos generados por la computadora. (Lanier fue consultor en la película Minority Report de Spielberg, una película que muestra una tecnología parecida.) La tele-inmersión combina las técnicas de la RV con las nuevas tecnologías de la visión que permiten a los participantes moverse por espacios virtuales compartidos, manteniendo sus propios puntos de vista. Los sistemas actuales necesitan unas gafas especiales; puede que las versiones futuras no. Los sistemas de tele-inmersión, cuando se acoplen a Internet2 de gran velocidad, permitirán a ingenieros, arqueólogos y artistas, además de otros, disfrutar de colaboraciones a larga distancia en espacios de trabajo virtuales compartidos. La tele-inmersión puede permitir a músicos y actores proporcionar actuaciones interactivas, y puede reducir significativamente la necesidad de los viajes de negocios dentro de una década.

LA REALIDAD AMPLIADA Otro apéndice prometedor de la investigación en RV es la realidad ampliada (RA); el uso de pantallas de computadora que añaden información virtual a las percepciones sensoriales de una persona. A diferencia de la RV, la RA complementa, en lugar de reemplazar, el mundo que vemos. La línea que se superpone en TV en el campo de fútbol para marcar un fuera de juego es un ejemplo de RA, pero el futuro ofrece muchas otras aplicaciones prácticas. Con la RA, un mecánico de reparaciones podría ver instrucciones superpuestas en una parte de la máquina; un cirujano podría ver en el cuerpo del paciente su interior mientras los escáneres de ultrasonido examinan los órganos internos; y un bombero podría ver la estructura de un edificio en llamas. El investigador de RA Steven K. Feiner predice que «la información superpuesta de los sistemas RA será parte de lo que esperamos ver en el trabajo y en el juego: etiquetas y pautas cuando no queramos perdernos, recordatorios cuando no queramos olvidar algo y, quizá, nuestro personaje de dibujos animados preferido saltando de un arbusto para contarnos un chiste cuando queramos divertirnos. Cuando las interfaces de usuario de computadora estén potencialmente en cualquier parte donde miremos, esta penetrante mezcla de realidad y virtualidad se convertirá en el medio principal para una nueva generación de artistas, diseñadores y cómicos del futuro.»

VIDEOS

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Cuadro de doble entrada.
URL: https://youtu.be/nU8tScHyoYs?t=22s
Duración: 4 min 15 s.
Autor(a): Lizbeth Esquivel, Yomali García, Crystal Murrieta, Yamileth Romero.
Año: 2014.
Licencia: YouTube estándar.

Video 8: Cómo hacer un cuadro de doble entrada.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: [Programa] Alterados por Pi: Algoritmos.
URL: https://youtu.be/-hHieORYNaI?t=1m50s
Duración: 4 min 34 s.
Autor(a): Canal Encuentro (Argentina).
Expositor(a): Adrián Paenza.
Año: 2008.
Licencia: YouTube estándar.

Video 9: ¿Qué es un algoritmo?

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: [Programa] Alterados por Pi: Algoritmos.
URL: http://www.encuentro.gov.ar/sitios/encuentro/programas/ver?rec_id=105740
Duración: 3 min 6 s.
Autor(a): Canal Encuentro (Argentina).
Expositor(a): Adrián Paenza.
Año: 2008.
Licencia: YouTube estándar.

Video 10: Problema propuesto.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Introducción a Algoritmos.
URL: https://youtu.be/PWgVXuQKrdo
Duración: 8 min 12 s.
Autor(a): David Chura.
Año: 2009.
Licencia: YouTube estándar.

Video 11: Introducción a los algoritmos.

ACTIVIDAD N° 1

Explica algoritmos usando el entorno empresarial y elabora flujograma para representar un al-goritmo

INSTRUCCIONES:

1. Lee y analiza, todos los contenidos de tema N° 1

2. Como apoyo visualice el siguiente video: https://www.youtube.com/watch?v=PWgVXuQKrdo

3. Explique brevemente la definición de los algoritmos y su utilidad en el entorno empresarial.

4. Elabora un algoritmo (Pseudocódigo y Diagrama de flujo) que permita calcular el sueldo bruto, sueldo neto y el importe de impuestos a pagar de un trabajador.

5. Los datos que deben ser ingresados al algoritmo son:

- Nombre del Empleado
- Número de horas trabajadas.
- Costo por hora
- Impuesto a pagar en porcentaje (ejemplo 18%)

6. Los datos a reportar son:

- Sueldo Bruto
- Sueldo Neto
- Monto a pagar por Impuestos

TEMA N° 2
ESTRUCTURA

Los algoritmos que hemos estado desarrollando hasta el momento han consistido en simples secuencias de instrucciones; sin embargo, existen tareas más complejas que no pueden ser resueltas empleando un esquema tan sencillo, en ocasiones es necesario repetir una misma acción un número determinado de veces o evaluar una expresión y realizar acciones diferentes en base al resultado de dicha evaluación.

Para resolver estas situaciones existen las denominadas estructuras de control que poseen las siguientes características:

- Una estructura de control tiene un único punto de entrada y un único punto de salida.
- Una estructura de control se compone de sentencias o de otras estructuras de control.

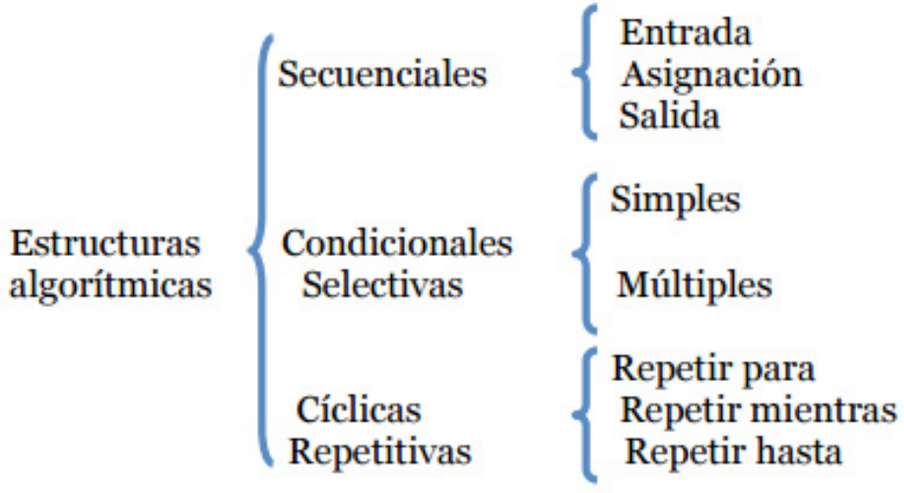
Tales características permiten desarrollar de forma muy flexible todo tipo de algoritmos aun cuando sólo existen tres tipos fundamentales de estructuras de control:

Secuencial.

Selectiva.

Repetitiva

Figura Nº 20: Estructura e Control



Fuente http://campus.cva.itesm.mx/

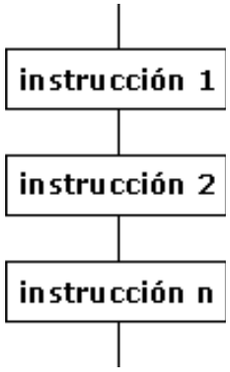
1. ESTRUCTURA SECUENCIAL, SELECTIVA Y REPETITIVAS

1.1 Estructura Secuencial

Una estructura secuencial es aquella en la que una acción (instrucción) sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el final del

proceso. La estructura secuencial tiene una entrada y una salida. Su representación gráfica se muestra a continuación.

Figura Nº 21: Estructura Secuencial



Fuente: Luis Joyanes: "Fundamentos de Programación"

La estructura secuencial se caracteriza por realizar tres acciones:

- Lectura: Consiste en recibir desde un dispositivo de entrada o de un archivo, un valor.
- Asignación: Consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de la variable (identificador) que recibe el valor. Formas de asignación:
  - o Simple: Consiste en pasar un valor constate a una variable.
  - o Contador: Consiste en usarla como un verificador del número de veces que se realiza un proceso.
  - o Acumulador: Consiste en usarla como un sumador en un proceso.
  - o De trabajo: Donde puede recibir el resultado de una operación matemática que involucre muchas variables.
- Escritura: Consiste en obtener información por medio de un dispositivo de salida como es la impresora, un panel de control, etc.

Ejemplo 01:

Escribir un algoritmo que permita calcular el área y el perímetro de un rectángulo

Pseudocódigo:

Inicio

b = 8

h = 2

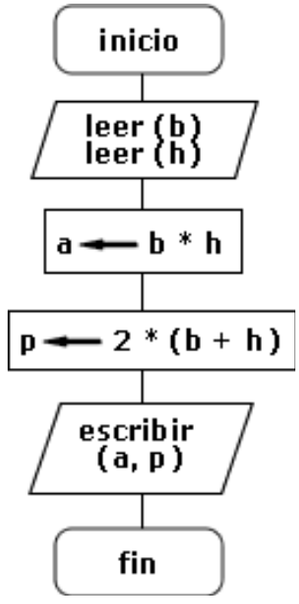
a = b * h

p = 2 * (b + h)

escribir (a, p)

*fin***Diagrama de Flujo:**

Figura Nº 22: Diagrama de flujo Estructura Secuencial



Fuente: Luis Joyanes: "Fundamentos de Programación"

1.2 Estructura Selectiva

No todos los problemas pueden resolverse empleando estructuras secuenciales. Cuando se necesita tomar una decisión aparecen las estructuras selectivas para poder resolver el problema.

Las estructuras selectivas se utilizan para tomar decisiones lógicas; también se suelen denominar también estructuras condicionales, de decisión o alternativas.

Se utilizan cuando en el desarrollo de la solución de un problema se debe de tomar una decisión para establecer un proceso o un camino alternativo a seguir. Esta toma de decisión (expresada en el diagrama de flujo con un rombo) se basa en la evaluación de una o más condiciones que señalan alternativas o consecuencias esto es, el camino (rama) a seguir. Una toma de decisión se puede realizar en cascada: se toma una decisión (rama a seguir) y dentro de esa rama se toma otra decisión y así sucesivamente. Esto implica diseñar un árbol de decisiones. Las estructuras algorítmicas selectivas que se utilizan para la toma de decisiones lógicas se clasifican como:

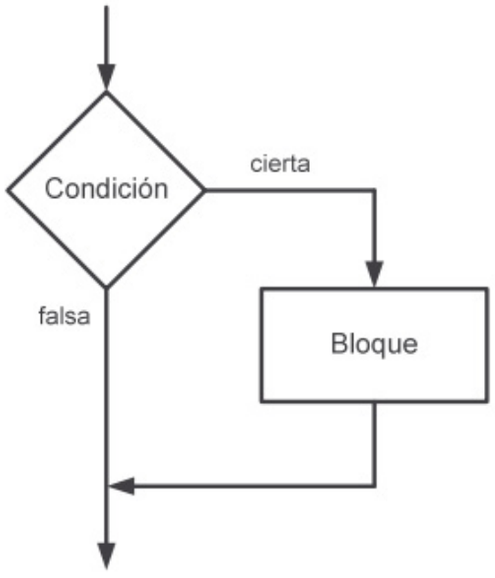
- Simple: SI … ENTONCES …
- Doble: SI … ENTONCES … SI NO …
- Múltiple: EN CASO DE … ENTONCES …

1.2.1 Estructura Selectiva Simple

Esta estructura permite evaluar una expresión lógica y en función de dicha evaluación ejecutar una acción (o composición de acciones) o no ejecutarla; también se la suele denominar SI-ENTONCES.

Cuando preguntamos en la condición siempre realizaremos las acciones en el caso de ser verdadera la respuesta, caso contrario no se realizará ninguna acción su representación gráfica la podemos apreciar en la figura 23.

Figura N° 23: Estructura Selectiva Simple



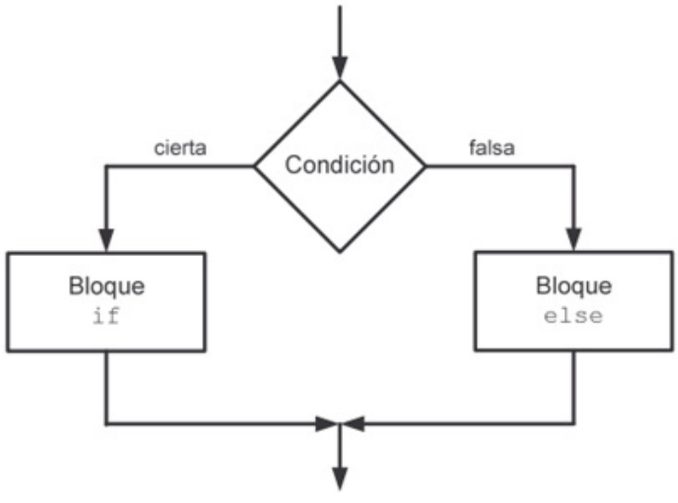
Fuente http://elvex.ugr.es/

1.2.2 Estructura Selectiva Doble

La estructura alternativa doble es similar a la anterior con la salvedad de que en este tipo de estructura se indican acciones no sólo para la rama "verdadera" sino también para la "falsa"; es decir, en caso de la expresión lógica evaluada sea cierta se ejecutan una acción o grupo de acciones y en caso de que sea falsa se ejecuta un grupo diferente.

En este caso cuando se consulte a la condición si es verdadera se ejecutarán una serie de acciones y en el caso de que sea falso también se ejecutaran otra serie de acciones, es decir, dependiendo de la decisión se realizaran una u otra acción. Su representación gráfica al podemos ver en la figura 24.

Figura N° 24: Estructura Selectiva Doble

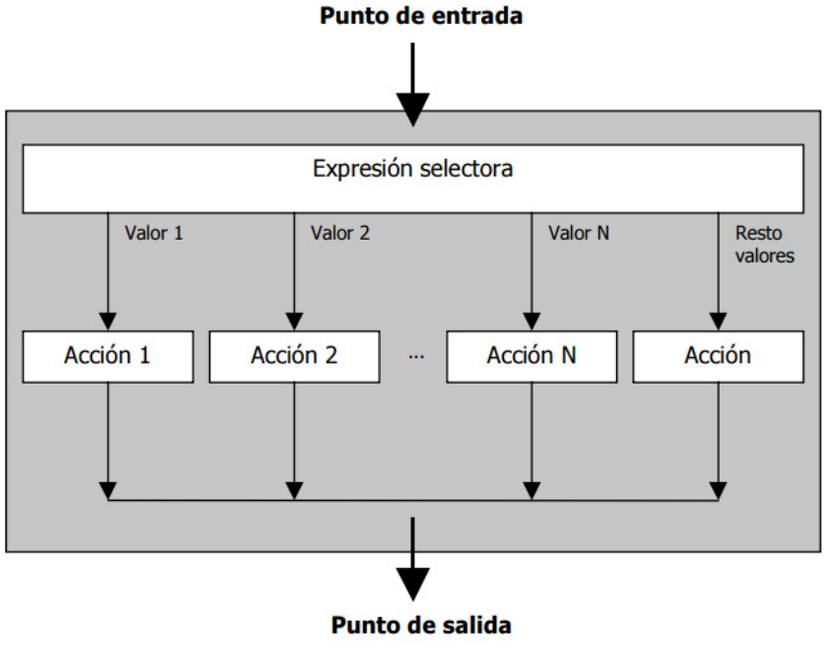


Fuente http://elvex.ugr.es/

1.2.3 Estructura Selectiva Múltiple

Esta estructura evalúa una expresión que pueda tomar n valores (enteros, caracteres y lógicos pero nunca reales) y ejecuta una acción o grupo de acciones diferente en función del valor tomado por la expresión selectora.

Figura N° 25: Estructura Selectiva Multiple



Fuente http://di002.edv.uniovi.es/

Ejemplo 01:

Escribir un algoritmo que Ingresar el sueldo de una persona, si supera los S/. 3000 soles, mostrar un mensaje en pantalla indicando que debe abonar impuestos.

Pseudocódigo:

inicio

real: sueldo

leer (sueldo)

si (sueldo > 3000) **entonces**

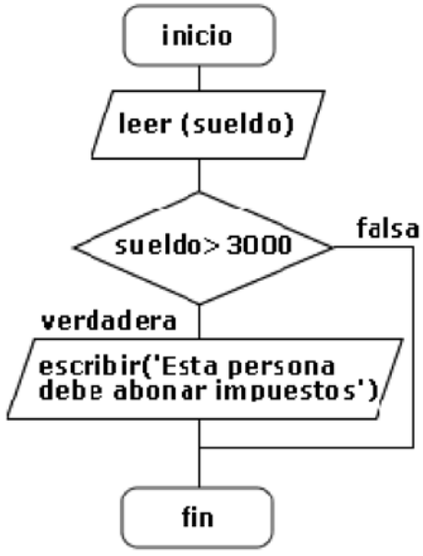
escribir('Esta persona debe abonar impuestos')

fin_si

fin

Diagrama de flujo:

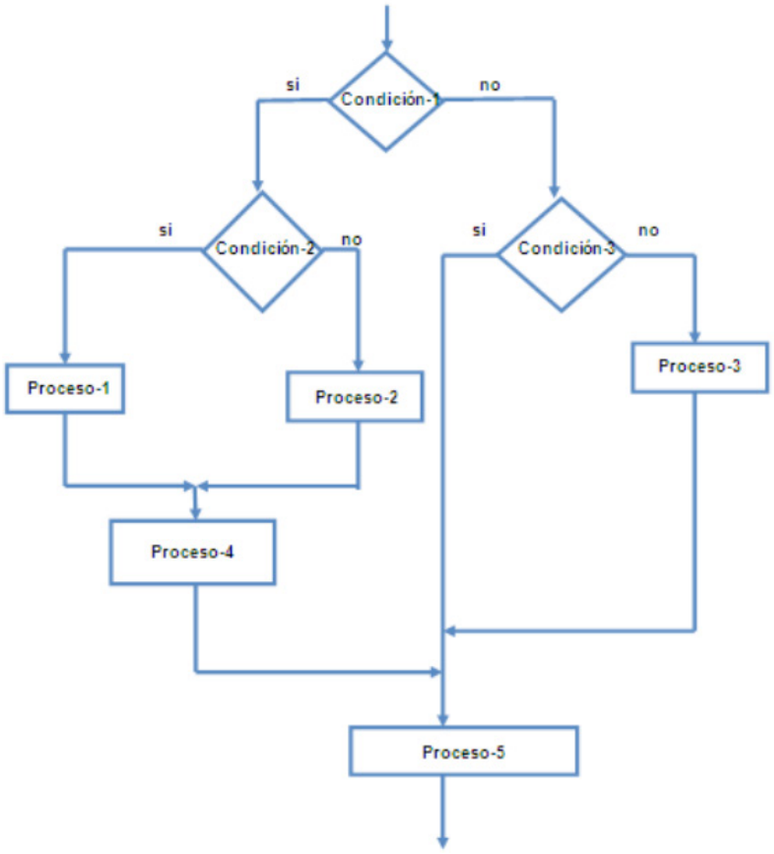
Figura Nº 26: Diagrama Estructura Selectiva



Fuente: Luis Joyanes: "Fundamentos de Programación"

También dado las circunstancia se puede anidar las sentencias y tener lo que se llama anidamiento en cascada, es decir uan estructura selectiva puede estar incluida en otra y así sucesivamente según la lógica de la solución, en la siguiente figura 26 podemos apreciar un ejemplo.

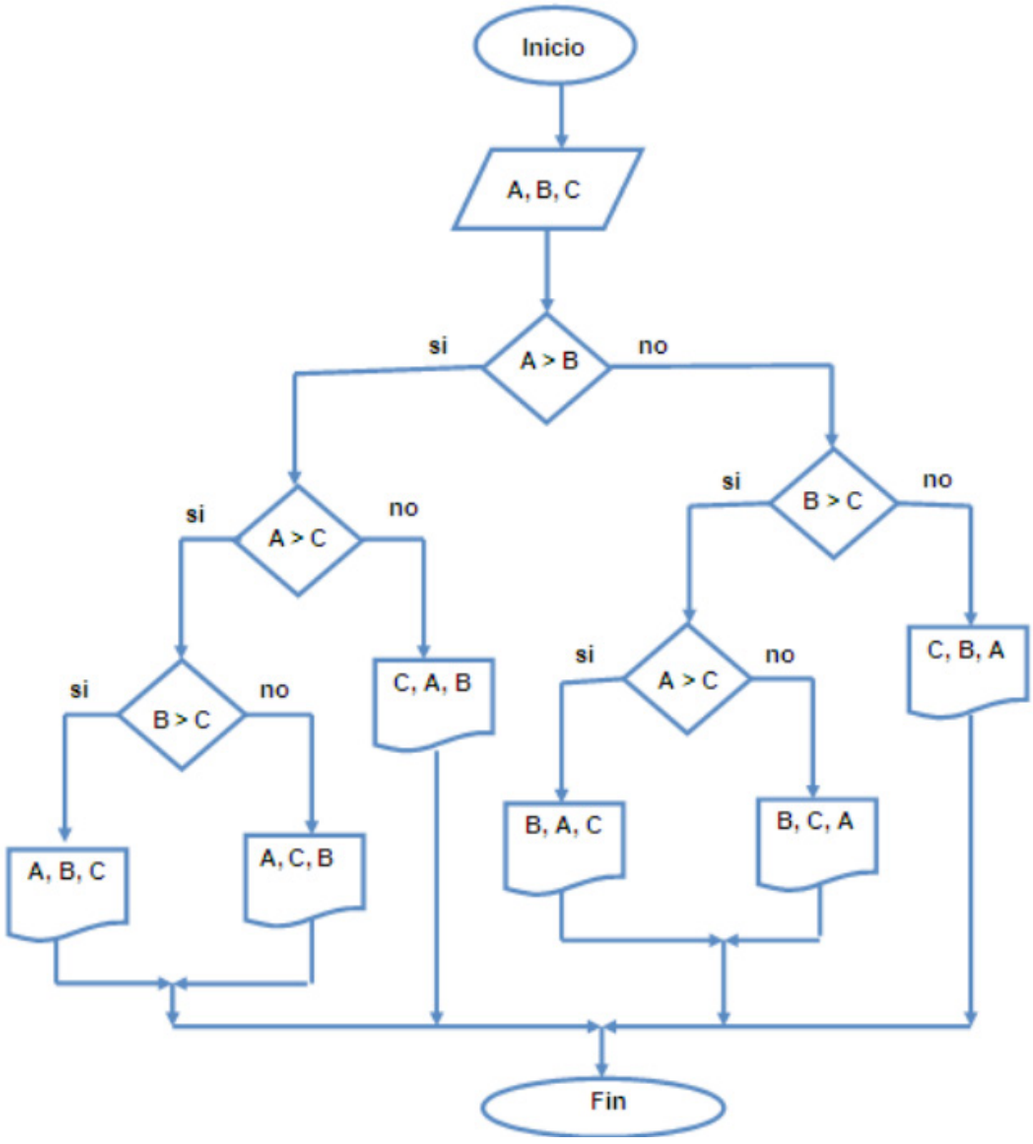
Figura Nº 27: Estructura Selectiva en Cascada



Fuente http://campus.cva.itesm.mx/

En la siguiente figura 28 se muestra un ejemplo de este tipo de anidamiento en el cual ordena tres números de mayor a menor.

Figura N° 28: Ejemplo Estructura Selectiva en Cascada



Fuente http://campus.cva.itesm.mx/

1.3 Estructura Repetitiva

Las computadoras están diseñadas para aquellas aplicaciones en las cuales una operación o conjunto de ellas deben repetirse muchas veces. Un tipo muy importante de estructura es el algoritmo necesario para repetir una o varias acciones por un número determinado de veces, a esta estructura se la llama *Estructura Repetitiva*.

Las estructuras repetitivas se utilizan cuando se desea que una instrucción o bloque de instrucciones se repita un número determinado de veces o hasta que una condición de finalización se cumpla.

Las estructuras que repiten una secuencia de instrucciones un número determinado de veces se denominan ***bucles***, y se llama ***iteración*** al hecho de repetir la ejecución de una secuencia de acciones. ***Iterar*** es repetir una vez el bucle.

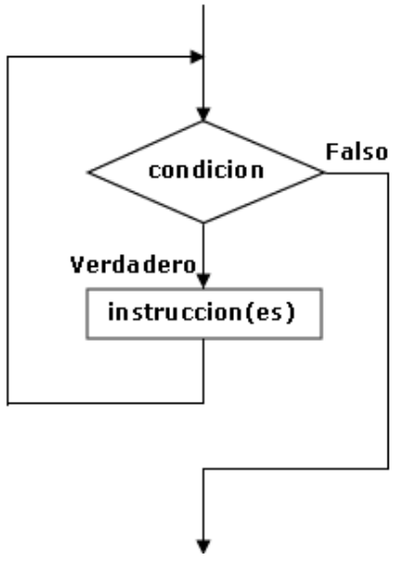
Se debe tener en cuenta lo siguiente para la construcción de una estructura repetitiva:

- ***El cuerpo del bucle***: Es el grupo de instrucciones que se van a repetir. Dentro del cuerpo del bucle debe existir una instrucción que modifique la condición lógica de finalización.
- ***Las sentencias de inicialización***. Son instrucciones que inicializan contadores y acumuladores.
- ***Las condiciones para la terminación del bucle***: Expresiones lógicas que controlan la finalización del bucle.

1.3.1 Estructura Repetitiva Mientras:

El proceso de una estructura repetitiva *Mientras* es el siguiente: para que ingrese al cuerpo del bucle tiene que evaluarse una condición, si esta es verdadera se ingresa y se realizan todas las instrucciones que están dentro del cuerpo del bucle; terminado la última instrucción se vuelve a comprobar la condición; se seguirá realizando el bucle mientras la condición siga siendo *verdadera* y si en un momento es *falsa* sale del bucle.

Figura N° 29: Estructura Repetitiva Mientras



Fuente: Luis Joyanes: "Fundamentos de Programación "

Ejemplo 01:

Escribir un algoritmo que lea las 40 notas finales del curso ABC, e informe cuantos alumnos han aprobado y cuantos desaprobaron. Dato: (Nota >= 10.5 Aprobado)

Pseudocódigo:

inicio

entero: c, ca, cd

real: notaFinal

c = 0

ca = 0

cd = 0

mientras (c < 40) hacer

c = c + 1

leer (notaFinal)

si (notaFinal >= 10.5) entonces

ca = ca + 1

sino

fin_si

fin_mientras

cd = cd + 1

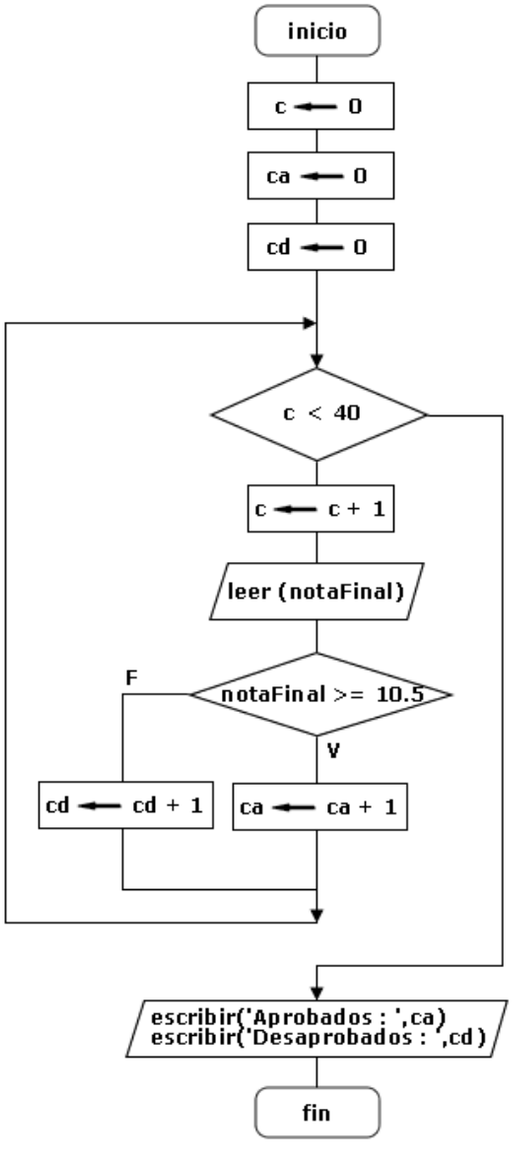
fin

escribir('Aprobados : ',ca)

escribir('Desaprobados : ',cd)

Diagrama de flujo:

Figura N° 30: Diagrama de Flujo Estructura Repetitiva Mientras



Fuente: Luis Joyanes: "Fundamentos de Programación "

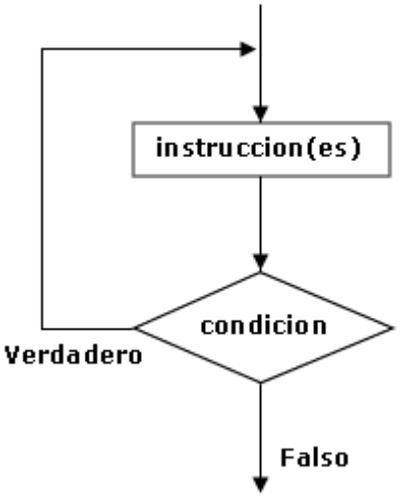
1.3.2 Estructura Repetitiva Hacer Mientras:

Existen muchas situaciones en las que se desea que un bucle se ejecute al menos una vez antes de comprobar la condición de repetición, para ello se puede utilizar la estructura repetitiva Hacer - Mientras. Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutara el bloque repetitivo.

Cuando una instrucción Hacer-Mientras se ejecuta, lo primero que sucede es la ejecución del bucle (todas las instrucciones) y a continuación se evalúa la expresión booleana de la condición. Si se evalúa como *verdadera*, el cuerpo del bucle se repite y se vuelve a evaluar la condición, si sigue siendo *verdadera* se seguirá repitiendo el bucle hasta que la condición sea *falsa*.

Dentro del bucle existirá una instrucción que en un cierto momento hará que la condición sea *falsa*.

Figura N° 31: Estructura Repetitiva Hacer Mientras



Fuente: Luis Joyanes: "Fundamentos de Programación "

Ejemplo 01:

Escribir un algoritmo que permita calcular el promedio del salón (Existe 40 alumnos)

Pseudocódigo:

inicio

entero: nota, c, suma

real: prom

c = 0

suma = 0

hacer

c = c + 1

leer(nota)

suma = suma + nota

mientras (c < 40)

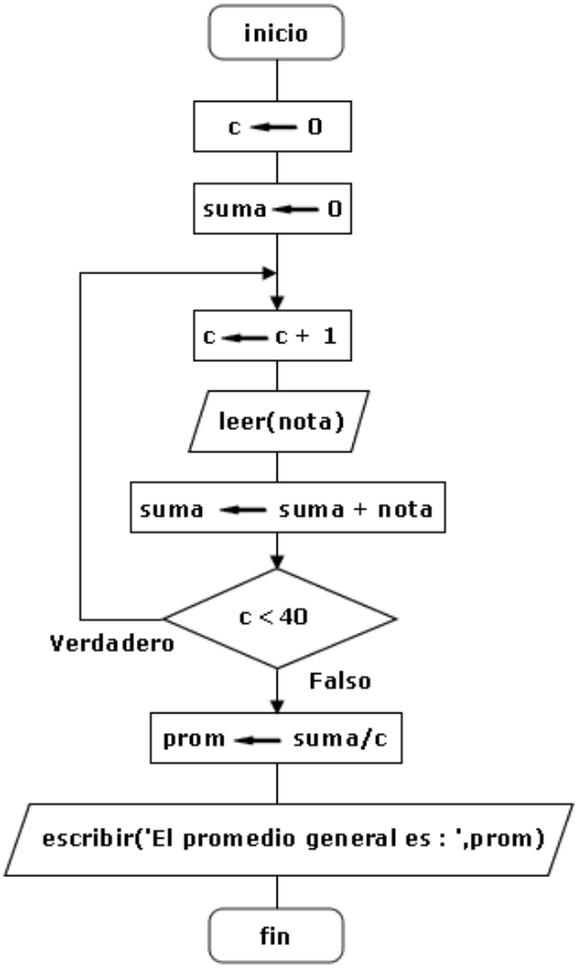
prom = suma/c

escribir('El promedio general es : ',prom)

fin

Diagrama de flujo:

Figura N° 32: Diagrama de Flujo Estructura Repetitiva Hacer Mientras



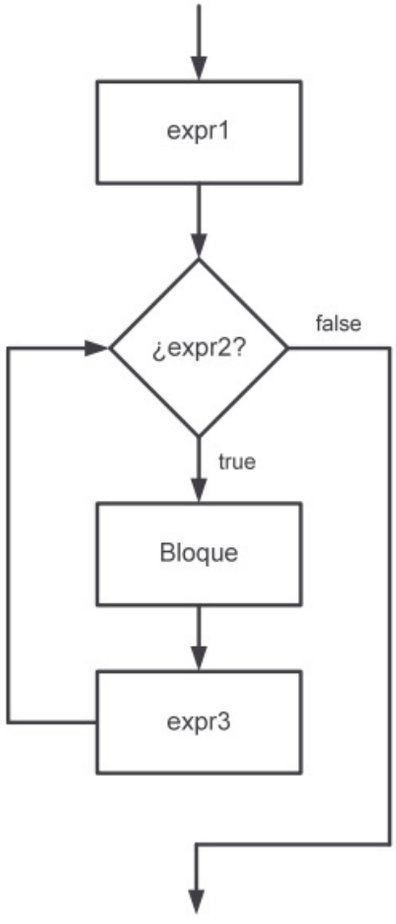
Fuente: Luis Joyanes: "Fundamentos de Programación "

1.3.3 Estructura Repetitiva Para:

La estructura para nos sirve ya que con ella se puede ejecutar un bucle que se repita determinado número de veces.

Esta sentencia requiere que conozcamos el número de veces que se desea ejecutar la sentencia del interior del bucle.

Figura N° 33: Estructura Repetitiva Para



Fuente: http://elvex.ugr.es/

LECTURA SELECCIONADA N°. 2:

INTELIGENCIA INTEGRADA Y COMPUTACIÓN OMNIPRESENTE

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez

Las computadoras están desapareciendo sin excepción dentro de otras herramientas. Los aparatos de información, como teléfonos móviles, faxes y dispositivos GPS, llevan a cabo su tarea especializada mientras ocultan a sus usuarios los detalles tecnológicos. Docenas de electrodomésticos y otras herramientas ocultan computadoras en su interior. Incluso nuestros coches procesan megabytes de información a medida que vamos conduciendo.

Algunas de estas computadoras de nuestros automóviles son invisibles; otras son más obvias. Varias compañías han introducido computadoras de abordo que puede reproducir CD y DVD, reconocer comandos hablados, alertar al conductor de la recepción de correos electrónicos, leer esos mensajes en voz alta, almacenar y recuperar contactos y citas, marcar números de teléfono, recitar direcciones usando sistemas de navegación basados en GPS, informar de problemas mecánicos e, incluso, seguir la pista de vehículos estropeados. Los investigadores de IBM han desarrollado un «pasajero artificial» para hacer más seguro para el conductor este cambio. Este agente inteligente es capaz de llevar una conversación, vigilando por si aparecen signos de fatiga en el conductor. En caso de encontrarlas, puede cambiar la emisora de radio, abrir una ventana e, incluso, rociar al conductor con agua fría. En 2001, Volkswagen AG se convirtió en la primera empresa de fabricación de coches en producir en cadena un vehículo con conexión a Internet (adecuadamente, la VW eGeneration fue vendida inicialmente sólo por la Red).

Muy pronto, las computadoras entrarán a formar parte de nuestro vestuario. Muchas de las computadoras que se pueden poner son cinturones para la recopilación activa de información. Pero los investigadores del MIT y otros muchos están cosiendo CPU, teclados y touchpads en nuestras ropas, convirtiéndolas en nodos inalámbricos de Internet. Estos pertrechos digitales no son sólo prendas de moda de alta tecnología; cuando se llevan junto con una pantalla de retina podrían convertirse en una herramienta inestimable para todos aquellos trabajos que precisan de actividad y de conectividad.

En Japón, las computadoras han llegado incluso a los cuartos de baño. Algunos fabricantes japoneses de adornos venden inodoros inteligentes controlados por computadora. Algunos modelos obtienen y almacenan automáticamente información sobre la presión sanguínea, el pulso, la temperatura, la orina y el peso. Esta información puede mostrarse en un monitor LCD, acumularse durante meses e, incluso, transmitirse por módem a un servicio médico. Los usuarios de estos inodoros inteligentes obtienen un minichequeo siempre que van al baño. Las características de monitorización corporal ofrecen a estos inodoros una nueva función, función que, indudablemente, podría salvar vidas.

Cuando esto ocurre, no cabe duda que estamos entrando en una era de computadoras omnipresentes. Durante varios años, investigadores de Xerox PARC, la universidad de Cambridge, Olivetti y otros han estado experimentando con tecnologías que harán que las computadoras estén todavía más presentes en nuestras vidas. Mark Weiser, del PARC, describe una oficina experimental equipada con dispositivos inteligentes «Puertas que sólo se abren al portador adecuado de uno de estos identificadores, habitaciones que saludan a la gente por su nombre, llamadas telefónicas que puedan redirigirse automáticamente a cualquier lugar en el que se encuentre su destinatario, terminales que recuperan las preferencias de cualquiera que esté sentado en ellos y diarios de citas escritos por ellos mismos».

VIDEOS

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Lógica de programación: Estructura selectiva simple.
URL: https://youtu.be/T9sg17TVgo4?t=3s
Duración: 4 min 47 s.
Autor(a): Facomsys.
Expositor(a): Fernando Arroyo Alarcón.
Año: 2010.
Licencia: YouTube estándar.

Video 12: Estructura selectiva simple.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Estructura de control while (ciclos repetitivos).
URL: https://youtu.be/ERTzfGoCXds?t=1s
Duración: 4 min 14 s.
Autor(a): CL-Security.
Año: 2012.
Licencia: YouTube estándar.

Video 13: Ciclos repetitivos

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: La estructura repetitiva hacer mientras (do while)
URL: https://youtu.be/REQkHnZTS18?t=22s
Duración: 12 min 27 s.
Autor(a): BinaryCode
Año: 2014.
Licencia: YouTube estándar.

Video 14: Estructura repetitiva hacer mientras.

ACTIVIDAD N° 2

Elabora un flujogramas que represente a un algoritmo utilizando estructuras: secuencial, selectivas, repetitivas

INSTRUCCIONES:

1. Lee y analiza, todos los contenidos de tema N° 2.
2. Como apoyo visualice el siguiente video:
   - https://www.youtube.com/watch?v=T9sg17TVgo4
   - https://www.youtube.com/watch?v=ERTzfGoCXds
   - https://www.youtube.com/watch?v=REQkHnZTS18
3. Escribir un algoritmo en pseudocódigo y en Diagrama de flujo que permita calcular el promedio de notas de 5 alumnos del colegio "Miguel Grau " de la asignatura de programación, cada alumno tiene 3 nota, se pide:
   - Calcular el promedio de cada alumno, los pesos de las notas son Nota 01: 30%, nota 02: 30% y nota 03: 40%
   - Calcular el promedio del salón.
4. Revisar su trabajo y enviarlo al aula virtual.

PRUEBA DE DESARROLLO N° 1

INSTRUCCIONES:

Lea cuidadosamente cada enunciado y responda según se requiera (Remarque/escriba con color azul su respuesta).

1. Escribir un algoritmo en pseudocódigo y en Diagrama de flujo que permita el ingreso de 5 números positivos y determine cuántos son mayores a 50 y cuántos son menores a 15. **(4 puntos)**

2. Escribir un algoritmo en pseudocódigo y en Diagrama de flujo que permita calcular la siguiente función matemática: **(4 puntos)**

3. Escribir un algoritmo en pseudocódigo y en Diagrama de flujo que permita calcular el promedio de 5 notas con la siguiente formula: **(4 puntos)**

Prom = N1(20%) +N2(15%)+N3(15%)+N4(25%)+N5(25%)

4. Escribir un algoritmo en pseudocódigo y en Diagrama de flujo que permita calcular el promedio de 5 alumnos, cada alumno tiene 3 notas, se pide: **(8 puntos)**

- El promedio de cada alumno.
- El promedio del salón.
- El promedio mayor.
- El promedio menor.

GLOSARIO DE LA UNIDAD II

A

ALGORITMO
Secuencia de pasos ordenados que pretenden para solucionar un problema.

B

BUCLE
Ciclo repetitivo para ejecutar n veces.

D

DIAGRAMA
Dibujo que sirve para resolver un problema.

P

PROCESAR
Someter datos o materiales a una serie de operaciones programadas.

V

VARIABLE
Magnitud que puede tener un valor cualquiera de los comprendidos en un conjunto

BIBLIOGRAFÍA DE LA UNIDAD II

- Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación.* México: Pearson Educación.
- Joyanes, Luis. (2008). Fundamentos de Programación. Madrid: Mc Graw Hill.
- Joyanes, Luis. (2007). Estructura de datos en C++. Madrid: Mc Graw Hill.
- Lecca Eduardo. (2000) El poder de turbo C++. Perú: Mundigraf.

AUTOEVALUACIÓN Nº 2

INSTRUCCIONES: Lea cuidadosamente cada enunciado y responda según se requiera (Remarque/escriba con color azul su respuesta).

1. Es una secuencia ordenada de pasos ordenados con la finalidad de dar solución a un problema.

a. Algoritmo.

b. Secuencia.

c. Proceso.

d. Estructura Si ... Entonces.

e. Estructura Mientras

2. Es un tipo de estructura selectiva.

a. Estructura selectiva simple.

b. Estructura selectiva relativamente simple

c. Estructura selectiva superior.

d. Estructura Si ... Entonces.

e. Estructura Mientras

3. La sentencia For es un tipo de estructura.

a. Estructura repetitiva.

b. Estructura selectiva relativamente simple

c. Estructura selectiva superior.

d. Estructura secuencial.

e. Estructura repetitiva secuencial simple.

4. La sentencia While es un tipo de estructura.

a. Estructura repetitiva.

b. Estructura selectiva relativamente simple

c. Estructura selectiva superior.

d. Estructura secuencial.

e. Estructura repetitiva secuencial simple.

5. La sentencia IF es un tipo de estructura.

   a. Estructura Selectiva.

   b. Estructura repetitiva relativamente simple

   c. Estructura repetitiva superior.

   d. Estructura secuencial.

   e. Estructura repetitiva secuencial simple.

UNIDAD III

INTRODUCCIÓN A LA PROGRAMACIÓN

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD III

UNIDAD III

TEMA N° 2

CONTENIDOS	ACTIVIDADES FORMATIVAS (habilidades y actitudes)	SISTEMA DE EVALUACIÓN (Técnicas y Criterios)
TEMA Nº 1: Introducción a la programación. 1 Etapas en la construcción de un programa. 2 Estructura de un programa 3 Programación estructurada 4 Programación orientada a objetos 5 Ejemplos de programas **TEMA Nº 2: Funciones.** 1 Definición y sintaxis 2 Componentes de una función 3 Variables locales y Globales 4 Tipos de Funciones	• Identifica las etapas de construcción de un programa y Diferencia entre programación estructurada y programación orientada a objetos. • Reconoce la importancia del uso de las funciones en la programación y **construye un programa teniendo como base los algoritmos para la solución de problemas** • **Prueba de desarrollo de codificación en C++.**	**Procedimientos e indicadores de evaluación permanente** • Entrega puntual de trabajos realizados. • Calidad, coherencia y pertinencia de contenidos desarrollados. • Prueba individual. • Actividades desarrolladas en sesiones tutorizadas. **CRITERIOS DE EVALAUCION DE UN PROGRAMA EN LENGUAJE C ++** • Permite ingresar 25 notas. • Utiliza funciones en la solución del problema. • Reportar la cantidad de notas aprobadas. • Reporta la cantidad de notas desaprobadas. • Reporta la cantidad de notas iguales a 20. • Reporta la cantidad de notas iguales a 05.

RECURSOS:

Vídeos o imágenes:

Tema Nº 1 :

¿Como empezar a Programar?

https://www.youtube.com/watch?v=9idgIGmQvAQ&list=PLw8RQJQ8K1ySN6bVHYEpDoh-CKVkL_uOF

Primer Programa Hola Mundo

https://www.youtube.com/watch?v=wgiBStujBCw&index=2&list=PLw8RQJQ8K1ySN6bVHYEpDoh-CKVkL_uOF

Tema N° 2

Introducción a Funciones en C++

https://www.youtube.com/watch?v=ZYCTqYvDEI8

Lectura complementaria:

Lectura Seleccionada Nº 1

Steve wozniak, steve jobs y el garaje que vio crecer las manzanas (apples)

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

Lectura Seleccionada Nº 2

Tim berners-lee teje la web para todos

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

INSTRUMENTO DE EVALUACIÓN	• Prueba Objetiva. • Prueba de Desarrollo
BIBLIOGRAFÍA (BÁSICA Y COMPLEMENTARIA)	**BÁSICA** JOYANES AGUILAR, Luis. *Fundamentos de programación. Algoritmos, estructuras de datos y objetos*, Tercera Edición. Madrid: Editorial McGraw-Hill, 2003. **COMPLEMENTARIA** PRIETO, A., LLORIS, A. y TORRES, *Introducción a la Informática*, Tercera Edición, Madrid: Editorial McGraw-Hill, 2005.
RECURSOS EDUCATIVOS DIGITALES	LÓPEZ GARCÍA, Juan Carlos, "Algoritmos y Programación", Año 2009 [ref. de 09 de noviembre de 2009].Disponible en Web:<http://www.eduteka.org/GuiaAlgoritmos.php>

TEMA N° 1: INTRODUCCIÓN A LA PROGRAMACIÓN

Un programa es una secuencia de instrucciones que indican las acciones que han de ser ejecutadas por una computadora.

1.-ETAPAS EN LA CONSTRUCCIÓN DE UN PROGRAMA

El proceso de solución de un problema con una computadora conduce a la escritura de un programa y a su ejecución en la misma. Aunque el proceso de diseñar programas es, esencialmente creativo, se puede considerar una serie de fases o pasos comunes que generalmente deben seguir todos los programadores

- **Análisis del problema:** El problema se analiza teniendo presente la especificación de los requisitos dados por el cliente de la empresa o por la persona que encarga el programa.
- **Diseño del algoritmo:** Una vez analizado el problema, se diseña una solución que conducirá a un algoritmo que resuelve el problema.
- **Codificación:** La solución se escribe en la sintaxis del lenguaje de alto nivel y se obtiene un programa fuente que se compila a continuación.
- **Ejecución Verificación y Depuración:** El programa se ejecuta, se comprueba rigurosamente y se eliminan todos los errores.
- **Mantenimiento:** El programa se actualiza y modifica, cada vez que sea necesario de modo que se cumpla con las necesidades de cambio del usuario.
- **Documentación:** Escritura de las fases del ciclo de vida del software. esencialmente el análisis, diseño y codificación, unidos a manuales de usuario y de referencia.

1.1.-Lenguaje de Programación

Un lenguaje de programación es un lenguaje que nos permite comunicarnos con una computadora.

Es una conversión para escribir descripciones que pueden ser adecuadas. Uno de los elementos más importantes en la tarea de programación es el lenguaje que vamos a usar; el lenguaje elegido fuerza al programador a pensar de una forma determinada influyendo en la calidad de los programas desarrollados.

Es una notación para escribir programas a través de los cuales podemos comunicarnos con el Hardware y dar así las órdenes adecuadas para la realización de 1 determinado proceso.

Los distintos niveles de programación existentes nos permiten acceder al Hardware de tal forma que según utilicemos un nivel u otro así tendremos que utilizar un determinado lenguaje ligado a sus correspondientes traductores.

En el siguiente esquema se representan los distintos niveles de acceso a la máquina teniendo en cuenta que por el único que se accede al Hardware directamente es por el lenguaje máquina por el resto accedemos a un lenguaje virtual que considera el lenguaje del nivel en que estemos como su lenguaje máquina.

Los niveles de programación los podemos dividir según la posibilidad que presentan respecto al Hardware en:

- Nivel bajo (Lenguaje máquina): Este lenguaje solo es entendido por la máquina: Depende del modelo del ordenador, es difícil de programar.
- Nivel medio(Esamblador):Es equivalente al lenguaje máquina, asocia nemónicos a las operaciones que entiende el CPU.
- Nivel Alto (Gestión, Científicos, propósito general, específicos): Permite que el programador exprese el procesamiento de datos de forma simbólica, sin tener en cuenta los detalles específicos de la máquina.

2.-ESTRUCTURA DE UN PROGRAMA:

2.1.-Elementos básicos de un Programa

En programación se debe separar la diferencia entre diseño del algoritmo y su implementación en un lenguaje específico. Por ello se debe distinguir claramente entre los conceptos de programación y el medio en que ellos se implementan en un lenguaje específico.

Los lenguajes de programación tienen elementos básicos que se utilizan como bloques constructivos, asi como reglas para las que esos elementos se combinan. Estas reglas se denominan sintaxis del lenguaje. Solamente las instrucciones sintácticamente correctas pueden ser interpretadas por la computadora y los programas que contengan errores de sintaxis son rechazados por la máquina.

Los elementos básicos constitutivos de un programa son:

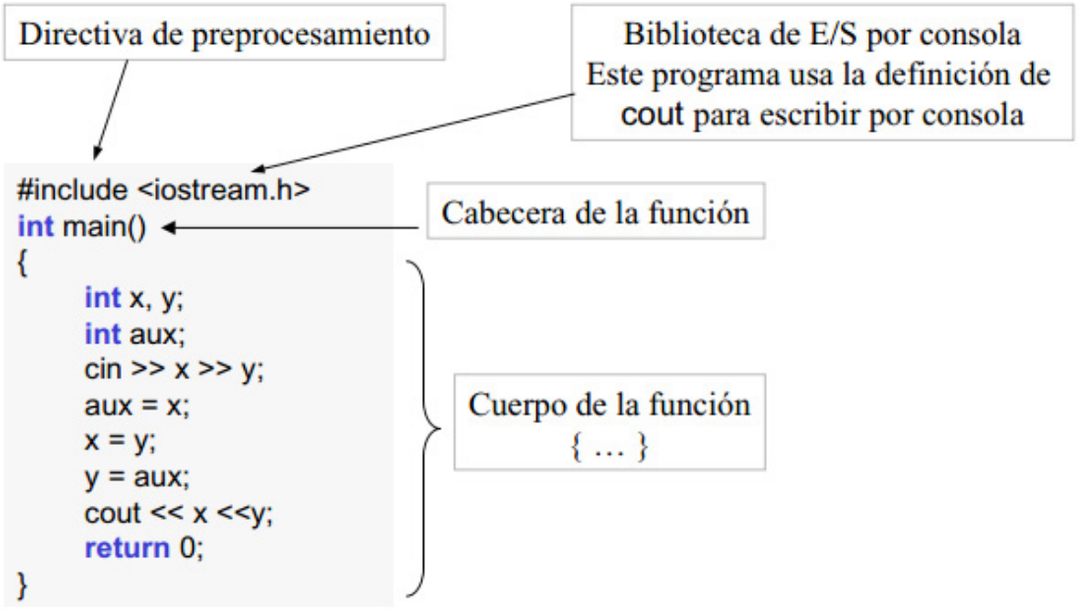
- Palabras reservadas (inicio, fin , si-entonces)
- Identificadores (procedimientos, funciones, nombre del programa)
- Caracteres especiales
- Constantes
- Variables
- Expresiones
- Instrucciones

Una estructura general sería:

```
Encabezamiento
main( )
{       Variables locales
        Sentencias
}f1( )
{       variables locales
        sentencias
}
...
...
...

fn( )
{
        variables locales
        sentencias
}
```

Para entender mejor veamos una estructura de un programa en el lenguaje C++.

Figura N° 34: Estructura de Programa en C++



Fuente: http://www.nebrija.es/

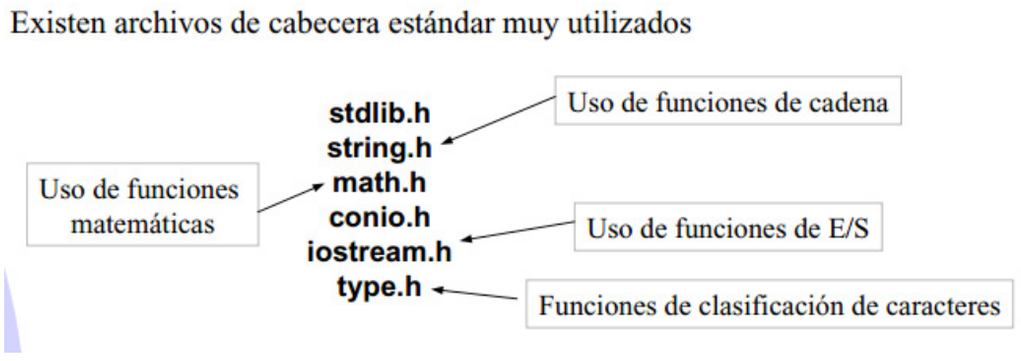
Analizaremos cada una de las partes:

- Directivas del preprocesador

  Los compiladores de C++ proporcionan bibliotecas de funciones. Cada biblioteca de funciones tiene asociada un archivo de definición que se denomina cabecera. Para utilizar algo de una biblioteca en un programa, hay que colocar al principio del programa una directiva de preprocesamiento seguida de la cabecera de la biblioteca.

  Las directivas más utilizadas son: #Include y #define.

Figura N° 35: Directiva del Procesador

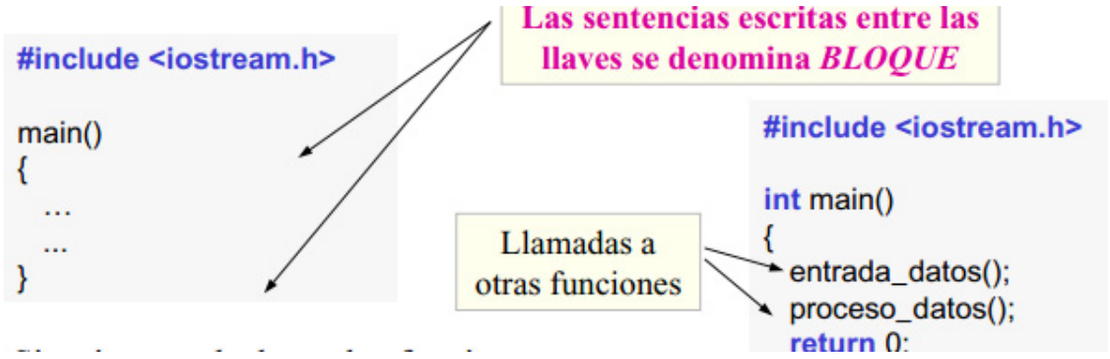


Fuente: http://www.nebrija.es/

- La función Main()

  Una función C++ es un subprograma que devuelve un valor, un conjunto de valores o realiza una tarea específica. Todo programa C++ tiene una única función main() que es el punto inicial de entrada al programa.

Figura N° 36: Función principal Main



Fuente: http://www.nebrija.es/

- CARACTERÍSTICAS DEL LENGUAJE C++:

  Al momento de iniciar un programa debemos tener en claro lo siguiente:

  Se distingue entre mayúsculas y minúsculas.

  Palabras clave: siempre en minúsculas.

  Todas las sentencias y declaración de variables terminan en punto y coma.

  La ejecución siempre comienza con la función main()

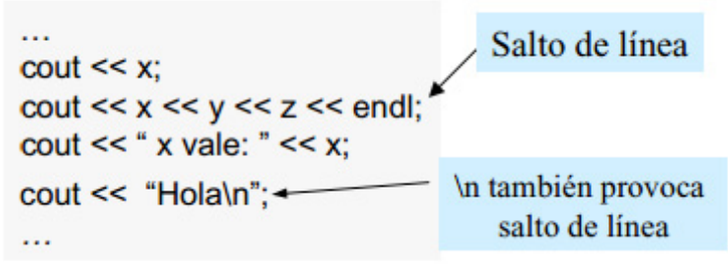
- Instrucciones de Entrada / Salida

  En C++ la entrada y salida se lee y escribe en flujos. Cuando se incluye la biblioteca iostream.h en el programa, se definen automáticamente dos flujos:

  Flujo cin (se utiliza para la entrada de datos)

  Flujo cout (se utiliza para la salida de datos)

Figura N° 37: Función principal Main



Fuente: http://www.nebrija.es/

C++ utiliza secuencias de escape para visualizar caracteres que no están representados por los símbolos tradicionales. Las más utilizadas las mostramos en la siguiente tabla:

\n Retorno de carro y avance de línea

\t Tabulación

\a Alarma

Ahora veremos los tipos de datos numéricos en C++

El tipo de dato numérico entero es un subconjunto finito de los números enteros del mundo real. Pueden ser positivos o negativos.

En C++ los tipos de datos numéricos enteros son los siguientes:

TIPO DE DATO	DESCRIPCIÓN	NÚMERO DE BYTES TÍPICO	RANGO
short	Entero corto	2	-32768 a 32767
int	Entero	4	-2147483648 a +2147483647
long	Entero largo	4	-2147483648 a +2147483647
char	Carácter	1	-128 a 127

Fuente: http://ejercicioscpp.blogspot.com/

Con los tipos enteros pueden utilizarse los calificadores signed y unsigned. Estos calificadores indican si el número tiene signo o no. Si se usan solos, sin indicar el tipo de dato se asume int.

Por ejemplo, las siguientes declaraciones son equivalentes:

unsigned int x; equivale a: unsigned x;

Usando estos calificadores podemos tener los siguientes tipos enteros:

TIPO DE DATO	DESCRIPCIÓN	NÚMERO DE BYTES TÍPICO	RANGO
signed short	Entero corto	2	-32768 a 32767
unsigned short	Entero corto sin signo	2	0 a 65535
signed int	Entero	4	-2147483648 a +2147483647
unsigned int	Entero sin signo	4	0 a 4294967295
signed long	Entero largo	4	-2147483648 a +2147483647
unsigned long	Entero largo sin signo	4	0 a 4294967295
signed char	Carácter	1	-128 a 127
unsigned char	Carácter sin signo	1	0 a 255

Fuente: http://ejercicioscpp.blogspot.com/

3.-PROGRAMACIÓN ESTRUCTURADA:

La programación estructurada es una teoría de programación que consiste en construir programas de fácil comprensión, es especialmente útil, cuando se necesitan realizar correcciones o modificaciones después de haber concluido un programa o aplicación. Al utilizar la programación estructurada, es mucho más sencillo entender la codificación del programa, que se habrá hecho en diferentes secciones.

Se basa en una metodología de desarrollo de programas llamada refinamiento sucesivo: Se plantea una operación como un todo y se divide en segmentos más sencillos o de menor complejidad, una vez terminado todos los segmentos del programa, se procede a unificar las aplicaciones realizadas por el grupo de programadores. Si se ha utilizado adecuadamente la programación estructurada, esta integración debe ser sencilla y no presentar problemas al integrar la misma, y de presentar algún problema, será rápidamente detectable para su corrección.

La representación gráfica de la programación estructurada se realiza a través de diagramas de flujo, el cual representa el programa con sus entradas, procesos y salidas.

La programación estructurada propone segregar los procesos en estructuras lo más simple posibles, las cuales se conocen como secuencia, selección e interacción, que están disponibles en todos los lenguajes modernos de programación imperativa en forma de sentencias, combinando esquemas sencillos se pueden llegar a construir sistemas amplios y complejos pero de fácil entendimiento.

La programación estructurada es un método disciplinado de escribir programas que sean claros, que se demuestre que sean correctos y fáciles de modificar.

La programación estructurada consiste en dividir los programas en módulos y se basa en el desarrollo de programas que van de lo general a lo particular, es decir, del conjunto al elemento, es decir de un todo a lo específico.

Para la solución de un problema en particular, se inicia considerando las funciones que tiene que cumplir el programa en general y después se va desmembrando estas funciones en subfunciones más pequeñas hasta llegar al caso último o más particular y que ya no se pueda subdividir en casos más pequeños. Una vez que ya se tiene el programa desmembrado en de lo general a lo particular, se empieza a programar estas funciones pequeñas, particulares o módulos, de esta manera, siempre podremos construir nuevos módulos o unidades insertando el nombre del módulo donde corresponda y desarrollándolo a parte.

La modificación de los módulos es más fácil y se pueden referenciar cuantas veces se requiera, con lo que se ahorra tiempo en la programación, un programa tiene un diseño estructurado si cumple las dos siguientes condiciones:

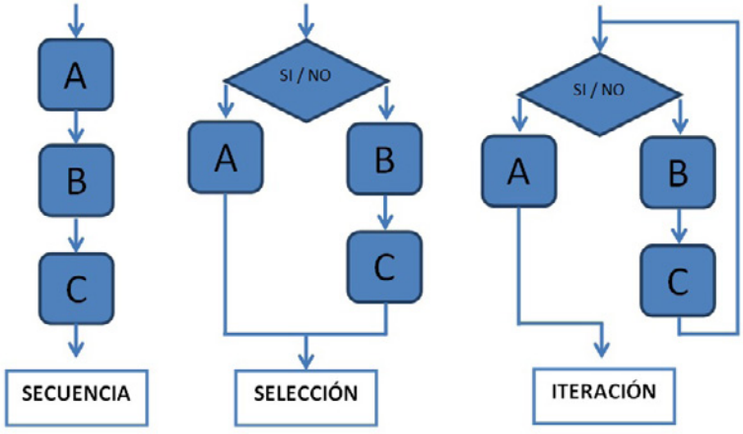
- El teorema de Estructura.
- Está debidamente documentado

El teorema de Estructura dice que "un programa cumple el teorema de estructura si y sólo (ó) si es propio y contiene únicamente las tres estructuras básicas de control" que son la secuencial, la alternativa y la repetitiva, un programa es propio si y sólo si cumple: que tenga un solo punto de entrada y un solo punto de salida y que entre dos puntos de control del programa exista al menos un camino.

La programación estructurada es un estilo con el cual él se busca que el programador elabore programas sencillos y fáciles de entender, la programación estructurada hace uso de tres estructuras básicas de control que son: Estructura Secuencial, Estructura Selectiva y la Estructura Repetitiva (ó Iterativa)

La programación estructurada se basa un teorema fundamental, el cual afirma que cualquier programa, no importa el tipo de trabajo que ejecute, puede ser elaborado utilizando únicamente las tres estructuras básicas.

Figura N° 38: Estructuras básicas



Estructuras básicas de la programación estructurada

EL creciente empleo de los computadores ha conducido a buscar un abaratamiento del desarrollo de software, paralelo a la reducción del costo del hardware obtenido gracias a los avances tecnológicos. Los altos costos del mantenimiento de las aplicaciones en producción normal también han urgido la necesidad de mejorar la productividad del personal de programación.

En la década del sesenta salieron a la luz publica los principios de lo que más tarde se llamó Programación Estructurada, posteriormente se liberó el conjunto de las llamadas "Técnicas para mejoramiento de la productividad en programación" (en inglés Improved Programming Technologies, abreviado IPTs), siendo la Programación Estructurada una de ellas.

Los programas computarizados pueden ser escritos con un alto grado de estructuración, lo cual les permite ser más fácilmente comprensibles en actividades tales como pruebas, mantenimiento y modificación de los mismos. Mediante la programación Estructurada todas las bifurcaciones de control de un programa se encuentran estandarizadas, de forma tal que es posible leer la codificación del mismo desde su inicio hasta su terminación en forma continua, sin tener que saltar de un lugar a otro del programa siguiendo el rastro de la lógica establecida por el programador, como es la situación habitual con codificaciones desarrolladas bajo otras técnicas.

3.1 SEGMENTACION

Para la comprensión de un programa se haga en forma fácil y rápida es necesario que, al revisarlo, uno no tenga que hojear una gran cantidad de páginas para entender cuál es el trabajo que realiza. Una regla practica para lograr estos fines es establecer que cada segmento del programa no exceda, en longitud, a una página de codificación, o sea, alrededor de 50 líneas (el significado que se asigna al termino segmento, en este trabajo, no tiene ninguna relación con su significado en relación a las funciones de sistemas operativos o sistemas maneadores de Bases de Datos).

La segmentación no es solamente particionar un programa en trozos cuya longitud sea de unas 50 líneas; esta técnica debe cumplir con ciertas características fundamentales:

a. La segmentación reflejara la división del programa en partes que se relacionen entre sí en forma jerárquica, formando una estructura de árbol. Esta organización puede ser representada gráficamente por un diagrama de procesos, lo que hace más sencillo comprender la relación existente entre un segmento y el resto del programa. Adicionalmente, podemos indicar que, el segmento en la cumbre de la estructura jerárquica contendrá las funciones de control de más alto nivel, mientras que los segmentos inferiores en esta organización contendrán funciones detalladas.

b. Una segmentación bien diseñada deberá mostrar, claramente, las relaciones existentes entre las distintas funciones de manera que sea fácil comprender lo que debe hacer el programa y asegurar que efectivamente lo realice. Este hecho, garantizara que los cambios que se efectúen a una parte del programa, durante la programación original o su mantenimiento, no afecten al resto del programa que no ha sufrido cambios.

c. En una segmentación bien realizada la comunicación entre segmentos se lleva a cabo de una manera cuidadosamente controlada. Algunos autores recomiendan que los segmentos consistan en procedimientos y la única comunicación existente entre ellos sea a través de una lista de parámetros, esto reduce la oportunidad de que interactúen entre ellos de una manera indeseada e inentendible.

3.2 IDENTACION

El uso de la identación es importante debido a que, cuando se es consistente en su utilización, facilita la lectura del programa al mostrar en una forma gráfica las relaciones existentes entre las distintas instrucciones.

DIRECTRICES PARA IDENTAR

Debe comprenderse claramente que las líneas siguientes solo pretenden presentar unas directrices de identación, sin pretender que estas sean las únicas reglas a seguir en este proceso, cada centro de procesamiento deberá establecer sus propias convenciones, sin que sea motivo de preocupación la diferencia respecto a las sugerencias dadas aquí, lo importante es que se establezcan unas normas y se cumplan de manera consistente.

Las siguientes son sugerencias para el desarrollo de una política de identación en un centro de procesamiento:

- En los lenguajes donde se permite el uso de etiquetas, estas deben colocarse lo más externas posibles, por ejemplo comenzando en la columna 2, y deben estar separadas por una línea (siempre que lo permita el lenguaje en uso).
- Se obtiene consistencia si todas las instrucciones se comienzan en una misma columna, por ejemplo en la columna 4 o cualquier otra ubicada a su derecha.
- En los lenguajes en que se hagan declaraciones sobre las variables a utilizar, la información quedara más claramente representada si los atributos declarados se alinean en forma vertical.
- El uso de líneas en blanco ayuda a mostrar con más claridad las relaciones existentes entre distintos ítems agrupados en las declaraciones.
- Las instrucciones son mucho más fáciles de localizar y de cambiar si no se escribe más de una instrucción por línea.
- La visión de control de las estructuras lógicas o de los bloques se clarifica si las instrucciones controladas son idénticas por alguna cantidad constante. Se sugiere una identación de tres espacios.

4.-PROGRAMACIÓN ORIENTADA A OBJETOS

La programación orientada a objetos es otra forma de descomponer problemas. Este nuevo método de descomposición es la descomposición en objetos; vamos a fijarnos no en lo que hay que hacer en el problema, sino en cuál es el escenario real del mismo, y vamos a intentar simular ese escenario en nuestro programa.

La programación orientada a objetos (POO) es una forma de programación en computadoras que tiene surge los años 70 pero tiene un desarrollo sorprendente los años 90 al utilizarlo en las microcomputadoras. Se diferencia de la programación clásica o estructurada en que las instrucciones hacen referencia a los elementos del entorno. Esos elementos representan "objetos"; y todos los datos y todas las acciones que se hagan con ellos o sobre ellos, están encapsuladas u ocultas en el objeto.

La programación orientada a Objetos básicamente define una serie de conceptos y técnicas de programación para representar acciones o cosas de la vida real basada en objetos, a diferencia de otras formas de programación como por ejemplo la estructurada, con la POO trabajamos de manera distinta vinculando diferentes conceptos tales como clases, objetos, métodos, propiedades, estados, herencia, encapsulación entre otros, generando cada vez interrelaciones en nuestro desarrollo en pro del funcionamiento del sistema principal, definiendo el programa como un conjunto de estos objetos relacionados entre sí.

Los elementos de la POO, pueden entenderse como los materiales que necesitamos para diseñar y programar un sistema:

Clases:

Las clases son los modelos sobre los cuáles se construirán nuestros objetos. Las clases son uno de los principales componentes de un lenguaje de programación, pues en ellas ocurren todos los procesos lógicos requeridos para un sistema, en si podemos definirlas como estructuras que representan objetos del mundo real, tomando como objetos a personas, lugares o cosas, en general las clases poseen propiedades, comportamientos y relaciones con otras clases del sistema

Métodos:

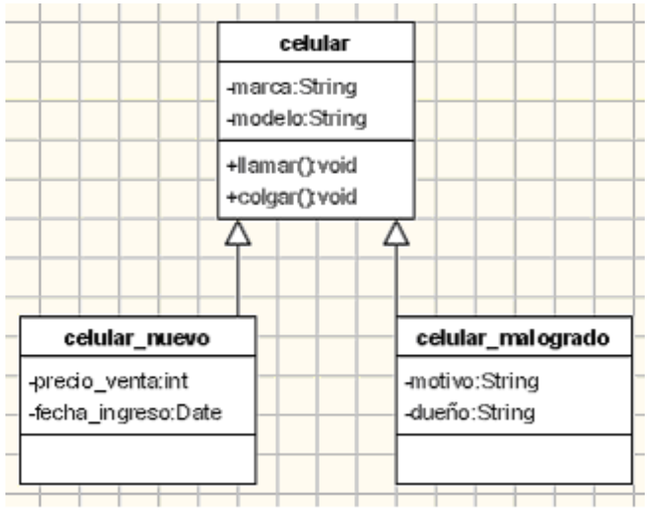
Los métodos son *funciones* (como las que vimos), solo que técnicamente se denominan métodos, y representan acciones propias que puede realizar el objeto (y no otro):

Objeto:

Las clases por sí mismas, no son más que modelos que nos servirán para crear objetos en concreto. Podemos decir que una clase, es el razonamiento abstracto de un objeto, mientras que el objeto, es su materialización. A la acción de crear objetos, se la denomina *instanciar una clase* y dicha instancia, consiste en asignar la clase, como valor a una variable:

Los objetos representan una entidad concreta o abstracta del mundo real, en programación básicamente se le conoce como la instancia de una clase en si es lo que da el sentido a estas.

Figura N° 39: Programación Orientada a Objetos

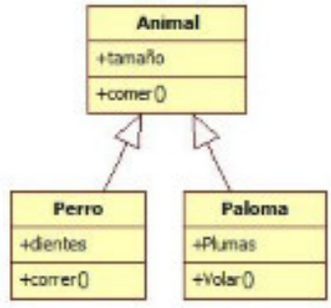


Fuente http://codejavu.blogspot.com/2013/05/conceptos-de-programacion-orientada.html

Herencia:

La herencia en java representa lo que conocemos de herencia en el mundo real, básicamente mediante esta obtenemos las características o rasgos comunes de nuestros padres o abuelos, en java es el mismo enfoque permitiendo la creación de nuevas clases basadas en clases ya existentes, con las cuales podemos obtener las características de las clases padres, heredando campos, atributos, métodos o funcionalidades.

Figura N° 40: Diagrama de Herencia



Fuente: http://codejavu.blogspot.com/2013/05/conceptos-de-programacion-orientada.html

Encapsulamiento: Este concepto es uno de los más importantes en términos de seguridad dentro de nuestra aplicación, la encapsulación es la forma de proteger nuestros datos dentro del sistema, estableciendo básicamente los permisos o niveles de visibilidad o acceso de nuestros datos

Polimorfismo:

Este tal vez sea uno de los conceptos de la programación orientada a objetos más usados pero muchas veces sin saber que se aplica ya que el concepto inicialmente puede ser un poco confuso, básicamente mediante el polimorfismo programamos de forma general en lugar de hacerlo de forma específica, se usa cuando se trabajen con la herencia y objetos de características comunes los cuales comparten la misma superClase y árbol jerárquico, al trabajar con este concepto optimizamos y simplificamos en gran medida nuestro trabajo.

4.1 Análisis y diseño Orientado a objetos

Para el desarrollo de software orientado a objetos no basta usar un lenguaje orientado a objetos. También se necesitará realizar un análisis y diseño orientado a objetos.

El modelamiento visual es la clave para realizar el análisis OO. Desde los inicios del desarrollo de software OO han existido diferentes metodologías para hacer esto del modelamiento, pero sin lugar a duda, el Lenguaje de Modelamiento Unificado (UML) puso fin a la guerra de metodologías.

Según los mismos diseñadores del lenguaje UML, éste tiene como fin modelar cualquier tipo de sistemas (no solamente de software) usando los conceptos de la orientación a objetos. Y además, este lenguaje debe ser entendible para los humanos y máquinas.

Actualmente en la industria del desarrollo de software tenemos al UML como un estándar para el modelamiento de sistemas OO. Fue la empresa Racional que creó estas definiciones y especificaciones del estándar UML, y lo abrió al mercado. La misma empresa creó uno de los programas más conocidos hoy en día para este fin; el Racional Rose, pero también existen otros programas como el Poseidon que trae licencias del tipo community edition que permiten su uso libremente.

El UML consta de todos los elementos y diagramas que permiten modelar los sistemas en base al paradigma

orientado a objetos. Los modelos orientados a objetos cuando se construyen en forma correcta, son fáciles de comunicar, cambiar, expandir, validar y verificar. Este modelamiento en UML es flexible al cambio y permite crear componentes plenamente reutilizables.

5.-EJEMPLOS DE PROGRAMAS

Después de haber revisado los conceptos sobre los programas es el momento para empezar a diseñar los mismos:

Ejemplo 01: Escribir Un programa que permita el ingreso de dos número y reporte la suma de ellos.

Solución:

Figura N° 41: Programa en C++ Nro. 01

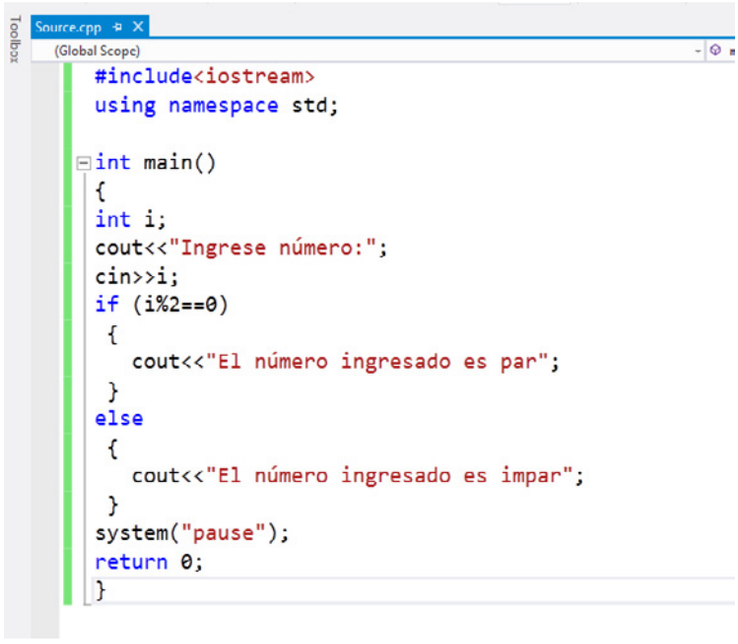
```
Source.cpp*
(Global Scope)
#include<iostream>
using namespace std;

int main()
{
int a,b,sum;
          cout<<"Ingrese 1er número:  ";
          cin>>a;
          cout<<"Ingrese 2do número:  ";
          cin>>b;
          sum = a+b;
          cout<<"la suma es: "<< sum;

system("pause");
return 0;
}
```

Fuente: Propia

Ejemplo 02: Escribir Un programa que permita el ingreso de un número y nos reporte si el número es par o impar.

Solución

Figura N° 42: Programa en C++ Nro. 02

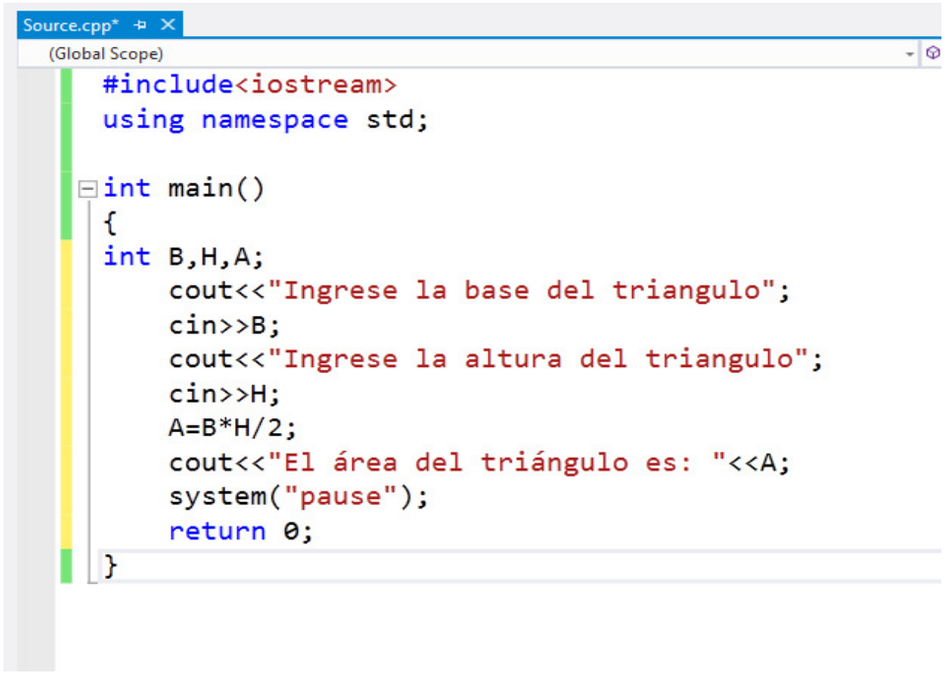


Fuente: Propia

Ejemplo 03: Escribir Un programa que calcular el área de un triángulo.

Solución

Figura N° 43: Programa en C++ Nro. 03



Fuente: Propia

LECTURA SELECCIONADA Nº. 1:

STEVE WOZNIAK, STEVE JOBS Y EL GARAJE QUE VIO CRECER LAS MANZANAS (APPLES)

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

Steve Wozniak y todas esas personas no previeron que se trataba de la revolución de la computadora personal: una revolución que él ayudó a empezar. Wozniak, un brillante ingeniero conocido por sus amigos como Woz, trabajó durante cierto tiempo como técnico de calculadoras en Hewlett-Packard; fue rechazado como ingeniero porque carecía de un título universitario. De la noche a la mañana diseñó y construyó un sistema informático a escala reducida que podía ajustarse al presupuesto de los aficionados domésticos. Cuando la completó en 1975, se la ofreció a HP, pero la rechazaron.

Wozniak mostró su invento al Homebrew Computer Club en Palo Alto, donde se encontró con la imaginación de otro joven que había abandonado la universidad, Steve Jobs. El visionario Jobs persuadió a Wozniak para que dejara su trabajo en 1976 y formaran una compañía que nació en el garaje de Job. Presentaron la máquina como el Apple I. Con la ayuda y la financiación del empresario A. C. Markkula, los dos Steves convirtieron Apple en un próspero negocio. Wozniak creó el Apple II, una máquina más refinada, y durante el proceso inventó el primer sistema operativo en disco para una computadora personal. Al poner toda la potencia de una computadora al alcance del individuo, el Apple II se hizo popular en empresas, hogares y, especialmente, en escuelas. Apple se convirtió en la primera empresa de la historia americana en aparecer en la lista Fortune 500 en menos de cinco años. Con poco más de veinte años, Jobs dirigía un gigante corporativo. Pero los malos tiempos estaban a punto de llegar para Apple.

Cuando IBM presentó su PC en 1982, ensombreció la presencia de Apple en el mundo empresarial, donde la gente estaba acostumbrada a trabajar con mainframes IBM. Otras compañías desarrollaron clones del PC, tratando al IBM PC como un estándar (estándar que Apple se negó a aceptar). Inspirado por una visita al PARC (Centro de investigación de Palo Alto, Palo Alto Research Center) de Xerox, Jobs trabajó con un equipo de ingenieros Apple para desarrollar el Macintosh, la futurística computadora en la que Jobs tenía puestas sus esperanzas de aventajar a IBM. Cuando Jobs insistió en focalizar la mayor parte de los recursos de Apple en el proyecto, Wozniak se resignó a perseguir otros intereses.

Las empresas dejaron de lado a la Mac, y los accionistas de Apple se sintieron cada vez más molestos con el peculiar modo de dirigir la empresa de Jobs. En 1985, un año y medio después de que el Macintosh fuera presentado, Jobs fue despedido. Volvió con NeXT, una empresa que producía caras estaciones de trabajo y software. También compró Pixar, la empresa de animación por computadora que más tarde capturaría la atención del público con la película Toy Story, el primer largometraje generado totalmente por computadora. Después de que la suerte de Apple declinara a causa de una cadena de CEOs, la compañía compró NeXT en 1997 e invitó a un mayor, pero más sabio, Jobs a retomar las riendas. Y estuvo de acuerdo en compartir su tiempo entre Pixar y Apple. Bajo su dirección, Apple ha recobrado su rama innovadora, lanzando al mercado una elegante línea de productos.

Aunque su cuota del mercado es pequeña, Apple mantiene una fanática base de clientes, y en la actualidad se está centrando en el mercado de consumo, creativo y educativo.

VIDEOS

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: ¿Cómo empezar a Programar?
URL: https://youtu.be/9idgIGmQvAQ?t=3m40s
Duración: 2 min 46 s.
Autor(a): EmpiezaAProgramar.
Expositor(a): Emilio Bello.
Año: 2013.
Licencia: YouTube estándar.

Video 15: Instalando el Entorno de Desarrollo Integrado (IDE).

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Estructura de Programación.
URL: https://youtu.be/odNK-_PpaxM?t=9s
Duración: 3 min 13 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

Video 16: Estructura de un programa en el lenguaje C++.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Cout.
URL: https://youtu.be/YWSz-odjKss?t=10s
Duración: 5min 30 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

Video 17: Nuestro primer programa en C++

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Cout y cadenas.
URL: https://youtu.be/hfSGGnA8S6k?t=10s
Duración: 4 min 30 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

Video 18: Flujo cout.

Video 19: Tipos de datos.

Datos del Video seleccionado

Título o Tema: Tipos de Datos.
URL: https://youtu.be/vqLFK_CYDb4?t=10s
Duración: 2 min 35 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

Video 20: Las variables.

Datos del Video seleccionado

Título o Tema: Variables.
URL: https://youtu.be/-hXasTUfxQw?t=7s
Duración: 5 min 6 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

Video 21: Flujo cin.

Datos del Video seleccionado

Título o Tema: Ingreso de Datos.
URL: https://youtu.be/DGbWkPXaeRU?t=7s
Duración: 5 min 31 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

Video 22: Estructuras de decisión: IF

Datos del Video seleccionado

Título o Tema: Instrucciones Compuestas.
URL: https://youtu.be/i179tR35cPw?t=30s
Duración: 4 min 7 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

ACTIVIDAD N° 1

Identifica las etapas de construcción de un programa y Diferencia entre programación estructurada y programación orientada a objetos.

INSTRUCCIONES:

1. Lee y analiza, todos los contenidos de tema N° 1
2. Como apoyo visualiza los siguientes videos:

¿Cómo empezar a Programar?

- https://www.youtube.com/watch?v=9idgIGmQvAQ&list=PLw8RQJQ8K1ySN6bVHYEpDoh-CKVkL_uOF

Primer Programa Hola Mundo

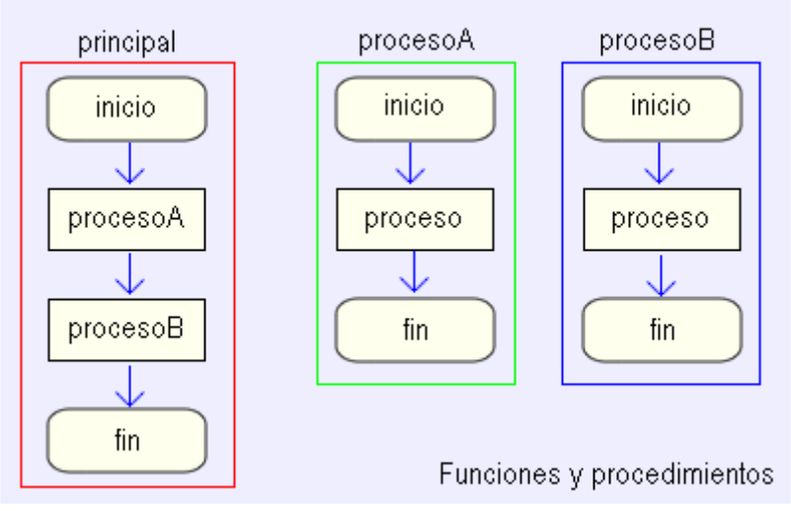
- https://www.youtube.com/watch?v=wgiBStujBCw&index=2&list=PLw8RQJQ8K1ySN6bVHYEpDoh-CKVkL_uOF

3. Describe las etapas de desarrollo de un programa.
4. Escribe en forma resumida y concreta las ventajas y desventajas de la Programación Estructurada.
5. Escribe en forma resumida y concreta las ventajas y desventajas de la Programación Orientada a Objetos.

TEMA N° 2
FUNCIONES

Las funciones son un conjunto de instrucciones que realizan una tarea específica. En general toman ciertos valores de entrada, llamados parámetros y proporcionan un valor de salida o valor de retorno.

Figura Nº 44: Funciones



Fuente: http://es.wikibooks.org/wiki/Programaci%C3%B3n_en_C%2B%2B/Funciones#/media/File:Funciones.png

1.-DEFINICIÓN Y SINTAXIS

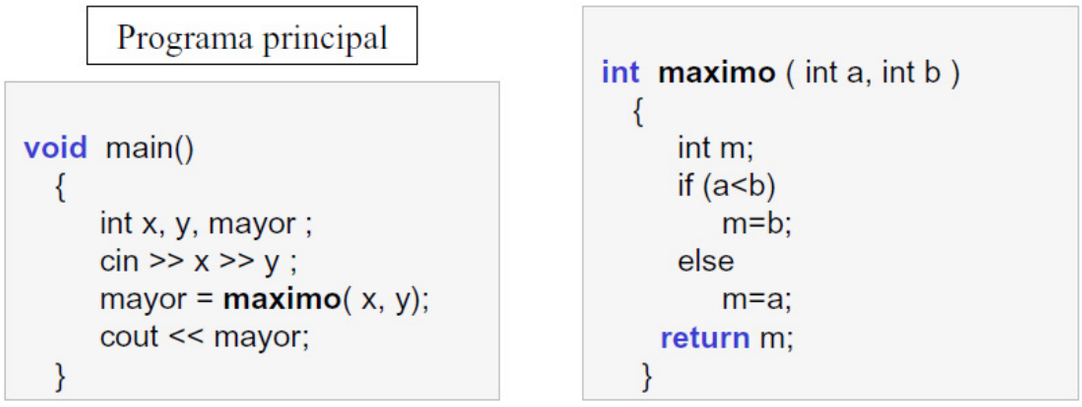
En programación, una función es una sección de un programa que calcula un valor de manera independiente al resto del programa. En conclusión, una función es un mini programa: tiene una entrada, un proceso y una salida.

La sintaxis para definir una función es la siguiente:

```
<tipo> <nombre> ( [Parámetros] )
{

        código;

        valor de retorno;

}

        Ejemplo de una función en C++:
        int suma(int a, int b)
{
        int c;
        c = a + b;
        return c;
}
```

Una vez que se ha diseñado y codificado una función, se puede usar. Para usar una función, debemos llamarla o invocarla. Una llamada, produce la ejecución de las instrucciones que se encuentran en el cuerpo.

Figura N° 45: Ejemplo Programa en C++

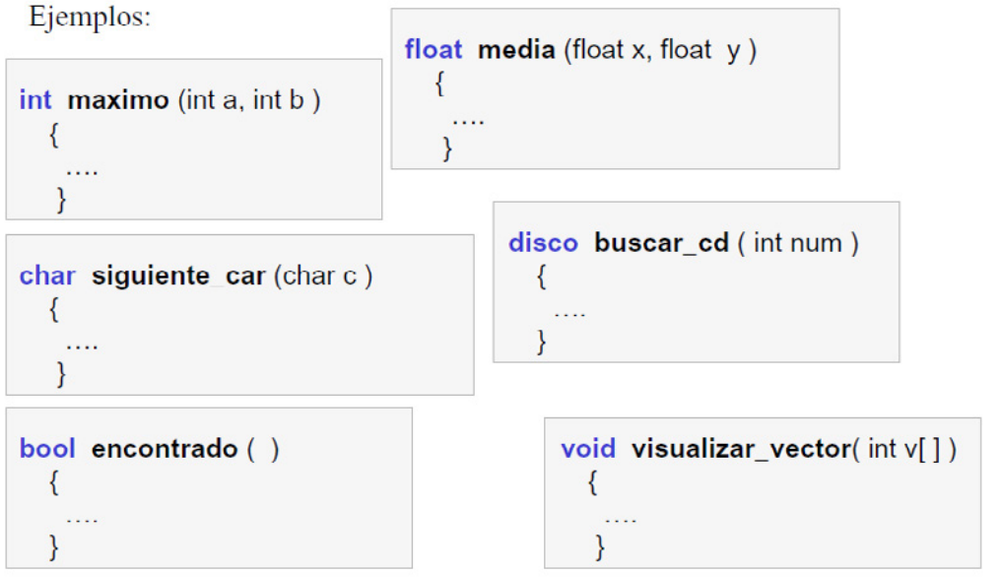


Fuente: http://www.nebrija.es

En una función se debe determinar el tipo de dato que va a devolver dicha función, existen varios tipos de datos como:

- Int (devuelve un valor entero)
- Char (devuelve un caracter)
- Float(devuleve un flotante)

Figura N° 46: Tipos devueltos por Funciones



Fuente: http://www.nebrija.es

2.- COMPONENTES DE UNA FUNCIÓN

Una función tiene tres componentes importantes:

- los parámetros, que son los valores que recibe la función como entrada;
- el código de la función, que son las operaciones que realiza la función; y
- el resultado o valor de retorno, que es el valor final que entrega la función.

Parámetros

Normalmente, las funciones operan sobre ciertos valores pasados a las mismas ya sea como constantes literales o como variables, aunque se pueden definir funciones que no reciban parámetros. Existen dos formas en C++ de pasar parámetros a una función; por referencia o por valor.

El hecho es que si en una declaración de función se declaran parámetros por referencia, a los mismos no se les podrá pasar valores literales ya que las referencias apuntan a objetos (variables o funciones) residentes en la memoria; por otro lado, si un parámetro es declarado para ser pasado por valor, el mismo puede pasarse como una constante literal o como una variable.

Los parámetros pasados por referencia pueden ser alterados por la función que los reciba, mientras que los parámetros pasados por valor o copia no pueden ser alterados por la función que los recibe, es decir, la función puede manipular a su antojo al parámetro, pero ningún cambio hecho sobre este se reflejará en el parámetro original.

Parámetros por valor

La función cuadrado () es un clásico ejemplo que muestra el paso de parámetros por valor, en ese sentido la función cuadrado() recibe una copia del parámetro n. En la misma función se puede observar que se realiza un cálculo (n*n), sin embargo el parámetro original no sufrirá cambio alguno, esto seguirá siendo cierto aun cuando dentro de la función hubiera una instrucción parecida a n = n * n; o n*=n;.

```
double Cuadrado(double n)

{
        return n*n;
}
```

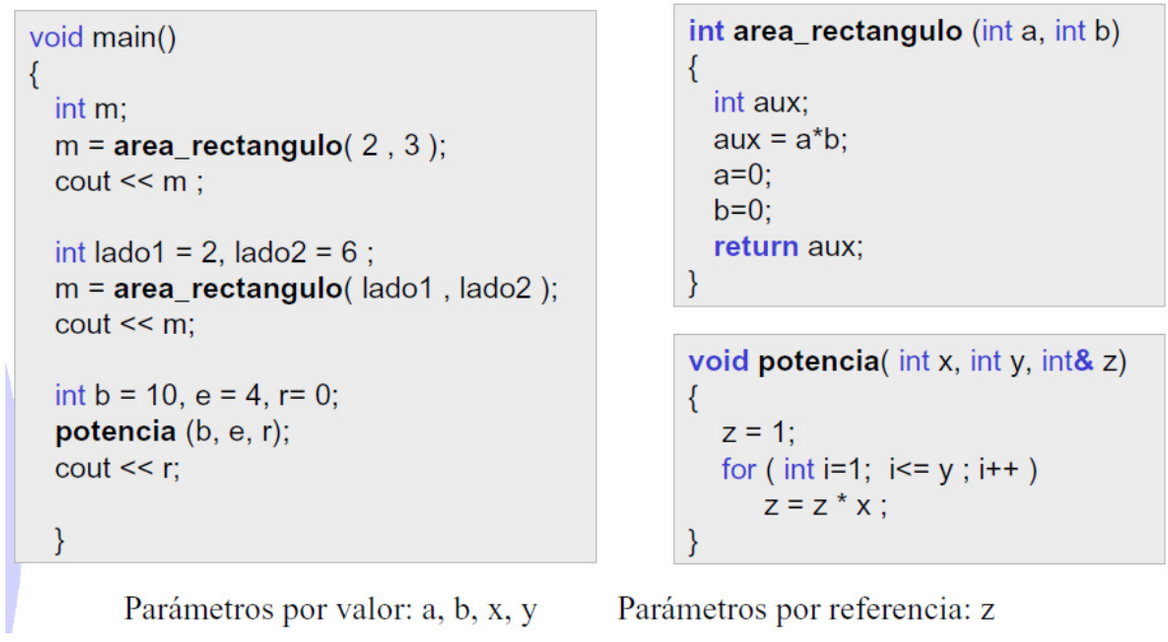
Parámetros por referencia

Para mostrar un ejemplo del paso de parámetros por referencia, vamos a retomar el caso de la función cuadrado, salvo que en esta ocasión cambiaremos ligeramente la sintaxis para definir la misma. Veamos

```
double cuadrado2(double &n)

{
        n *= n;
        return n;
}
```

Al poner a prueba las funciones cuadrado() y cuadrado2() se podrá verificar que la primera de estas no cambia el valor del parámetro original, mientras que la segunda sí lo hace. Veamos otro ejemplo del paso de parámetros:

Figura N° 47: Paso de parámetros



Fuente: http://www.nebrija.es

Características importantes relativas a funciones:

a. La instrucción return:

   i. Fuerza la salida inmediata de la función.

   ii. sirve para devolver un valor. Dicho valor puede ser constante, variable ó una expresión.

b. No se pueden declarar (DECLARAR es diferente a USAR) unas funciones dentro de otras.

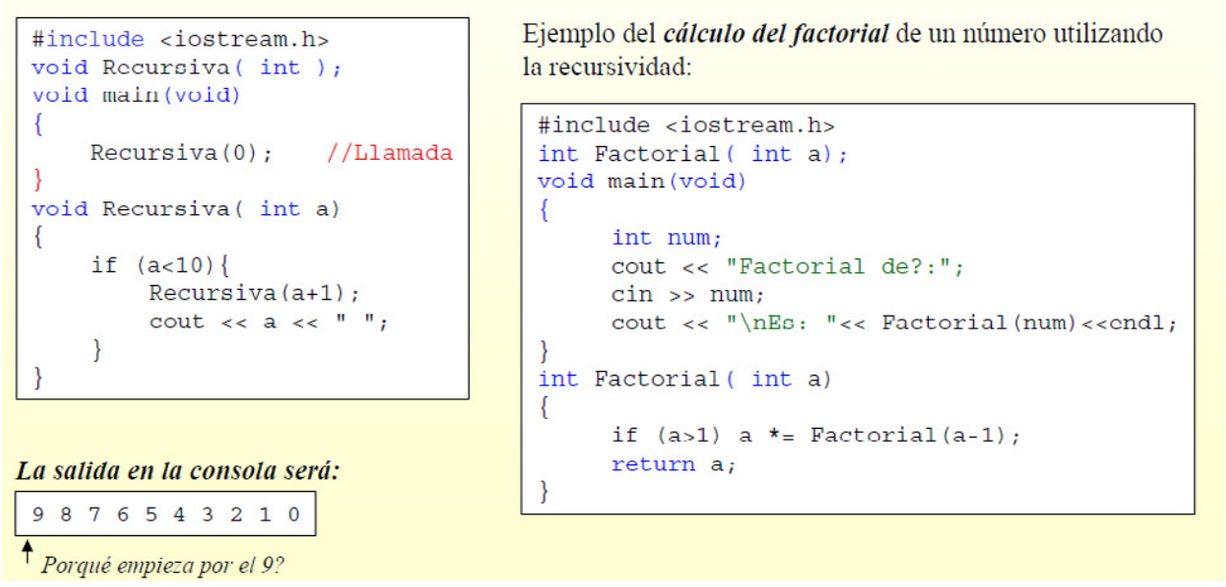
c. Las constantes, variables y tipos de datos declarados en el cuerpo de la función son locales a la misma y no se pueden utilizar fuera de ella.

d. El cuerpo de la función encerrado entre llaves, no acaba en ';'.

Funciones Recursivas:

El C++ es un lenguaje de programación que admite la recursividad, esto es, funciones que pueden llamarse a sí mismas. Cuando una función es llamada por sí misma, se crea un nuevo juego de parámetros y variables locales, pero el código ejecutable es el mismo. Es muy importante definir las condiciones dentro de la función para que la recursividad finalice y no genere un bucle infinito. Veamos un ejemplo:

Figura Nº 48: Función Factorial



```
#include <iostream.h>
void Recursiva( int );
void main(void)
{
    Recursiva(0);    //Llamada
}
void Recursiva( int a)
{
    if (a<10){
        Recursiva(a+1);
        cout << a << " ";
    }
}
```

Ejemplo del ***cálculo del factorial*** de un número utilizando la recursividad:

```
#include <iostream.h>
int Factorial( int a);
void main(void)
{
     int num;
     cout << "Factorial de?:";
     cin >> num;
     cout << "\nEs: "<< Factorial(num)<<endl;
}
int Factorial( int a)
{
     if (a>1) a *= Factorial(a-1);
     return a;
}
```

La salida en la consola será:

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

Porqué empieza por el 9?

Fuente: http://www.unav.es/adi/UserFiles/File/80971550/4.Funciones.pdf

3.-VARIABLES LOCALES Y GLOBALES

Las variables que son declaradas en una función se llaman variables locales. Ellas existen sólo mientras la función es llamada. Una vez que la función entrega su resultado, las variables locales dejan de existir, y no pueden ser utilizadas desde el resto del programa. Los parámetros de la función también son variables locales.

Las variables que son declaradas al inicio del programa se llaman variables globales, y pueden ser utilizadas en cualquier parte del programa, incluso dentro de una función.

3.1 Variable local

Una variable local es aquella cuyo ámbito se restringe a la función que la ha declarado se dice entonces que la variable es local a esa función. Esto implica que esa variable sólo va a poder ser manipulada en dicha sección, y no se podrá hacer referencia fuera de dicha sección. Cualquier variable que se defina dentro de las llaves del cuerpo de una función se interpreta como una variable local a esa función.

Cuando una variable x es local a una función func1, significa que la función func1 es la propietaria de dicha variable, y puede acceder a ella y modificarla. Si cualquier otra función del programa necesita conocer el valor de la variable x, es la función func1 la que debe transferir el valor de x a través del paso de argumentos en la llamada a la función. Si además esta función desea modificar el valor de dicha variable, entonces tendrá que devolver el nuevo valor a func1, y será func1 quien se encargue de asignar el valor devuelto a su variable x.

3.2 Variable Global

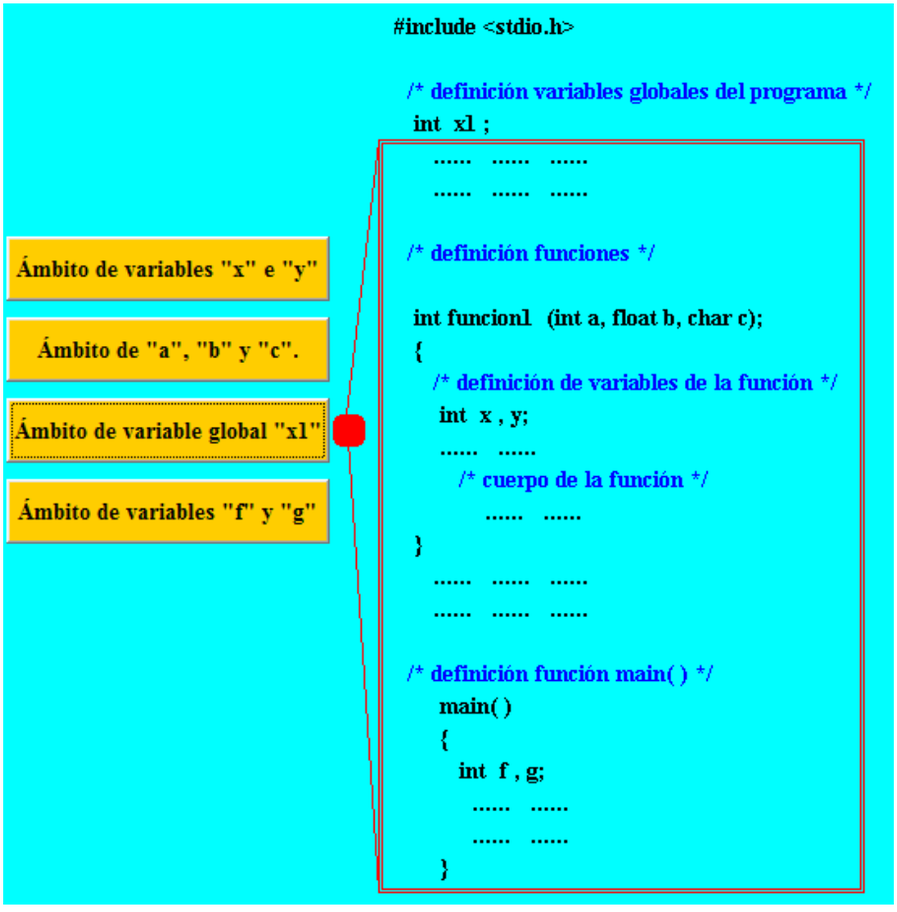
Una variable global es aquella que se define fuera del cuerpo de cualquier función, normalmente al principio del programa, después de la definición de los archivos de biblioteca (#include), de la definición de constantes simbólicas y antes de cualquier función.

El ámbito de una variable global son todas las funciones que componen el programa, cualquier función puede acceder a dichas variables para leer y escribir en ellas. Es decir, se puede hacer referencia a su dirección de memoria en cualquier parde del programa.

Ejemplo: Se muestra la diferencia entre las variables locales y globales, con la consiguiente diferenciación en los ámbitos que abarcan:

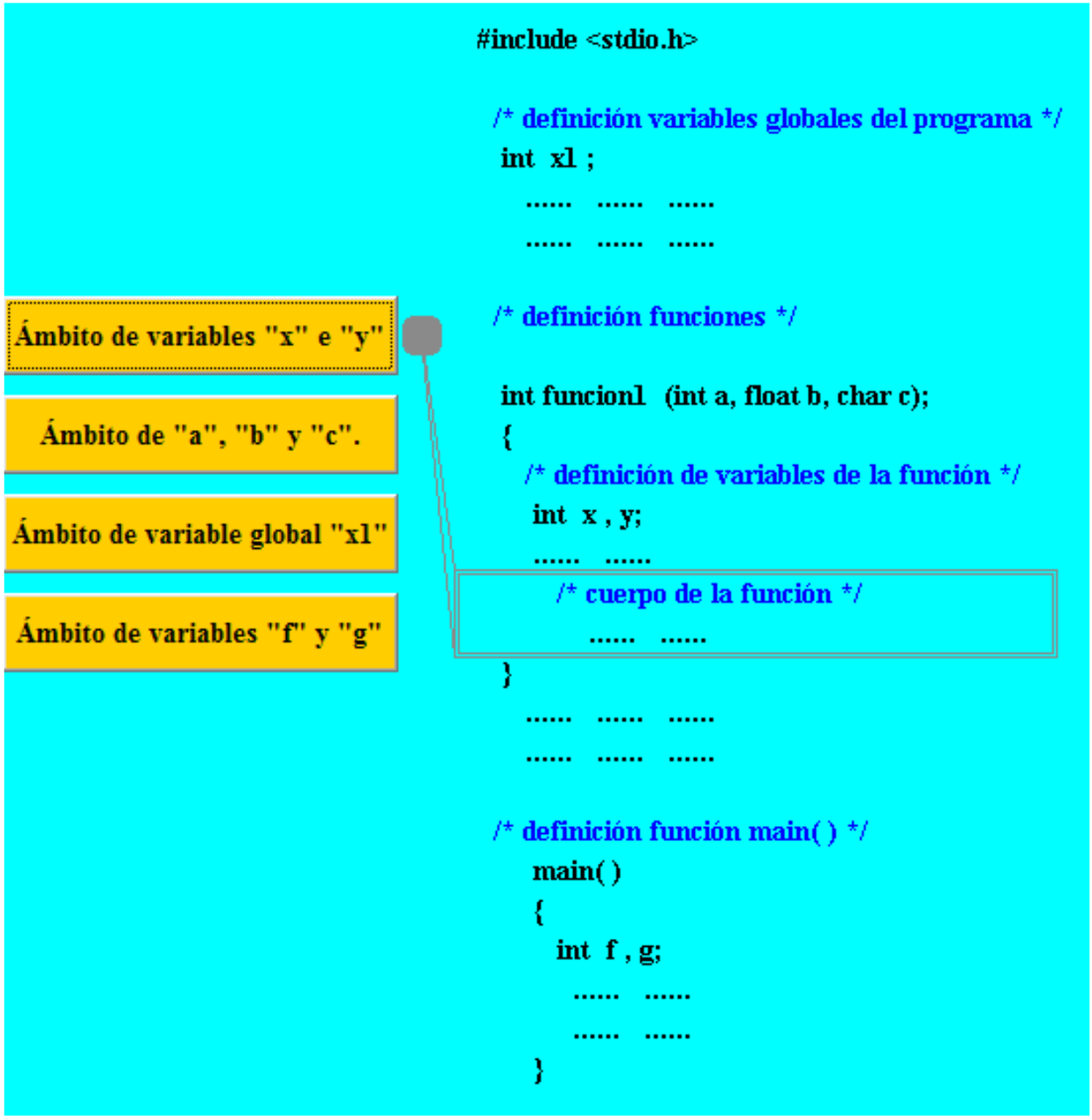
En las siguientes figuras veremos la diferencia del ámbito de acción de las variables locales y globales

Figura N° 49: Ámbito de variables globales X1



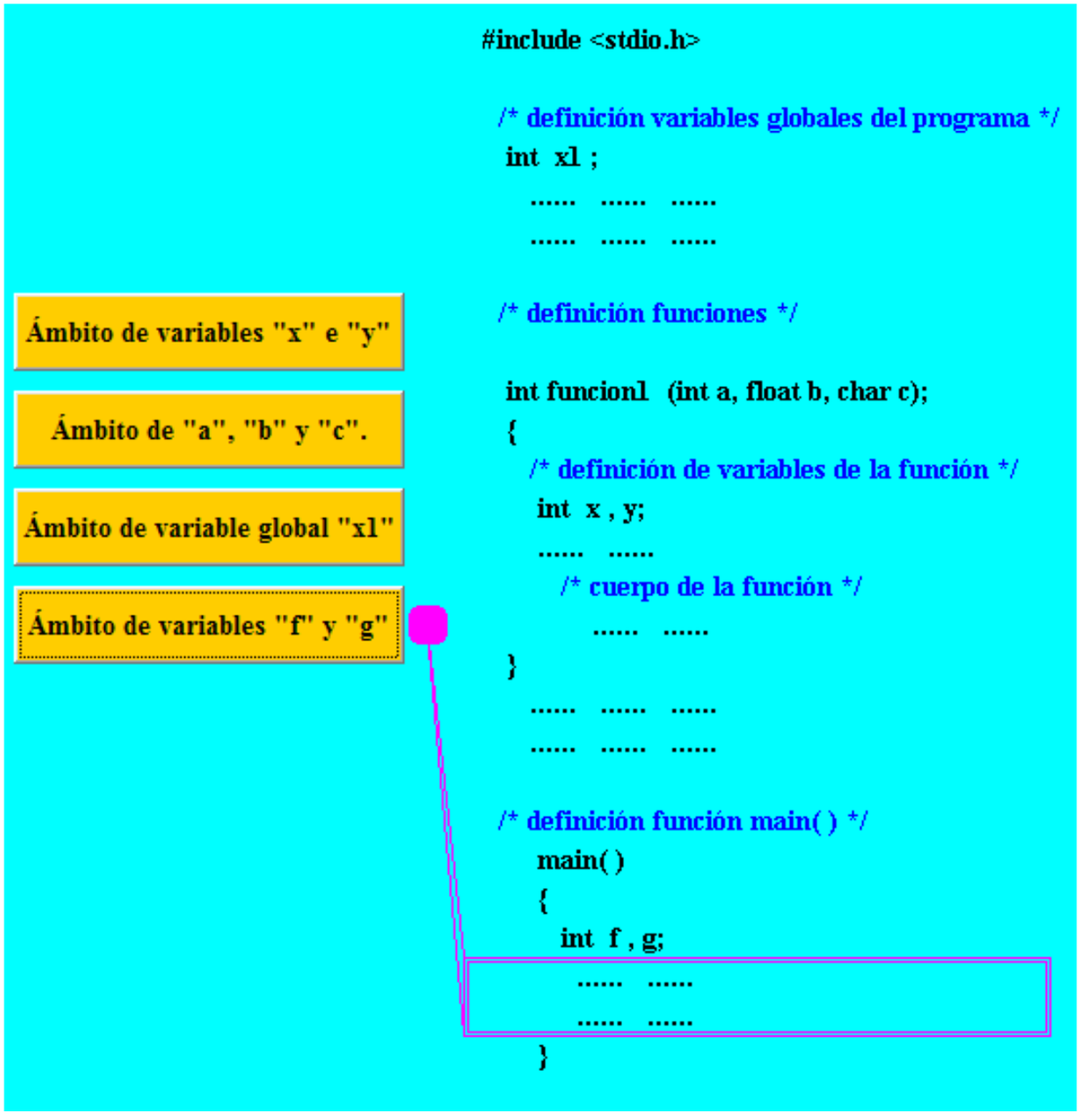
Fuente: http://decsai.ugr.es/

Figura N° 50: Ámbito de variables locales x e y



Fuente: http://decsai.ugr.es/

Figura Nº 51: Ámbito de variables locales f y g



Fuente: http://decsai.ugr.es/

En el lenguaje C++ existen algunos caracteres especiales que se usan frecuentemente. Estos caracteres tienen una representación especial:

Figura N° 52: Caracteres especiales de C++

Código	Significado	Valor ASCII (decimal)	Valor ASCII (hexadecimal)
'\n'	nueva línea	10	0x0A
'\r'	retorno de carro	13	0x0D
'\f'	nueva página (*form feed*)	2	x0C
'\t'	tabulador horizontal	9	0x09
'\b'	retroceso (*backspace*)	8	0x08
'\''	comilla simple	39	0x27
'\"'	comillas	4	0x22
'\\ '	barra invertida	92	0x5C
'\? '	interrogación	63	0x3F
'\nnn'	cualquier carácter (donde nnn es el código ASCII expresado en octal)		
'\xnn'	cualquier carácter (donde nn es el código ASCII expresado en hexadecimal)		

Fuente: http://platea.pntic.mec.es/

Ejemplo 01: Escribir un programa que permita sumar dos números, la solución debe utilizar funciones.

Figura N° 53: Programa con funciones

(Global Scope)

```
#include<iostream>
using namespace std;
int sumar(int a, int b)
{
return a+b;
}
int main()
{
int x,y, sum;
    cout<<"Ingrese el valor de x";
    cin>>x;
    cout<<"Ingrese el valor de y";
    cin>>y;
    sum = sumar(x,y);
    cout<<"La suma es: "<<sum;
    system("pause");
    return 0;
}
```

4 .-TIPO DE FUNCIONES

Los tipos de funciones en C++ son 4, aunque en realidad son las combinaciones de las 2 cosas que una función puede hacer.

Una función, como les decía, puede hacer (o no) dos cosas: Recibir datos y Retornar datos. De esto surgen los cuatro tipos de funciones:

- No reciben ni retornan
- Reciben y no retornan
- No reciben y retornan
- Reciben y retornan

Vamos a hacer un programa que sume dos números, usando los cuatro tipos de funciones:

4.1.-No reciben ni retornan

Las más sencillas. Para usarlas sólo tenemos que saber cómo crearlas y cómo llamarlas. Una función se crea de esta forma general:

tipo nombre(){}

El 'tipo' se refiere al tipo de dato (int, float, void, char) y en las funciones que no retornan siempre es void.

El 'nombre' es el nombre de la función: cualquiera que empiece con una letra, que sea significativo y que no sea una palabra reservada.

Para llamarlas sólo hay que escribir el nombre de la función seguido de sus paréntesis y un punto y coma (;).

Así nuestro programa sería:

```
#include
using namespace std;
void sumar()
{
        int num1, num2, r;
        cout <> num1;
        cout <> num2;
        r = num1 + num2;
        cout << "La suma es " << r;
}

int main()
{
        sumar();
}
```

Como ven, todo lo que habríamos puesto en nuestro main mejor los pusimos en una función y desde el main la llamamos. Una función siempre tiene que ir antes del main.

4.2 Reciben y No Retornan

¿Cómo haríamos para pedir los dos números en el main y que la función haga la suma? Para eso tenemos que hacer una función capaz de recibir datos, entonces la sintaxis cambia un poco:

tipo nombre(tipo_var1 nombre_var1, tipo_var2 nombre_var2){}

Dentro del paréntesis tenemos otros aspectos:

'tipo_var1' se refiere al tipo de la variable que nuestra función va a recibir.

'nombre_var1' se refiere al nombre de esa variable.

Si queremos recibir una variable hasta ahí es suficiente, si queremos otra variable ponemos una coma (,) y declaramos la siguiente variable.

Para llamar la función hay que poner las variables que vamos a enviar dentro del paréntesis en el mismo orden en que las declaramos en la función:

nombre(var1, var2);

Ejemplo:

```
#include
using namespace std;

void sumar(int num1, int num2)
{
        int r;
        r = num1 + num2;
        cout << "La suma es " << r;
}

int main()
{
        int num1, num2;
        cout <> num1;
        cout <> num2;
        sumar(num1, num2);
}
```

Pedimos los dos números en el main, los enviamos a la función, ésta las sumas y los muestra.

Una función de este tipo que hemos usado muchas veces es el odiado por muchos, amados por otros,

4.3 Retornan y No Reciben

¿Y si ahora queremos lo contrario? Pedir los números en la función, pero mostrar el resultado en el main. Para eso necesitamos una función que retorne.

Recibir es enviar datos del main a la función. Retornar es enviar datos de la función al main. Para retornar datos hay que hacer dos cosas: no usar void como tipo y usar return.

El 'tipo' tiene que ser del tipo de variable que queremos retornar, si nuestra variable retorna una variable int, pues el tipo de la función es int.

Para indicar qué variable estamos retornando usaremos la palabra return seguido de la variable. Usualmente esto va al final de la función.

Para llamar a la función hay que preparar un colchón en donde caiga la variable que está retornando.

var = nombre();

La variable que está retornando nuestra función se va a almacenar en la variable 'var'. Este es un buen momento para recordarles que las variables declaradas entre dos llaves {} únicamente existen entre esas dos llaves. O sea que la variable 'var' de la función no es la misma que la variable 'var' de la función; sin embargo la var del main está adquiriendo el valor de la var del main. Un poco confuso lo se, no se preocupen.

Ejemplo:

```
#include
using namespace std;

int sumar()
{
   int num1, num2, r;
   cout <> num1;
   cout <> num2;
   r = num1 + num2;
   return r;
}
int main()
{
   int r;
   r = sumar();
   cout << "La suma es " << r;
}
```

4.4 Reciben y Retornan

Ahora queremos que nuestra función únicamente sume, el main se va a encargar de pedir los números y sumar los resultados. Para eso necesitamos que nuestra función reciba las variables y además retorne el resultado.

Es sólo cuestión de combinar las funciones que reciben y no retornan con las que retornan y no reciben.

Ejemplo:

```
#include
using namespace std;
int sumar(int num1, int num2)
{
   int r;
   r = num1 + num2;
   return r;
}
int main()
{
   int num1, num2, r;
   cout <> num1;
   cout <> num2;
   r = sumar(num1, num2);
   cout << "La suma es " << r;
}
```

LECTURA SELECCIONADA N°. 2:

TIM BERNERS-LEE TEJE LA WEB PARA TODOS

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

Internet ha sido en mucho tiempo un poderoso medio de comunicación y almacenamiento de información valiosa. Pero hasta hace poco, pocas eran las personas que dominaban los códigos crípticos y los lenguajes desafiantes necesarios para desbloquear los tesoros de Internet. La Red quedaba efectivamente fuera de los límites de la mayoría de personas del mundo. Tim Berners-Lee lo cambió todo cuando él solo inventó la World Wide Web y nos la entregó a todos nosotros.

Tim Berners-Lee nació en Londres en 1955. Sus padres se conocieron mientras programaban el Ferranti Mark I, la primera computadora comercial. Animaron a su hijo a que pensara de forma original. Él se enamoró de la electrónica e incluso construyó una computadora a base de piezas de repuesto y un conjunto de TV cuando estudiaba físicas en Oxford.

Berners-Lee tomó un trabajo de ingeniería de software en el CERN, el laboratorio europeo de física de partículas de Ginebra, Suiza. Mientras se encontraba allí, desarrolló un programa que le ayudara a rastrear todas sus notas aleatorias. Intentó que el programa, llamado Enquire, tratara con información de una manera similar a como lo hacía el cerebro. Enquire era un sistema primitivo de hipertexto que permitía que los documentos de su computadora se enlazaran mediante números y no mediante los clics del ratón. (En 1980, las PC no tenían ratones.) Berners-Lee quiso extender el concepto de Enquire para poder enlazar documentos almacenados en otras computadoras con la suya. Su idea era crear un sistema de hipertexto abierto y distribuido sin límites para que los científicos de cualquier lugar pudieran unir sus trabajos.

Durante los siguientes años, él solo construyó un sistema completo para cumplir su sueño. Diseñó el esquema URL para asignar a cada documento de Internet una dirección única. Desarrolló HTML, el lenguaje para codificar y visualizar documentos de hipertexto en la Web. Creó HTTP, el conjunto de reglas que permitía que los documentos de hipertexto se enlazaran a través de Internet. Y construyó el primer navegador software para la visualización de esos documentos desde localizaciones remotas.

En 1991, sometió el primer artículo que describía la Web a una conferencia; los organizadores de la misma lo rechazaron porque la Web les parecía demasiado simple. Pensaban que las ideas de Berners-Lee estaban un paso por detrás en comparación con los sistemas de hipertexto que habían desarrollado Ted Nelson, Doug Engelbart y otros, 25 años antes.

Ahora es fácil ver que la simplicidad de la Web era una fuerza, no una debilidad. En lugar de intentar hacer suyo su conjunto de invenciones, Berners-Lee lo dejó gratuitamente a disposición del público. De repente, la inmensa superficie de Internet estaba abierta a casi cualquiera que pudiera apuntar y hacer clic con un ratón. Otros programadores añadieron capacidades multimedia a la Web, y su popularidad se extendió como un virus. En unos pocos años, Internet se transformó de una fortaleza prohibida de comandos y códigos crípticos en un entorno multimedia que invitaba a las masas.

Cuando creó la Web, Tim Berners-Lee creó un nuevo medio de comunicación. Pocas

Personas en la historia han tenido un impacto tan grande en nuestra manera de comunicarnos. En palabras del escritor Joshua Quittner, los logros de Tim Berners-Lee son «casi Gutenbergianos». Tim Berners-Lee trabaja ahora en una modesta oficina del MIT, donde encabeza el Consorcio de la World Wide Web (W3C). El W3C es una organización de normalización dedicada a ayudar a que la Web evolucione en las direcciones positivas, en lugar de disgregarse en facciones incompatibles. El trabajo de Tim Berners-Lee y el W3C ayudará a garantizar que la World Wide Web continúe perteneciendo a todos.

VIDEOS

Video 25: Alcance de una variable.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Alcance de una Variable.
URL: https://youtu.be/1KdmeUtx5d8?t=34s
Duración: 5 min 15 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

ACTIVIDAD N° 2

Reconoce la importancia del uso de las funciones en la programación a través del desarrollo de un programa,

INSTRUCCIONES:

Lea cuidadosamente cada enunciado y responda según se requiera.

1. Lee y analiza, todos los contenidos de tema N° 2.

   - Como apoyo visualiza el siguiente video: Introducción a Funciones en C++

   https://www.youtube.com/watch?v=ZYCTqYvDEI8

2. Investiga para ampliar la información recibida.

3. Desarrolla un programa que permita realizar las operaciones de suma, resta , multiplicación y división de números enteros, cada operación debe realizarse con funciones.

4. El Main() (Programa principal) Debe invocar a las funciones desarrolladas en le ítem anterior.

PRUEBA DE DESARROLLO N° 2

Escribir un programa en lenguaje C++ que desarrolle lo siguiente:

INSTRUCCIONES:

1. El programa debe permitir ingresar 25 notas (las notas deben ser validadas con el ingreso solo de notas de cero a veinte, cualquier otro valor no debe considerarse).
2. Se debe utilizar funciones en la solución del problema.
3. El programa debe reportar la cantidad de notas aprobadas.
4. El programa debe reportar la cantidad de notas desaprobadas.
5. El programa debe reportar la cantidad de notas iguales a 20.
6. El programa debe reportar la cantidad de notas iguales a 05.

GLOSARIO DE LA UNIDAD III

F

FUNCION
Proceso que devuelve un valor previamente calculado.

P

PARAMETRO
Dato o factor que se toma como necesario para analizar o valorar una situación.

PROGRAMA
Operaciones que en un orden determinado ejecutan algunas máquinas.

V

VARIABLE
Magnitud que puede tomar un valor cualquiera.

BIBLIOGRAFÍA DE LA UNIDAD III

- Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). *Introducción a la Computación.* México: Pearson Educación.
- Joyanes, Luis. (2008). Fundamentos de Programación. Madrid: Mc Graw Hill.
- Joyanes, Luis. (2007). Estructura de datos en C++. Madrid: Mc Graw Hill.
- Lecca Eduardo. (2000) El poder de turbo C++. Perú: Mundigraf.

AUTOEVALUACIÓN Nº 3

INSTRUCCIONES:

Lea cuidadosamente cada enunciado y responda según se requiera (Remarque/escriba con color azul su respuesta).

1. Es un lenguaje de programación que se utiliza para crear y desarrollar los programas.

   a. Lenguaje C++

   b. Lenguaje A++

   c. Sql server

   d. Windows.

   e. Microsoft.

2. La _____ siempre devuelve un valor como resultado de uno o varias operaciones.

   a. Función

   b. Procedimiento

   c. Vector

   d. Matriz.

   e. Tipo de dato.

3. La _____ almacena un valor que puede ser asignado durante el desarrollo del programa.

   a. Variable

   b. Procedimiento

   c. Función

   d. Matriz.

   e. Tipo de dato.

4. Es la instancia de una clase previamente definida.

   a. Objeto

   b. Variable

   Función

d. Clase.

e. Tipo de dato.

5. Es el molde a partir del cual se crean los objetos.

a. Clase

b. Objeto

c. Función

d. variable

e. Tipo de dato.

UNIDAD IV

ARREGLOS

DIAGRAMA DE PRESENTACIÓN DE LA UNIDAD IV

CONTENIDOS	ACTIVIDADES FORMATIVAS (HABILIDADES Y ACTITUDES)	SISTEMA DE EVALUACIÓN (TÉCNICAS Y CRITERIOS)
TEMA Nº 1: Arreglos Unidimensionales. 1 Arreglos unidimensionales o vectores. 2 Operaciones con arreglos unidimensionales **TEMA Nº 2: Arreglos Bidimensionales.** 1 Arreglos bidimensionales o matrices 2 Operaciones con arreglos bidimensionales	• Implementa y organiza la información en datos estructurados **usando arreglos unidimensionales a través del desarrollo de un programa con vectores.** • Implementa y organiza la información en datos estructurados **usando arreglos bidimensionales a través del desarrollo de un programa con matrices.**	**Procedimientos e indicadores de evaluación permanente** • Entrega puntual de trabajos realizados. • Calidad, coherencia y pertinencia de contenidos desarrollados. • Prueba individual. • Actividades desarrolladas en sesiones tutorizadas **Criterios para evaluar programas con vectores y programas con Matrices:** • N° de Elementos del vector • Calculo de valores • Características de la matiz • Diagonal principal de la matriz • Correcta estructura de datos • Ejecución correcta del programa. • Reportes correctos del programa

RECURSOS:

Vídeos o imágenes:

Tema Nº 1 :

Arreglos Unidimensionales en C++

https://www.youtube.com/watch?v=QjR6UwMPyPw

Tema Nº 2

Arreglos bidimensionales en C++

https://www.youtube.com/watch?v=5LoigeLCeoo

Lectura complementaria:

Lectura Seleccionada Nº 1

Alan Kay inventa el futuro

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

Lectura Seleccionada Nº 2

Alan Turing, inteligencia militar y máquinas inteligentes

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

INSTRUMENTO DE EVALUACIÓN	• Prueba Objetiva. • Prueba de Desarrollo
BIBLIOGRAFÍA (BÁSICA Y COMPLEMENTARIA)	**BÁSICA** JOYANES AGUILAR, Luis. Fundamentos de programación. Algoritmos, estructuras de datos y objetos, Tercera Edición. Madrid: Editorial McGraw-Hill, 2003. **COMPLEMENTARIA** PRIETO, A., LLORIS, A. y TORRES, Introducción a la Informática, Tercera Edición, Madrid: Editorial McGraw-Hill, 2005.
RECURSOS EDUCATIVOS DIGITALES	LÓPEZ GARCÍA, Juan Carlos, "Algoritmos y Programación", Año 2009 [ref. de 09 de noviembre de 2009].Disponible en Web:<http://www.eduteka.org/GuiaAlgoritmos.php>

TEMA N° 1: ARREGLOS UNIDIMENSIONALES..

Los arreglos (arrays) permiten almacenar vectores y matrices. Los arreglos unidimensionales sirven para manejar vectores y los arreglos bidimensionales para matrices. Sin embargo, las matrices también pueden almacenar mediante arreglos unidimensionales y por medio de apuntadores a apuntadores.

La palabra unidimensional no indica que se trata de vectores en espacios de dimensión; indica que su manejo se hace mediante un subíndice.

1.-ARREGLOS UNIDIMENSIONALES O VECTORES

Un arreglo unidimensional es un tipo de datos estructurado que está formado por una colección finita y ordenada de datos del mismo tipo. Es la estructura natural para modelar listas de elementos iguales. Los datos que se guarden en los arreglos todos deben ser del mismo tipo.

El tipo de acceso a los arreglos unidimensionales es el acceso directo, es decir, podemos acceder a cualquier elemento del arreglo sin tener que consultar a elementos anteriores o posteriores, esto mediante el uso de un índice para cada elemento del arreglo que nos da su posición relativa.

Para implementar arreglos unidimensionales se debe reservar espacio en memoria.

Los arreglos nos permiten hacer un conjunto de operaciones para manipular los datos guardados en ellos, estas operaciones son: ordenar, buscar, insertar, eliminar, modificar entre otras.

Los arreglos son una colección de variables del mismo tipo que se referencian utilizando un nombre común. Un arreglo consta de posiciones de memoria contigua. La dirección más baja corresponde al primer elemento y la más alta al último. Un arreglo puede tener una o varias dimensiones. Para acceder a un elemento en particular de un arreglo se usa un índice.

El formato para declarar un arreglo unidimensional es:

tipo nombre_arr [tamaño]

Por ejemplo, para declarar un arreglo de enteros llamado a con diez elementos se hace de la siguiente forma:

int A[10];

En el lenguaje C++, todos los arreglos usan cero como índice para el primer elemento. Por tanto, el ejemplo anterior declara un arreglo de enteros con diez elementos desde A[0] hasta A[9]

La forma como pueden ser accesados los elementos de un arreglo, es de la siguiente forma:

A[2] = 15;

Asigna 15 al 3er elemento del arreglo a

num = A[2];

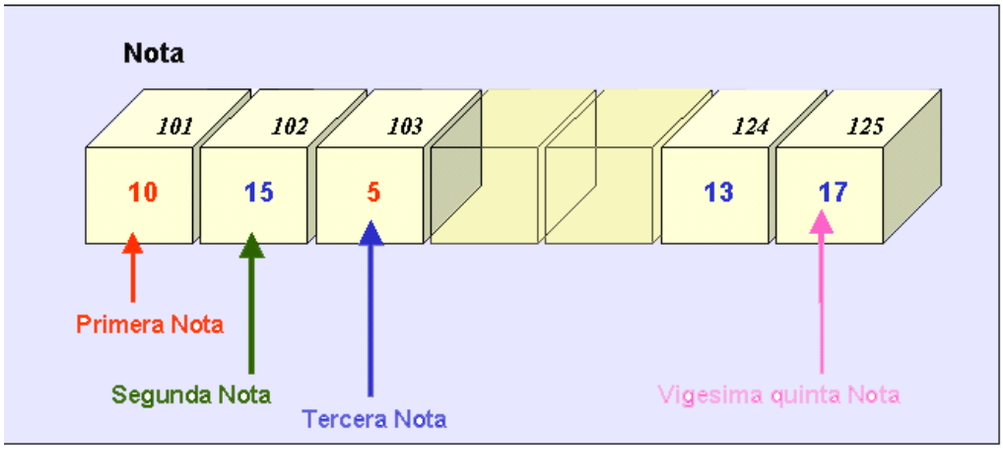
Asigna el contenido del 3er elemento a la variable num.

El lenguaje C++ no realiza comprobación de contornos en los arreglos. En el caso de que sobrepase el final du-

rante una operación de asignación, entonces se asignarán valores a otra variable o a un trozo del código, esto es, si se dimensiona un arreglo de tamaño N, se puede referenciar el arreglo por encima de N sin provocar ningún mensaje de error en tiempo de compilación o ejecución, incluso aunque probablemente se provoque un error en el programa.

Se debe ser responsable de asegurar que todos los arreglos sean lo suficientemente grandes para guardar lo que pondrá en ellos el programa.

Figura N° 54: Arreglo Unidimensional



Fuente: Luis Joyanes "Fundamentos de Programación"

Ejemplo 01: Escribir un programa que permita ingresar 5 notas a un vector y luego debe reportar estas notas.

Figura N° 55: Programa con Vectores Nro 01

```
(Global Scope)
#include <iostream>
using namespace std;
int main()
{

int notas[5];
int i;
/* Ingresar notas al vector */
for (i=0;i<5;i++)
{
   cin>>notas[i];
}
/* Mostrar las notas del vector */
for (i=0;i<5;i++)
{
cout<<notas[i]<<endl;
}
system("pause");
return 0;
}
```

Fuente: Calderón Sedano Carlos Alberto

Ejemplo 02: Escribir un programa que permita carga el arreglo llamado vector con los cuadrados de los números del 1 al 10 y luego los visualiza.

Figura N° 56: Programa con vectores Nro. 2

```
(Global Scope)
#include <iostream>
using namespace std;
int main()
{

int vector[10];
int i;

for (i=1;i<11;i++)
{
   vector[i-1]=i*i;
}

for (i=0;i<10;i++)
{
cout<<vector[i]<<endl;
}
system("pause");
return 0;
}
```

Fuente: Calderón Sedano Carlos Alberto

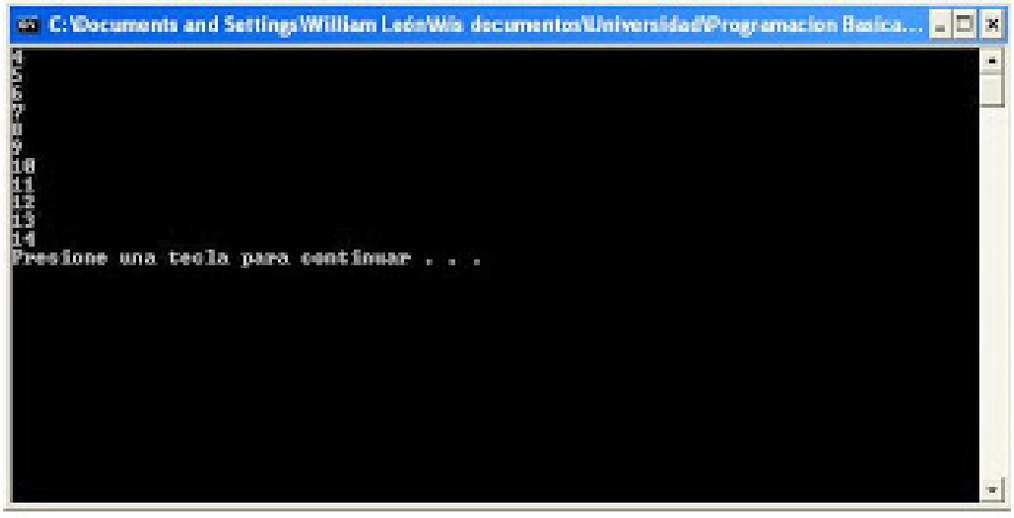
Ejemplo 03: Escribir un programa que llene un arreglo con los números enteros comprendidos entre 4 y 14.

Figura N° 57: Programa con vectores Nro. 3

```
#include <iostream>
#include <stdlib.h>

using namespace std;

int main(int argc, char *argv[])
{
  int a[11];
  int n=4;
  for(int i=0;i<11;i++)
  {
    a[i]=n++;
      cout<<a[i]<<endl;

  }

  system("PAUSE");
  return 0;
}
```

Fuente: http://ejerpbas.blogspot.com/

Figura N° 58: Ejecución del Programa con vectores Nro. 3



Fuente: http://ejerpbas.blogspot.com/

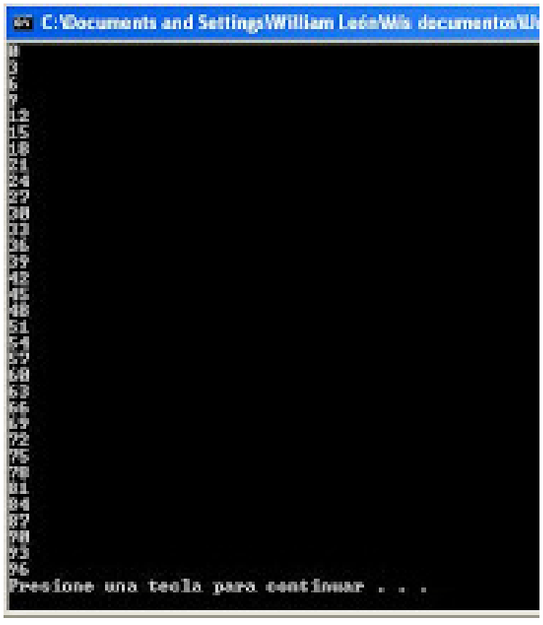
Ejemplo 04 Escribir un programa que llene un arreglo con los números comprendidos entre 0 y 100 divisibles por 3

Figura N° 59: Programa con vectores Nro. 4

```
#include <iostream>
#include <stdlib.h>

using namespace std;

int main(int argc, char *argv[])
{
  int x=0,y=0;
  int m[33];
do{
if(x%3==0 && x!=100){
m[y++]=x;
}
}while(x++<100);
for(y=0;y<33;y++){
cout << m[y] << endl;
}
  system("PAUSE");
  return 0;
}
```

Fuente: http://ejerpbas.blogspot.com/

Figura N° 60: Ejecución Programa con vectores Nro. 4



Fuente: http://ejerpbas.blogspot.com/

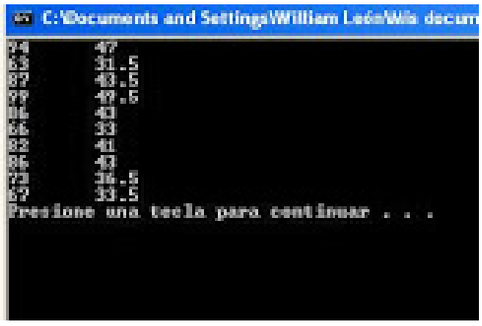
Ejemplo 05: Escribir un programa que llene un arreglo de 10 números enteros aleatorios comprendidos entre 50 y 100, copie en otro arreglo esos números multiplicados por 0,5 y muestre ambos arreglos.

Figura N° 61: Programa con vectores Nro. 5

```
#include <iostream>
#include <ctime>

using namespace std;

int main()
{
    int o[10];
    float c[10];
    srand(time(0));

    for(int i=0;i<10;i++)
        {
            o[i]=50+(rand()%51);
        }

for(int i=0;i<10;i++)
    {
        c[i]=o[i]*0.5;
    }

for(int i=0;i<10;i++){
    cout << o[i] << "\t" << c[i] << endl;
    }
system("pause");
return 0;
}
```

Fuente: http://ejerpbas.blogspot.com/

Figura N° 62: Ejecución del Programa con vectores Nro. 5



Fuente: http://ejerpbas.blogspot.com/

2.-OPERACIONES CON ARREGLOS UNIDIMENSIONALES

Las operaciones válidas en arreglos son las siguientes:

Lectura / Escritura.

Asignación.

Actualización.

Inserción.

Eliminación.

Modificación.

Ordenación.

Búsqueda.

Veamos la descripción de algunos de ellos:

Lectura.

El proceso de lectura de un arreglo consiste en leer un valor en cada uno de sus componentes. Supongamos que se desea leer todos los elementos de un arreglo unidimensional Nombres[200] en forma consecutiva, utilizaríamos un ciclo repetitivo de la siguiente manera:

```
String  Nombres[200]
Hacer Desde C=0 Hasta 199
        Leer Nombres[C]
Fin Desde
```

Escritura.

El caso de la operación de escritura es similar al de lectura. Solo que en vez de leer el componente del arreglo lo escribimos. Supongamos que se desea escribir los componentes del arreglo unidimensional Nombres[200] en forma consecutiva. Los pasos a seguir son:

```
String Nombres[200]
Hacer Desde C=1 Hasta 199
        Nombres[C] ← C
Fin Desde
```

Al variar el valor de C se escribe el elemento del arreglo Nombres[200], correspondiente a la posición indicada por C, en otras palabras, cuando C=1 se almacenara el valor de C en la posición del arreglo Nombres[1], y así consecutivamente para todo el arreglo.

Asignación.

Antes de utilizar un arreglo es necesario inicializarlo, para inicializar todos los elementos de una vez, se colocan dentro de una estructura for que va del primer elemento al último que contiene el arreglo. Para asignar un valor a un elemento del arreglo se hace por ejemplo:

Calificaciones[0] ← 100

Cuando se usan arreglos, una operación común es usar una variable índice para acceder a los elementos de un arreglo. Suponiendo que la variable índice I contiene el valor 3, la siguiente instrucción asigna el valor 400 avalores[3]:

Valores[I] ← 400

No es posible asignar directamente un valor a todo el arreglo; sino que se debe asignar el valor deseado en cada componente. Con una estructura repetitiva se puede asignar un valor a todos los elementos del arreglo.

Por ejemplo:

```
Int Numeros[500]
Hacer Desde I= 0 Hasta 499
        Numeros[I] ← 3
Fin Desde
```

Actualización

La actualización es una operación que se realiza frecuentemente en los arreglos. La cantidad de actualizaciones es directamente proporcional al problema que se intenta resolver. A diferencia de las otras operaciones estudiadas, la actualización lleva implícita otras operaciones como inserción y eliminación.

Con el propósito de realizar una actualización de manera eficiente, es importante conocer si el arreglo está o no ordenado, si sus componentes respectan algún orden (ascendiente o decreciente). Cabe destacar que las operaciones de inserción, eliminación y modificación serán tratadas de forma separada para arreglos ordenados y desordenados

Finalmente, es importante señalar que la operación de búsqueda se utiliza como auxiliar en las operaciones de inserción, eliminación y modificación. Por esta razón se presenta la búsqueda secuencial en a. desordenados.

Inserción

Al insertar un elemento X en un arreglo unidimensional ordenado se debe tener en cuenta lo siguiente:

Verificar que exista espacio

Encontrar la posición que ocupará el nuevo elemento

Cuando se detecte la posición, recorrer todos los elementos

Se asignará el valor de X a la posición encontrada

Cuando el valor a insertar es mayor que el último elemento del arreglo, no habrá desplazamiento.

Generalmente, se verifica si el elemento a insertar no existe en el arreglo. De lo contrario no se lleva realiza la inserción ya que no interesa tener información repetida.

Antes de presentar el algoritmo de inserción en arreglos ordenados, veremos la función de búsqueda auxiliar, para arreglos ordenados, que se utilizará en el proceso de inserción y eliminación.

Eliminación

Para eliminar un elemento X de un arreglo unidimensional ordenado V se debe;

Buscar la posición al elemento a eliminar. Si el resultado de la función es un valor positivo, significa que el elemento se encuentra en V y por lo tanto se puede eliminar. De lo contrario no se realiza ninguna operación.

Modificación

Consiste en reemplazar un componente del arreglo con otro valor y debemos hacer:

Se busca el elemento en el arreglo

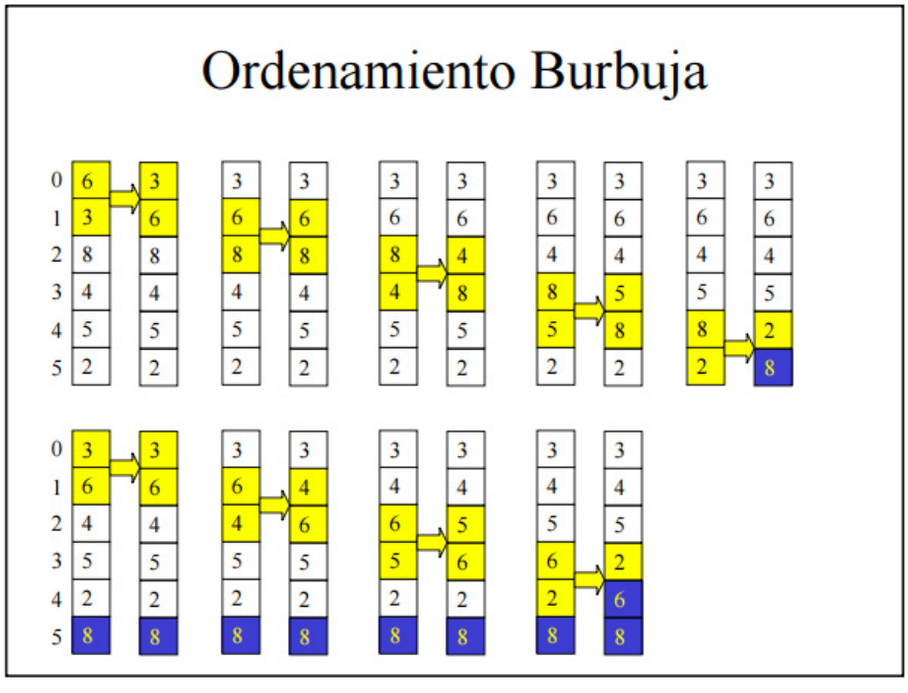
Si se encuentra verificar que al modificar no se altere el orden

Si se altera entonces es necesario eliminar el elemento a modificar y luego insertar el nuevo elemento en la posición correspondiente.

Ordenación

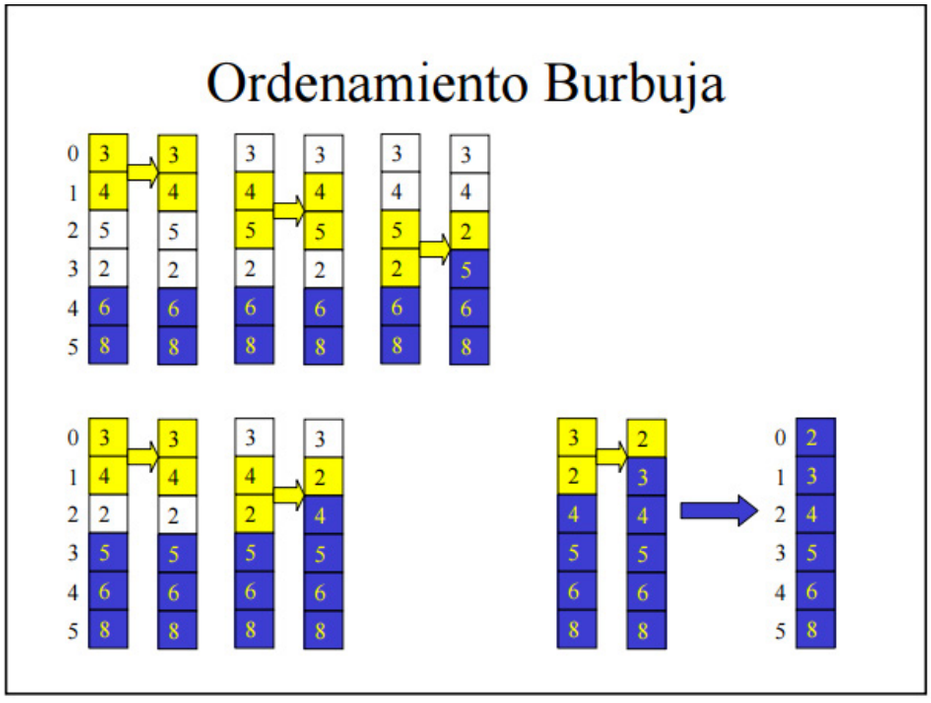
Ordenamiento Burbuja (bublesort): Se compara elementos adyacentes y empujamos los valores más livianos hacia arriba (los más pesados van quedando abajo). Idea de la burbuja que asciende, por lo liviana que es.

Figura N° 63: Ordenamiento Método Burbuja Nro. 1



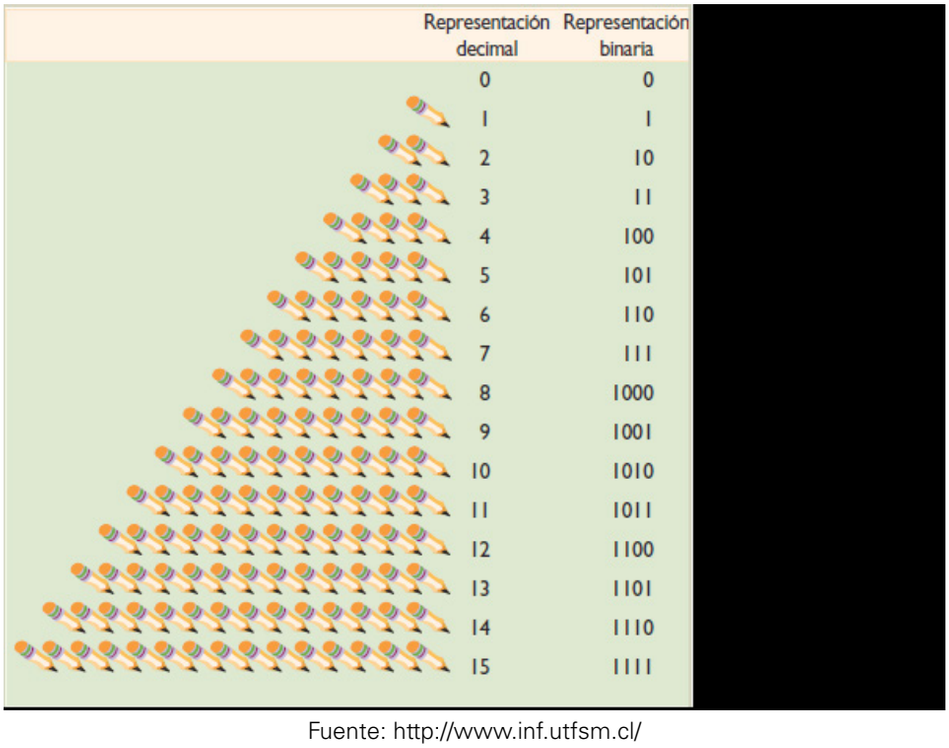
Fuente: http://www.inf.utfsm.cl/

Figura N° 64: Ordenamiento Método Burbuja Nro. 2



Fuente: http://www.inf.utfsm.cl/

Figura N° 65: Algoritmo Método Burbuja



Fuente: http://www.inf.utfsm.cl/

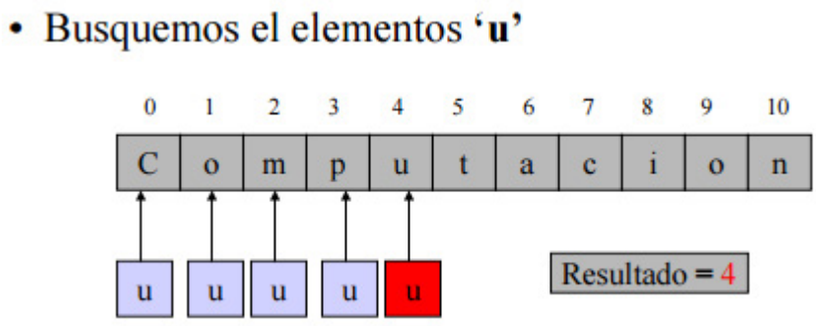
Búsqueda:

Una búsqueda es el proceso mediante el cual podemos localizar un elemento con un valor específico dentro de un conjunto de datos. Terminamos con éxito la búsqueda cuando el elemento es encontrado.

Búsqueda secuencial: A este método también se le conoce como búsqueda lineal y consiste en empezar al inicio del conjunto de elementos, e ir a través de ellos hasta encontrar el elemento indicado ó hasta llegar al final de arreglo. Este es el método de búsqueda más lento, pero si nuestro arreglo se encuentra completamente desordenado es el único que nos podrá ayudar a encontrar el dato que buscamos.

Consiste en ir comparando el elemento que se busca con cada elemento del arreglo hasta cuándo se encuentra.

Figura N° 66: Búsqueda secuencial



Fuente: http://www.inf.utfsm.cl/

Búsqueda del menor

```
menor = a[0];
for (i=1;i<n;i++)
if ( a[i]<menor )
menor=a[i];
```

Búsqueda del mayor

```
mayor= a[n-1];
for (i=0;i<n-1;i++)
if ( a[i]>mayor )
mayor=a[i];
```

Búsqueda de elemento

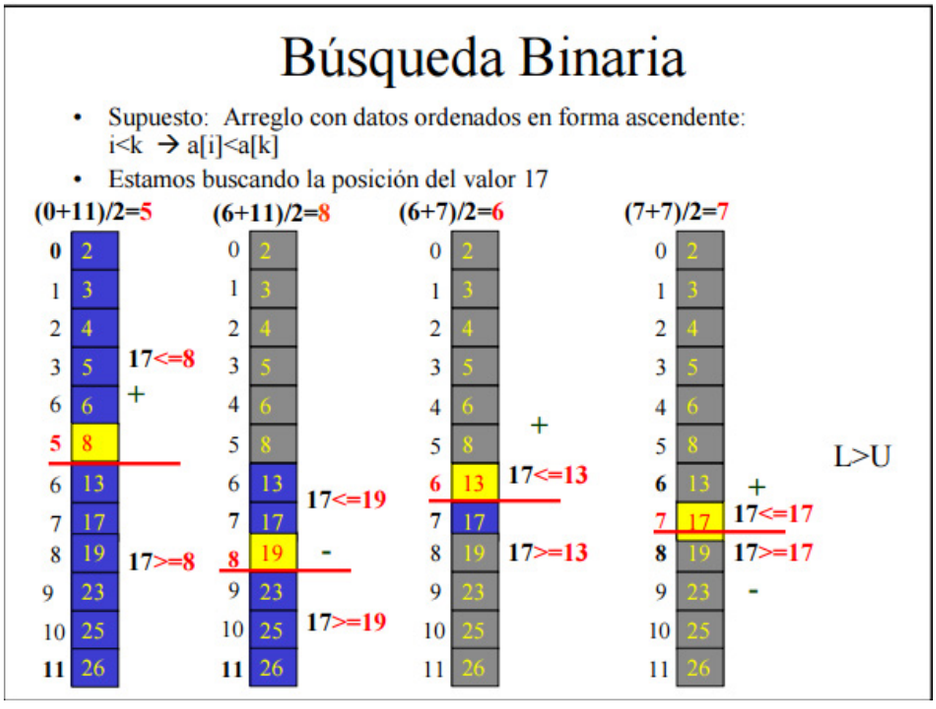
```
encontrado=-1;
for (i=0;i<n;i++)
if ( a[i]==elemento_buscado )
encontrado=i;
```

Búsqueda binaria: Las condiciones que debe cumplir el arreglo para poder usar búsqueda binaria son que el arreglo este ordenado y que se conozca el numero de elementos. Este método consiste en lo siguiente: comparar el elemento buscado con el elemento situado en la mitad del arreglo, si tenemos suerte y los dos valores coinciden, en ese momento la búsqueda termina. Pero como existe un alto porcentaje de que esto no ocurra, repetiremos los pasos anteriores en la mitad inferior del arreglo si el elemento que buscamos resulto menor que

el de la mitad del arreglo, o en la mitad superior si el elemento buscado fue mayor. La búsqueda termina cuando encontramos el elemento o cuando el tamaño del arreglo a examinar sea cero.

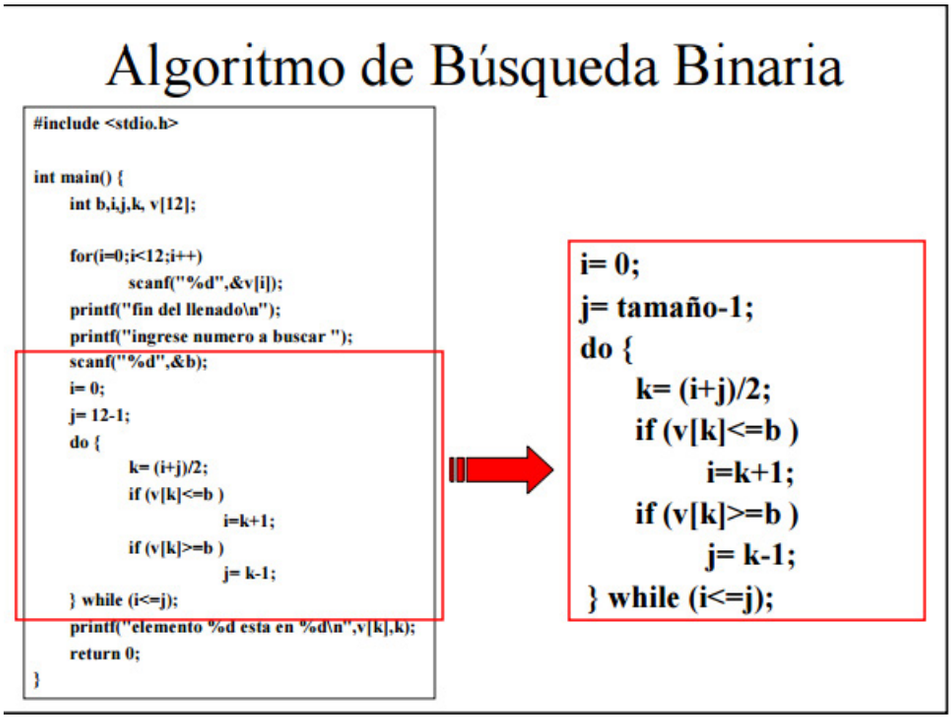
Una búsqueda más eficiente puede hacerse sobre un arreglo ordenado. Una de éstas es la Búsqueda Binaria. La Búsqueda Binaria, compara si el valor buscado está en la mitad superior o inferior. En la que esté, subdivido nuevamente, y así sucesivamente hasta encontrar el valor.

Figura N° 67: Búsqueda Binaria



Fuente: http://www.inf.utfsm.cl/

Figura N° 68: Algoritmo búsqueda binaria



Fuente: http://www.inf.utfsm.cl/

Búsqueda por hash: La idea principal de este método consiste en aplicar una función que traduce el valor del elemento buscado en un rango de direcciones relativas. Una desventaja importante de este método es que puede ocasionar colisiones.

VENTAJAS

Se pueden usar los valores naturales de la llave, puesto que se traducen internamente a direcciones fáciles de localizar.

Se logra independencia lógica y física, debido a que los valores de las llaves son independientes del espacio de direcciones.

No se requiere almacenamiento adicional para los índices.

DESVENTAJAS

No pueden usarse registros de longitud variable.

El archivo no esta clasificado

No permite llaves repetidas

Solo permite acceso por una sola llave

LECTURA SELECCIONADA N° 1:

ALAN KAY INVENTA EL FUTURO

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

Alan Kay ha estado inventando el futuro la mayor parte de su vida. Kay fue un niño prodigio que compuso música, construyó un clavicordio y apareció en la NBC. El genio de Kay no se reflejó en sus cursos, y tuvo problemas debido a la rígida estructura de las escuelas a las que asistió. Después del instituto, trabajó como guitarrista de jazz y como programador de las Fuerzas Aéreas.

Su proyecto Ph.D. fue uno de los primeros sobre microcomputadoras, uno de los varios que Kay había estado desarrollando. En 1968, Kay estaba entre la audiencia cuando Douglas Engelbart dejó sin habla a la comunidad científica con una futurística demostración de informática interactiva. Inspirado por esta demostración, Kay formó un equipo de investigadores en Xerox PARC (Centro de investigación de Palo Alto en California) para construir la computadora del futuro.

Trabajando en la trastienda de una tienda de computadoras llamada Alto, Kay desarrolló una pantalla gráfica con iconos y ventanas que se solapaban (el tipo de pantalla que se convirtió en un estándar dos décadas más tarde). También abanderó la idea de una interfaz de usuario más amigable. Para comprobar este punto, Kay solía llevar a su hijo al laboratorio. Con su ayuda, Kay desarrolló el primer programa para pintar y Smalltalk, el antepasado de los lenguajes de programación orientados a objetos.

En esencia, el equipo de Kay desarrolló la primera computadora personal, una máquina de escritorio monousuario diseñada para uso interactivo. Pero Kay, que acuñó el término de «computadora personal», no veía a Alto como una de ellas. En su mente, una computadora personal debería acompañar a su dueño a cualquier parte, sirviéndole como calculadora, calendario, procesador de textos, máquina gráfica, dispositivo de comunicación y herramienta de referencia. Su visión de lo que él denominó Dynabook es ahora, tres décadas después, lo que podemos ver en nuestras computadoras de bolsillo.

Xerox falló al intentar convertir Alto en un éxito comercial. Pero cuando Steve Jobs de Apple visitó el PARC, quedó impresionado por lo que vio. Bajo su supervisión, un equipo de ingenieros y programadores llevaron a la práctica las ideas de Xerox, junto con algunas propias, y desarrollaron la Macintosh, la primera computadora económica en incorporar muchas de las innovadoras ideas de Kay. Kay se convirtió en desarrollador de Apple, donde definió la Macintosh como «la primera computadora personal lo bastante buena como para criticar». En la actualidad, casi todas las PC tienen interfaces de usuario basadas en los primeros estudio de Kay.

Tras 12 años en Apple, Kay entró como investigador en Disney, donde desarrolló Squeak, una herramienta de programación gráfica para niños. Kay describe su MO (modus operandi) como «empezar a trabajar con los usuarios finales, normalmente niños, e intentar pensar en tipos de experiencias que puedan ayudarles a crecer de diversas formas». En 2002, Kay se unió al grupo de investigadores de Hewlett Packard, mientras continúa con su trabajo con el Viewpoints Research Institute, una organización sin ánimo de lucro que trabaja para mejorar la educación en general y para la comprensión de sistemas complejos.

Kay continúa su cruzada para los usuarios, especialmente para los pequeños. Dice que, al igual que ocurre con el lápiz y el papel, «algo no es válido si un niño no puede usarlo». En un reciente proyecto de investigación en el que colaboró, tanto él como investigadores del MIT trabajaron con escolares para diseñar formas de vida artificial en ambientes artificiales creados dentro de la computadora. Como muchos de sus proyectos, Vivarium tiene muy poca relación con el mercado informático actual. Este tipo de estudio no siempre produce beneficios. Pero para Alan Kay, es la forma de inventar el futuro.

VIDEOS

Video 26: Crear arreglos unidimensionales.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Crear arreglo.
URL: https://youtu.be/iRfrRcx5ziE?t=38s
Duración: 8 min 16 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

ACTIVIDAD N° 1

Implementa y organiza la información en datos estructurados usando arreglos unidimensionales a través de un programa con vectores.

INSTRUCCIONES:

1. Lee y analiza, todos los contenidos de tema N° 2.
2. Como apoyo para desarrollar la actividad visualiza el siguiente video: Arreglos Unidimensionales en C++:

- https://www.youtube.com/watch?v=QjR6UwMPyPw

3. Elabora un programa en el lenguaje de programación C++ con las siguientes especificaciones:

   a. La cantidad de elementos del vector debe ser 50.

   b. Los valores ingresados pueden ser números decimales

   c. Se debe calcular el promedio de los 25 primeros valores.

   d. Se debe calcular la suma de los valores de las posiciones pares.

   e. Se debe calcular la multiplicación de los 10 últimos valores.

TEMA N° 2: ARREGLOS BIDIMENSIONALES

Los arreglos bidimensionales son tablas de valores. Cada elemento de un arreglo bidimensional está simultáneamente en una fila y en una columna.

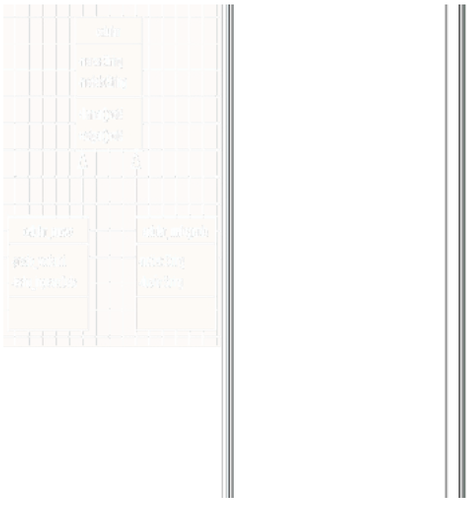
En matemáticas, a los arreglos bidimensionales se les llama matrices, y son muy utilizados en problemas de Ingeniería.

En un arreglo bidimensional, cada elemento tiene una posición que se identifica mediante dos índices: el de su fila y el de su columna.

1 ARREGLOS BIDIMENSIONALES (MATRICES)

Es una estructura de datos estática y de un mismo tipo de datos, y de longitud fija que almacena datos de forma matricial. De igual forma que los arreglos unidimensionales, el almacenamiento de los datos en la memoria se realiza de forma secuencial y son accedidos mediante índices. Los arreglos bidimensionales son también conocidos como matrices. Por lo tanto se llama matriz de orden "m×n" a un conjunto rectangular de elementos dispuestos en filas "m" y en columnas "n", siendo m y n números naturales. Las matrices se denotan con letras mayúsculas: A, B, C, ... y los elementos de las mismas con letras minúsculas y subíndices que indican el lugar ocupado: a, b, c, ... Un elemento genérico que ocupe la fila i y la columna j se escribe i,j. Si el elemento genérico aparece entre paréntesis también representa a toda la matriz: A (i,j).

Figura Nº 69: Arreglo Bidimensional



Fuente http://christophermontenegro-aci220-2.blogspot.com/

Los arreglos bidimensionales son aquellos que tienen dos dimensiones y, en consecuencia se manejan con dos índices, se puede ver también como un arreglo de arreglos. Un arreglo bidimensional equivale a una tabla con múltiples filas y múltiples columnas.

Figura Nº 70: Arreglo Bidimensional con índices

columnas

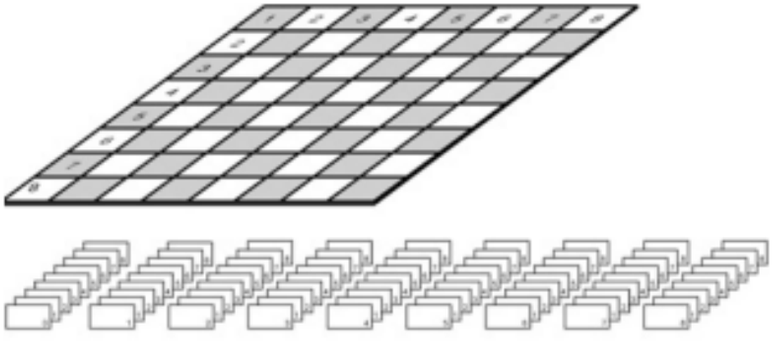
filas	0	1	2	3	4
0	123	25	56	3	45
1	32	44	56	45	67
2	23	100	56	48	67

Arreglo bidimensional de 3 filas y 5 columnas

Fuente: http://www.cs.buap.mx/

Cada dimensión está representada por un subíndice en la matriz. Por tanto, una matriz bidimensional tiene dos subíndices; una matriz tridimensional tiene tres subíndices; y así sucesivamente. Una matriz puede tener cualquier número de dimensiones, aunque las matrices más utilizadas son las de dos dimensiones. Un buen ejemplo de matriz es un tablero de ajedrez. Una dimensión representa las ocho filas; la otra dimensión representa las ocho columnas.

Figura Nº 71: Tablero de ajedrez



Fuente: http://www.tel.uva.es/

La declaración de una matriz en lenguaje C++ que represente un tablero de ajedrez podría ser:

```
int tablero[8][8];
```

1.1 Declaración y Acceso de Matrices

Declaración: La sintaxis para declarar una matriz es:

Tipo <Nombres>[Dimensión_fila] [Dimensión_columna]

Por ejemplo:

```
int Matriz [15][15]
```

Acceso:

Inserción: Para insertar valores en una matriz podemos utilizar el siguiente código

Por ejemplo:

M [3] [2] ←- 9

Extracción: Para extraer datos el código será:

Por ejemplo:

X ←M [3] [2]

Ejemplo 01 Escribir un programa que permita ingreso de una matriz de orden 3x3 y luego muestre sus datos.

Figura N° 71: Programa en C++ Matrices Nro. 01

```
Source.cpp*
(Global Scope)
#include <iostream>
using namespace std;
int main()
{
int matriz[3][3];
int i,j;
/* Ingresar valores a la matriz */
for (i=0;i<3;i++)
   for (j=0;j<3;j++)
       cin>>matriz[i][j];

 /* Mostrar los valores */
for (i=0;i<3;i++)
   for (j=0;j<3;j++)
       cout<<matriz[i][j];

system("pause");
return 0;
}
```

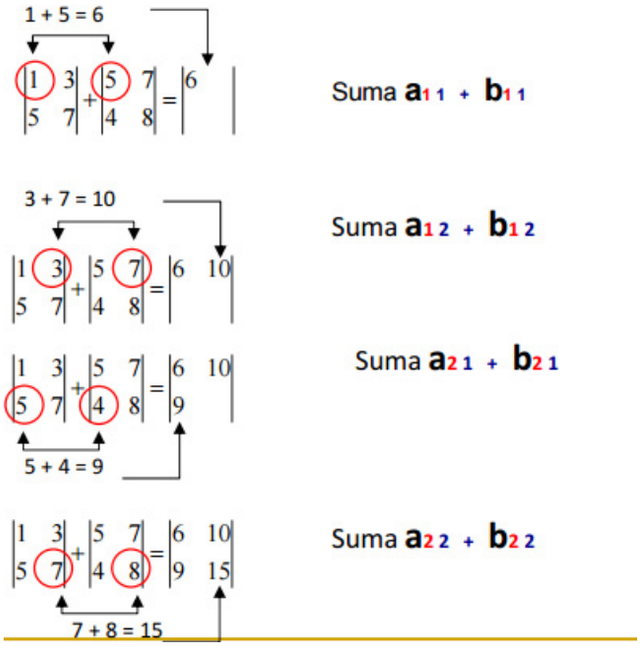
Fuente: Calderón Sedano Carlos Alberto

2. OPERACIONES CON ARREGLOS BIDIMENSIONALES

Veamos algunas operaciones básicas con matrices como son:

Suma de matrices: Para realizar el código respectivo primero recordemos el procedimiento para sumar dos matrices:

Figura N° 72: Suma de Matrices



Fuente: http://www.cs.buap.mx/

Ejemplo 01 Escribir un programa que permita ingreso dos matrices y realiza la suma respectiva.

Figura N° 73: Programa en C++ Matrices Nro. 02

```
(Global Scope)                                                    ma
#include <iostream>
using namespace std;
int main()
 {
 int matriz1[3][3], matriz2[3][3], suma[3][3];
  int i,j;
 /* Ingresar valores a la matriz 01 */
 for (i=0;i<3;i++)
    for (j=0;j<3;j++)
        cin>>matriz1[i][j];
 /* Ingresar valores a la matriz 02 */
 for (i=0;i<3;i++)
    for (j=0;j<3;j++)
        cin>>matriz1[i][j];
  /* Sumar las matrices */
 for (i=0;i<3;i++)
    for (j=0;j<3;j++)
        suma[i][j]= matriz1[i][j] + matriz2[i][j];

 system("pause");
 return 0;
 }
```

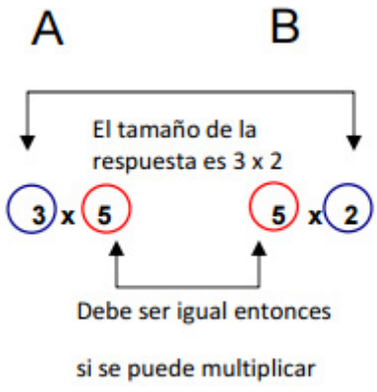
Fuente: Calderón Sedano Carlos Alberto

Resta de matrices: el procedimiento para la resta es exactamente igual al de la suma solo cambiando los símbolo menos (-) por el mas (+).

Multiplicación de matrices

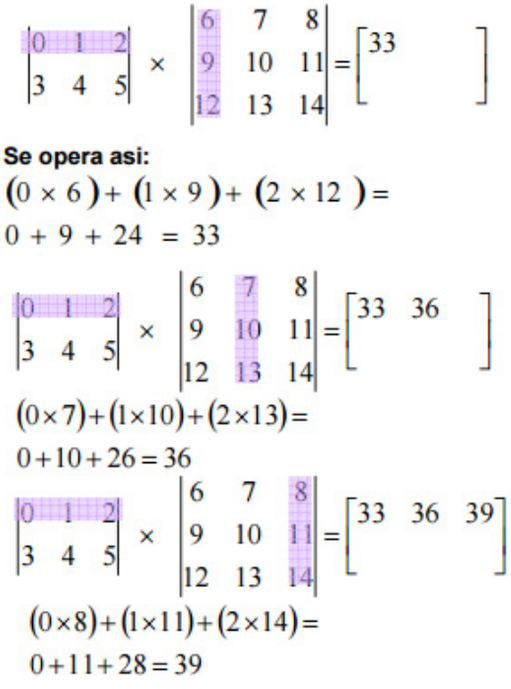
Primero debe verificarse el número de filas y columnas de las dos matrices A y B, el resultado se almacena en la matriz C, de la siguiente forma:

Figura N° 74: Multiplicación de Matrices Nro. 01



Fuente: http://www.cs.buap.mx/

Figura N° 74: Multiplicación de Matrices Nro. 02



Fuente: http://www.cs.buap.mx/

Algoritmo para multiplicar dos matrices

Pseudocódigo

1. Leer las variables de entrada m, p, n

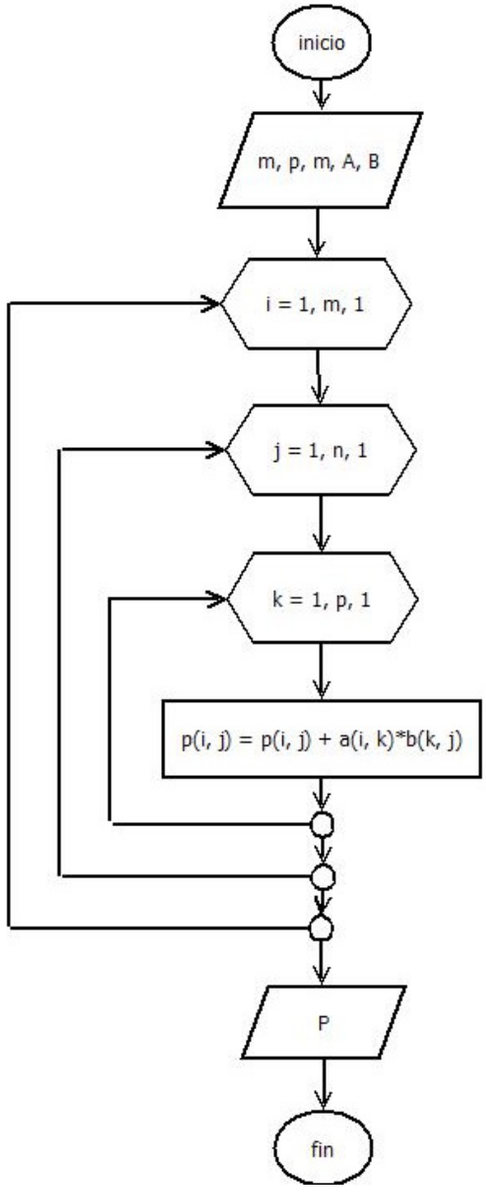
2. Leer A y B

3.

```
Desde i=1 hasta m, con paso 1, hacer:
    Desde j=1 hasta n, con paso 1, hacer:
        Desde k=1 hasta p, con paso 1, hacer:
        P(i, j) = p(i, j) + a (i, k) * b(k, j)
        Fin bucle en k
    Fin bucle en j
Fin bucle en i
```

Diagrama de Flujo

Figura N° 75: Diagrama de flujo: Multiplicación de Matrices



Fuente http://www.wikispaces.com

El código en C++ para multiplicar dos matrices:

```
int main()
{    int i, j, k, a[d][d], b[d][d], c[d][d];
   cout << "MATRIZ A." << endl; // Introduce los elementos de la matriz A
   for(i = 0 ; i < d ; i++)
        {
      for(j = 0 ; j < d ; j++)
           {
              cout << "Introduzca el valor del elemento [" << i << "]["
                        << j << "]: ";
              cin >> a[i][j];
           }
  }
   cout << endl;
   for(i = 0 ; i < d ; i++)
        { // Imprime los elementos de la matriz A
      for(j = 0 ; j < d ; j++)
           {
         cout << a[i][j] << " ";
         if(j == 2)
           cout << endl;
     }
   }
   cout << endl;

 cout << "MATRIZ B." << endl; // Introduce los elementos de la matriz B
   for(i = 0 ; i < d ; i++)
         {
      for(j = 0 ; j < d ; j++)
           {
         cout << "Introduzca el valor del elemento [" << i << "]["
                << j << "]: ";
         cin >> b[i][j];
      }
   }
   cout << endl;

   for(i = 0 ; i < d ; i++)
        { // Imprime los elementos de la matriz B
      for(j = 0 ; j < d ; j++)
           {
         cout << b[i][j] << " ";
         if(j == 2)
           cout << endl;
      }
   }

 for(i=0;i<d;i++)
 { /* Realiza el producto de matrices y guarda
                el resultado en una tercera matriz*/
    for(j=0;j<d;j++)
    {
```

```
        c[i][j]=0;
        for(k=0;k<d;k++)
            {
           c[i][j]=c[i][j]+(a[i][k]*b[k][j]);
        }
    }
}

cout << endl << "MATRIZ C (Matriz A*B)." << endl;
cout << endl;

for(i=0;i<d;i++)
{ // Imprime la matriz resultado
    for(j=0;j<d;j++)
    {
        cout << c[i][j] << " ";
        if(j==2)
        cout << endl;
    }
}
system("PAUSE");
return 0;
}
```

Búsqueda en Matrices

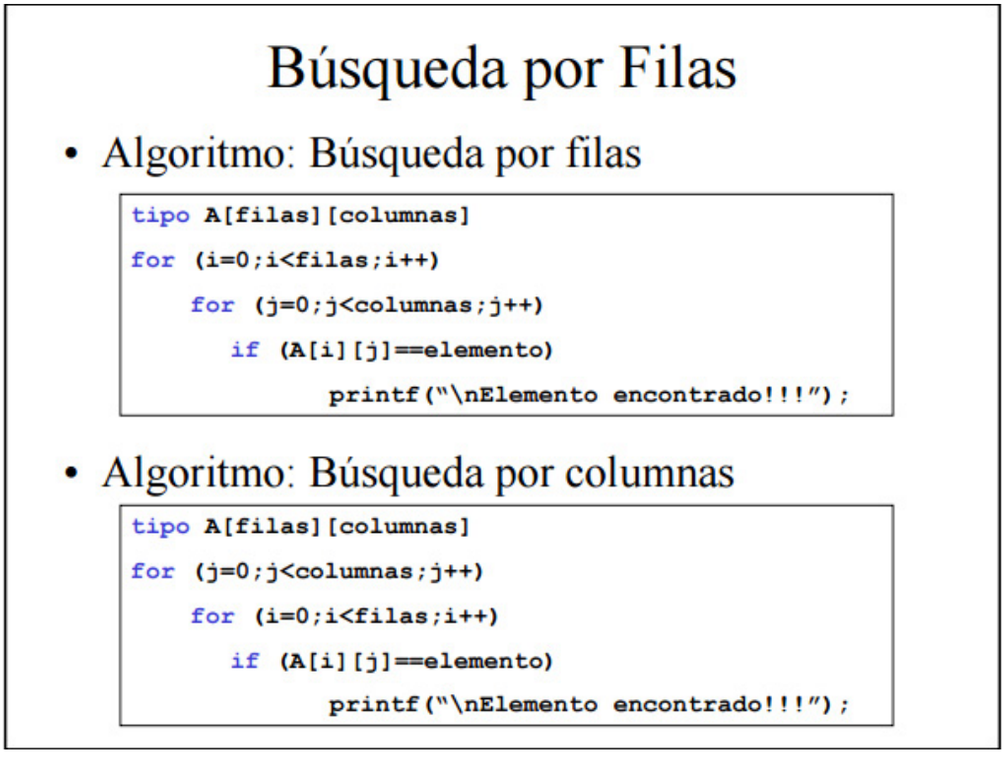
Para buscar un elemento en un arreglo de dos dimensiones (el menor o el mayor), podemos suponer que uno de ellos es el menor (mayor), o mejor suponer un valor muy alto (o muy bajo), para luego contrastarlo uno a uno cada elemento, es decir una búsqueda secuencial.

Figura N° 76: Valor Máximo y Mínimo en Matrices

```
#include <stdio.h>
#define N 3

int main() {
    int i,j,max,min, a[N][N];
    for(i=0; i<N; i++)
        for(j=0; j<N; j++)
            a[i][j] = rand();
    max=-1000;
    min= 1000;
    for(i=0;i<N;i++)
        for(j=0;j<N;j++) {
            if (a[i][j]>max)
                max = a[i][j];
            if (a[i][j]<min)
                min = a[i][j];
        }
    printf("el maximo es %d y el minimo es %d\n",max,min);
    return 0;
}
```

Fuente: http://www.inf.utfsm.cl/

Figura N° 77: Búsqueda en Matrices

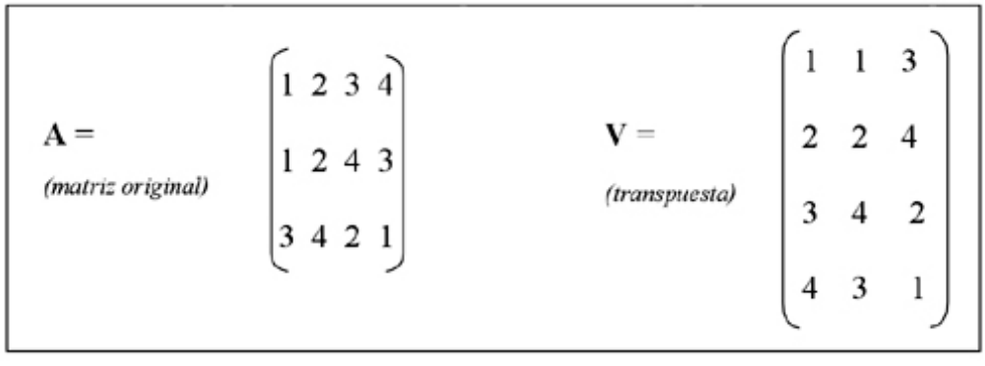


Fuente: http://www.inf.utfsm.cl/

Transpuesta de una matriz

Una matriz está compuesta de filas (reng) y columnas (col), la matriz transpuesta es una matriz que cambia las columnas por las filas en el siguiente ejemplo vemos la matriz A y su transpuesta la matriz V.

Figura N° 78: Transpuesta de una Matriz



Fuente: http://ceciliaurbina.blogspot.com/

El código para obtener la transpuesta de una matriz en el lenguaje C++sería:

```
int i,j;
int temp;
   For (int i=0; i<n; i++)
   {
      For (int j = i+1;j<n;j++)
      {
         temp:=A[i,j];
         A[i,j]:=A[j,i];
         A[j,i]:=temp;
       }
   }
```

LECTURA SELECCIONADA N° 2:

ALAN TURING, INTELIGENCIA MILITAR Y MÁQUINAS INTELIGENTES

Autor: Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez.

Alan M. Turing, el matemático británico que diseñó la primera computadora digital electrónica funcional del mundo en la década de 1940, puede haber sido el pensador más importante en la historia de la informática. Mientras era estudiante en Princeton en 1936, Turing publicó «On Computable Numbers», un documento que sentó el fundamento teórico de la moderna ciencia de la informática. En ese documento, describía una «máquina Turing» que podría leer instrucciones de una cinta de papel perforada y ejecutar las operaciones críticas de una computadora. El documento también establecía los límites de la ciencia de la informática demostrando matemáticamente que algunos problemas simplemente no pueden ser solucionados por cualquier clase de computadora.

Después de recibir su doctorado en 1938, Turing tuvo una oportunidad de llevar su teoría a la realidad. El gobierno británico, anticipando una invasión de las tropas de Hitler, reunió un equipo de matemáticos e ingenieros que tenía la misión secreta de descifrar el código militar alemán. Bajo el liderazgo de Turing y otros, el grupo construyó Colossus, una máquina de propósito único considerada por muchos como la primera computadora digital electrónica. Desde que se terminó Colossus en 1943 hasta el final de la guerra, descifró satisfactoriamente los códigos nazis: un hecho ocultado por el gobierno británico hasta mucho después de haber terminado la guerra. Muchos expertos creen que Colossus fue finalmente el responsable de la derrota de los nazis.

Turing inició el campo de la inteligencia artificial (AI) con un documento de 1950 titulado «Computing Machinery and Intelligence», en el que proponía una prueba concreta para determinar si una máquina era inteligente. En los años siguientes, Turing defendió la posibilidad de emular el pensamiento humano a través de la computación. Incluso, fue coautor del primer programa que jugaba al ajedrez.

Turing era una persona poco convencional y extremadamente sensible. En 1952 fue

arruinado, profesional y socialmente, cuando le arrestaron. Aparentemente, este genio de 41 años se suicidó en 1954, años antes de que el gobierno hiciera públicos sus héroes de guerra. Cuatro décadas después de su muerte, el trabajo de Turing todavía tiene relevancia entre los científicos informáticos, matemáticos y filósofos. La arquitectura de las computadoras actuales está basada en las ideas de Turing. El premio más alto en informática, el Premio Turing, hace honor a su nombre. Es imposible saber lo que podría haber contribuido de haber vivido durante esas décadas.

Alan Turing consumió mucha de su corta vida en intentar responder a la pregunta «¿Las máquinas pueden pensar?». Sigue siendo la pregunta central de la inteligencia artificial (IA, o AI si nos atenemos a la expresión en inglés; Artificial Intelligencel), el campo de la informática dedicado a conseguir que las computadoras perciban, razonen y actúen de la forma que, hasta ahora, está reservada a los humanos. Pero incluso hoy, los que creen que las computadoras no pueden «pensar» tienen que admitir que la investigación en IA ha producido resultados impresionantes: computadoras que se pueden comunicar en lenguajes humanos; sistemas que proporcionan asistencia técnica instantánea en medicina, ciencia y otros campos; jugadores electrónicos de ajedrez de calidad mundial; y robots que pueden sustituir a los humanos en cantidad de tareas.

VIDEOS

Video 27: Crear arreglos bidimensionales.

Este material de video ha sido seleccionado solo y únicamente con fines de estudio académico y todos sus derechos corresponden a sus autores en el ámbito local, regional e internacional.

Datos del Video seleccionado

Título o Tema: Crear Arreglos Bidimensionales.
URL: https://youtu.be/6F6iwj848EU?t=10s
Duración: 9 min 44 s.
Autor(a): ProgramarFácil.
Año: 2011.
Licencia: YouTube estándar.

ACTIVIDAD N° 2

Implementa y organiza la información en datos estructurados usando arreglos bidimensionales a través del desarrollo de un programa con matrices.

Instrucciones:

1. Lee y analiza, todos los contenidos de tema N° 2.

- Como apoyo para desarrollar la actividad visualiza el siguiente video: Arreglos bidimensionales en C++ https://www.youtube.com/watch?v=5LoigeLCeoo

2. Investiga para ampliar la información recibida.
3. Escribe un programa que permita el ingreso de una matriz de orden 3x3, se debe ingresar sólo números positivos.
4. El programa debe reportar la diagonal principal de la matriz.

PRUEBA DE DESARROLLO N° 3

INSTRUCCIONES: Lea cuidadosamente cada enunciado y responda según se requiera.

1. Escribir un programa en lenguaje C++ que permita restar 2 matrices. **(05 puntos)**
2. Escribir un programa en lenguaje C++ que ingrese un vector de 30 números enteros diferentes y que pueda eliminar un número ingresado por el usuario. **(05 puntos)**
3. Escribir un programa en lenguaje C++ que permita ingresar un vector de 100 posiciones, una vez ingresado los 100 números se debe mostrar al número menor: **(05 puntos)**
4. Escribir un programa en lenguaje C++ que permita ingresar una matriz cuadrada y debe imprimir la diagonal principal.**(05 puntos)**

GLOSARIO DE LA UNIDAD IV

I

INDICE
Valor posicional.

M

MATRIZ
Estructura de dato de una dimensión de dos dimensiones

V

VECTOR
Estructura de dato de una dimensión.

BIBLIOGRAFÍA DE LA UNIDAD IV

- Beekman,, George.,Pacheco, Rosbinda.,Tábora Alez. (2008). Introducción a la Computación. México: Pearson Educación.
- Joyanes, Luis. (2008). Fundamentos de Programación. Madrid: Mc Graw Hill.
- Joyanes, Luis. (2007). Estructura de datos en C++. Madrid: Mc Graw Hill.
- Lecca Eduardo. (2000) El poder de turbo C++. Perú: Mundigraf.

AUTOEVALUACIÓN Nº 4

INSTRUCCIONES:

Lea cuidadosamente cada enunciado y responda según se requiera (Remarque/escriba con color azul su respuesta).

1. Un _____ representa a un arreglo bidimensional

   a. Matriz.

   b. Vector

   c. Clase

   d. Objeto.

   e. Microsoft.

2. Un _____ representa a un arreglo unidimensional

   a. Vector.

   b. Herencia

   c. Clase

   d. Objeto.

   e. Microsoft.

3. La diagonal principal de una matriz se puede calcular solo si la matriz es:

   a. Cuadrada

   b. Rectangular

   c. Cualquier tipo de matriz

   d. Abstracta

   e. De n dimensiones

4. En el lenguaje de programación C++ se comienza a contar desde::

   a. Cero

   b. Uno

c. Dos

d. Tres

e. Cuatro

5. El ordenamiento Burbuja también es conocido como::

a. bubleSort

b. SearchSort

c. BuscarValor

d. FindSort

e. Sort

ANEXO:
CLAVES DE LAS AUTOEVALUACIONES

Respuestas de la Autoevaluación de la Unidad I

NÚMERO	RESPUESTA
1	A
2	A
3	A
4	A
5	A
6	A
7	A
8	A
9	A
10	A

Respuestas de la Autoevaluación de la Unidad II

NÚMERO	RESPUESTA
1	A
2	A
3	A
4	A
5	A

Respuestas de la Autoevaluación de la Unidad III

NÚMERO	RESPUESTA
1	A
2	A
3	A
4	A
5	A

Respuestas de la Autoevaluación de la Unidad IV

NÚMERO	RESPUESTA
1	A
2	A
3	A
4	A
5	A

UNIVERSIDAD CONTINENTAL

Este manual autoformativo es el material didáctico más importante de la presente asignatura, desarrollada para la modalidad virtual. Elaborado por el docente, orienta y facilita el autoaprendizaje de los contenidos y el desarrollo de las actividades propuestas en el sílabo.

Los demás recursos educativos del Aula virtual complementan y se derivan del manual. Los contenidos multimedia ofrecidos utilizan vídeos, presentaciones, audios y clases interactivas que se corresponden con los contenidos del presente manual.

La modalidad te permite estudiar desde el lugar donde te encuentres y a la hora que más te convenga. Basta conectarte a Internet e ingresar al campus virtual para encontrar todos tus servicios: aulas, vídeoclases, presentaciones animadas, biblioteca de recursos, muro y las tareas, siempre acompañado de tus docentes y amigos.

El modelo educativo de la Universidad Continental virtual es innovador, interactivo e integral, conjugando el conocimiento, la investigación y la innovación. Su estructura, organización y funcionamiento están de acuerdo con los estándares internacionales. Es innovador, porque desarrolla las mejores prácticas del *e-learning* universitario global; interactivo, porque proporciona recursos para la comunicación y colaboración síncrona y asíncrona con docentes y estudiantes; e integral, pues articula contenidos, medios y recursos para el aprendizaje permanente y en espacios flexibles.

MANUALES **AUTOFORMATIVOS**