Introducción a la Programación

CON R Y RSTUDIO

Edwin Fernando Mejía Peñafiel
Johanna Enith Aguilar Reyes
Nancy Elizabeth Chariguamán Maurisaca
Rodrigo Rigoberto Moreno Pallares

CIDE Editorial
www.cidepro.org

Edwin Fernando Mejía Peñafiel
Johanna Enith Aguilar Reyes
Nancy Elizabeth Chariguamán Maurisaca
Rodrigo Rigoberto Moreno Pallares

INTRODUCCIÓN A LA PROGRAMACIÓN CON R Y RSTUDIO

INTRODUCTION TO PROGRAMMING WITH R AND RSTUDIO

Edwin Fernando Mejía Peñafiel
Johanna Enith Aguilar Reyes
Nancy Elizabeth Chariguamán Maurisaca
Rodrigo Rigoberto Moreno Pallares

Introducción a la programación
Con R y Rstudio

Introduction to programming
With R and Rstudio

Autores:

Edwin Fernando Mejía Peñafiel
Escuela Superior Politécnica de Chimborazo
Estadística - Ciencias
efmejia@espoch.edu.ec
https://orcid.org/0000-0001-6888-4621

Nancy Elizabeth Chariguamán Maurisaca
Escuela Superior Politécnica de Chimborazo
Facultad de Ciencias
nchariguaman@espoch.edu.ec
https://orcid.org/0000-0002-7345-0710

Johanna Enith Aguilar Reyes
Escuela Superior Politécnica de Chimborazo
Facultad de Ciencias
johannae.aguilar@espoch.edu.ec
https://orcid.org/0000-0002-1230-2503

Rodrigo Rigoberto Moreno Pallares
Escuela Superior Politécnica de Chimborazo
Facultad de Mecánica
rodrigo.moreno@espoch.edu.ec
https://orcid.org/0000-0003-1877-6942

Primera Edición, octubre 2022

Introducción a la programación
Con R y Rstudio

ISBN: 978-9942-607-11-9 (eBook)
ISSN: 2600-5719 (electronic)
https://doi.org/10.29018/978-9942-607-11-9

Editado por:
Centro de Investigación y Desarrollo Profesional

Babahoyo, Ecuador
Móvil - (WhatsApp): (+593) 9 8 52-92-824
www.cidepro.org
E-mail: editorial@cidepro.org

Este texto ha sido sometido a un proceso de evaluación por pares externos con base en la normativa editorial de CIDEPRO.

Diseño y diagramación:
CIDEPRO Editorial

Diseño, montaje y producción editorial:
CIDEPRO Editorial

Hecho en Ecuador
Made in Ecuador

ÍNDICE

CAPÍTULO 1

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

CAPÍTULO 5

PRÓLOGO

El objetivo de este escrito es proporcionar un punto de partida para los programadores, estadísticos y todas las personas que desean aprender algoritmos, diagramas de flujo y la programación en R, se ha escogido el lenguaje R para dar una manera de hacer énfasis en la automatización y así entender lo que significa programar.

Un informático, típicamente un programador, tiene en sus manos un conjunto de recursos de programación: materiales y herramientas, que al combinarlos de acuerdo con su técnica y habilidad, transforma un problema del mundo real en un programa que se resuelve automáticamente en el computador. Aunque esencialmente tecnológica, en el caso de la estadística, el estadístico tiene ese mismo papel: los distintos ambientes de la estadística bajo R ponen delante de él un conjunto de recursos que de acuerdo con sus conocimientos, habilidades y sensibilidades, combinará para producir obras: programas y sistemas, que, en la superficie, serán funcionales o no funcionales; pero que, a un nivel más profundo, podrían también ser juzgadas como estéticamente atractivas o repulsivas. Uno de los factores que más decisivamente influyen en el cómo un creador combina los elementos a su mano, es el gusto y la pasión que imprime en su tarea. Por consiguiente, si de la lectura de este texto se logra encender en ti, querida lectora o lector, una pasión que te lleve a producir verdaderas obras de arte, los autores estaremos más que satisfechos por haber cumplido el propósito de esta pequeña obra.

PREFACE

The objective of this paper is to provide a starting point for programmers, statisticians and all people who wish to learn algorithms, flowcharts and programming in R, the R language has been chosen to give a way to emphasize automation and thus understand what it means to program.

A computer scientist, typically a programmer, has in his hands a set of programming resources: materials and tools, which when combined according to his technique and skill, transforms a real-world problem into a program that is automatically solved on the computer. Although essentially technological, in the case of statistics, the statistician has that same role: the various statistical environments under R place before him a set of resources that, according to his knowledge, skills and sensibilities, he will combine to produce works: programs and systems, which, on the surface, will be functional or non-functional; but which, at a deeper level, could also be judged as aesthetically appealing or repulsive. One of the factors that most decisively influence how a creator combines the elements at hand is the taste and passion he brings to his task. Therefore, if the reading of this text manages to ignite in you, dear reader, a passion that leads you to produce true works of art, the authors will be more than satisfied for having fulfilled the purpose of this small work.

INTRODUCCIÓN

En la actualidad el ser humano se enfrenta a distintas situaciones: de aprendizaje, de retroalimentación y en muchas ocasiones dificultades que con la experiencia o por la elección de la alternativa apropiada, va dando solución.

Es por ello que a menudo se emplea cierta metodología para la solución de los problemas en lugar de actuar de forma imprevista, siendo una característica relevante el análisis de los mismos. La humanidad de forma natural emplea en la vida cotidiana ciertas conductas que son rutinarias, siguen un orden, una secuencia y pretenden alcanzar un objetivo. Este conjunto de acciones rutinarias que se llevan a cabo y forman parte ya de la vida cotidiana del ser humano, se conocen como algoritmos, los cuales son aplicables en los ámbitos que así se necesiten.

El ámbito de mayor de aplicación y de primordial importancia es en la solución de problemas mediante computadora. Donde el elemento base para lograr dicha solución es el algoritmo propio. Desarrollar un algoritmo involucra tener un conocimiento base sobre las características y elementos que debe contener, con el fin de cumplir sus cualidades: finito, definido y preciso.

Es importante señalar que en el ámbito de la programación antes de resolver el problema mediante la computadora se recomienda realizar primero el algoritmo, ya que es aquí donde se encuentra la solución universal de la problemática en cuestión.

CONCEPTOS GENERALES

Capítulo 1

CONCEPTOS GENERALES

Informática

Es la ciencia que estudia el tratamiento automático y racional de la información. Más específico: es el conjunto de conocimientos científicos y técnicos que hacen posible analizar la información por medio de ordenadores electrónicos. La palabra "informática" se deriva del francés informatique acuñado por el ingeniero Philippe Dreyfus en 1962. En español es la contracción de INFORmación autoMÁTICA y en inglés se le conoce como computer science (ciencia de la computación).

El computador

Figura 1. Computador

Definición: es un dispositivo electrónico capaz de recibir, almacenar y procesar información de una forma útil. Una computadora está programada para realizar operaciones lógicas o aritméticas de forma automática. En la figura 1 se muestra un computador con sus

componentes principales como son: teclado, mouse, monitor y case donde encontramos los componentes electrónicos del mismo.

Componentes del computador

En un computador, existen tres componentes generales, los cuales permiten el funcionamiento correcto. Podemos decir que cada uno cumple con una característica fundamental que es, permitir que el usuario pueda generar sus distintos documentos electrónicos, programas y software de distinta índole.

Los componentes internos son los que componen el hardware de nuestro equipo, y serán los encargados de manejar la información que nosotros introducimos o la que descargamos desde Internet. Serán los que nos harán posible almacenar datos, almacenar programas o mostrar por una pantalla los trabajos que realizamos. Los componentes básicos como se muestran en la figura 2 son:

- Hardware (HW)
- Software (SW)
- Firmware (FW)

Figura 2. Componentes del Computador

Hardware

Los componentes hardware son los dispositivos físicos que componen el computador, posibilitando el funcionamiento y complementando el Software. Entre las partes que lo forman están: Dispositivos de entrada, dispositivos de salida y dispositivos de entrada/salida.

Dispositivos de entrada

Dentro del ámbito informático, podemos decir que los dispositivos de entrada o dispositivos de alimentación (input) son los que nos permiten ingresar datos o información al computador.

Figura 3. Dispositivos de entrada

Los dispositivos de entrada son los siguientes:

- Teclado: compuestas por teclas agrupadas para actuar como placas mecánicas que envían información a la computadora.
- Mouse: dispositivos que mueve un puntero en la pantalla para seleccionar información.
- Micrófono: traductor electro acústico.
- Webcam: pequeña cámara digital conectada a la computadora.
- Lápiz óptico: periférico de entrada para la computadora, puede ser usado para apuntar objetos en el monitor.
- Escáner: se usa para introducir imágenes en papel a la computadora.
- Lector de código de barras: escáner que lee un código de barra y transmite el cogido a la pantalla.
- Joystick: dispositivo de control que se usa para una computadora o video consola para ejecutar los movimientos.

Dispositivos de salida

Los dispositivos de salida son también conocidos con el nombre de periféricos ya que no están integrados en el ordenador. La función principal de los dispositivos de salida es la de ofrecer al usuario información que, previamente, ha sido procesada en el ordenador. Es decir, el computador nos brinda información al mundo real.

Figura 4. Dispositivos de salida

Los dispositivos de salida tenemos en la figura 4 y son los siguientes: Impresora: Es uno de los dispositivos de salida más utilizados por los usuarios. Su principal función es la de realizar la transformación de un documento de texto o imagen que podemos ver en pantalla a un formato físico en papel.

Entre los tipos de impresoras que podemos encontrar, destacan las siguientes:

Impresoras de inyección de tinta: suelen ser las más comunes, especialmente en el ámbito doméstico. Funcionan mediante la

distribución de tinta que va conformando en el papel la imagen que aparece en pantalla, tal cual como se muestra en la figura 5.

Figura 5. Impresora de inyección de tinta

Impresoras láser: suelen tener un tamaño más grande y su uso está destinado para empresas que utilizan grandes volúmenes de impresiones. Funcionan con tóner y aunque proporcionan una extraordinaria nitidez y velocidad, tienen un precio superior. Como se observa en la figura 6:

Figura 6. Impresora láser

Impresoras matriciales: es un periférico del ordenador que tiene la función de transcribir o marcar un documento desde el ordenador hasta el papel. Para ello, la cabeza de impresión se desplaza de un lado a otro utilizando pines que marcan con tinta el papel para formar el gráfico que se imprime, como se ve en la figura 7:

Figura 7. Impresora matricial

Monitor: también conocido como pantalla, muestra la información de tu equipo como imágenes y textos, que son generados gracias a una tarjeta de video que se encuentra en el interior de la torre del computador. La resolución de los monitores depende de la cantidad de píxeles que estos tengan.

Figura 8. Tipos de monitores

Un monitor puede clasificarse, según la tecnología empleada para formar las imágenes en:

- Tubo de rayos catódicos o CRT (Cathode Ray Tube).
- Pantalla de cristal líquido o LCD (Liquid Crystal Display).
- Pantalla de plasma o PDP (Plasma Display Panel).
- TFT LCD (Thin Film Transistor: transistor de películas finas).
- Pantalla LED (Light Emitting Diode: diodo emisor de luz).
- OLED (Organic Light-Emitting Diode: diodo orgánico de emisión de luz).
- AMOLED (Active Matrix OLED: OLED de matriz activa).
- Super AMOLED (Super Active Matrix Organic Light-Emitting Diode: Súper AMOLED).

En tanto, según el estándar, un monitor puede clasificarse en:

- Monitor numérico
- MDA
- CGA
- EGA
- analógico,
- VGA
- SVGA

En cuanto a los colores que usan los monitores pueden ser:

- Monitor monocromático.
- Monitor policromático.

En cuanto a si es solamente un Periférico de salida (S) o Periférico de entrada/salida (E/S):

- Monitor no táctil: S
- Pantalla táctil (touch screen): E/S
- Multitáctil (multitouch): E/S

Se pueden clasificar en función de la forma de la pantalla:

- Pantalla plana: La pantalla no presenta curvas
- Pantalla curva: La pantalla está ligeramente curvada para dar una sensación de inmersión más grande

Un monitor se puede clasificar con base en una serie de factores, y con estos podemos comparar cuál nos conviene más en función de:

Pulgadas: este factor nos permite elegir el tamaño de un monitor, ya que más pulgadas será sinónimo de mayor superficie de pantalla. Normalmente las pulgadas de un monitor oscilan entre las 17 y las 32.

Resolución: Respecto a este factor, a más resolución mayor será el detalle con el que se reproducirán las imágenes en el monitor. Ejemplos de resolución son el HD (720p), FULL HD (1080p) o 4K (2160p), entre otras.

Tecnología: En este apartado se abarca la tecnología que utiliza el monitor, pudiendo ser táctil, de pantalla curva, entre otras características específicas extras.

Por tanto, si nos ceñimos a estas características podremos determinar el tamaño, la calidad de visualización y la tecnología que usa el monitor. Esto determinará el precio final del monitor cuanto mayor sean las pulgadas y resolución, además de la tecnología integrada en el mismo.

En la tipología de monitores hay un factor aún más determinante que marca la diferencia. Esta se trata del tipo de pantalla que posee. Algunas clases de pantallas que nos podemos encontrar en el mercado son:

- Plasma (Plasma Display Panel).
- LCD (Liquid Crystal Display).
- LED (Light Emitting Diode).

Estas clases de pantallas dan paso a otras variantes como las 'TFT', originarias de las propias LCD, o las 'OLED' y 'AMOLED', las cuales son versiones mejoradas de las LED. Cabe destacar que la calidad de las pantallas además de todos los factores y características expuestas, pueden verse afectadas por otras adicionales. (Llamas J., 2021)

Auriculares: Son transductores que reciben una señal eléctrica de un tocador de medios de comunicación o el receptor y usan altavoces colocados en la proximidad cercana a los oídos para convertir la señal en ondas sonoras audibles. Como se observa en la figura 9:

Figura 9. Auriculares con micrófono incorporado.

Proyector: Un proyector de video, video proyector, cañón proyector o data show es un aparato óptico que recibe una señal de video y proyecta la imagen correspondiente en una pantalla de proyección usando un sistema de lentes, permitiendo así mostrar imágenes fijas o en movimiento. Como se observa en la figura 10:

Figura 10. Proyector

Altavoces: Son los dispositivos que le dan salida de audio al computador, gracias a ellos podemos escuchar el sonido de la música o video que estés reproduciendo. Dependiendo del modelo los puedes conectar al puerto USB o al de audio. Algunos computadores traen los

altavoces incorporados dentro del monitor. Tal cual como se observa en la figura 11:

Figura 11. Altavoces del computador

Plotters: Un plotter es una impresora que interpreta las órdenes de un ordenador para hacer dibujos vectoriales en papel mediante una pluma o lápiz. A diferencia de una impresora normal, el plotter puede dibujar líneas continuas directamente a partir de gráficos vectoriales, mientras que las primeras imprimen mediante un conjunto de puntos.

Los plotters son mucho más lentos que las impresoras porque los movimientos que lleva a cabo su pluma son muy diferentes que la impresión por puntos.

Los plotters de impresión son muy utilizados por arquitectos y diseñadores. Además, los plotters permiten imprimir imágenes de gran tamaño y son la herramienta con la que se trabaja en vinilos, ya sean textiles o de rotulación. (Mastoner, 2021)

En la figura 12 se muestra un plotter:

Figura 12. Plotter

¿Cómo funciona una impresora plotter?

Un plotter depende del software de imágenes que utilice el ordenador para producir la imagen final. Hay que indicar las coordenadas en las que se desea que aparezca la imagen en el papel, algo que el software moderno permite fácilmente mediante el trazado de líneas e imágenes.

Una vez se haya realizado el esquema de la imagen, el ordenador envía las coordenadas a la impresora, que interpreta el código y calcula la ruta más eficiente para trazar las líneas en el papel mediante la pluma. Por lo general, los plotters usan HPGL2 de Hewlett-Packard o DMPL de Houston Instruments. (Mastoner, 2021).

¿Cuál es la diferencia entre una impresora normal y una impresora plotter?

Las impresoras tradicionales usan un tóner para imprimir puntos en el papel o material que se esté usando. Las líneas rectas que producen

no están tan claras o rectas como deberían por este motivo. Como una impresora plotter usa herramientas como plumas, su impresión de líneas es mucho más certera.

Además, con un plotter se puede imprimir en materiales más allá del papel. Algunos diseñadores de moda imprimen sobre tela y en publicidad es muy común imprimir sobre aluminio, cartón o plástico. No hay que olvidar que las impresoras plotter son más grandes que las impresoras normales. (Mastoner, 2021)

Tipos de plotters

Plotters de pluma

Los cabezales de estos plotters permiten colocar seis u ocho plumillas, bolígrafos, rotuladores o lo que deseemos. La cercanía con el papel se puede regular. Fueron los primeros plotters en salir al mercado y se utilizaban para dibujar planos en ramas destinadas a la arquitectura o topografía, pero a día de hoy todavía se siguen utilizando. Son más lentos, por lo que tardan más en realizar un trabajo pero ofrecen gran calidad y suavidad en las curvas. (Mastoner, 2021)

Plotters de inyección de tinta

Este es uno de los plotters más extendidos. Se parecen a una impresora normal, pero de gran tamaño. Por lo general, su tamaño base es A1. Se podrían definir como una impresora de chorro de tinta, pero a gran formato. La diferencia entre este tipo de plotter y el resto es que ofrecen impresiones de gran calidad en cuanto a colores. (Mastoner, 2021)

Plotters electrostáticos, láser o térmicos

Son más caros que otros, tienen un tamaño de punto menor y resisten mejor la luz y el paso del tiempo. Su acabado recuerda a la impresión de un fax y no suelen imprimir a color. Suelen ser más caros que el resto, pero la calidad que ofrecen es similar. Sin embargo, aunque no dibujan a color y el resultado final no es de mayor calidad, son más resistentes a la luz y al paso del tiempo. (Mastoner, 2021)

Plotters de corte

Si se cambia la pluma por una cuchilla, se pueden crear diseños de corte. Algunos plotters de corte permiten regular la presión que ejerce la cuchilla. Estos se utilizan para cortar vinilos, tela, carteles, señales personalizadas y muchos otros tipos de productos. Algunos modelos dan la posibilidad de cortar materiales más gruesos como cartulinas o cartones. Su diferencia con el resto es que pueden ser de muchos tamaños y tipos: de mesa o de rodillo, de corte tangencial, arrastre o cabezal excéntrico y funcionan tanto por arrastre como por fricción. (Mastoner, 2021)

Tipos de plotter según su diseño

Ya hemos adelantado los tipos de diseños de los plotters anteriormente, pero aquí te contamos los dos tipos de plotters que hay en el mercado.

Plotter plano o de mesa

El documento no se mueve. Se mantiene estático sobre una superficie plana y es la pluma la que se mueve sobre el papel de forma horizontal y verticalmente. El tamaño del gráfico está determinado por el tamaño de la superficie de trazado.

Plotter de tambor o rodillo

En este tipo de plotters se enrolla el papel a un rodillo o tambor. Este gira, lo que permite que se dibuje la imagen a su paso. (Mastoner, 2021).

¿Qué puedes hacer con un plotter?

Antes de pasar a determinar qué se puede hacer con un plotter de corte, vamos a clasificar los plotters de corte de vinilo.

Por un lado, están los plotters de corte domésticos. Se utilizan principalmente para las manualidades como el scrapbooking y pueden cortar materiales de bastante grosor, como la goma eva o el cuero. Sin embargo, a pesar de que están destinados a uso doméstico, también son ideales para negocios que están empezando y que tienen bajo volumen de producción.

Por otro lado, están los plotters de corte profesionales. Destinados a negocios con alto volumen de producción, se utilizan para cortar materiales de tiradas largas. Los plotters profesionales están pensados para trabajar el vinilo, pero ofrecen muchas más prestaciones, además de poder trabajar con mayor precisión.

Una vez que tenemos clara la diferencia entre el uso doméstico y el profesional, estas son algunas tareas que se pueden hacer con un plotter de corte vinilo:

- Se pueden hacer estampaciones en camisetas, pantalones, gorras o bolsas de tela con vinilo textil.

- Carteles o frases para paredes, puertas, cristales. Cualquier vinilo de rotulación.
- Transferir imágenes mediante papel transfer.
- Recortar diseños que se hayan impreso en tintas de sublimación, inkjet, láser o tintas ecosolventes, para crear pósters, etiquetas o pegatinas.

Como ves, las impresoras plotters se centran en la calidad del trazo y en el tamaño de la impresión. No están dirigidas al gran público, sino a profesionales como arquitectos, diseñadores gráficos y publicistas. Además, su gran tamaño impide que cualquier persona pueda meter una de estas impresoras en casa. Sin embargo, los profesionales que las utilicen a diario encontrarán muy útiles sus capacidades. (Mastoner, 2021)

Dispositivos de entrada/salida

Sirven básicamente para la comunicación de la computadora con el medio externo. Los periféricos de entrada/salida son los que utiliza el ordenador tanto para enviar desde el computador hacia el mundo real, como para recibir información del mundo real al computador. Una de sus funciones es la de almacenar o guardar, de forma permanente o virtual, todo aquello que hagamos con el ordenador para que pueda ser utilizado por los usuarios u otros sistemas.

Otra forma de definirlos es como aquel dispositivo que se utiliza para grabar los datos de una computadora de forma permanente o temporal. Una unidad de disco por ejemplo junto con los discos que graba, es un dispositivo de almacenamiento.

Como ejemplos de periféricos de entrada/salida tenemos:

- Disco duro
- Memorias
- Grabadoras - lectoras
- Pantalla táctil
- Pizarras digitales
- Periféricos de comunicación

Disco duro: Un disco duro es un dispositivo de almacenamiento necesario para conservar tus archivos y datos en el largo plazo. Siempre que guardas un archivo en la computadora, lo guardas en el disco duro de la computadora. Un disco duro se parece a un archivador para tus archivos digitales.

Existen 2 tipos:

- Unidad de disco duro - Hard Drive Disk o HDD
- Unidad de estado sólido - Solid State Drive o SSD

Unidad de disco duro – Hard Drive Disk o HDD: Los discos duros, también conocidos como HDD, son un componente informático que sirve para almacenar de forma permanente tus datos. Esto quiere decir, que los datos no se borran cuando se apaga la unidad como pasa en los almacenados por la memoria RAM. La primera empresa en comercializarlos fue IBM en 1956.

Están compuestos de piezas mecánicas, de ahí que a veces se le llame discos duros mecánicos, y utilizan el magnetismo para grabar tus datos y archivos. Se compone de uno o varios discos rígidos unidos por un

mismo eje y que giran a gran velocidad dentro de una caja metálica. En cada plato y en cada una de sus caras, un cabezal de lectura/escritura lee o graba tus datos sobre los discos.

Cuanto más finos sean los discos mejor será la grabación, y cuanto más rápido giran a mayor velocidad se transmiten los datos, tanto a la hora de leerlos como al escribirlos. Por lo general, la velocidad de los discos duros suele ser de 5400 o 7200 RPM (revoluciones por minuto), aunque en algunos discos basados en servidores pueden llegar a hasta 15.000 RPM

En cuanto al tamaño, las cajas de los discos duros mecánicos pueden ser de 2,5" o de 3,5". Su precio puede variar dependiendo de este tamaño, pero sobre todo de su capacidad de almacenamiento. De hecho, la gran ventaja de estos discos duros con respecto a los SSD es que son bastante más económicos. (Fernández Y., 2021)

En la figura 13 se muestra un HDD:

Figura 13. Varios HDD de 10TB

Unidad de estado sólido – Solid State Drive o SSD: Las unidades de estado sólido o SSD (Solid State Drive) son una alternativa a los discos duros. La gran diferencia es que mientras los discos duros utilizan componentes mecánicos que se mueven, las SSD almacenan los archivos en microchips con memorias flash interconectadas entre sí. Por lo tanto, casi podríamos considerarlos como una evolución de las memorias USB.

Los SSD suelen utilizar memorias flash basadas en NAND, que como también son no-volátiles mantienen la información almacenada cuando el disco se desconecta. No tienen cabezales físicos para grabar los datos, en su lugar incluyen un procesador integrado para realizar operaciones relacionadas con la lectura y escritura de datos.

Estos procesadores, llamados controladores, son los que toman las "decisiones" sobre cómo almacenar, recuperar, almacenar en caché y limpiar los datos del disco, y su eficiencia es uno de los factores que determinan la velocidad total de la unidad. Además, al no depender del giro de un componente físico, también se logra una unidad más silenciosa que los discos mecánicos.

En cuanto al tamaño, estos discos suelen ser de 2,5", y tienen un diseño casi idéntico al de los discos duros mecánicos, lo que ayuda a que puedan encajar en las mismas carcasas y ranuras donde van montados los discos duros convencionales en un ordenador. (Fernández Y., 2021)

En la figura 14 se muestra un SDD:

INTEL TRABAJA EN UN SSD DE 20TB

TOSHIBA ASEGURA PODER HACER DE 2.5" DE 85TB.

Figura 14. Discos sólidos próximos a salir en el mercado.

En la tabla 1 se muestra las diferencias existentes entre un HDD vs SDD:

Tabla 1. Diferencias entre un HDD y un SDD

Principales ventajas	Ssd	Hdd
Capacidad	En general entre 256 gb y 4 tb	En general entre 1 y 10 tb
Consumo	Menor consumo	Mayor consumo
Coste	Bastante más caros	Mucho más económicos
Ruido	Más silencioso por no tener partes móviles	Algo más ruidoso por tener partes móviles
Vibraciones	No vibra por no tener partes móviles	El giro de sus discos puede provocar leves vibraciones

Fragmentación	No tiene	Puede darse
Durabilidad	Sus celdas pueden reescribirse un número limitado de veces	Con partes mecánicas que pueden dañarse con movimientos
Tiempo de arranque de so	7 Segundos	16 Segundos
Transferencia de datos	En general, entre 200 y 550 mb/s	En general entre 50 y 150 mb/s
Afectado por el magnetismo	No	El magnetismo puede eliminar datos

Fuente: (Fernández Y., 2021)

En tabla 1 se puede observar cuales son las principales diferencias entre ambas tecnologías de almacenamiento. La principal diferencia tiene que ver con capacidades máximas y precio. Hay que tener en cuenta que las SSD son mucho más modernas, por lo que es normal que su precio sea notablemente superior. A día de hoy, una SSD de 250 GB puede valer lo mismo que un HDD de 3 TB.

Por ser unidades de almacenamiento sin partes móviles como sus antecesores, las SSD tienen algunas ventajas notables como son provocar un menor ruido y vibración. También hay que dejar claro que no podemos decir que los discos duros mecánicos sean tampoco sumamente ruidosos, por lo que es una diferencia no tan importante.

La que sí que es una diferencia notable es la de la velocidad. En nuestra comparativa entre ambas tecnologías vimos cómo un SSD iniciaba un sistema operativo en menos de la mitad del tiempo que un HDD de 7.200 rpm, y que triplicaba ampliamente sus velocidades de escritura y lectura de datos. En nuestra prueba utilizamos un HDD que leía y escribía datos a 150 MB/s, y una SSD que leía y escribía a 545 MB/s y 525 MB/s.

Evidentemente, estos datos concretos de escritura y lectura de datos dependen siempre de los diferentes modelos que hay en el mercado. Pero por lo general los discos duros SSD siempre suelen ser mucho más rápido que los mecánicos. De ahí precisamente su alto precio.

La gran preocupación entorno a los SSD siempre ha estado en torno a su durabilidad, sobre todo por la poca que tuvieron las primeras unidades en llegar al mercado. La vida útil de los discos de estado sólido depende directamente de la cantidad de datos que vas escribiendo en él, ya que cada celda de un banco de sus memorias flash sólo puede ser escrita un número determinado de veces.

Un estudio realizado por Tech Report concluyó que un disco SSD bastante estándar, concretamente un Samsung 850 Pro, podía durar hasta los 2,4 Petabytes de datos escritos, lo que equivale a 2457,6 Terabytes. Por lo tanto, la duración de uno de estos discos depende de cuánto tardes en escribir y reescribir el disco duro hasta llegar a esa cantidad, que seguro que es más de los 3 a 5 años de garantía que suelen ofrecer los fabricantes.

Uno de los inconvenientes de los discos SSD frente a los HDD en durabilidad es que tiene una mayor tasa de fallos. Aun así esto va mejorando generación a generación, y tiene otras ventajas como una mejor resistencia a los golpes. Recuerda, los HDD tienen piezas mecánicas, por lo que un golpe podría propiciar que se rompan o desgasten antes acortando su vida útil.

En cualquier caso y para evitar sorpresas, tanto en los SSD como en unos HDD que pueden durar alrededor de 10 años o más, lo mejor es monitorizar regularmente la salud de tu disco duro. Aquí tienes una colección de herramientas con las que puedes hacerlo, de manera que sepas cuándo podría estar a punto de sufrir un error que ponga en peligro tus datos. (Fernández Y., 2021)

¿Qué disco duro elegir?

Si sueles descargar muchos contenidos de Internet y necesitas grandes cantidades de almacenamiento, o si cuentas con un presupuesto bajo, lo recomendado es que sigas recurriendo a los HDD. También son un buen recurso para los discos duros externos, donde suele primar la capacidad de almacenamiento por encima de la velocidad.

En cambio los SSD son recomendables si quieres tener un ordenador mucho más rápido. De hecho, su velocidad puede hacer que un PC con algunos años vaya mucho más rápido sin tener que invertir en otros componentes. También es recomendado si sueles trabajar en la edición de contenidos multimedia o eres un amante de los videojuegos, ya que los procesos de carga se acelerarán gracias a ellos.

En la mayoría de los casos sin embargo lo recomendable es combinar ambos tipos de disco duro. En una torre doméstica, por ejemplo, puedes utilizar un SSD en el disco C: para instalar allí el sistema operativo y que vaya todo más rápido. Lo acompañas de un HDD como disco secundario y tendrás una unidad perfecta en la que almacenar todos los archivos pesados que tengas en el ordenador. (Fernández Y., 2021)

Memorias: La memoria es uno de los componentes fundamentales para el correcto funcionamiento de nuestra PC, ya que su existencia permite que la computadora pueda arrancar, se procesen los datos, se ejecuten las instrucciones para los distintos programas y demás.

Por otro lado, cuanto mayor es la cantidad de memoria que posea una PC, mayor será el rendimiento y la mejora en la performance del equipo. (Guille V., 2021)

En la figura 15 se observan los tipos de memorias existentes en nuestro computador.

Figura 15. Memorias del computador.

No obstante, una computadora trabaja con cuatro tipos de memorias diferentes, que sirven para realizar diversas funciones. Estas son la memoria RAM, la memoria ROM, la memoria SRAM o caché y la memoria Virtual o de Swap.

Tipos de memoria

La memoria RAM

La más importante es la denominada memoria RAM (Random Access Memory), ya que nuestra computadora no podría funcionar sin su existencia.

En la RAM se guarda distinto tipo de información, desde los procesos temporales como modificaciones de archivos, hasta las instrucciones que posibilitan la ejecución de las aplicaciones que tenemos instaladas en nuestra PC.

Por tal motivo, es utilizada constantemente por el microprocesador, que accede a ella para buscar o guardar temporalmente información referente a los procesos que se realizan en la computadora.

Dentro de las memorias RAM existen distintos tipos de tecnologías que se diferencian principalmente por su velocidad de acceso y su forma física. Entre ellas encontramos las DRAM, SDRAM, RDRAM, entre otras.

Las denominadas DRAM (Dynamyc Random Acces Memory) han sido utilizadas en las computadoras desde los primeros años de la década de los 80’s, y aún en la actualidad continúan utilizándose. Se

trata de uno de los tipos de memorias más económicas, aunque su mayor desventaja está relacionada con la velocidad de proceso, ya que es una de las más lentas, lo que ha llevado a los fabricantes a modificar su tecnología para ofrecer un producto mejor.

En cuanto al tipo de tecnología SDRAM, derivada de la primera, comenzó a comercializarse a finales de la década de los 90's, y gracias a este tipo de memoria se lograron agilizar notablemente los procesos, ya que puede funcionar a la misma velocidad que la motherboard a la que se encuentra incorporada.

Por su parte, la tecnología RDRAM es una de las más costosas debido a su complejidad de fabricación, y sólo se utilizan en procesadores grandes, tales como los Pentim IV y superiores.

Otra de las diferencias entre las distintas memorias RAM se halla en el tipo de módulo del que se trate, que pueden ser SIMM (Single in line Memory Module), DIMM (Double Memory Module) y RIMM (Rambus in line Memory Module), dependiendo de la cantidad de pines que contenga y del tamaño físico del módulo. (Guille V., 2021)

La memoria ROM

Además de la memoria RAM, las computadoras trabajan con la memoria denominada ROM, Read Only Memory, que como su nombre lo indica se trata de una memoria sólo de lectura, ya que la mayoría de estas memorias no pueden ser modificadas debido a que no permiten su escritura.

La memoria ROM viene incorporada a la motherboard y es utilizada por la PC para dar inicio a la BIOS, lo cual es básicamente un programa que posee las instrucciones adecuadas para guiar a la computadora durante el arranque.

Entre sus funciones, la BIOS comienza con el proceso denominado POST (Power On Self Test) durante el cual inspeccionará todo el sistema para corroborar que todos sus componentes funcionan adecuadamente para dar lugar al arranque.

Para ello, la BIOS consulta un registro en el que se halla toda la información referente al hardware que tenemos instalado en nuestra PC, para comprobar que todo se encuentre en orden. Dicho registro es denominado CMOS Setup.

Si bien mencionamos que en muchos casos la memoria ROM no puede ser modificada, en la actualidad gran cantidad de motherboards incorporan nuevos modelos de ROM que permiten su escritura, para que el usuario pueda realizar cambios en la BIOS con el fin de mejorar su funcionamiento. (Guille V., 2021)

La diferencia fundamental que existe entre la memoria RAM y la ROM radica en la velocidad, ya que la ROM al tratarse de un tipo de memorial secuencial necesita recorrer todos los datos hasta hallar la información que está buscando, mientras que la RAM trabaja de manera aleatoria, lo que hace que acceda a la información específica de manera directa.

Este factor hace que la velocidad de la RAM sea notablemente superior. Asimismo, la capacidad de ésta es mayor a la de la memoria ROM, y a diferencia de esta última, la RAM no viene integrada al motherboard, lo que permite que el usuario pueda expandir la cantidad de memoria RAM de su PC. (Guille V., 2021)

La memoria caché

Otro de los tipos de memoria utilizados por las computadoras es la denominada SRAM, más conocida como memoria caché.

Tanto el procesador como el disco rígido y la motherboard poseen su propia memoria caché, que básicamente resguarda distintas direcciones que son utilizadas por la memoria RAM para realizar diferentes funciones, tales como ejecutar programas instalados en la PC.

El proceso que realiza la memoria caché es guardar las ubicaciones en el disco que ocupan los programas que han sido ejecutados, para que cuando vuelvan a ser iniciados el acceso a la aplicación logre ser más rápido. (Guille V., 2021)

Existen tres tipos de caché diferentes

- El caché L1 que se encuentra en el interior del procesador y funciona a la misma velocidad que éste, y en el cual se guardan instrucciones y datos.
- El caché L2 que suelen ser de dos tipos: interno y externo. El primero se encuentra dentro de la motherboard, mientras que el

segundo se halla en el procesador pero de manera externa, lo que lo hace más lento que el caché L1.

- El caché L3 que sólo vienen incorporado a algunos de los microprocesadores más avanzados, lo que resulta en una mayor velocidad de procesos.

La memoria de Swap

En algunas computadoras, sobre todo en aquellas que poseen sistema operativo Microsoft Windows o Linux, también encontraremos la denominada memoria virtual o de Swap.

Este tipo de memoria, que funciona de manera similar a la caché, es creada por Windows o Linux para ser utilizada exclusivamente por el sistema operativo. En el caso de Linux esta denominada memoria swap generalmente está ubicada en una partición diferente del disco, mientras que en el sistema de Microsoft es un archivo dentro del sistema operativo mismo.

En muchas ocasiones la memoria virtual suele producir ciertos problemas que ocasionan que la PC se cuelgue, ya que este tipo de memoria ha sido creada por el sistema dentro del disco rígido y a veces puede llegar a superar la capacidad de proceso.

En la ejecución de programas mediante la memoria virtual, sólo obtendremos como resultado que nuestra PC se vuelva más lenta, ya que le resta velocidad de proceso al disco rígido.

La mejor forma de evitar este inconveniente es expandir la cantidad de memoria RAM de nuestra PC, para que el sistema no necesite de la creación de memoria virtual extra, y por ende relentice los procesos durante nuestro trabajo. (Guille V., 2021)

Tipos de memorias RAM

Memorias DDR

De acuerdo al tipo de placa madre que utilicemos en nuestra PC, ésta estará provista de diferentes tipos de zócalos según su antigüedad, y puede que utilice memoria RAM DDR, DDR2, DDR3 ó DDR4.

Las siglas DDR son utilizadas para abreviar el concepto “Double Data Rate”, cuya definición es memoria de doble tasa de transferencia, y se trata de una serie de módulos que están compuestos por memorias síncronas, llamadas SDRAM, y si bien tienen el mismo tamaño de los DIMM de SDRAM, las DDR-SDRAM poseen mayor cantidad de conectores, ya que mientras la SDRAM normal tiene 168 pines, la DDR-SDRAM posee 184.

Las memorias DDR trabajan transfiriendo datos a través de dos canales diferentes, de manera simultánea y en un mismo ciclo de reloj con una transferencia de un volumen de información de 8 bytes en cada ciclo de reloj. No obstante son compatibles con procesadores más potentes en cuanto a ciclos de reloj.

En lo que respecta a la memoria DDR2 se trata básicamente de la segunda generación de DDR SDRAM, que ha logrado mejorar ciertos aspectos brindando mayor rapidez en los procesos simultáneos.

Al ser una tecnología más moderna, las DDR2 poseen notables diferencias con sus antecesoras, entre las cuales la más significativa tiene que ver con el valor de transferencia mínima, ya que mientras que en las DDR tradicionales es de 1600Mbps, en las DDR2 se duplica a 3200Mbps.

Esto le permite un mayor ancho de banda en los procesos, ya que las memorias DDR2 tienen mayor latencia porque trabajan con 4 bits por ciclo (2 de ida y 2 de vuelta) dentro de un mismo ciclo y bajo la misma frecuencia de una DDR convencional.

Lamentablemente las DDR y las DDR2 no son compatibles, por lo que si tienes una PC cuya motherboard posee zócalos para DDR no podrás utilizar Memorias DDR2, ya que estás últimas tienen 240 pines, lo que permite reducir su voltaje a 1.8V, mientras que las DDR utilizan un voltaje de 2.5V.

La reducción del voltaje en la segunda generación de memorias DDR ha incorporado una gran mejora, debido a que de esta manera se reduce considerablemente el consumo de energía y por ende la generación de calor.

El avance en el desarrollo de la tecnología de este tipo de memorias RAM produjo los módulos DDR3, cuyo fabricante más importante hasta el momento ha sido la empresa Samsung Electronics.

DDR3 incorpora importantes mejoras en el campo de las memorias DDR SDRAM, entre las que se destaca el hecho de que puede transferir

datos a una tasa de reloj efectiva de 800-1600 Mhz, superando en gran medida a las DDR anteriores, ya que las DDR2 tienen una tasa de 533-800 MHz y las DDR de 200-400 MHz.

Esto permite un mayor ancho de banda en los procesos, significativamente notable en el funcionamiento de la PC, además de haber duplicado su latencia a 8 bits, con el fin de aumentar su rendimiento, y duplicar su tasa de transferencia mínima a 6400Mbps, en comparación a las DDR2 que poseen una tasa de 3200Mbps.

Las DDR3 consumen sólo 1.5V, gracias a la implementación de la tecnología de fabricación de 80 nanómetros. Este cambio reduce el consumo de energía y la generación de calor, por lo que aumenta la velocidad en los procesos.

En cuanto al aspecto físico, si bien las DDR3 poseen 240 pines, es decir la misma cantidad que las DDR2, ambos tipos de memorias son incompatibles, ya que los pines han sido ubicados de manera diferente.

Las memorias DDR4 poseen una velocidad de 2.667 Mhz y su tasa de transferencia es de 21.300 Mbps. (Guille V., 2021)

Memorias GDDR

En el mercado, además de las típicas memorias RAM del tipo DDR, también podemos encontrarnos con una variante de la misma, llamada GDDR SDRAM (Graphics Double Data Rate Synchronous Dynamic RAM), la cual es un tipo de memoria que fue diseñada específicamente con el propósito de ser utilizada en el ámbito del renderizado de video,

habitualmente trabajando en equipo con la GPU de nuestra tarjeta gráfica.

Con este tipo de memorias, estaremos en condiciones de crear estructuras gráficas 3D muy complejas, para las cuales necesitamos gran cantidad de memoria. Sin embargo, con las memorias GDDR, que son mucho más rápidas, la cantidad de memoria requerida para estos procesos se reduce, lo que significa menos dinero y espacio, aunque el precio de las memorias GDDR no permite que puedan ser usadas por el usuario promedio con un presupuesto ajustado en sus implementaciones domésticas, ya que son mucho más caras de producir que las DDR, lo que se traduce en un precio mucho mayor. (Guille V., 2021)

Si bien las memorias de tipo GDDR comparte muchas de las características técnicas con las memorias de tipo DDR, lo cierto es que no son completamente iguales. En este sentido, las memorias GDDR, al estar optimizadas para su uso en el renderizado de video, prioriza el ancho de banda, no a la latencia. También las memorias GDDR trabajan respetando el estándar DDR especificado por la JEDEC, por lo cual es capaz de enviar dos bits o 4 por cada ciclo de reloj, si bien en este caso la memoria GDDR está optimizada para lograr frecuencias mayores y un ancho de bus más grande, lo que le permite minimizar el tiempo de acceso a las instrucciones almacenadas en la memoria.

Las memorias GDDR, al igual que las DDR, con el tiempo fueron evolucionando, por lo cual podemos encontrar múltiples variantes. A partir de este punto conoceremos los diferentes tipos de memorias GDDR en el mercado.

GDDR: El primer tipo de GDDR en el mercado. Su frecuencia efectiva de trabajo era de entre 166 y 950 MHz con una latencia de 4 a 6 ns.

GDDR2: En este tipo se mejoró la frecuencia de operación, que llegó a oscilar entre los 533 y 1000 MHz, y podían ofrecer un ancho de banda de entre 8,5 a 16 GB/s.

GDDR3: Especialmente utilizadas por algunos modelos de tarjetas gráficas de ATI y Nvidia, estas memorias pueden operar entre los 166 y 800 MHz.

GDDR4: Reemplazadas rápidamente por las GDDR5, sólo fueron utilizadas por algunos modelos de AMD.

GDDR5: Un tipo de memoria GDDR de los más extendidos en los últimos años. Es utilizada en tarjetas de video de gama media y alta de fabricantes como Nvidia, AMD y Radeon, entre otros. Estas memorias son capaces de ofrecer un ancho de bus cercanos a los 20 GB/s en buses de 32 bits y a los 160 GB/s en buses de 256 bit, pudiendo llegar la frecuencia de operación hasta los 8 Gbps. Cabe destacar que este tipo de memorias se instalan también en consolas de juegos como la Xbox One y la PS4.

GDDR5X: Esta memoria es básicamente una evolución de la tecnología GDDR5 que es utilizada en algunos modelos de tarjetas de video. Ofrecen una frecuencia de operación de 11 Gbps y un ancho de banda de 484 GB/s sobre un bus de 352 bit.

GDDR6: Hasta el momento, es la última versión de memoria GDDR disponible. Son capaces de ofrecer hasta una frecuencia de operación de 14 Gbps con un ancho de banda de 672 GB/s sobre un bus de 384 bit. Este tipo de memorias se utilizan en tarjetas de video de alta gama como las Nvidia Titan RX. (Guille V., 2021)

La memoria ROM: La memoria ROM es quizás el elemento de hardware más importante de computadoras y dispositivos portátiles como celulares, teléfonos inteligentes y tablets, entre muchos otros, ya que en este pequeño componente electrónico se almacena toda la información necesaria para que el dispositivo arranque y pueda cumplir con su función.

El término ROM es una abreviatura del término sajón “Read Only Memory” que en español significa “Memoria de solo lectura”, y como su nombre lo indica, este tipo de memoria almacena información a la cual sólo puede ser accedida, es decir no puede escribirse con nuevos datos, salvo mediante procedimientos especiales como cuando estamos actualizando una BIOS.

Básicamente, una memoria ROM es un chip que en su interior almacena la información necesaria para poder arrancar un dispositivo

electrónico como una computadora o un smartphone, y cuya principal característica es la de tener la capacidad de conservar los datos que contiene aun cuando no existan energía que la alimente, al contrario que las memorias RAM, las cuales si no son energizadas, pierden inmediatamente su contenido.

El término ROM en la actualidad se utiliza por convención, y provienen básicamente de cuando las memorias ROM se desarrollaban y salían de la factoría ya con los datos almacenados en ellas, y no existía ninguna forma de poder escribirlas.

Hoy en día es posible encontrar memorias que cumplen con la misma función de las antiguas ROM pero que sí se pueden escribir, llamadas EPROM y Flash EEPROM, sin embargo escribir en este tipo de memorias es una tarea complicada y que no se puede hacer directamente, salvo con herramientas y procedimientos especiales, que la mayoría de las veces no están al alcance del usuario promedio.

Estas memorias EPROM y Flash EEPROM pueden escribirse multitud de veces, lo que favorece, por ejemplo, que actualizar la BIOS de una computadora pueda ser una tarea frecuente y que no presente problemas. Tal es la adopción de este tipo de memorias para cumplir con el rol de ROM que prácticamente no podremos encontrar en el mercado dispositivos que contenga ROM del tipo más antiguo desde finales de la primera década del siglo XXI.

Para qué sirve la memoria ROM

Las memorias ROM en los dispositivos cumplen con la importante función de almacenar en su interior el código que se necesita para arrancar los diferentes módulos que componen una computadora, es decir todo lo que se requiere para comenzar a trabajar con ella. Asimismo la memoria ROM cumple con la función de iniciar el sistema operativo de la PC en que se encuentra instalado.

Además de utilizarse para la gestión del proceso de arranque de la PC, la memoria ROM se usa para el chequeo inicial del sistema y diversas rutinas de control de dispositivos de entrada y salida.

La capacidad que ofrece la memoria ROM de poder conservar los datos aunque no se encuentre energizada, la hace ideal para el trabajo de iniciar una computadora, ya que los datos almacenados en la memoria ROM no se alterna ni degradan en ausencia de electricidad que la alimente, es decir siempre son los mismos, por lo cual el dispositivo que gestionan siempre se comportará de la misma manera. (Guille V., 2021)

Tipos de memoria ROM

Con el paso de los años, las memorias ROM han ido evolucionando para adaptarse a las nuevas tecnologías. En la actualidad, existen tres tipos básicos de memoria ROM.

ROM (Read Only Memory)

Este tipo de memoria ROM o "Memoria de solo lectura" fue la primera que se desarrolló y fabricó, y la información que debía almacenarse en ella se grababa usando un procedimiento que implicaba la utilización de una placa de silicona y una máscara. Este tipo de memorias ROM ya no se utilizan, siendo reemplazadas por las memorias que se detallan a continuación.

PROM (Programmable Read Only Memory)

Las memorias PROM, también conocidas como "Memoria Programable de Sólo Lectura", vieron la luz a fines de los 70s, y su programación, es decir la carga de los datos que debían contener, se efectuaba quemando unos determinados componentes electrónicos, llamados diodos, con una sobrecarga de tensión mediante un dispositivo conocido como "Programador ROM". Los diodos afectados con la carga corresponden a "0", mientras que los demás corresponden a "1".

EPROM (Erasable Programmable Read Only Memory)

Las memorias del tipo EPROM, también conocidas como "Memoria Programable y Borrable de Sólo Lectura", son básicamente memorias del tipo PROM pero que tienen la particularidad de poder borrarse. El modo de programar estas memorias es a través de rayos de luz ultravioleta que penetran en el circuito a través de una ventana en el encapsulado del chip. En el momento en que el chip se somete a la luz ultravioleta, todos los bit vuelven a su estado"1".

EEPROM (Electrically Erasable Programmable Read Only Memory)

Las memorias EEPROM conocidas también por el nombre "Memoria Programable de Sólo Lectura Borrable Eléctricamente", son, al igual que las memorias PROM, borrables, sin embargo este procedimientos en las memorias EEPROM es más sencillo, ya que se puede realizar mediante una determinada corriente eléctrica.

Cabe destacar que las memorias EEPROM ofrecen una variante llamada Flash EEPROM, que utiliza menos componentes, específicamente un solo transistor, en lugar de los 2 o 3 que utiliza la memoria EPROM. Además ofrece la posibilidad de leer registro por registro, en vez de una lectura de página completa como la memoria EEPROM.

Diferencias entre memorias RAM y ROM

Como sabemos, existen dos tipos de memoria en una computadora, la memoria ROM y la memoria RAM, y cada una de ellas cumple con una función muy distinta. La memoria RAM, o memoria de acceso aleatorio, es aquella memoria a la que accede el sistema operático para buscar los datos que están usando tanto el usuario como el sistema operativo, ya que es un método mucho más rápido que buscarlos en el disco rígido.

La memoria RAM se puede leer y escribir múltiples veces, sin embargo la RAM es temporal, ya que los datos que contiene se borran inmediatamente ante la falta de energía, es decir cuando pierde el suministro eléctrico.

En cambio, la memoria ROM no es afectada por el suministro eléctrico, lo que convierte a este tipo de memoria en el medio ideal para almacenar los datos necesarios para que un dispositivo funcione. Además, la condición de no ser escribible, por lo menos por los medios habituales que tiene disponible el usuario promedio, garantiza que mantendrá los datos que contiene en cualquier situación, por lo cual el dispositivo siempre encenderá y seguirá la misma rutina.

La memoria caché

La memoria caché nació cuando se descubrió que las memorias ya no eran capaces de acompañar a la velocidad del procesador, haciendo que muchas veces este último se quedara "esperando" por los datos que debía entregar la memoria RAM para poder concluir sus tareas, perdiendo mucho rendimiento.

Si en la época del 386, año 1991, la velocidad de las memorias ya era un factor limitante, imagina este problema hoy, con los procesadores que tenemos actualmente.

Para solucionar este problema, se comenzó a usar la memoria caché, un tipo ultra-rápido de memoria que sirve para almacenar los datos que son más frecuentemente utilizados por el procesador, evitando, la mayoría de las veces, tener que recurrir a la comparativamente lenta memoria RAM.

Sin la memoria caché, la performance del sistema estaría limitada a la velocidad de la memoria, pudiendo caer hasta un 95%.

Los tipos de memoria caché

Se utilizan dos tipos de memoria caché, llamados caché primario, o caché L1 (level 1), y caché secundario, o caché L2 (level 2). La memoria caché primaria está insertada en el mismo procesador y es tan rápida como para acompañarlo en velocidad. Siempre que un nuevo procesador es desarrollado, es preciso desarrollar también un tipo más rápido de memoria caché para acompañarlo. Como este tipo de memoria es extremadamente cara (llega a ser centenares de veces más cara que la memoria RAM convencional) se usa sólo una pequeña cantidad de ella. Para complementar, se utiliza también un tipo de memoria caché un poco más lenta, la cual se llama caché secundario, que por ser mucho más barata, permite usar mayor cantidad.

¿Cómo instalar memoria caché?

Primero, debes asegurarte que la placa madre permita la instalación de memoria caché. Las placas madre que permiten la instalación, poseen un socket llamado COAST donde se coloca el módulo de memoria caché. Generalmente se necesita cambiar los jumpers de configuración del tamaño de la memoria caché. La posición correcta de los jumpers se deberá consultar en el manual de la placa. Si luego de esta configuración la PC no enciende, significa que el módulo de memoria caché está fallado o es incompatible con la placa madre. En este caso, el módulo debe ser cambiado. Cuando esté todo funcionando, se deberá habilitar el caché de memoria en la BIOS de la PC.

¿Chequear la existencia de memoria caché en la PC?

Hay varios programas para este fin. Uno de ellos se llama PC-Config, es shareware y puede ser bajado gratis en Internet. Además de probar el caché, este programa no brinda información importante sobre la PC, tales como el tipo de memoria instalada y el tipo de chipset

- Grabadoras – lectoras:
- Pantalla táctil:

Pizarras digitales: Una Pizarra Digital Interactiva se define como un sistema tecnológico integrado por un ordenador, un video-proyector y un dispositivo de control de puntero que permite proyectar en una superficie interactiva contenidos digitales en un formato adecuado para que puedan ser visualizados en grupo. Se puede interactuar directamente sobre la superficie de proyección, permitiendo escribir directamente sobre ella y controlar los programas informáticos con un puntero o incluso con los dedos.

Proporciona la posibilidad de interactuar a distancia sobre las imágenes proyectadas aportando, entre otros, los siguientes beneficios al modelo de enseñanza y aprendizaje:

- Facilita la participación de los alumnos. Aumenta su interés por presentar materiales y trabajos, ya que por todos es conocida la facilidad y rápida adaptación de los alumnos a las nuevas tecnologías. Este es un factor de motivación importante.

- Aumenta la atención del estudiante en la materia. Se presentan formatos atractivos y visuales, gráficos que favorecen este aspecto.
- Existe mayor interacción permitiendo visualizar conceptos difíciles y complejos.
- Potencia el aprendizaje visual de los alumnos.
- El profesor dispone de más recursos de aprendizaje y tiene cubiertas las necesidades especiales de visión, audición etc.

El funcionamiento de la pizarra interactiva consiste básicamente en que la pizarra transmite al ordenador las instrucciones correspondientes y el ordenador envía al proyector de video las instrucciones para que éste proyecte sobre la pizarra el resultado, permitiendo que la persona que maneja el equipo pueda ver en tiempo real lo que hace sobre la pizarra y cómo lo interpreta el ordenador.

El uso de las Pizarras Digitales Interactivas proporciona grandes beneficios a los docentes, ya que se acomoda a los diferentes modos de enseñanza y permite combinar el trabajo individual con el trabajo en grupo. Se trata de un recurso que despierta el interés del docente por las tecnologías TIC, animando a su desarrollo profesional con nuevas estrategias pedagógicas.

Para los alumnos se presenta como un recurso atractivo, que aumenta su motivación, presentado con vivos colores y figuras, utilizando videos, simulaciones e imágenes con las que es posible interactuar.

Para aquellos alumnos con necesidades especiales, las pizarras digitales proporcionan interesantes beneficios en el proceso de aprendizaje, como la utilización simultánea del lenguaje de signos, el aumento del tamaño de los textos e imágenes, la posibilidad de manipular objetos y símbolos, etc.

Sin duda, las Pizarras Digitales Interactivas ofrecen grandes posibilidades de avance en el mundo de la enseñanza, acercando a toda la comunidad educativa a las nuevas tecnologías. (CGA, 2016)

En la figura 16 se muestra una pizarra digital:

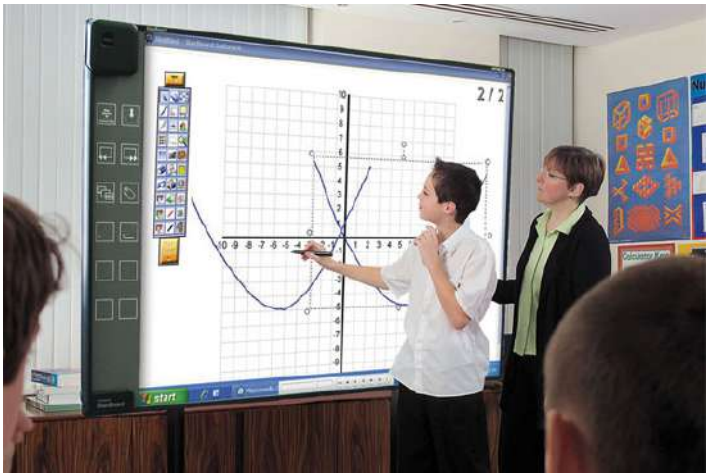


Figura 16. Pizarra digital

Periféricos de comunicación

Son aquellos que facilitan la interacción entre dos o más ordenadores, o entre una computadora y otro periférico externo a esta. Como ejemplos tenemos:

- Fax-modem
- Tarjeta de red
- Concentrador
- Tarjeta inalámbrica
- Bluetooth

Software

Software son los dispositivos lógicos que componen el ordenador. Software es todo el conjunto intangible de datos y programas de la computadora. Entre las partes que lo forman están: Sistemas operativos y programas.

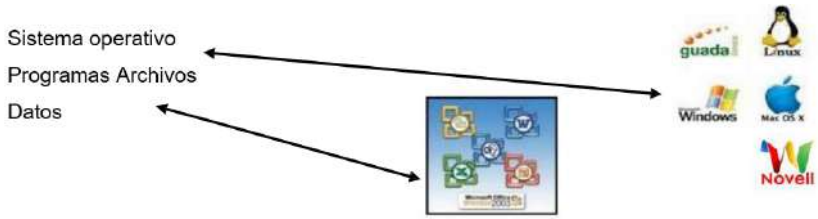


Figura 17. Software y su clasificación

Sistemas operativos

Un sistema operativo es un conjunto de programas que permite manejar la memoria, disco, medios de almacenamiento de información y los diferentes periféricos o recursos de nuestra computadora, como son el teclado, el mouse, la impresora, la placa de red, entre otros.

Los periféricos utilizan un driver o controlador y son desarrollados por los fabricantes de cada equipo. Encontramos diferentes sistemas operativos como Windows, Linux, MAS OS, en sus diferentes versiones. También los teléfonos y tablets poseen un sistema operativo.

Dentro de las tareas que realiza el sistema operativo, en particular, se ocupa de gestionar la memoria de nuestro sistema y la carga de los diferentes programas, para ello cada programa tiene una prioridad o jerarquía y en función de la misma contará con los recursos de nuestro sistema por más tiempo que un programa de menor prioridad.

El sistema operativo se ocupa también de correr procesos. Llamamos proceso a la carga en memoria de nuestro programa, si no está cargado en memoria nuestro programa simplemente "no corre". (Tecnología Inclusiva, 2021)

Clasificación de los sistemas operativos

- Sistemas operativos de software libres
- Sistemas operativos de software propietario, privado o de paga

En la figura 18 se representa el significado de cada uno de los software mencionados:

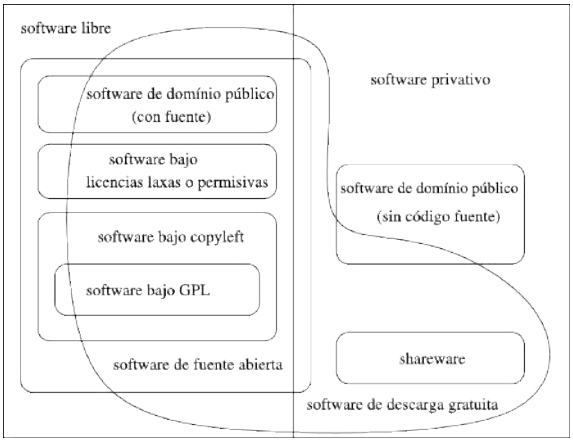


Figura 18. Software libre y software privado

Sistemas operativos de software libre

Los sistemas operativos libres son aquellos sistemas que permiten a las personas operar sus computadoras sin restricciones de uso, desarrollo y mejoramiento. Un sistema operativo es el conjunto de programas informáticos que permite la gestión efectiva del hardware (el equipo tangible) y el software (intangibles) del computador.

Es una especie de macro programa que permite al usuario utilizar su computador para realizar las tareas que desea. Un sistema operativo permite administrar y escalar tareas. Mantiene la integridad del sistema. (Yánez D., 2020)

Cuando se habla de sistema operativo libre se hace referencia a sistemas que permiten estas libertades específicas:

- Usar el programa con cualquier propósito.
- Estudiar el funcionamiento del programa y hacerle adaptaciones.
- Distribuir copias.
- Mejorar el programa y hacer públicas esas mejoras.

Se considera que fue Richard Stallman quien inició en 1984 el movimiento mundial del software libre con su proyecto GNU.

En 1985 Stallman creó la Fundación del Software Libre (FSF) y desarrolló la Licencia pública general de GNU (GNU GPL), que ofrecía un marco legal a la difusión libre del software. En el año 1991 Linus Torvalds desarrolló el núcleo para los sistemas operativos GNU/Linux.

El desarrollo y la distribución de este tipo de software ha sido objeto de polémica por las implicaciones económicas que puede tener. (Yánez D., 2020)

Sistemas operativos linux

Es uno de los sistemas operativos libres más populares dentro del área informática a nivel mundial y que se encuentra en los niveles más altos de popularidad por su ambiente de seguridad y familiaridad con los administradores de sistemas operativos. En la figura 19 se muestran algunos de estos sistemas operativos:

Figura 19. Sistemas operativos Linux

Sistemas operativos de software propietario, privado o de paga

Fueron creados por empresas para su uso comercial. Tales empresas son sus propietarias y cobrar por utilizarlo y distribuirlo y aquellos que lo diseñaron y crearon ocultan su código original para evitar que se altere.

Sólo la empresa propietaria tiene derecho a modificar el software. Además, queda prohibida su distribución sin un previo pago. Windows es un sistema operativo propietario de la empresa Microsoft. Otro ejemplo de sistema operativo propietario es el Mac OS X, propiedad de Apple.

Sistemas operativos windows

Windows es un sistema operativo, es decir, un programa de software que admite funciones básicas, como la administración de archivos y la ejecución de aplicaciones, y que usa dispositivos periféricos, como la impresora, el monitor, el teclado y el mouse. En el pasado, Windows podía considerarse como un software que residía solo en tu dispositivo. Ahora con Windows 10, las partes importantes de Windows se basan en la nube e interactúan con los servicios en línea. En la figura 20 podemos observar los diferentes sistemas operativos Windows:

Figura 20. Sistemas Operativos Windows

Sistemas operativos mac

MacOS es un sistema operativo diseñado por Apple que está instalado en todos los equipos creados por la compañía Apple Inc., y son conocidos generalmente como Mac.

El sistema operativo es aquello que te permite realizar todas las tareas en un computador, como jugar, escuchar música, ver y editar imágenes, entre muchas otras cosas.

A diferencia del sistema operativo Windows que puede ser usado en equipos de diferentes fabricantes (DELL, Lenovo, etc.), macOS está diseñado específicamente para computadores fabricados por Apple. Esto implica que el hardware y el software son totalmente compatibles, por este motivo el ordenador tiene un mejor funcionamiento y puede procesar información más rápido. (GCF, 2021). En la figura 21 se observa los diferentes tipos de sistemas operativos MacOS:

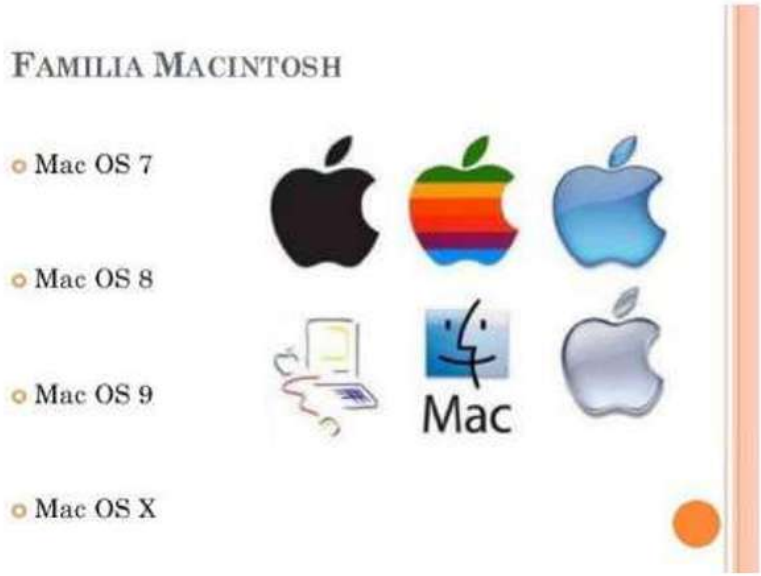


Figura 21. Sistemas Operativos MacOS

Sistema operativo novell

Desde 1.983, Novell es el líder del mercado en los sistemas operativos de Red. Desde su aparición hasta la actualidad ha sido mejorado permitiendo; una instalación mucho más sencilla, mayor potencia y seguridad, pero sobre todo flexibilidad. Novell permite conectarse a prácticamente cualquier sistema, posibilitando la creación de sistemas distribuidos.

El éxito de Novell se debe sobre todo a ser uno de los pocos sistemas operativos de red con soporte para MS-DOS. Esto ha permitido que sin grandes modificaciones, todo el software basado en MS-DOS esté ahora disponible en sus correspondientes versiones de red.

Novell no es un sistema operativo barato. Pero los recursos que proporciona han permitido que sea uno de los estándares dentro del mundo de la informática.

El futuro de Novell apunta a la gestión de sistemas operativos de red soportando el estándar de OSI de ISO, además de la integración de las distintas topologías y cableados bajo un mismo sistema operativo. (Carrasco T., 2016).

Inicialmente, Novell, realizó un sistema operativo que se llamaba Netware, y estaba fabricado para el procesador Motorola MC 68000. En 1983 aparecerá el XT y el sistema operativo MS-DOS, y en ese momento, Novell reescribe el sistema operativo. Conforme ha avanzado la informática ha avanzado el sistema operativo Novell

Netware. En la figura 22 se observa los diferentes tipos de sistemas operativos Novell:

1983	NetWare	Primer software LAN basado en la arquitectura cliente/servidor.
1986	Advanced NetWare	Disponia de un soporto mas amplio para hardware de red.
	Advanced NetWare 286	Para equipos basados en el procesador 80286 de Intel. Tenia la capacidad multitarea.
1989	NetWare 386 v 3.0	SO de 32 bits para el procesador 80386 de Intel
1990	NetWare v 3.1	Mayor rendimiento que las versiones anteriores
1991	NetWare v 3.11	Soportaba servicios de archivos de DOS, Macintosh, Windows y UNIX
	NetWare LITE	Sistema operativo de punto a punto, basado en DOS
1994	PERSONAL NetWare	Implementado por plataformas que no solo se dedican a los servicios de archivos
1998	NetWare v 5.0	Promoviendo herramientas para unir plataformas en una sola herramienta
2001	NetWare v 6.0	De 64 bits

Figura 22. Sistemas operativos Novell

Recientemente Novell acaba de anunciar la muerte de NetWare en favor de LINUX. Novell compró hace varios años la empresa alemana SUSE AG con el fin de posicionarse en el emergente mercado de LINUX. Durante estos años Novell estuvo vendiendo y desarrollando el producto Open Enterprise Server, el cual, básicamente es un servidor de red.

Novell ha decidido no desarrollar más productos basados en NetWare y por lo tanto se puede concluir que NetWare ha muerto. No obstante, Novell sigue a delante como empresa al ser una de las veinte empresas de informática más grandes del mundo centrada en LINUX. La

ventaja del servidor OES de Novell basado en LINUX es que todos los productos que existían para Novell: edirectory, iprint, groupwise, etc… están migrados a esta plataforma.

Se calcula que actualmente todavía hay 90 millones de personas utilizando NetWare en el mundo.

Programas

Un programa es un conjunto de pasos lógicos escritos en un lenguaje de programación que nos permite realizar una tarea específica. El programa suele contar con una interfaz de usuario, es decir, un medio visual mediante el cual interactuamos con la aplicación. Algunos ejemplos son la calculadora, el navegador de Internet, un teclado en pantalla para el celular, etc.

Hoy encontramos programas o aplicaciones que pueden ejecutarse en una computadora, notebooks, tablets y celulares. Estas aplicaciones pueden ser escritas en diferentes lenguajes de programación. Como ejemplos encontramos C, Java, PHP, Python, entre otros. Estos programas corren sobre un sistema operativo, por ejemplo, Windows, Linux, Mac OS y Android entre otros.

Los programas para poder correr se deben cargar en la memoria, el responsable de esta tarea es el sistema operativo. Un programa puede diseñarse para una computadora o para otro tipo de dispositivos pero su programación suele realizarse en una computadora utilizando un entorno de desarrollo integrado (en ingles IDE). Este programa

cuenta con herramientas que permiten convertir nuestro código en un programa funcional. Estás herramientas son el compilador, el “linker” y el depurador (debugger).

Existen otras herramientas que facilitan nuestro trabajo, por ejemplo para documentar o llevar registro de lo que hacemos (doxygen), para compartir nuestro trabajo y realizarlo en forma colaborativa (SVN / GIT). De esta forma, un equipo de trabajo puede desarrollar diferentes partes de un programa y luego integrarlas en forma más simple.

En la figura 23 se observa varios de los programas que se utilizan dentro del área de software como son los lenguajes de programación:

Figura 23. Lenguajes de programación.

También dentro de los programas podemos encontrar software libre y software propietario, en la figura 24 observamos programas libres dentro de una suite de ofimática:

Figura 24. Suite de ofimática libre

En la figura 25 observamos los programas dentro de la suite ofimática propietario:

Figura 25. Suite de ofimática propietario

Firmware

El firmware, también conocido como soporte lógico inalterable, es el programa básico que controla los circuitos electrónicos de cualquier dispositivo. Es parte del hardware porque siempre está integrado en la electrónica, pero no deja de ser un programa informático, por lo que también es software. Entonces se habla de: BIOS, SETUP, CMOS.

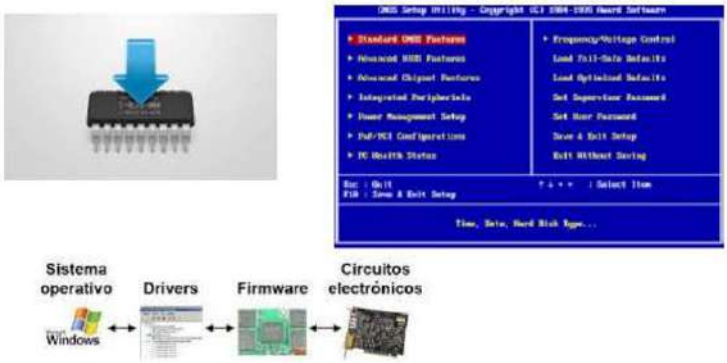


Figura 26. BIOS del computador

El semiconductor complementario de óxido metálico o complementary metal-oxide-semiconductor (CMOS), es una pequeña cantidad de memoria en una placa base del equipo que almacena la configuración del Sistema básico de entrada/salida o Basic Input-Output System (BIOS). El BIOS es el software almacenado en el chip de memoria en la placa base. Instruye a la computadora sobre cómo realizar una serie de funciones básicas como el arranque y el control del teclado. El BIOS también se usa para identificar y configurar el hardware en el equipo.

Si no hay problemas de arranque o visualización, borrar el CMOS puede ayudar a recuperar las placas porque eso restaura la configuración predeterminada del BIOS.

Explicamos un poco más esto, así: En primer lugar, la CMOS es la memoria donde se almacena la configuración de la BIOS, como la hora y los ajustes. Por este motivo, cuando quitamos la pila CMOS, se resetea la BIOS, perdiéndose toda la información. Frecuentemente, se relaciona la CMOS con la memoria RAM, ya que es un tipo de chip de memoria que guarda información.

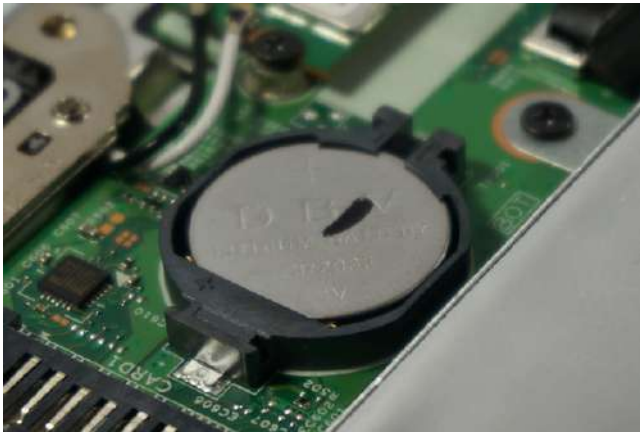


Figura 27. CMOS del Computador.

Sin embargo, los chips RAM corrientes pierden la información almacenada cuando se apaga el PC. La CMOS la retiene porque su chip está continuamente alimentado por su pila. Así que podemos decir que la CMOS es una memoria no volátil.

En segundo lugar, la BIOS es un chip que está en la placa base y se trata de memoria ROM (de sólo lectura). Dicho esto, tiene que ver con el software que lee la información que la CMOS guarda. Mediante la BIOS podemos configurar muchísimos aspectos de nuestro sistema: overclock, cambiar la hora, el idioma de la BIOS, la prioridad de arranque, actualizar el firmware, etc.

Para acceder a la BIOS tenemos que pulsar una tecla cuando encendemos el PC y el logo del fabricante de la placa base aparece. En ese caso, accedemos a la BIOS y podremos realizar ajustes, los cuales se guardarán en la CMOS.

El SETUP es una interface para ajustar manualmente varios parámetros del BIOS. Consta de varias pestañas por las que podemos desplazarnos usando el teclado (no el ratón). Cada pestaña muestra información o parámetros que se pueden ajustar.

Internet y redes de computadores

Internet

Proviene de "INTERconnected NETworks" ("redes interconectadas"): básicamente se trata de millones de computadoras conectadas entre sí en una red mundial. Es una red global en la que se unen todas las redes que utilizan protocolos y que son compatibles entre sí. Es una "red de redes" en donde están interconectadas ordenadores de todo el mundo para compartir información y recursos.

Redes de computadores

Es un conjunto de computadores conectados entre sí, para poder compartir datos y recursos entre ellos.

Clasificación de las redes de computadores

Red de área personal (pan)

Se trata del tipo de red informática más pequeño y básico. Una red PAN se compone de un módem, un ordenador o dos, teléfonos, impresoras, tablets, etc.

PAN es la red que normalmente nos encontramos en oficinas pequeñas o residencias particulares. Son administradas por una sola persona o por una empresa desde un único dispositivo.

Red de área local (lan)

Seguro que sabes lo que son las LAN. Se trata de un tipo de red muy común y muy usada que conecta un grupo de ordenadores o dispositivos ubicados en una misma estancia para compartir información y recursos.

Red de área local inalámbrica (wlan)

Al funcionar como una LAN, las WLAN utilizan tecnología de red inalámbrica, como Wi-Fi. Sus usos generales son los mismos que los de una red LAN, la única diferencia es que la WLAN no depende de cables físicos para conectarse a la red.

Red de área del campus (CAN)

Más grandes que las LAN, pero más pequeñas que las que veremos a continuación, estos tipos de redes se ven típicamente en universidades.

Se suelen distribuir en varios edificios que están cerca unos de otros para que los usuarios puedan compartir recursos.

Red de área metropolitana (MAN)

Son redes más grandes y que abarcan más que las LAN y que las CAN. Este tipo de redes abarcan un área geográfica determinada, normalmente un pueblo o ciudad.

Su mantenimiento e instalación corre a cargo de una empresa o del propio ayuntamiento.

Red de área amplia (WAN)

Las redes WAN son las que conectan los ordenadores que se encuentran a distancias físicas considerables.

Permiten que los dispositivos se conecten de forma remota entre sí a través de una gran red para comunicarse incluso cuando están a kilómetros de distancia.

Internet es el ejemplo más básico de una WAN, que conecta todos los dispositivos con acceso a él a lo largo y ancho del mundo.

Red de área de almacenamiento (SAN)

Las SAN son redes informáticas de alta velocidad que conectan grupos compartidos de dispositivos de almacenamiento a varios servidores.

Estos tipos de redes no dependen de una LAN o WAN. Estas redes alejan los recursos de almacenamiento de la red y los colocan en su propia red de alto rendimiento.

Red de área local óptica pasiva (POLAN)

Como alternativa a las LAN tradicionales basadas en conmutadores, la tecnología POLAN se integra en el cableado para superar las preocupaciones sobre la compatibilidad con los protocolos Ethernet tradicionales.

POLAN es una arquitectura de LAN de punto a multipunto que emplea divisores ópticos para multiplicar la señal de una hebra de fibra óptica monomodo para repartirla entre usuarios y dispositivos.

Red privada empresarial (EPN)

Este tipo de redes están construidas y son propiedad de empresas que desean conectar de forma segura sus diversas ubicaciones para compartir recursos informáticos.

Red privada virtual (VPN)

Una VPN permite a sus usuarios enviar y recibir datos como si sus dispositivos estuvieran conectados a la red privada, incluso si no lo están.

De esta forma, a través de una conexión virtual, los usuarios pueden acceder a una red privada de forma remota.

INTRODUCCIÓN A LA PROGRAMACIÓN

Capítulo 2

INTRODUCCIÓN A LA PROGRAMACIÓN

Algoritmo

En informática, un algoritmo es una secuencia de instrucciones secuenciales, gracias al cual pueden llevarse a cabo ciertos procesos y darse respuesta a determinadas necesidades o decisiones. Se trata de conjuntos ordenados y finitos de pasos, que nos permiten resolver un problema o tomar una decisión.

Los algoritmos no tienen que ver con los lenguajes de programación, dado que un mismo algoritmo o diagrama de flujo puede representarse en diversos lenguajes de programación, es decir, se trata de un ordenamiento previo a la programación.

Visto así, un programa no es otra cosa que una serie compleja de algoritmos ordenados y codificados mediante un lenguaje de programación para su posterior ejecución en un computador.

Los algoritmos también son frecuentes en la matemática y la lógica, y son la base de la fabricación de manuales de usuario, folletos de instrucciones, etc. Su nombre proviene del latín algoritmus y éste apellido del matemático persa Al-Juarismi. Uno de los algoritmos más conocidos de la matemática es el atribuido a Euclides, para obtener el máximo común divisor de dos enteros positivos, o el llamado “método de Gauss” para resolver sistemas de ecuaciones lineales. (Editorial Etecé, 2021)

Partes de un algoritmo

Todo algoritmo debe constar de las siguientes partes:

- Input o entrada. El ingreso de los datos que el algoritmo necesita para operar.
- Proceso. Se trata de la operación lógica formal que el algoritmo emprenderá con lo recibido del input.
- Output o salida. Los resultados obtenidos del proceso sobre el input, una vez terminada la ejecución del algoritmo. (Editorial Etecé, 2021)

¿Para qué sirve un algoritmo?

Dicho muy llanamente, un algoritmo sirve para resolver paso a paso un problema. Se trata de una serie de instrucciones ordenadas y secuenciadas para guiar un proceso determinado.

En las Ciencias de la computación, no obstante, los algoritmos constituyen el esqueleto de los procesos que luego se codificarán y programarán para que sean realizados por el computador. (Editorial Etecé, 2021)

Tipos de algoritmos

Existen cuatro tipos de algoritmos en informática:

- Algoritmos computacionales. Un algoritmo cuya resolución depende del cálculo, y que puede ser desarrollado por una calculadora o computadora sin dificultades.
- Algoritmos no computacionales. Aquellos que no requieren de los procesos de un computador para resolverse, o cuyos pasos

son exclusivos para la resolución por parte de un ser humano.

- Algoritmos cualitativos. Se trata de un algoritmo en cuya resolución no intervienen cálculos numéricos, sino secuencias lógicas y/o formales.
- Algoritmos cuantitativos. Todo lo contrario, es un algoritmo que depende de cálculos matemáticos para dar con su resolución. (Editorial Etecé, 2021)

Características de los algoritmos

Un algoritmo debe ofrecer un resultado en base a sus funciones.

Los algoritmos presentan las siguientes características:

- Secuenciales. Los algoritmos operan en secuencia, debe procesarse uno a la vez.
- Precisos. Los algoritmos han de ser precisos en su abordaje del tema, es decir, no pueden ser ambiguos o subjetivos.
- Ordenados. Los algoritmos se deben establecer en la secuencia precisa y exacta para que su lectura tenga sentido y se resuelva el problema.
- Finitos. Toda secuencia de algoritmos ha de tener un fin determinado, no puede prolongarse hasta el infinito.
- Concretos. Todo algoritmo debe ofrecer un resultado en base a las funciones que cumple.
- Definidos. Un mismo algoritmo ante los mismos elementos de entrada (input) debe dar siempre los mismos resultados. (Editorial Etecé, 2021)

Ejemplos de algoritmos

Un par de ejemplos posibles de algoritmo son: (Editorial Etecé, 2021

Algoritmo para elegir unos zapatos de fiesta:

1. INICIO
2. Entrar a la tienda y buscar la sección de zapatos de caballero.
3. Tomar un par de zapatos.
4. ¿Son zapatos de fiesta?
5. SI: (ir al paso 5) – NO: (volver al paso 3)
6. ¿Hay de la talla adecuada?
7. SI: (ir al paso 6) – NO: (volver al paso 3)
8. ¿El precio es pagable?
9. SI: (ir al paso 7) – NO: (volver al paso 3)
10. Comprar el par de zapatos elegido.
11. FIN

Algoritmo para calcular el área de un triángulo rectángulo:

1. INICIO
2. Hallar las medidas de la base (b) y altura (h)
3. Multiplicar: base por altura (b x h)
4. Dividir entre 2 el resultado (b x h) / 2
5. FIN

¿Qué es una constante?

Es un dato cuyo valor no puede cambiar durante la ejecución del programa. Recibe un valor en el momento de la compilación y este permanece inalterado durante todo el programa.

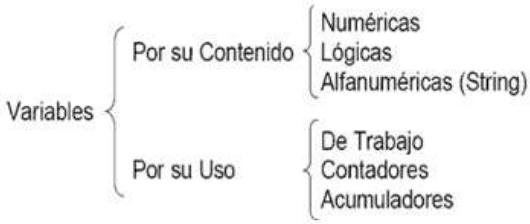
Ejemplo: pi, IVA, % Interés, etc.

¿Qué es una variable?

En programación, una variable es un espacio reservado en la memoria de nuestro ordenador para almacenar un dato que puede ser usado o modificado tantas veces como se desee. Además, las variables deben tener un nombre asociado al dato que almacenan. Es decir, podemos tener una variable llamada Nombre para almacenar el nombre (dato) de una persona, una variable llamada Edad para almacenar la edad (dato), etc.

*Ejemplo: sueldo = 380; Sueldo = sueldo + Horas extras *20;*

Tipos de variables



- Cada variable tiene un único nombre el cual no puede ser cambiado
- Dos o más variables pueden tener el mismo contenido, pero no el mismo nombre
- El nombre de una variable comenzará siempre por una letra,

Ejemplos de variables por su contenido – numéricas, se muestran en la tabla 2:

Tabla 2. Variables numéricas

	EJEMPLO1	EJEMPLO2
ENTERO (INT)	X= 2	C1 = 20000
REAL (FLOAT)	W = 2.12	W1 = 2350.20

Ejemplos de variables por su contenido – lógicas, se muestran en la tabla 3:

Tabla 3. Variables lógicas

X	Y	AND (^)	OR(v)	NOT(~)
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	0

2^2 = 4 -> Cuando tenemos 2 variables

Y si tenemos 3 variables, entonces: 2^3 = 8

Ejemplo de variables por su contenido – alfanuméricas se muestran en la tabla 4:

Tabla 4. Variables alfanuméricas

	EJEMPLO1	EJEMPLO2
CHAR	'1'	'A'
STRING	"1"	"RIOBAMBA"
INTEGER	1	10

Ejemplo de variables por uso – de trabajo, se muestran en la tabla 2.4 y tabla 5:

Tabla 5. Variables de trabajo - vectores

VECTOR

VEC1	0	1	2	3	4
	100	2	-4	500	12

X = VEC1(4)
X tendrá el valor de 12

Tabla 6. Variables de trabajo - matrices

MATRIZ

MAT1	0	1	2	3
0	10	30	12	4
1	16	1000	-124	-1
2	2	4	78	-150

Z= MAT1(1,2)
Z tendrá el valor de -124

Ejemplo de variables por su uso – contadores, se muestran en la tabla 7:

Tabla 7. Variables contadores

Contadores	Ejemplo 1	Ejemplo 2	Ejemplo3
Incremento	I=1 I=i+1 La variable i tendrá el valor de 2	J=0 J++ La variable j tendrá el valor de 1	K=2 ++K La variable k luego de ejecutar el proceso tendrá el valor de 3
Decremento	I=1 I=i-1 La variable i tendrá el valor de 1	J=0 J-- La variable j tendrá el valor de -1	K=2 --K La variable k luego de ejecutar el proceso tendrá el valor de 1

Nota.- Cuando ustedes. se encuentren con ++ se tiene que sumar un 1 a la variable. Y cuando se encuentren con – se tiene que restar un 1 a la variable.

Ejemplo de variables por su uso – acumuladores, se muestran en la tabla 8:

Tabla 8. Variables acumuladores

Acumuladores	Ejemplo 1	Ejemplo 2	Observación
Suma	Sum = 0 P = 20 Sum = sum + p La variable sum tendrá el valor de 20.	Acumulador para la suma de los n primeros números pares.	No olvidar que el acumulador de suma debe empezar en 0.
Multiplicación	Mul = 1 R = 4 Mul = mul*r La variable mul tendrá el valor de 4.	El factorial de un número. Este es un gran ejemplo de acumulador de multiplicación.	No olvidar que el acumulador de multiplicación debe empezar en 1.

Partes fundamentales en la vida de una variable.

- Declaración
- Iniciación
- Utilización

Estas son las partes fundamentales en la vida de una variable, primero se debe declararla para utilizarla, aunque en algunos lenguajes de programación como lenguaje M de Matlab y lenguaje R de RStudio no se las declare, en el resto de lenguajes si se los debe hacer.

Una variable se debe inicializarla para que pueda saber que valor tiene, ahora esto ocurre cuando usted pide al usuario que ingrese un valor por ejemplo, o también cuando debe iniciar un variable contador o un variable acumulador.

Finalmente la variable debe ser usada o utilizada ya sea en alguna operación dentro de una expresión algorítmica o como fórmula matemática.

Tipos de Datos

En la figura 28 se observa los tipos de datos a utilizar de los software como son PSeInt y DFD dentro de la asignatura fundamentos de programación tanto de la carrera de Estadística como en Ingeniería Ambiental de la Facultad de Ciencias de la Escuela Superior Politécnica de Chimborazo.

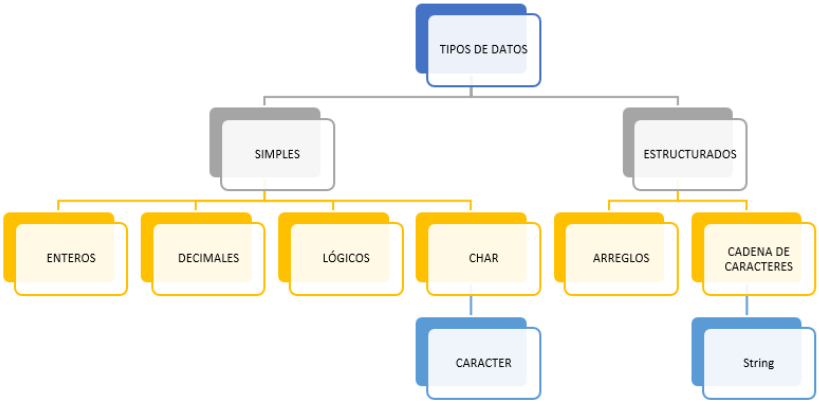


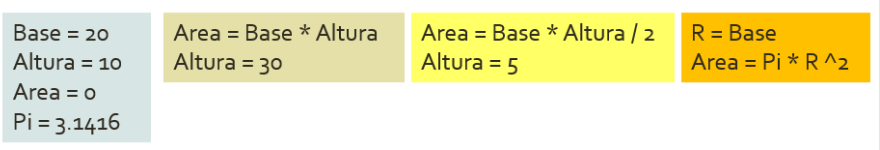
Figura 28. Tipos de datos

¿Qué diferencias hay entre una constante y una variable?

Constantes y variables son dos elementos esenciales en cualquier lenguaje de programación, puesto que ambos almacenan los datos que harán funcionar el programa.

La diferencia fundamental entre ambos, es que en el caso de las constantes, esos datos almacenados no podrán cambiar durante la ejecución del programa. En cambio, las variables podrán cambiar su contenido tantas veces sea necesario.

Ejemplos:



Comparativa entre variables y constantes en programación

En la tabla 9 se muestra una comparativa entre variables y constantes.

Tabla 9. Comparativa entre variables y constantes

Variables	Constantes
El valor de una variable puede cambiar durante la ejecución del programa	Las constantes no cambian sus valores durante la ejecución del programa
Se almacenan en la memoria principal del computador	Se almacenan en la memoria principal del computador

Habitualmente difieren en la nomenclatura. Se pueden escribir como se desee, siempre y cuando inicie con carácter.	En cambio, las constantes, en PSeInt y en DFD, se escriben en mayúsculas generalmente.

Operadores

Los operadores son aquellos símbolos especiales que se utilizan en algoritmos, diagramas de flujo y el lenguaje de programación e indican al compilador que debe realizarse una operación matemática o lógica que puede ayudar a resolver desde problemas financieros hasta problemas algebraicos.

Tipos de operadores

En la figura 29 se puede observar los tipos de operadores que vamos a usar dentro de los algoritmos y diagramas de flujo.

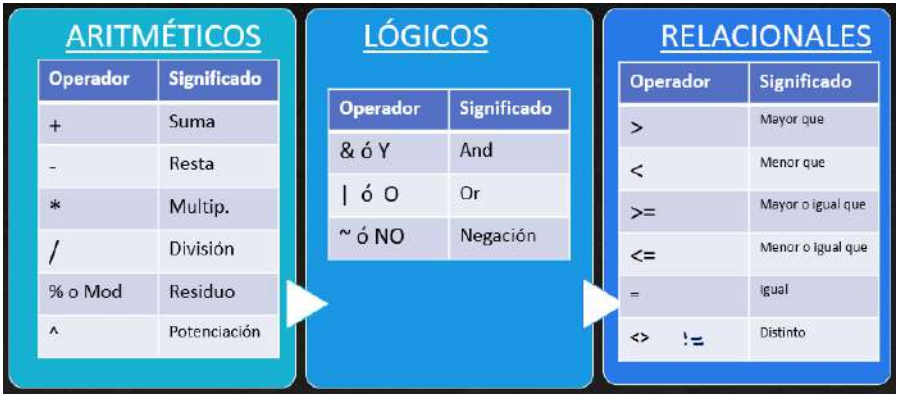


Figura 29. Tipos de operadores

Expresiones

En programación una expresión es una fórmula aritmética que permite calcular un valor, cuando se construye la formula se debe tener en cuenta la jerarquía de operadores.

Una expresión es una secuencia de operadores y operandos que describe un cálculo. Normalmente una expresión se evalúa en tiempo de ejecución.

Operadores

Un operador es el símbolo que determina el tipo de operación o relación que habrá que establecerse entre los operandos de una expresión para alcanzar un resultado.

Operando

Un operando es una de las entradas (argumentos o variables o valores) de un operador. También se les conoce como elementos a los que se aplica una operación.

Prioridad de los operadores. En la tabla 10 se puede observar cómo se debe resolver un ejercicio y que operador primero ejecutar.

Tabla 10. Prioridad de los operadores

Prioridad	Operador	Descripción
1	()	Paréntesis
2	^, √, mod(%), div(//)	Exponenciación, radicación, Módulo, Cociente

3	*, /	Multiplicación, División
4	+, -	Suma y resta

Elementos de una expresión: Se puede observar en la figura 30 y son:

- Operandos
- Operadores

Un operador es un símbolo o palabra que significa que se ha de realizar cierta acción entre dos o más valores, llamados operandos. Los operandos pueden ser variables o valores que previamente han sido asignados.

Ejemplo:

Figura 30. Operandos y operadores

Cuando dos operadores tienen el mismo nivel de prioridad, dentro de una expresión se evalúan de izquierda a derecha.

Cuando se desea asignar un orden específico de ejecución en una expresión aritmética, se debe emplear los paréntesis para agrupar, de esta manera, las operaciones que se encuentren dentro del paréntesis

serán las primeras en ejecutarse

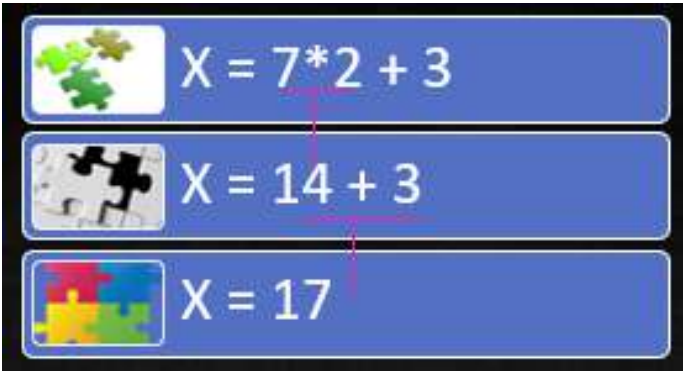


Figura 31. Ejemplo de expresión

En la figura 31, en la expresión 7*2+3, los valores 7,2 y 3 se denominan operandos, que son los valores que se evalúan o relacionan. El valor de la expresión 7*2 + 3 se conoce como resultado de la expresión, que es el producto de la relación entre los operandos condicionados por los operadores * y +.

Ejemplos de expresiones

Calcular el valor de C, si este se obtiene con la siguiente expresión:

$$C = 2^3 + 5*4 - 5^2$$

1. Resolvemos potencias (^)

C = 2^3 + 5 * 4 - 5^2
C = 8 + 5*4 + 25

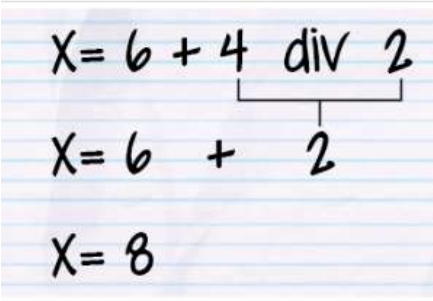
2. Resolvemos multiplicaciones (*)

C = 8 + 5*4 - 25
C = 8 + 20 - 25

3. Resolvemos sumas (+) y restas (-)

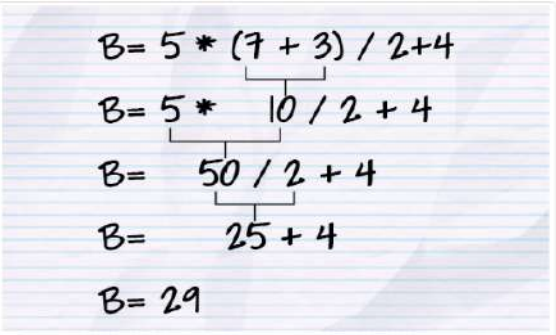
C = 8 + 20 - 25
C = 3

En este ejemplo como se puede observar primero ejecutamos la potencia poque es mayor prioridad, luego seguimos con multiplicaciones y finalmente con sumas y restas que son menor prioridad.



Ejemplo 1

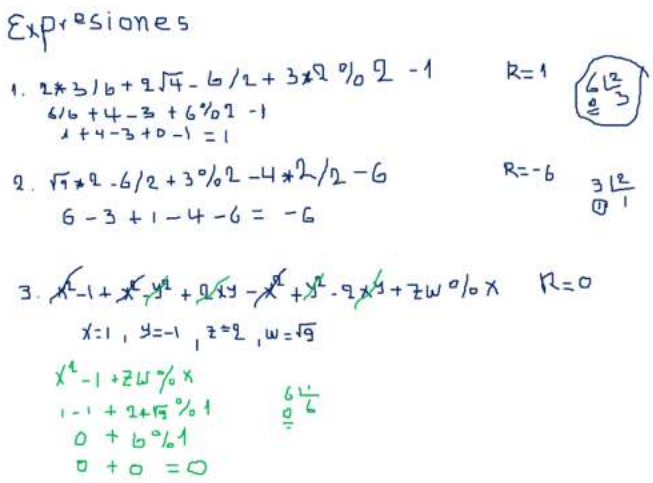
De la misma forma en estos ejemplos se van resolviendo basados en la prioridad de cada uno de los operadores.



Ejemplo 3

Cuando vaya a resolver estos ejercicios basados en prioridad de operadores debe tener en cuenta cuál de ellos tiene mayor prioridad sobre los otros.

Ejercicios de expresiones:



En este ejercicio se pone énfasis en la raíz y el residuo, recordar que la raíz cuadrada puede también resolver como una potencia elevada a la ½ y que el residuo es la parte que nos sobra debajo del dividendo, tener cuidado con esto.

Expresiones Algebraicas

Es un conjunto de símbolos, números o variables utilizados en matemáticas para representar relaciones aritméticas. El término algebraico es utilizado en matemáticas pero la computadora es incapaz de reconocer la simbología. Una expresión algebraica simple es:

$$\frac{(ax^2 + bx^2)}{2} = c$$

Expresiones algorítmicas

Es un conjunto de símbolos, números o variables que representan una instrucción específica y computable para el computador. Las expresiones algorítmicas son utilizadas en los lenguajes de programación para especificar claramente cuál es el orden y el tipo de operación a realizar por el computador. Una expresión algorítmica típica es:

$$(a * x\text{^}2 + b * x\text{^}2)/2 = c$$

Ejemplo de expresión algebraica a expresión algorítmica

Las siguientes expresiones algebraicas convierten a expresiones algorítmicas:

a) $\frac{3}{2}+\frac{4}{3}$

b) $\frac{1}{x-5}-\frac{3xy}{4}$

c) $\frac{1}{2}+7$

d) $7+\frac{1}{2}$

e) $\frac{a^2}{b-c}+\frac{d-e}{f-\frac{g*h}{j}}$

f) $\frac{m}{n}+p$

g) $m+\frac{n}{p-q}$

h) $\frac{a^2}{b^2}+\frac{c^2}{d^2}$

i) $\frac{m+\frac{n}{p}}{q-\frac{r}{s}}$

j) $\frac{3a+b}{c-\frac{d+5e}{f+\frac{g}{2h}}}$

k) $\frac{a^2+2ab+b^2}{\frac{1}{x^2}+2}$

Solución: aquí se muestra la expresión algorítmica de los ejercicios propuestos.

```
a) 3/2+4/3                  e) a*a/(b-c)+(d-e)/(f-g*h/j)      i) (m+n/p)/(q-r/s)
b) 1/(x-5)-3*x*y/4          f) m/n+p                          j) (3*a+b)/(c-(d+5*e)/(f+g/(2*h)))
c) 1/2+7                    g) m+n/(p-q)                      k) (a*a+2*a*b+b*b)/(1/(x*x))+2
d) 7+1/2                    h) a*a/(b*b)+c*c/(d*d)
```

Actividades

A continuación se presentan una serie de ejercicios que deberá resolver paso a paso según la jerarquía de los operadores involucrados en cada expresión. Es decir, debe resolver cada operación según la jerarquía, obteniendo resultados parciales, hasta llegar al resultado final, tal como lo realiza el computador.

1. Digamos que A=5, B=25 y C=10. ¿Puedes deducir cuál será el valor de la variable?

X en cada uno de los siguientes casos?

- X = A + B + C
- X = A + B * C
- X = A + B / C
- X = A + B mod C
- X = (A + B) div C
- X = A + (B / C)

2. Calcular el valor de la variable C en cada uno de los siguientes casos.

- C = 8 + 7 * 3 + 5 * 6
- C = -2 ^ 3
- C = (33 + 3 * 4) / 5

- C = 2 ^ 2 * 3
- C = 3 + 2 * (18 - 4 ^ 2)
- C = 16 * 6 - 3 * 2

3. Resuelva la expresión aritmética siguiente:

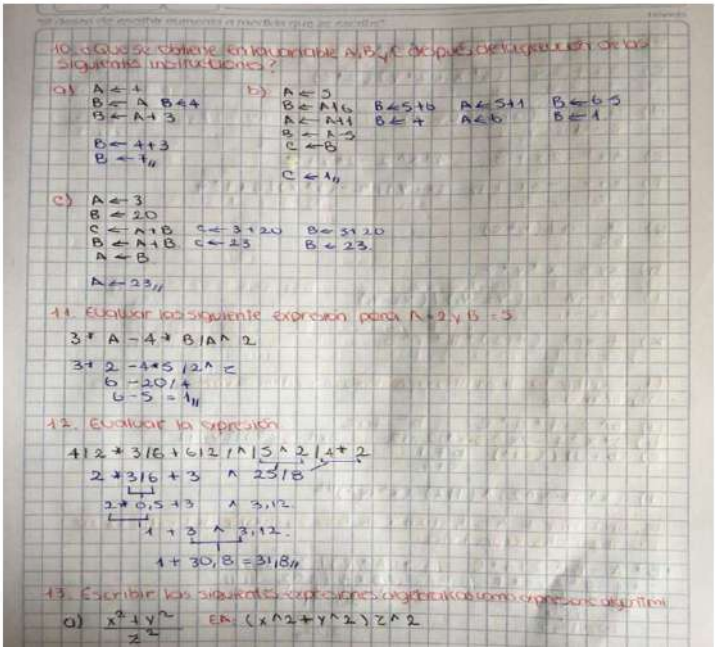
X = 5 Y = 9 K = 2 N = 7

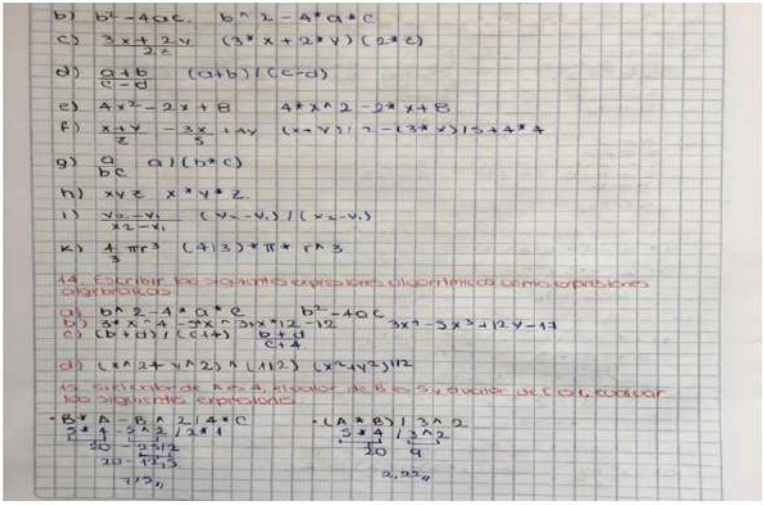
*Y = (X + N Mod 6 / 3) - (Y * 7 Mod K Mod (Y + 8 / 2 + X Mod 5)) * X + N*

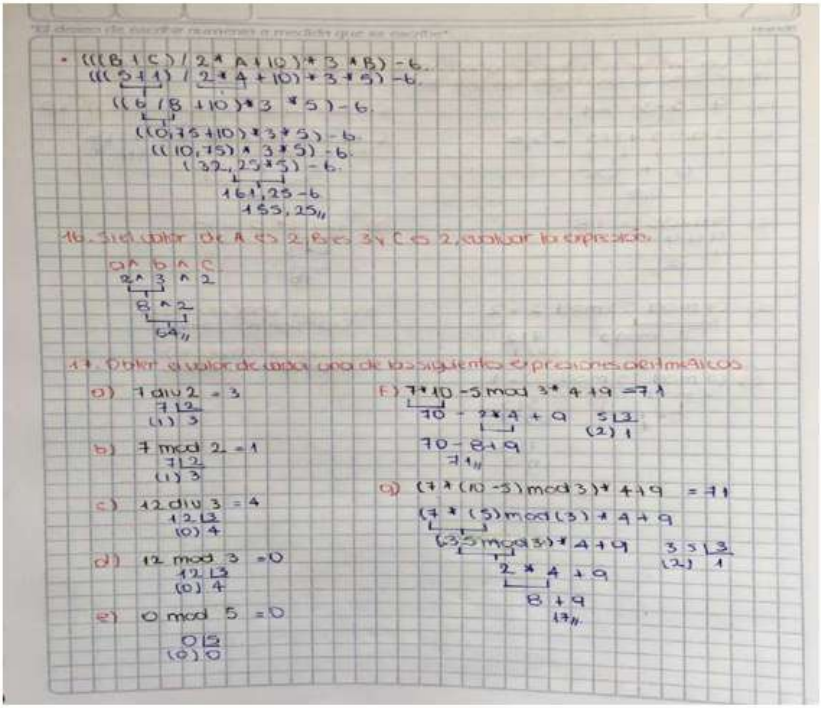
- **Ejercicio Resuelto.**
  Convierte en expresiones algoritmicas las siguientes expresiones algebraicas

a)	$a^2 + b^2$	a * a + b * b ó a^2 + b^2
b)	$(a + b)^2$	(a + b) * (a+b)
c)	$\sqrt[3]{b} + 34$	b^(1/3) + 34
d)	$\sqrt[3]{b + 34}$	(b + 34)^(1/3)
f)	$\frac{x + y}{u + \frac{w}{b}}$	(x+y) / (u + w / b)
g)	$x + \frac{y}{u} + \frac{w}{b}$	x + y/u + w/b
h)	$\frac{x}{y}(z + w)$	x/y * (z+w)

Ejercicios con operadores







Nótese como se van resolviendo paso a paso este tipo de ejercicios y de acuerdo a la prioridad de los operadores siempre.

Entrada y salida de datos

En esta situación de la entrada y salida de datos primero vamos a instalar el software PSeInt, el cual nos servirá para poder realizar los algoritmos dentro de los próximos capítulos.

PSeInt

Las características de este pseudo lenguaje fueron propuestas en 2001 por el responsable de la asignatura Fundamentos de Programación de

la carrera de Ingeniería Informática de la FICH-UNL. Las premisas son:

Sintaxis sencilla

- Manejo de las estructuras básicas de control
- Solo 3 tipos de datos básicos: numérico, carácter /cadenas de caracteres y lógico (verdadero-falso).

Estructuras de datos: arreglos

Instalación de PSeInt

1. Ingresamos a Google, como se observa en la figura 32 para descargar PSeInt:

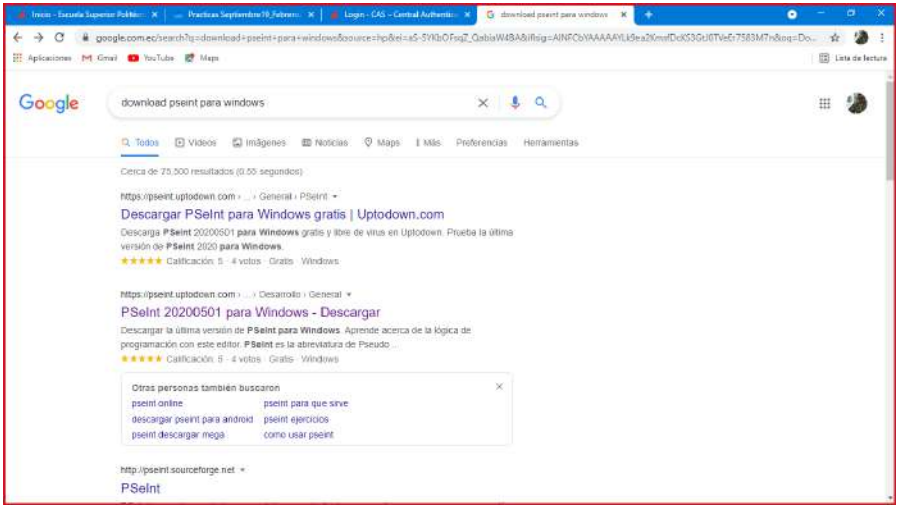


Figura 32. Descargar PSeInt

2. Hacemos clic en PSeInt 20200501 y tenemos la figura 33:

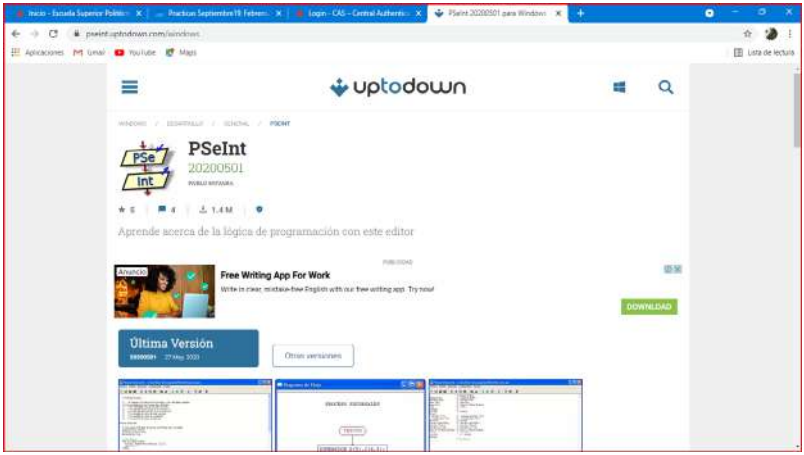


Figura 33. Sitio para descargar PSeInt

3. Clic en última versión y tenemos la figura 34:

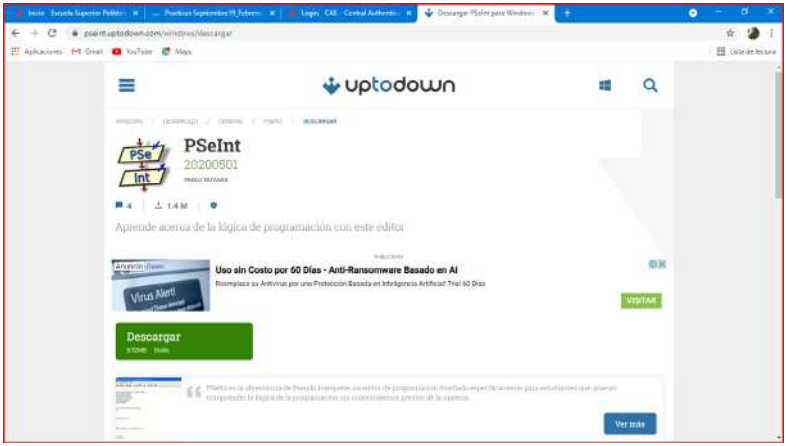


Figura 34. Instalador de PSeInt

4. Clic en descargar y obtenemos la figura 35:

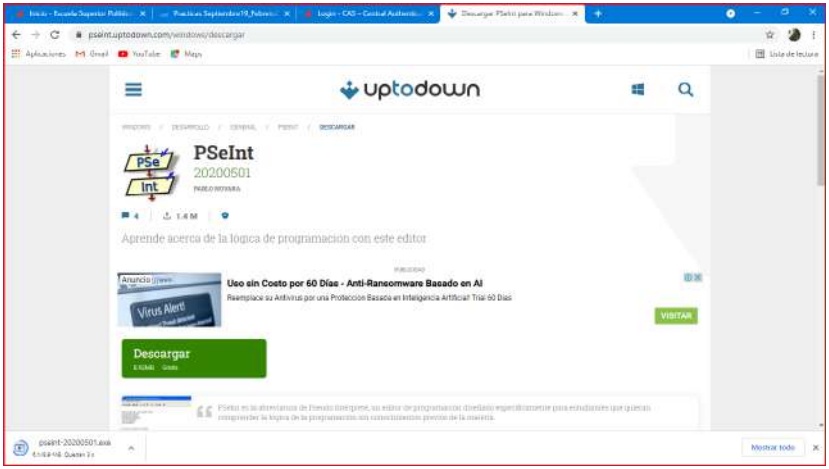


Figura 35. Descarga del archivo instalador

5. Clic en la pestaña del archivo descargado y clic en abrir, obtenemos la figura 36:

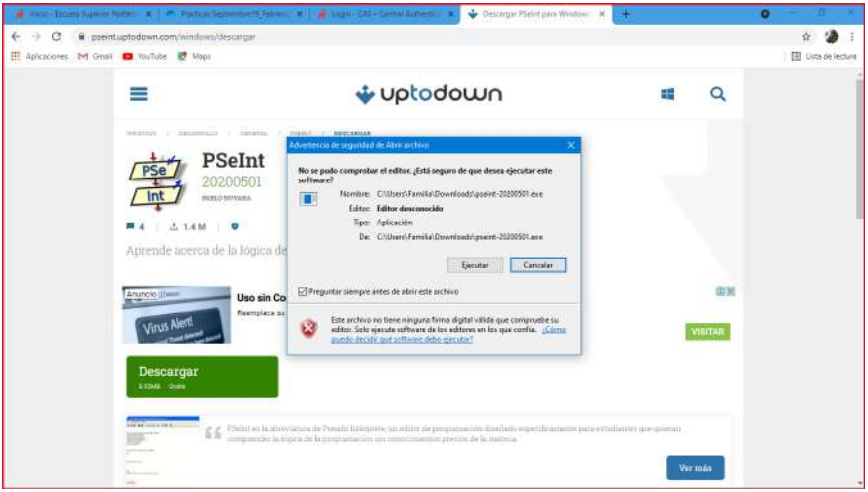


Figura 36. Inicio de la instalación de PSeInt

6. Clic en el botón ejecutar y tenemos la figura 37:

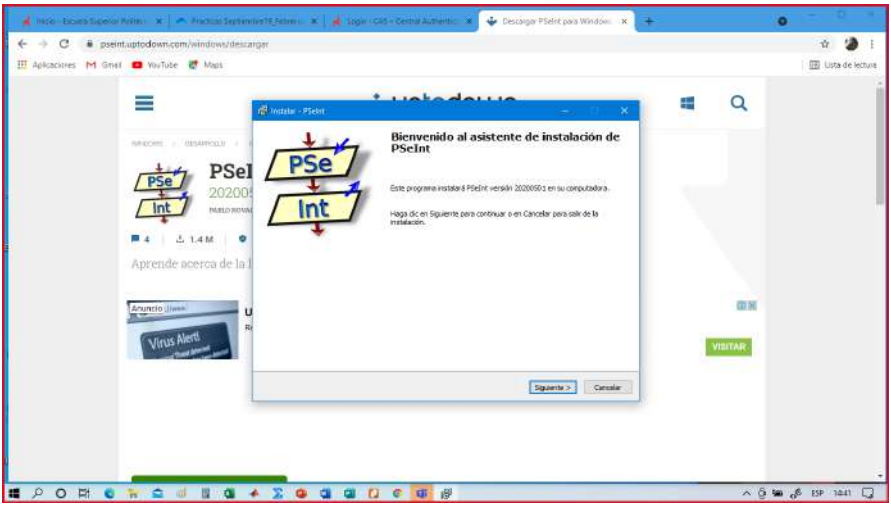


Figura 37. Bienvenido al asistente de instalación de PSeInt

7. Clic en siguiente y tenemos la figura 38:

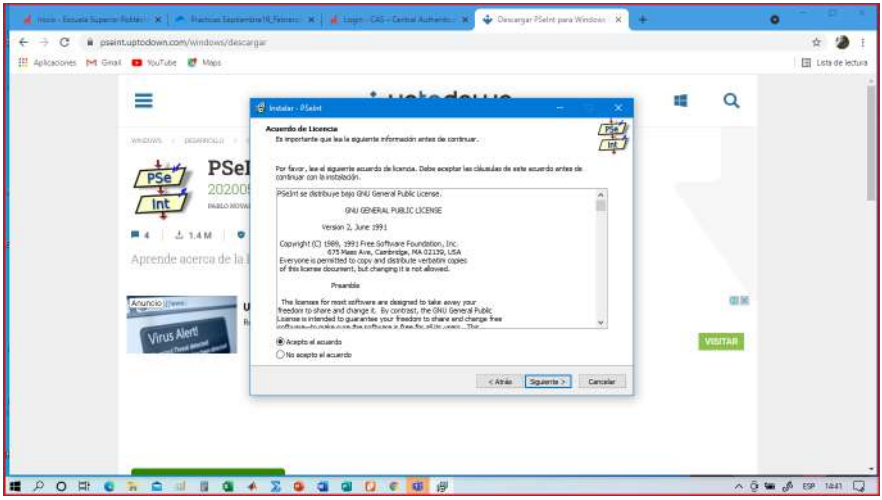


Figura 38. Acuerdo de licencia

8. Clic en "Acepto el acuerdo" y luego clic en siguiente, obtenemos la figura 39:

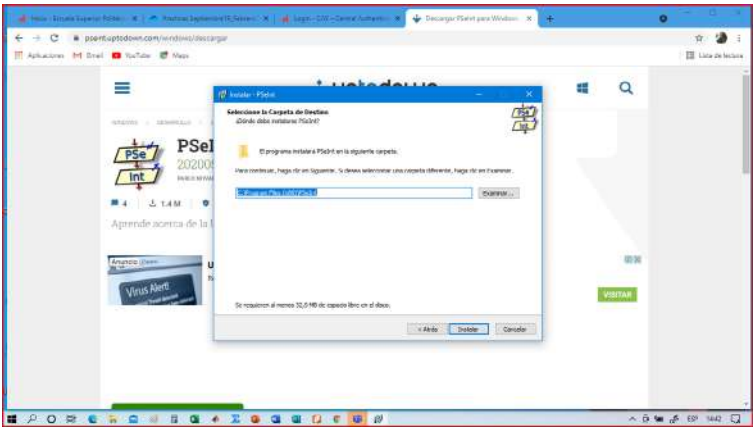


Figura 39. Instalar PSeInt en la carpeta escogida

9. Clic en Instalar y tenemos instalado nuestro software PSeInt.

10. En el icono que se encuentra en el escritorio de PSeInt dar doble clic y tenemos la figura 40:

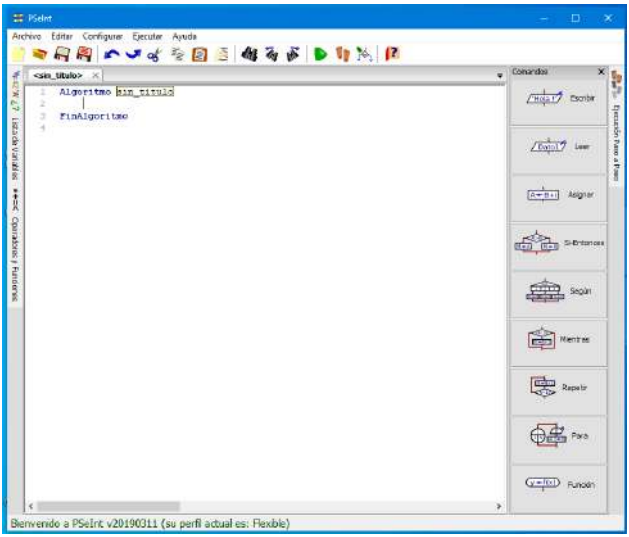


Figura 40. Interfaz gráfica de PSeInt

Mi primer ejemplo en PSeInt

Realizar un algoritmo que permita visualizar en pantalla la frase “Hola mundo en PSeInt”

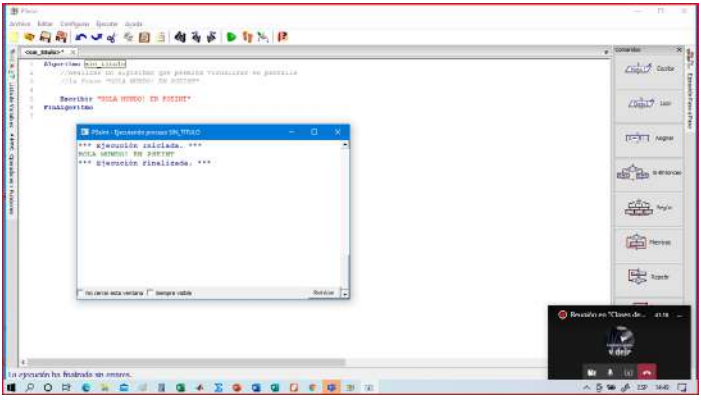


Figura 41. Algoritmo que visualiza la frase
“HOLA MUNDO! EN PSEINT”

Como podemos ver en la figura 41 tenemos el algoritmo que utiliza la palabra reservada “Escribir” la cual nos permite visualizar cualquier frase en pantalla dentro de PSeInt.

Segundo ejemplo en PSeInt

Realizar un algoritmo que permita sumar 2 números:

Para resolver este ejercicio se utiliza la metodología que se dijo anteriormente que es ENTRADA, PROCESO Y SALIDA.

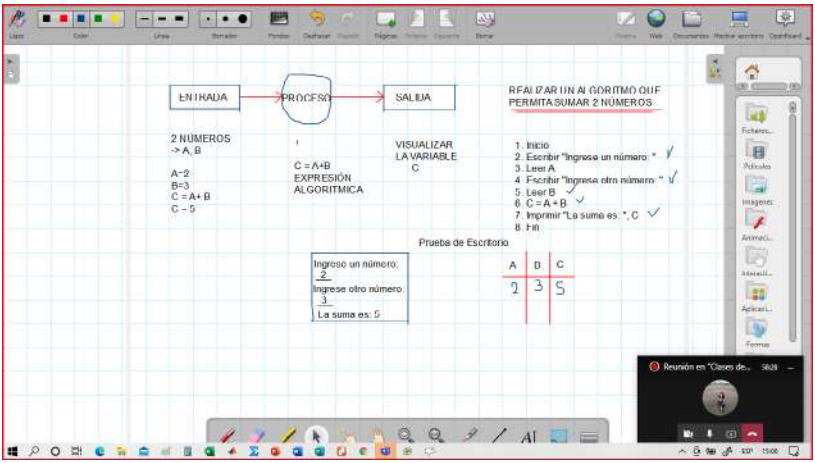


Figura 42. Metodología para un algoritmo que sume 2 números

Como podemos ver la figura 42 la metodología es muy fácil de realizar y en ella se muestra que es lo que tenemos a la ENTRADA, cuál es el PROCESO que vamos a realizar y finalmente que obtendremos a la SALIDA.

En la figura 43 tenemos la solución al algoritmo planteado:

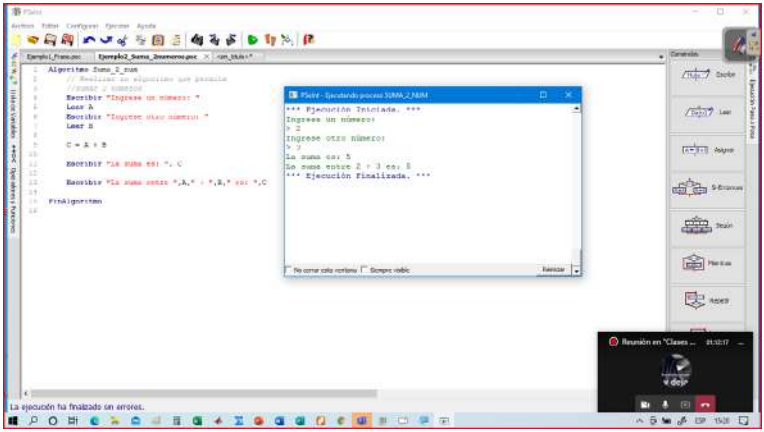


Figura 43. Solución al algoritmo de la suma de 2 números en PSeInt

Tercer ejemplo en PSeInt

Realizar un algoritmo que permita calcular el área de un rectángulo.

Para resolver este ejercicio se utiliza la metodología que se dijo anteriormente que es ENTRADA, PROCESO Y SALIDA.

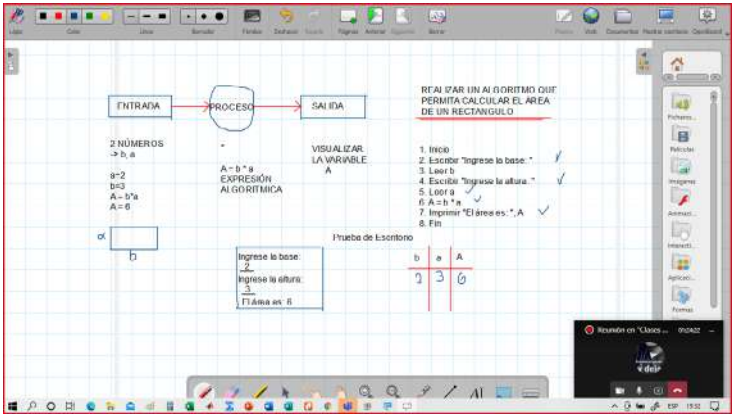


Figura 44. Metodología para un algoritmo que calcule el área de un rectángulo

Como podemos ver la figura 44 la metodología es muy fácil de realizar y en ella se muestra que es lo que tenemos a la ENTRADA como la base y la altura, cuál es el PROCESO que es el cálculo del área a través de la fórmula matemática A=b*a esto como una expresión algorítmica y finalmente que obtendremos a la SALIDA en este caso el área del rectángulo calculada.

En la figura 45 tenemos la solución al algoritmo planteado:

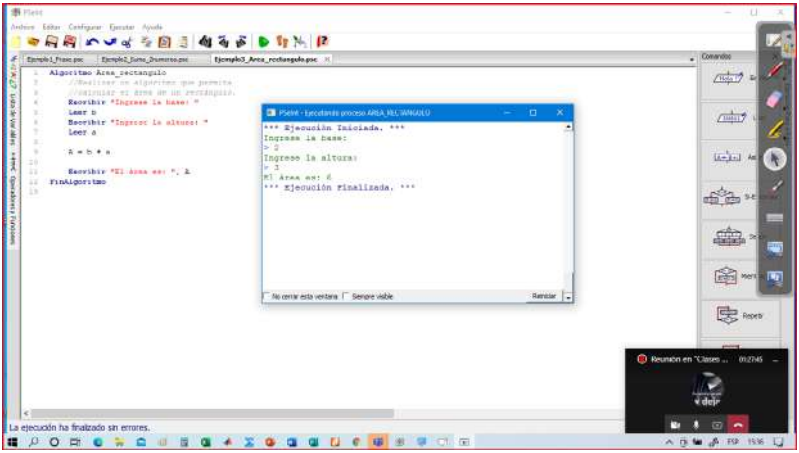


Figura 45. Solución al algoritmo del cálculo del Área de un rectángulo en PSeInt

Diagrama de flujo

Un diagrama de flujo de datos es una descripción gráfica de un algoritmo para la resolución de un problema. Son frecuentemente usados para describir algoritmos y programas de computador. Los diagramas de flujo de datos están conformados por figuras conectadas con flechas.

Para ejecutar un proceso descrito por un diagrama de flujo de datos se comienza por el INICIO y se siguen las flechas de figura a figura, ejecutándose las acciones indicadas por cada figura; el tipo de figura indica el tipo de paso que representa.

Los diagramas de flujo son frecuentemente usados debido a que pueden suprimir detalles innecesarios y tener un significado preciso, si son usados correctamente.

Figura 46. Diagrama de flujo

En la figura 46 se puede observar cómo se puede realizar un diagrama de flujo para resolver un problema planteado.

Ejemplo 1: Lavarnos los dientes es un procedimiento que realizamos varias veces al día. Veamos la forma de expresar este procedimiento como un DFD:

Figura 47. Diagrama de flujo para lavarnos los dientes

DFD – Software de diseño de diagramas de flujo

Dfd es un software diseñado para construir y analizar algoritmos de manera gráfica. Usted puede crear diagramas de flujo de datos para la representación de algoritmos de programación estructurada a partir de las herramientas de edición que para este propósito suministra el programa. Después de haber ingresado el algoritmo representado por el diagrama, podrá ejecutarlo, analizarlo y depurarlo en un entorno interactivo diseñado para éste fin. La interfaz gráfica de Dfd, facilita en gran medida el trabajo con diagramas ya que simula la representación estándar de diagramas de flujo en hojas de papel.

En la figura 48 podemos observar la forma de conseguir el software DFD y poder ejecutarlo en nuestra computadora.

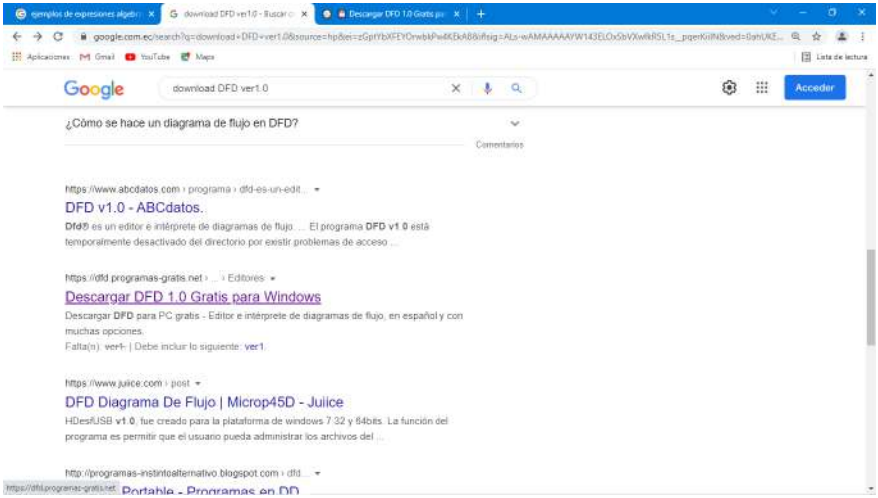


Figura 48. Archivo del software DFD

En la figura 49 se puede observar el sitio de donde obtener el archivo de ejecución del software DFD.

Figura 49. Sitio de descarga del archivo de ejecución de DFD

Al hacer clic en descargar nos lleva a la figura 50 de donde descargamos el archivo en formato zip que luego se debe descomprimir en alguna carpeta y listo:

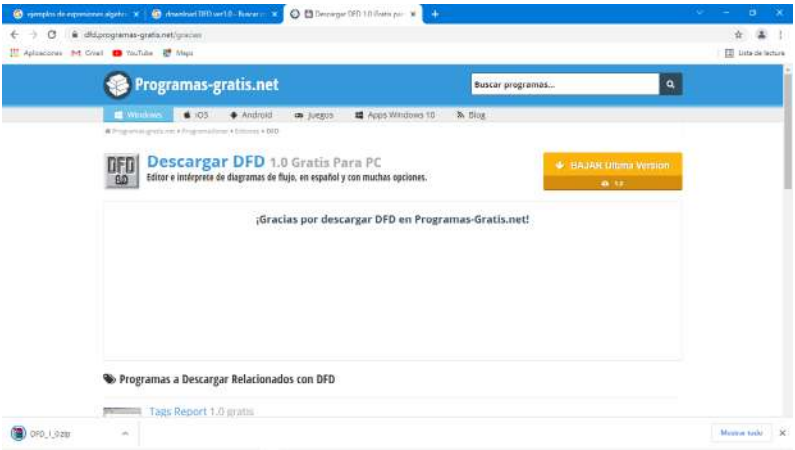


Figura 50. Archivo zip del software DFD

Luego de haberse descargado el archivo zip pasarlo a una carpeta así como se muestra en la figura 51.

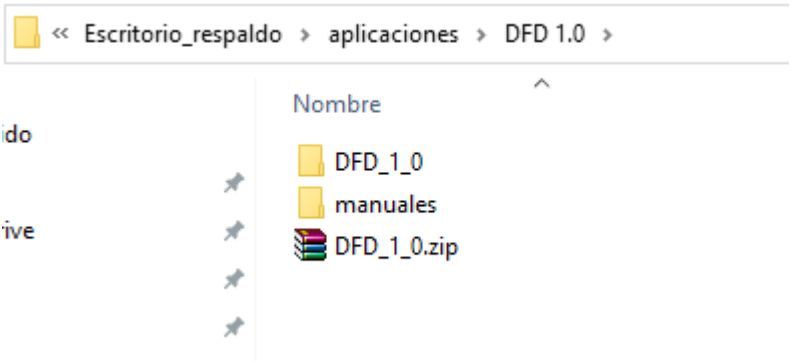


Figura 51. Archivo zip del software DFD
Copiado a una carpeta

Ahora debemos descomprimir el archivo y nos queda de la siguiente manera, tal cual como se muestra en la figura 52:

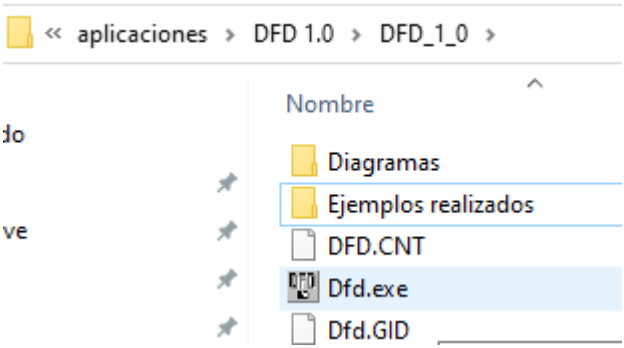


Figura 52. Archivo zip descomprimido y el archivo Dfd.exe

Ahora al hacer clic en el archivo Dfd.exe obtenemos la interfaz gráfica de DFD como se muestra en la figura 53.

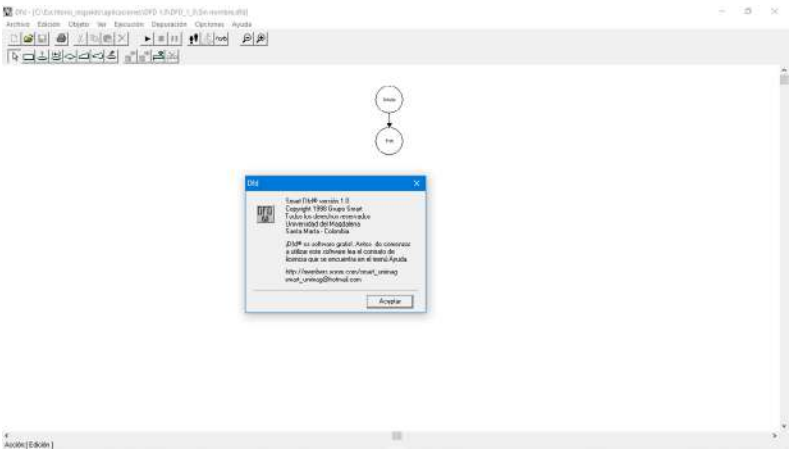


Figura 53. Interfaz gráfica de DFD

En la figura 54 se puede observar los objetos para ingreso y salida de información en el software DFD:

Figura 54. Objetos de entrada y salida
de información en DFD

Primer ejemplo en DFD

Realizar un diagrama de flujo que permita visualizar en pantalla el mensaje “HOLA MUNDO! EN DFD”

En la figura 55 tenemos la resolución del diagrama de flujo planteado.

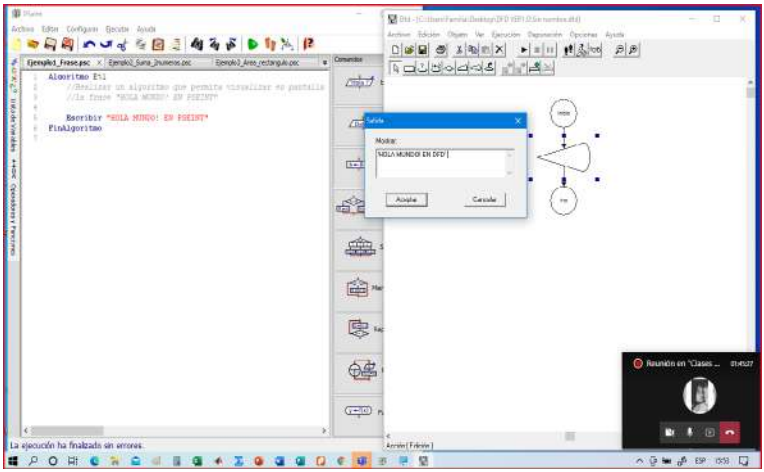


Figura 55. Diagrama de flujo para el mensaje “HOLA MUNDO! EN DFD”

En la figura 56 se observa la resolución y ejecución del diagrama de flujo planteado.

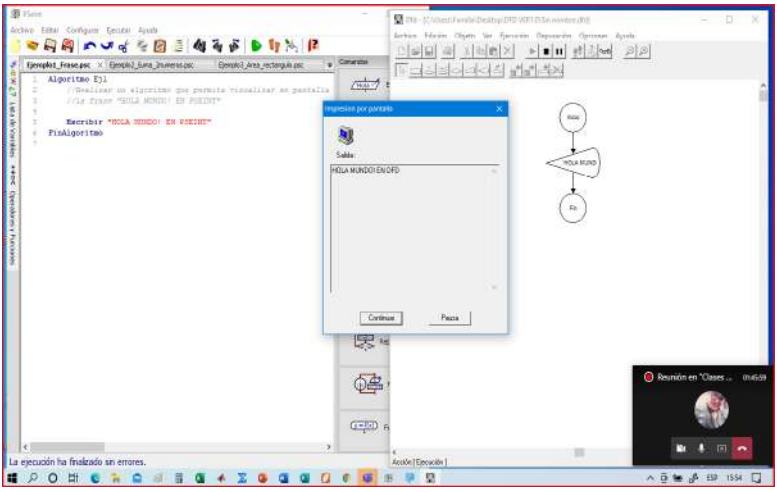


Figura 56. Diagrama de flujo para el mensaje
“HOLA MUNDO! EN DFD”, resolución y ejecución.

Segundo ejemplo en DFD

Realizar un diagrama de flujo para sumar 2 números.

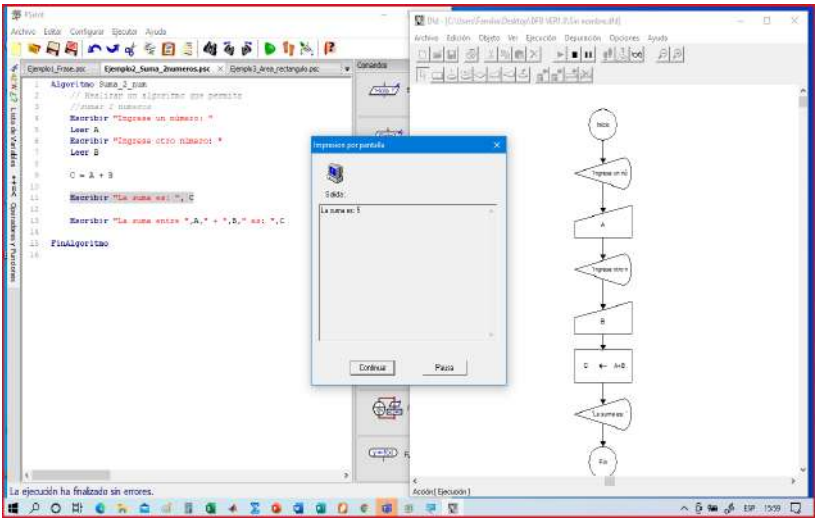


Figura 57. Diagrama de flujo para suma de 2 números

En la figura 57 se observa el diagrama de flujo su resolución y ejecución en DFD.

Tercer ejemplo en DFD

Realizar un diagrama de flujo para calcular el área de un rectángulo. (Tarea en casa)

Funciones usadas en PSeInt

Las funciones que tiene PSeInt para ayudarnos a resolver algunos ejercicios se muestran en la tabla 11.

Tabla 11. Funciones usadas en PSeInt

Función	**Significado**
RC(X) o RAIZ(X) Raíz Cuadrada de X	Raíz Cuadrada de X
ABS(X)	Valor Absoluto de X
LN(X)	Logaritmo Natural de X
EXP(X)	Función Exponencial de X
SEN(X)	Seno de X
COS(X)	Coseno de X
TAN(X)	Tangente de X
ASEN(X)	Arcoseno de X
ACOS(X)	Arcocoseno de X
ATAN(X)	Arcotangente de X
TRUNC(X)	Parte entera de X
REDON(X)	Entero más cercano a X
AZAR(X)	Entero aleatorio entre 0 y x-1

ESTRUCTURAS DE CONTROL

Capítulo 3

ESTRUCTURAS DE CONTROL

En programación, las estructuras de control permiten modificar el flujo de ejecución de las instrucciones de un programa.

Todos los lenguajes de programación modernos tienen estructuras de control similares. Básicamente lo que varía entre las estructuras de control de los diferentes lenguajes es su sintaxis; cada lenguaje tiene una sintaxis propia para expresar la estructura.

Con las estructuras de control se puede:

- De acuerdo al seguimiento de pasos, la estructura secuencial.
- De acuerdo con una condición, ejecutar un grupo u otro de sentencias (If-ThenElse), la estructura condicional.
- De acuerdo con el valor de una variable, ejecutar un grupo u otro de sentencias (Select-Case o Switch), estructura condicional múltiple.
- Ejecutar un grupo de sentencias mientras se cumpla una condición (While), estructura cíclica "Mientras". ▪ Ejecutar un grupo de sentencias hasta que se cumpla una condición (Do-While), estructura cíclica "Hacer Mientras".
- Ejecutar un grupo de sentencias un número determinado de veces (For), estructura cíclica "Para".

Tipos de estructuras de control

En la figura 58 se muestra los tipos de estructuras de control que tenemos dentro de la programación:

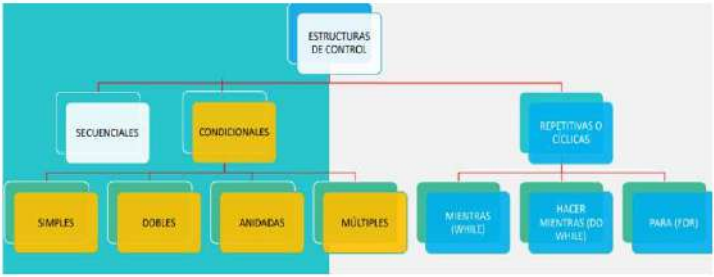


Figura 58. Tipos de estructuras de control

Estructura secuencial

Se caracteriza porque una acción se ejecuta detrás de otra. El flujo del programa coincide con el orden físico en el que se han ido poniendo las instrucciones. Dentro de este tipo podemos encontrar operaciones de inicio/fin, inicialización de variables, operaciones de asignación, cálculo, sumarización, etc. Este tipo de estructura se basa en las 5 fases de que puede tener todo algoritmo o programa, tal cual como se observa en la figura 59:

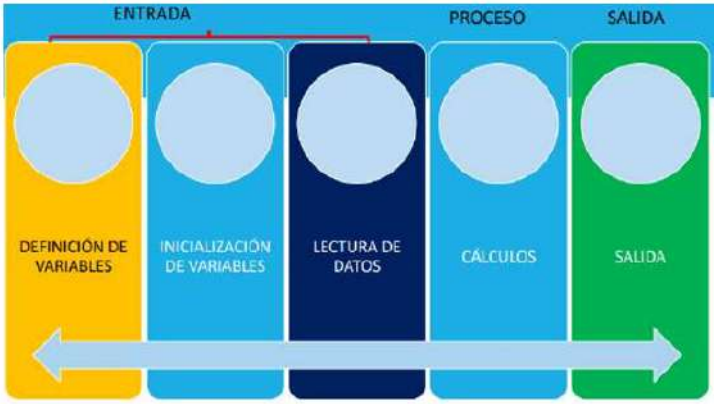


Figura 59. Fases de todo algoritmo

Ejemplo 1:

Realizar un algoritmo y diagrama de flujo que permita ingresar 3 notas de un estudiante y obtener la suma de las mismas y el promedio

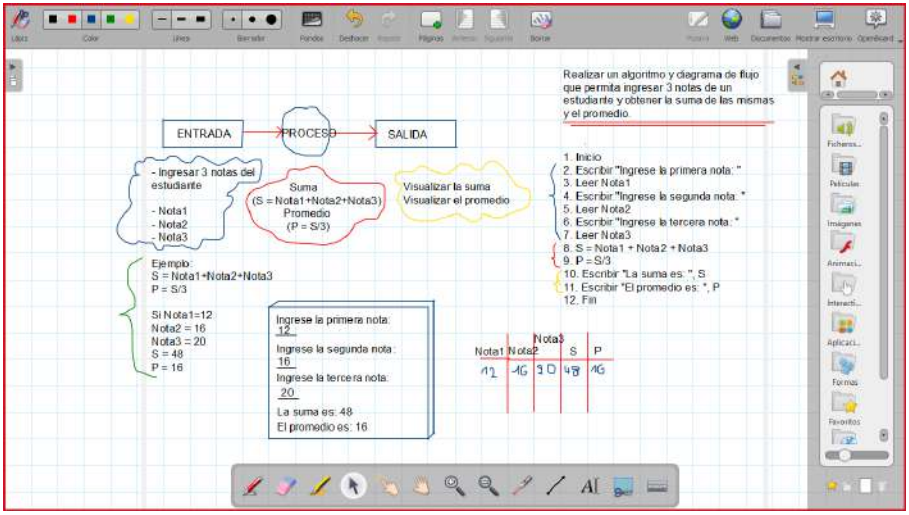


Figura 60. Metodología usada para el cálculo de notas

Como se puede observar en la figura 60 tenemos la metodología usada para resolver el ingreso de 3 notas y sus variables a la ENTRADA, luego tenemos las fórmulas a usar en el PROCESO y finalmente lo que vamos a visualizar a la SALIDA.

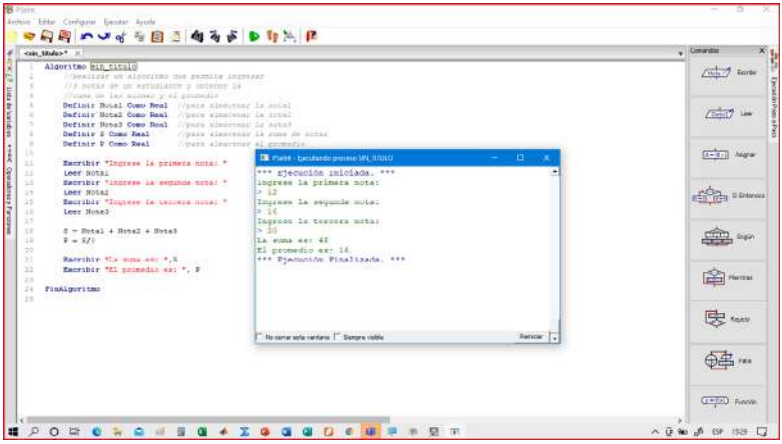


Figura 61. Resolución del algoritmo de las notas

En la figura 61 podemos observar los pasos para el algoritmo planteado en PSeInt junto a su ejecución en el mismo software.

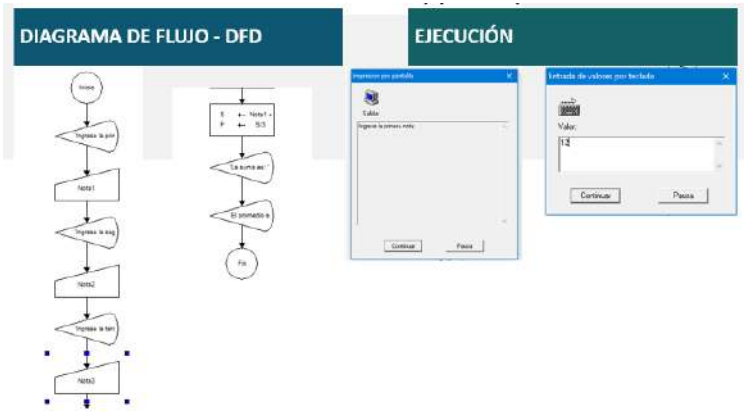


Figura 62. Diagrama de flujo para el algoritmo de notas

Se presenta en la figura 62 el diagrama de flujo para el cálculo de notas junto con su promedio.

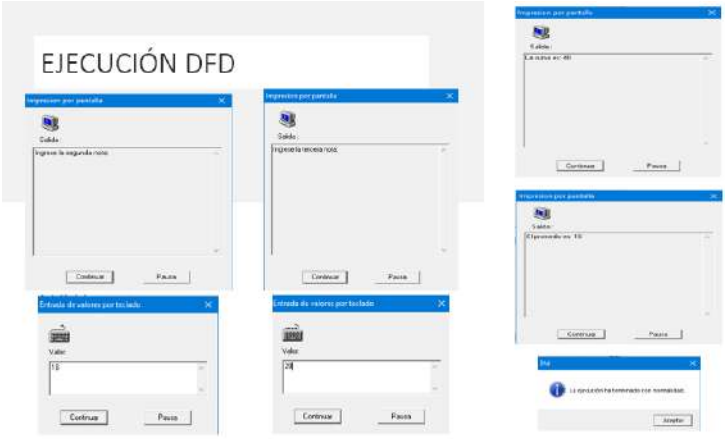


Figura 63. Ejecución del diagrama de flujo de notas

Ejemplo 2:

Realizar un algoritmo y diagrama de flujo que permita ingresar 3 números enteros y obtener el cuadrado, cubo y a la cuarta de cada uno en columnas.

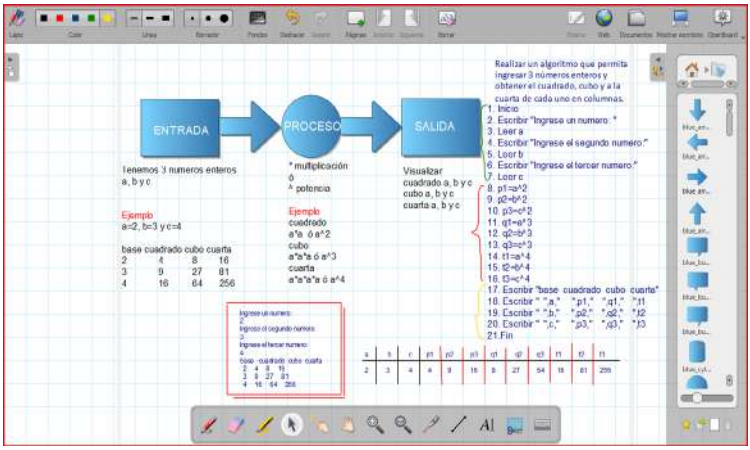


Figura 64. Metodología usada para el cálculo de cuadrado, cubo y cuarta de 3 números.

Como se puede observar en la figura 64. tenemos la metodología usada para resolver el ingreso de 3 números enteros y sus variables a la ENTRADA, luego tenemos las fórmulas a usar en el PROCESO de cálculo de cuadrado, cubo, cuarta y finalmente lo que vamos a visualizar a la SALIDA.

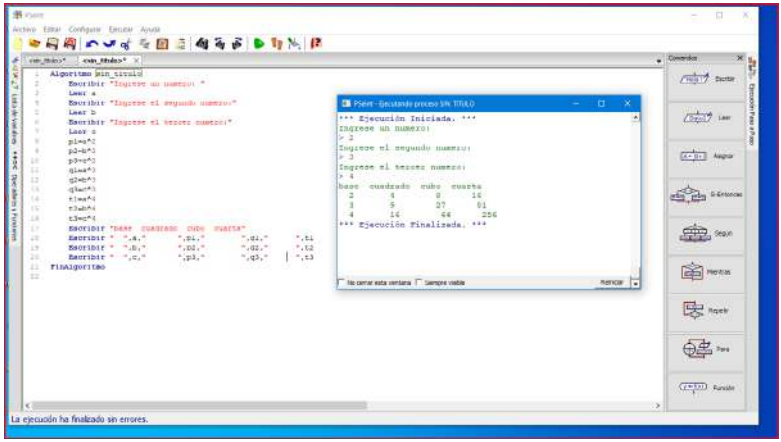


Figura 65. Resolución del algoritmo de cuadrado, cubo y cuarta de 3 números.

En la figura 65 podemos observar los pasos para el algoritmo planteado del ejemplo 2 en PSeInt junto a su ejecución en el mismo software.

Nota.- No olvide que las estructuras secuenciales siguen línea a línea la ejecución.

Estas estructuras nos permiten realizar algoritmos de acuerdo a un seguimiento estandarizado de código.

Ejercicios a realizar de estructuras secuenciales

a. Realizar un algoritmo y diagrama de flujo para los siguientes ejercicios:

1. Calcular el área de un rectángulo
2. Calcular el área de un triángulo
3. Calcular el sueldo de un empleado con las horas trabajadas, valor por hora trabajada, horas extras, valor por hora extra y el seguro del IESS.

   FÓRMULA:

*Sueldo = horas_trabajadas*valor_hora_trabajada + horas_extras*valor_hora_extra + Iess*

4. Calcular el Iva de un producto ingresando su costo. Al final se debe mostrar el costo ingresado, el iva y el precio.
5. Se pide que se calcule el porcentaje de alumnos y alumnas de un salón de clase.
6. Calcular la suma de 5 datos ingresados y su promedio.
7. Realice por su cuenta 4 ejercicios más.

b. Realice un algoritmo y diagrama de flujo para los siguientes ejercicios, junto con su metodología y prueba de escritorio.

1. Dada las horas trabajadas de una persona y el valor por hora. Calcular su salario e imprimirlo.
2. Dado un tiempo en segundos e ingresado por teclado, calcular los segundos restantes para convertirse exactamente en minutos, cuanto sobra en segundos y cuanto le falta para otro minuto.

3. Dado un tiempo en minutos, calcular los días, horas y minutos que le corresponden. (2710)
4. Un colegio desea saber qué porcentaje de niños y qué porcentaje de niñas hay en el curso actual. Muestre la cantidad de alumnos en porcentaje.
5. Exprese cierta cantidad de centavos en billetes y monedas de curso legal.
6. Convertir una medida dada en pies a sus equivalentes en:

- Yardas
- Pulgadas
- Centímetros
- Metros

Nota del ejercicio6.- (1 pie =12 pulgadas, 1 yarda = 3 pies, 1 pulgada = 2.54cm, 1m= 100cm). Leer el número de pies e imprimir el número de yardas, pies, pulgadas, centímetros y metros

7. Escribir un programa que determine la suma de las cifras de un entero positivo de 4 cifras

Ejemplo: 1463 se debe visualizar 14

Estructuras condicionales

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

constante, según se necesite. Existen cuatro tipos básicos, las simples, las dobles, las anidadas y las múltiples

Estructuras condicionales simples

Las estructuras condicionales simples se les conoce como:

- Tomas de decisión.
- Estas tomas de decisión tienen la siguiente forma, tal cual como se muestra en la figura 66.

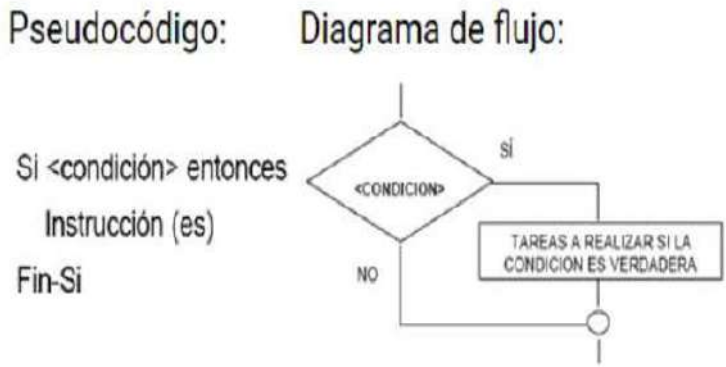


Figura 66. Seudocódigo y diagrama de flujo Estructura simple

Ejemplo 1:

Realizar un algoritmo y diagrama de flujo que permita ingresar la edad de una persona y si es mayor de 18 años visualizar el mensaje “Es Adulto.

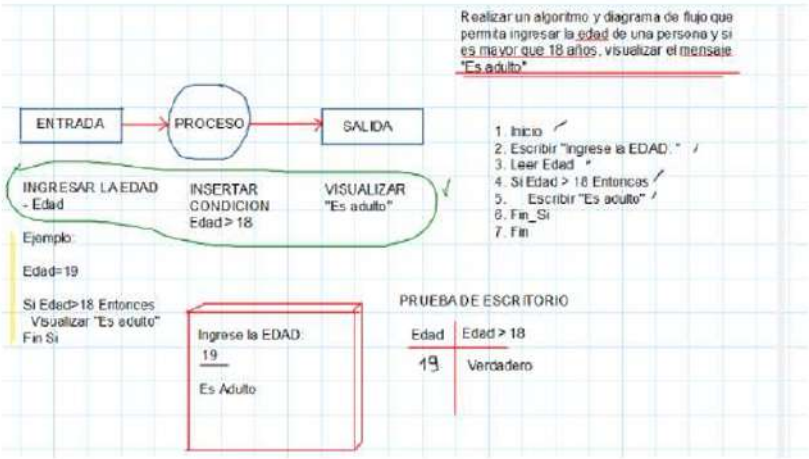


Figura 67. Metodología para resolver el ejemplo de la edad de una persona, para visualizar es adulto.

Como se puede observar en la figura 67 tenemos la metodología usada para resolver el ingreso de la Edad de una persona junto a su variable de ENTRADA, luego tenemos la condición a usar en el PROCESO y finalmente lo que vamos a visualizar a la SALIDA.

Luego tenemos el algoritmo en PSeInt:

Algoritmo Mensaje_adulto

Escribir "Ingrese su edad" Leer Edad
Si Edad > 18 Entonces
Escribir "Es Adulto y su edad es",Edad,"años"
FinSi
FinAlgoritmo

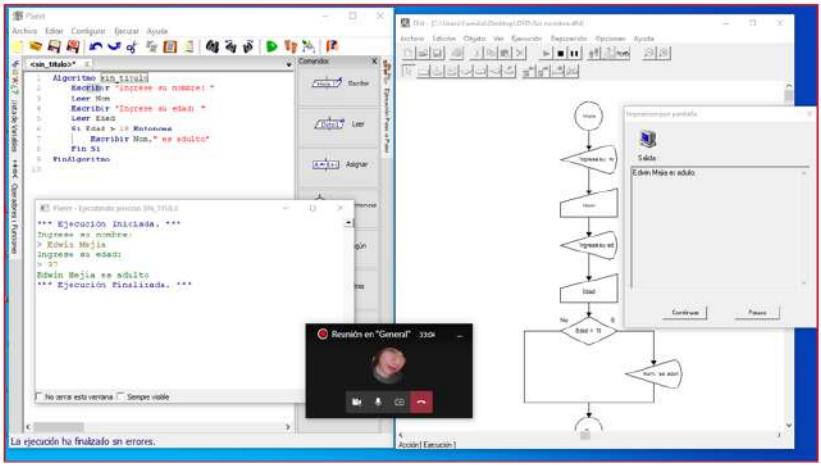


Figura 68. Algoritmo y diagrama de flujo para visualizar es adulto.

El mismo ejemplo anterior pero ahora ingresando el nombre de una persona.

- Aquí tenemos el algoritmo:

Algoritmo nombre_edad

Escribir "Ingrese su nombre" Leer nombre
Escribir "ingrese la edad de ",Nombre,":"

Leer edad

Si Edad > 18 Entonces
Escribir Nombre," es adulto y su edad es ",Edad," años"
Finsi
FinAlgoritmo

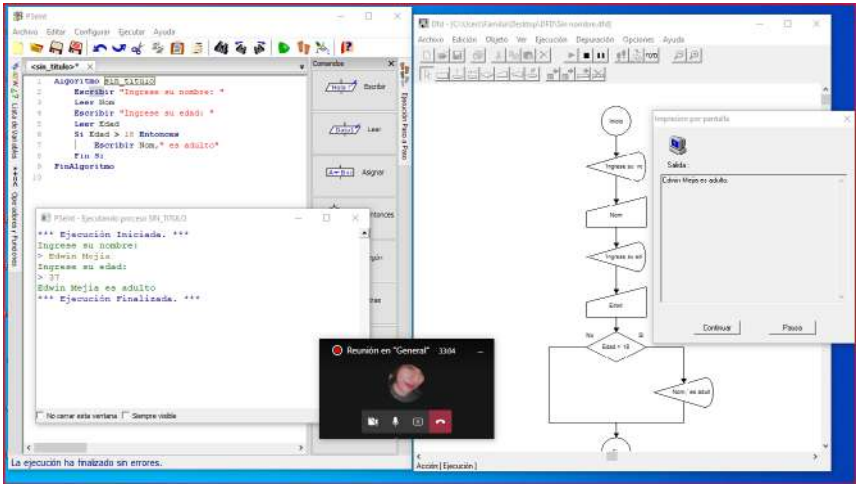


Figura 69. Algoritmo y diagrama de flujo para visualizar el nombre de una persona, junto al mensaje es adulto.

En la figura 69 podemos observar los pasos para el algoritmo planteado del ejemplo 1 de estructuras condicionales simples en PSeInt junto a su ejecución en el mismo software.

Nota.- No olvide que las estructuras condicionales simples solo se ejecutan por la parte Verdadera, la parte falsa no se ejecuta.

Estas estructuras nos permiten realizar algoritmos de acuerdo a una condición y si es VERDADERA la misma se ejecuta.

Estructuras condicionales dobles

Las estructuras condicionales dobles permiten elegir entre dos opciones o alternativas posibles en función del cumplimiento o no de una determinada condición. Se representa de la siguiente forma:

Figura 70. Seudocódigo y diagrama de flujo Estructura doble

Donde: Si: Indica el comando de comparación Condición: Indica la condición a evaluar Entonces: Precede a las acciones a realizar cuando se cumple la condición Instrucción(es): Son las acciones a realizar cuando se cumple o no la condición si no: Precede a las acciones a realizar cuando no se cumple la condición Dependiendo de si la comparación es cierta o falsa, se pueden realizar una o más acciones.

Ejemplo 1:

Algoritmo

//

Escribir 'Ingrese el total de materia prima en Kg que llego el primer semestre del año'

Leer Sem1

Escribir 'Ingrese el total de materia prima en Kg que llego el segundo semestre del año'
Leer Sem2

Mattotal=Sem1+Sem2

Escribir 'El total de materia prima comprado en el año fue de ' , Mattotal , 'Kg'
Si Mattotal > 500 Entonces

Escribir 'La cantidad de materia prima fue la requerida'
SiNo
Escribir 'Se necesita más materia prima'

Fin Si

Escribir 'Gracias por permitir ayudarle'

FinAlgoritmo

Ejemplo 2:

Realizar un algoritmo y diagrama de flujo que permita obtener el mayor de 2 números.

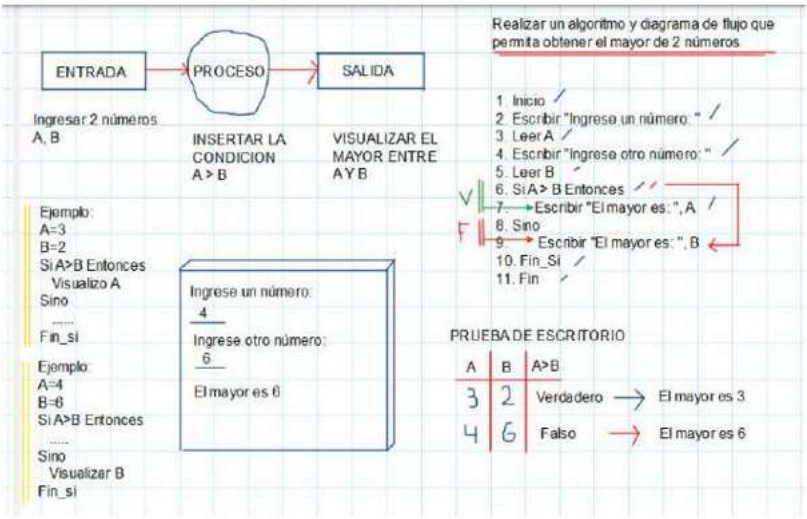


Figura 71. Metodología para resolver el ejemplo de mayor de 2 números.

Metodología y prueba de escritorio

ENTRADA

Debemos ver que necesitamos a la entrada, para ello analizamos: Necesitamos 2 variables para almacenar los 2 números.

- Num1 - Num2

PROCESO

Vamos a calcular el mayor de los 2 números

SALIDA

Visualizamos el resultado de haber obtenido el mayor de los 2 números.

Prueba de escritorio

Una prueba de escritorio es un tipo de prueba algorítmica, que consiste en la validación y verificación del algoritmo a través de la ejecución

de las sentencias que lo componen (proceso) para determinar sus resultados (salida) a partir de un conjunto determinado de elementos (entrada).

Num1	Num2	Camino	Mayor
5	2	V	5
3	7	F	7

Algoritmo mayor_2_numeros

Escribir "Ingrese un número:"

Leer A

Escribir "Ingrese otro número:"

Leer B

Si A > B Entonces

Escribir " El mayor es:", A

SiNo

escribir "El mayor es :",B

FinSi

FinAlgoritmo

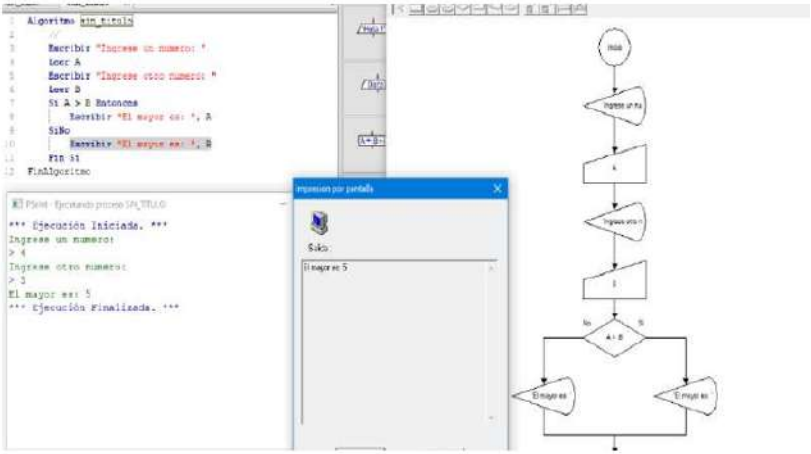


Figura 72. Algoritmo y diagrama de flujo para visualizar el Mayor de 2 números.

Estructuras condicionales anidadas y múltiples

Las estructuras condicionales comparan una variable contra otro(s) valor(es), para que, en base al resultado de esta comparación, se siga un curso de acción dentro del programa. Cabe mencionar que la comparación se puede hacer contra otra variable o contra una constante, según se necesite.

Una estructura condicional es anidada cuando por la rama del verdadero (SI) o el falso (NO) de una estructura condicional hay otra estructura condicional y así sucesivamente.

Estas tomas de decisión tienen la siguiente forma:

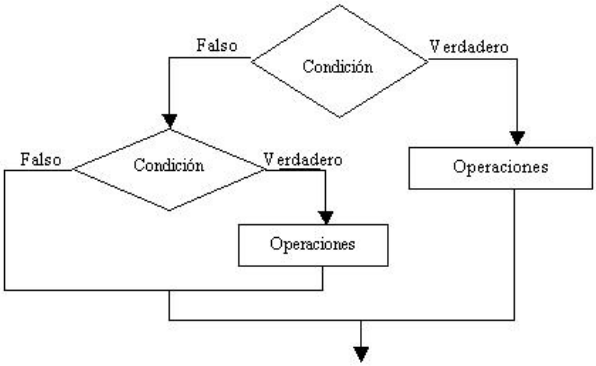


Figura 73. Diagrama de flujo Estructura anidada

Ejemplo 1:

Realizar un algoritmo y diagrama de flujo que permita ingresar la edad de una persona:

Si la edad es menor que 10 años visualizar "La persona es un niño",

Si la edad es mayor o igual que 10 y menor que 15 años visualizar "La persona es un joven",

Si la edad es mayor o igual que 15 y menor que 18 años visualizar "La persona es un adulto",

Si la edad es mayor o igual que 18 años visualizar "La persona es mayor de edad",

Si es mayor que 40 y menor que 60 años visualizar "La persona es adulta mayor" y

Si es mayor o igual que 60 años visualizar "La persona es un Anciano'

Figura 74. Metodología para resolver el ejemplo 1 con estructura anidada.

En la figura 75 se observa el algoritmo y diagrama de flujo para el ejemplo 1 de estructura anidada.

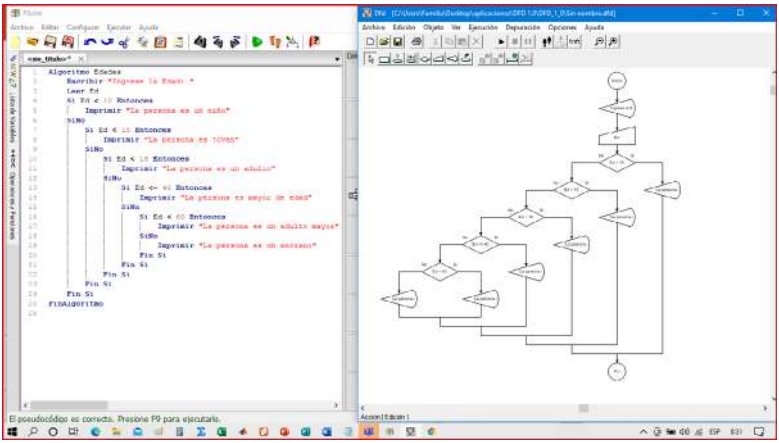


Figura 75. Algoritmo y diagrama de flujo para el ejemplo 1 de estructura anidada.

Multiples (según) o switch

La estructura múltiple o instrucción "Según" básicamente sirve para definir "casos" para cada valor que pueda tomar una variable; con el fin de ejecutar el bloque de código deseado cuando ese "caso" se cumpla.

Se representa de la siguiente forma como se observa en la figura 76:

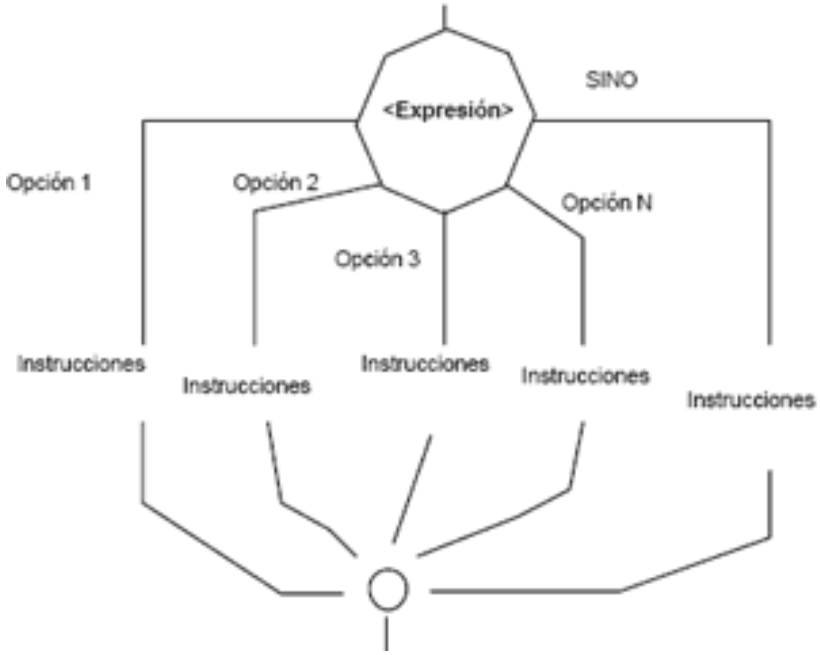


Figura 76. Diagrama de flujo de la estructura múltiple.

Las estructuras de comparación múltiples, es una toma de decisión especializada que permiten evaluar una variable con distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas.

Ejemplo

Realizar un algoritmo y diagrama de flujo que dado un número del 1 al 7, indique su equivalente en el día de la semana.

En la figura 77 se observa el algoritmo para el ejemplo planteado de la estructura múltiple:

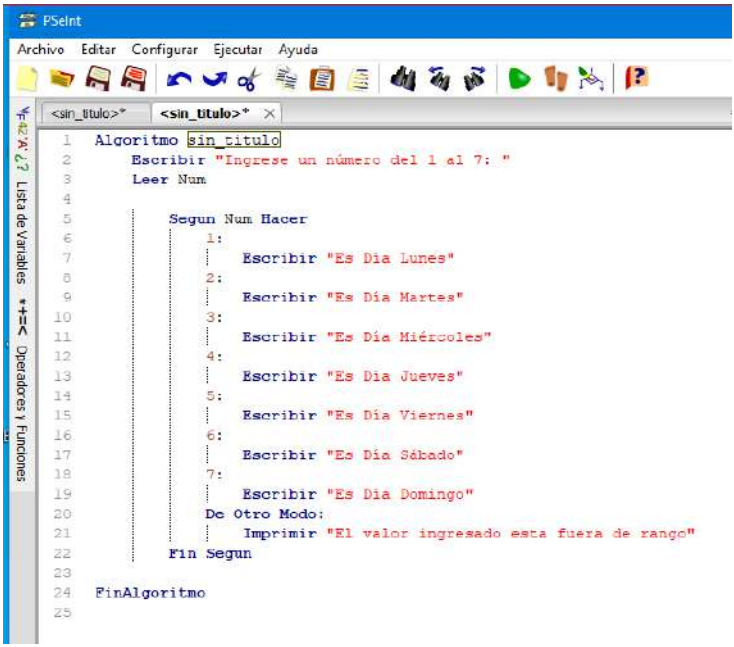


Figura 77. Algoritmo para el ejemplo de estructura múltiple.

En la figura 78. se observa el diagrama de flujo para el ejemplo planteado de la estructura múltiple:

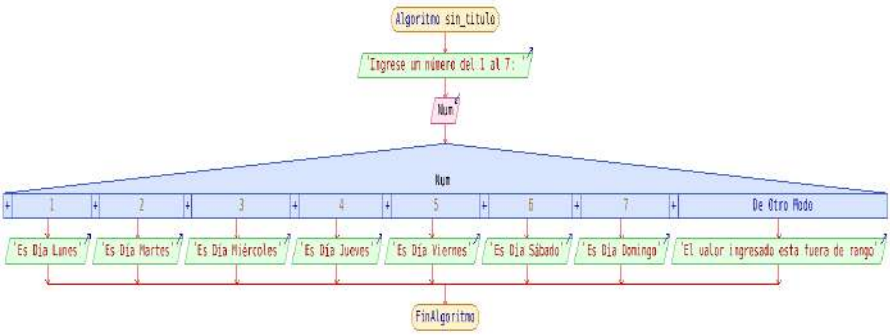


Figura 78. Diagrama de flujo para el ejemplo de la estructura múltiple.

Ejercicios de estructuras anidadas y estructuras múltiples

1. El problema consiste en determinar si UN estudiante (SÓLO UNO), aprueba, rinde examen supletorio o reprueba una asignatura, a través de la suma de las notas de los parciales en la carrera de ingeniería ambiental y el Exam Princ. Tienen oportunidad de rendir el examen supletorio aquellos que obtengan una suma mayor a 15 pero menor a 28, aprueba mayor o igual a 28 y reprueba menor que 15.

Solución:

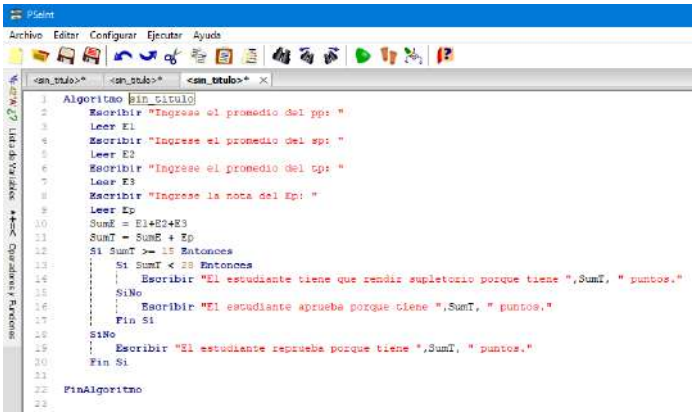


Figura 79. Algoritmo resuelto para el ejercicio 1

Como se puede observar en la figura 79. primero debemos ingresar los promedios de los parciales y el examen principal, luego sumamos y finalmente comparamos haciendo con estructuras anidadas.

En la figura 80 podemos observar el diagrama de flujo para el ejercicio 1 utilizando estructuras anidadas.

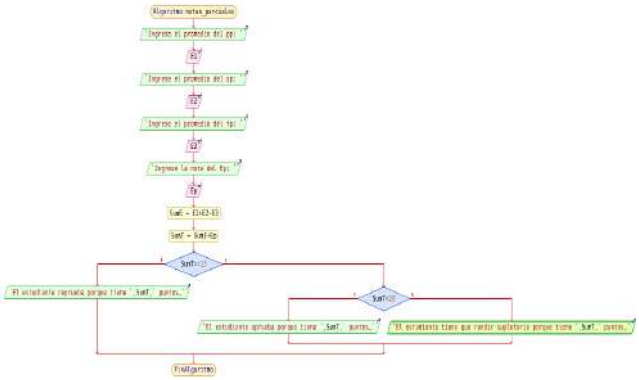


Figura 80. Diagrama de flujo para el ejercicio 1 usando estructura anidada.

2. Realizar un algoritmo que permita visualizar si una persona es niño, joven, adulto, maduro y anciano, basado en su edad.

Si la edad es menor o igual a 11 años es niño, si esta entre 12 y 17 es joven, si esta entre 18 y 27 es adulto, si esta entre 28 y 58 maduro y finalmente si es mayor o igual a 59 es anciano.

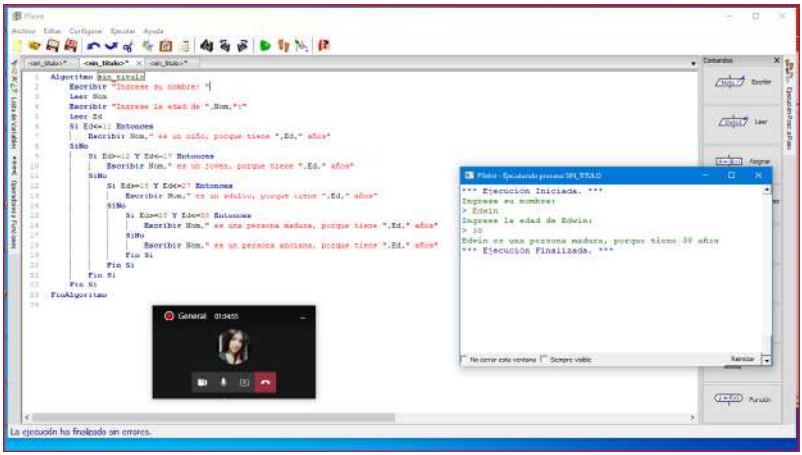


Figura 81. Algoritmo para el ejercicio 2 usando estructuras anidadas.

Como se puede observar en la figura 81 primero debemos ingresar primero el nombre de la persona y la edad de la misma, luego comparamos la edad haciendo con estructuras anidadas.

En la figura 82 podemos observar el diagrama de flujo para el ejercicio 2 utilizando estructuras anidadas.

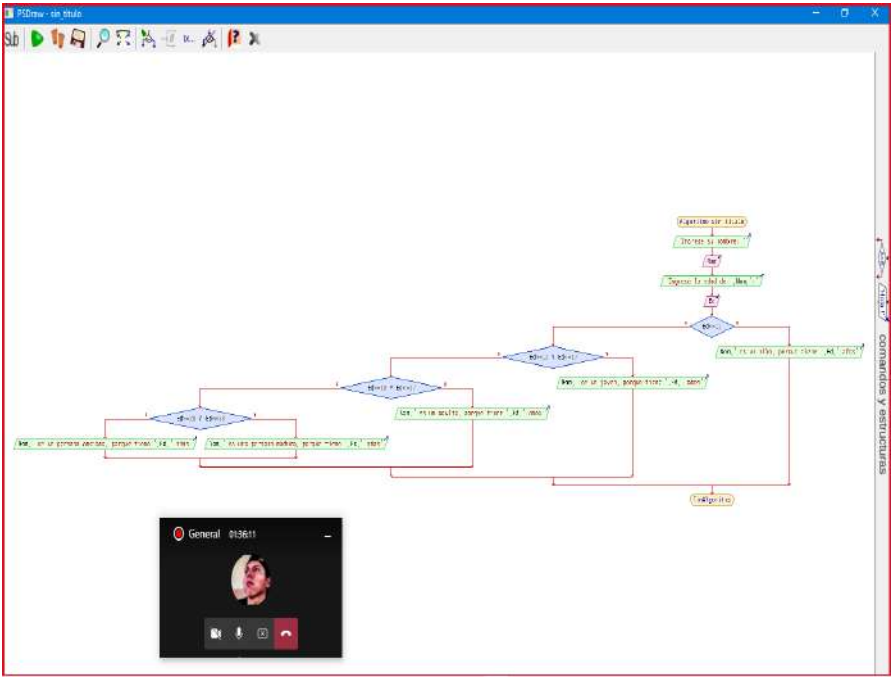


Figura 82. Diagrama de flujo para el ejercicio 2 usando estructuras anidadas.

1. Realizar un algoritmo que permita obtener un menú con 4 opciones:

- Suma de 2 números
- Mayor de 4 números
- Si es par o impar un número
- Salir

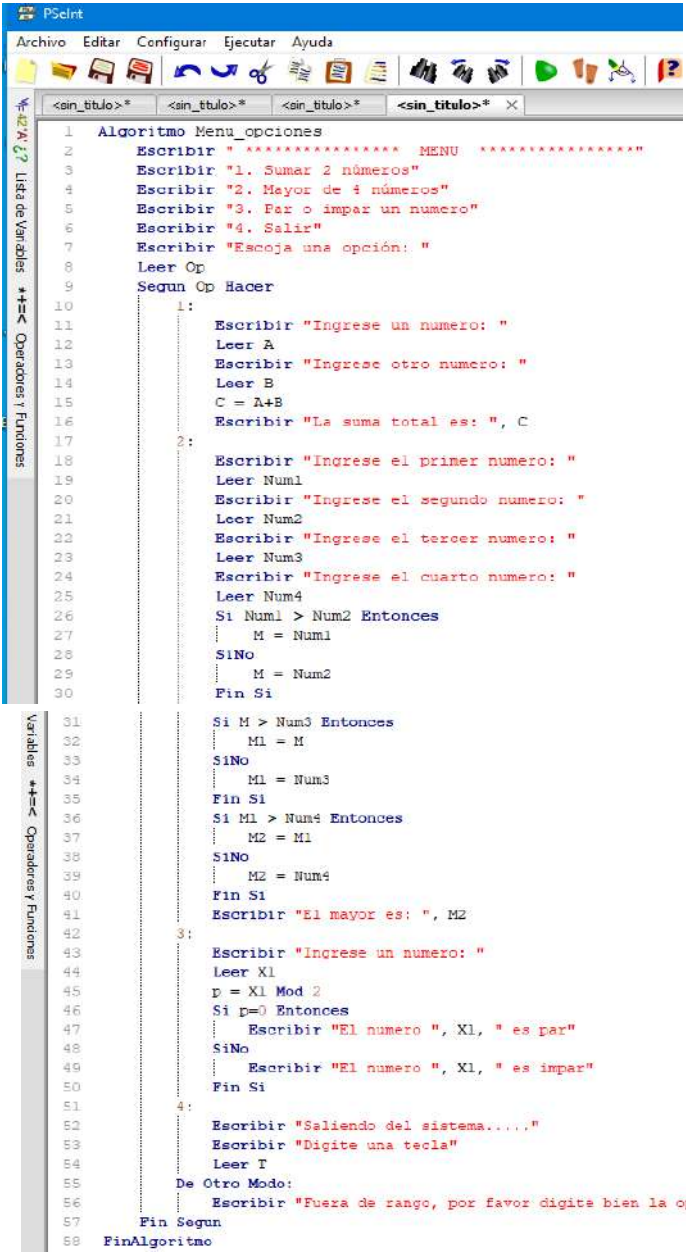


Figura 83. Algoritmo para el ejercicio 3 usando estructura múltiple.

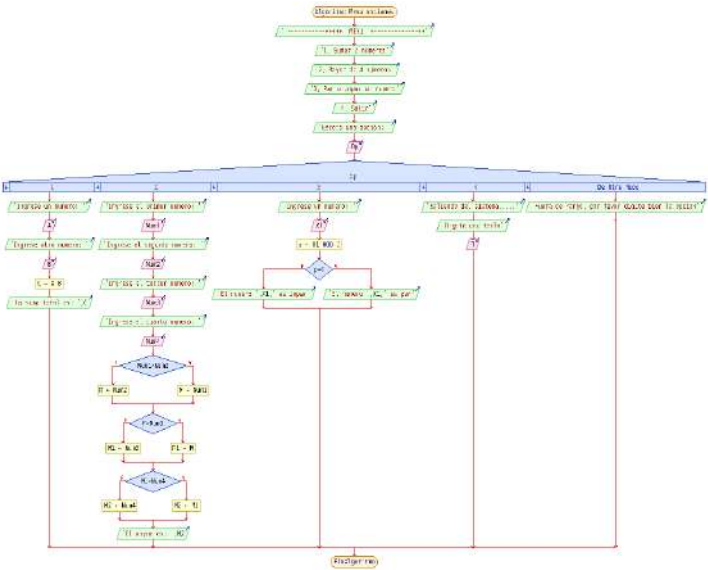


Figura 84. Diagrama de flujo para el ejercicio 3 usando estructura múltiple.

Ejercicios propuestos de estructuras simples, dobles y anidadas

Realizar un algoritmo y diagrama de flujo para los siguientes ejercicios con estructuras anidadas:

1. Se ingresan 5 números por teclado. Visualizar cual es el mayor y cuál es el menor de los números ingresados.
2. Se ingresa por teclado un valor entero, visualizar una leyenda que indique si el número es positivo, nulo o negativo además de verificar si es par o impar. Ejemplo: Ingreso el -3, entonces tendrá que visualizar el mensaje “El número es negativo impar”
3. Se ingresa un número entero positivo de hasta 3 cifras, visualizar indicando que tiene una, dos o tres cifras, en caso contrario visualizar “Error, está fuera de rango”.

4. Un postulante a un empleo, realiza un test de capacitación, se obtuvo la siguiente información: cantidad total de preguntas que se le realizaron y la cantidad de preguntas que contestó correctamente. Se pide confeccionar un programa que ingrese los dos datos por teclado e informe el nivel del mismo según el porcentaje de respuestas correctas que ha obtenido, y sabiendo que:

   Nivel máximo: Porcentaje>=90%.

   Nivel medio: Porcentaje>=75% y <90%.

   Nivel regular: Porcentaje>=50% y <75%.

   Fuera de nivel: Porcentaje<50%.

5. Se ingresan 10 números por teclado. Visualizar cual es el mayor y cuál es el menor de los números ingresados.
6. Realice un algoritmo que lea un número de tres cifras y determine si es igual al revés del número. Como el número tiene tres cifras, para que sea igual al revés, basta con que la cifra de las unidades sea igual a la cifra de las centenas. Por ejemplo: 353, 878, etc.
7. Una compañía dedicada al alquiler de automóviles cobra un monto fijo de $30000 para los primeros 300 km de recorrido. Para más de 300 km y hasta 1000 km, cobra un monto adicional de $ 1.500 por cada kilómetro en exceso sobre 300. Para más de 1000 km cobra un monto adicional de $ 1.000 por cada kilómetro en exceso sobre 1000 km. Los precios ya incluyen

el 12% del impuesto general a las ventas, IVA. Realice un algoritmo que determine el monto a pagar por el alquiler de un vehículo y el monto incluido del impuesto.

8. Realice un algoritmo que determine quienes son contemporáneos entre Juan, Mario y Pedro, debe ingresar su edad y comparar las mismas.
9. Realice un algoritmo que lea tres longitudes y determine si forman o no un triángulo. Si es un triángulo determine de que tipo de triángulo se trata entre: equilátero (si tiene tres lados iguales), isósceles (si tiene dos lados iguales) o escaleno (si tiene tres lados desiguales). Considere que para formar un triángulo se requiere que: "el lado mayor sea menor que la suma de los otros dos lados".

Ejercicios propuestos de estructuras múltiples

Realizar un algoritmo y diagrama de flujo para los siguientes ejercicios con estructuras MULTIPLES:

1. Realizar un algoritmo que permita obtener el siguiente MENU de opciones: 1. Suma de 3 números, 2. mayor de 3 números, 3. saber si es par o impar y 4. salir.
2. Realizar un algoritmo que obtenga el siguiente Menú: a) cálculo del área de un rectángulo, b) cálculo del área de un triángulo, c) cálculo del área de una circunferencia y d) salir.
3. Realizar un algoritmo para obtener el siguiente Menú: i) Lea un número entero de 3 cifras, y forme el mayor número posible con

las cifras del número ingresado. El número formado debe tener el mismo signo que el número ingresado, ii) Lea un número de tres cifras y determine si es igual al revés del número, iii) Lea un número entero positivo de hasta tres cifras y muestre un mensaje indicando si tiene 1, 2, o 3 cifras. Mostrar un mensaje de error si el número de cifras es mayor y iv) salir.

4. Realice un menú con las siguientes opciones: 1. Obtener las raíces de una ecuación de segundo grado, 2) calcule el perímetro de un romboide, 3) ingrese tres números y calcule cuadrado, cubo, cuarta y a la quinta de los mismos, 4) Salir.
5. Realice un menú de 5 opciones, plantear Ud. el ejercicio.

Estructuras repetitivas

Las estructuras de control repetitivas, son aquellas que permiten ejecutar un conjunto de instrucciones varias veces, de acuerdo al valor que genere la expresión relacional y/o lógica. Esto significa que una instrucción repetitiva permite saltar a una instrucción anterior para volver a ejecutarla.

Para que se usan

En Ingeniería estadística se utiliza para automatizar procesos estadísticos, innovar aplicaciones en la estadística y otros más.

Tipos de estructuras repetitivas

Existen 3 estructuras repetitivas y son:

- Mientras o while
- Para o For

- Repetir hasta que

Estructuras repetitivas mientras o while

La estructura repetitiva Mientras, es una estructura de control repetitiva que realizará la repetición de un conjunto de instrucciones si y solo si la condición es VERDADERA en caso contrario termina la ejecución de un conjunto de instrucciones, si la evaluación de la expresión relacional y/o lógica es FALSA.

Funcionamiento

En primer lugar se verifica la condición, si la misma resulta verdadera se ejecutan las operaciones que indicamos entre la palabra Mientras y las palabras Fin Mientras. En caso que la condición sea Falsa continúa con la instrucción siguiente al bloque Mientras. El bloque se repite MIENTRAS la condición sea Verdadera.

Importante: Si la condición siempre retorna verdadero estamos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación, nunca finalizará el programa. Para ello se debe controlar con una variable de tipo CONTADOR.

Estructura repetitiva mientras – sintaxis en pseint

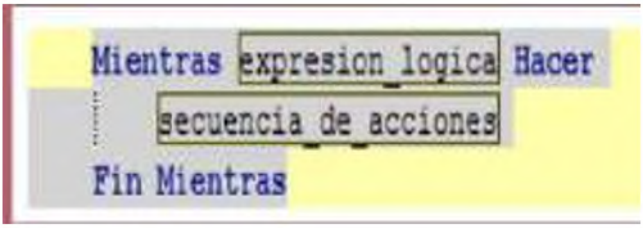


Figura 85. Sintaxis estructura mientras o while.

Variable contador

Son variables enteras que se incrementan (+) o decrementan (-) con un valor constante, por ejemplo una variable CONT, cuyo valor se incrementa en uno; se conoce como variable contador. La técnica es:

EJEMPLO: Inicializar una variable a cero o uno antes del ciclo repetitivo. Dentro del ciclo repetitivo, incrementar en uno la variable.

CONT <- 1;

Mientras (CONT <= 3) Hacer

sentencia1 ……..

CONT= CONT + 1 // No olvidar insertar esta línea por favor

Fin Mientras

Ejemplo de contador:

Por ejemplo si se desea sumar los números del 1 al 5, se necesitará una variable que genere esos números, es decir que empiece en 1 y llegue hasta el 5.

Figura 86. Ejemplo de variable contador.

En la figura 86 se puede observar que la variable que cumple el rol de contador, aparece tanto a la izquierda como a la derecha, por la propiedad destructiva de la asignación; así tomará el valor anterior, le adicionará o reducirá el valor constante y asignará el nuevo valor.

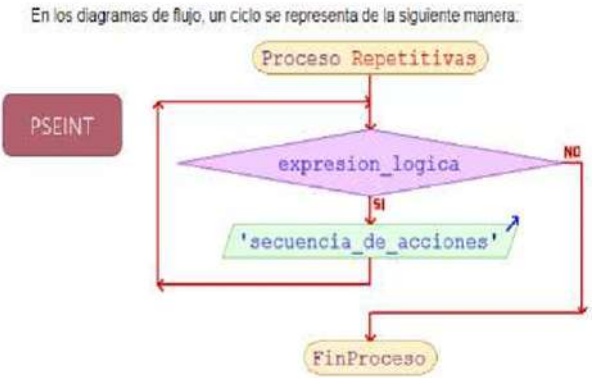


Figura 87. Diagrama de flujo para la estructura repetitiva mientras o while.

Como podemos observar en la figura 87 en el diagrama de flujo si la condición es verdadera el ciclo continúa o se repite las veces que sea necesario hasta que la condición sea falsa en cuyo caso sale de la estructura.

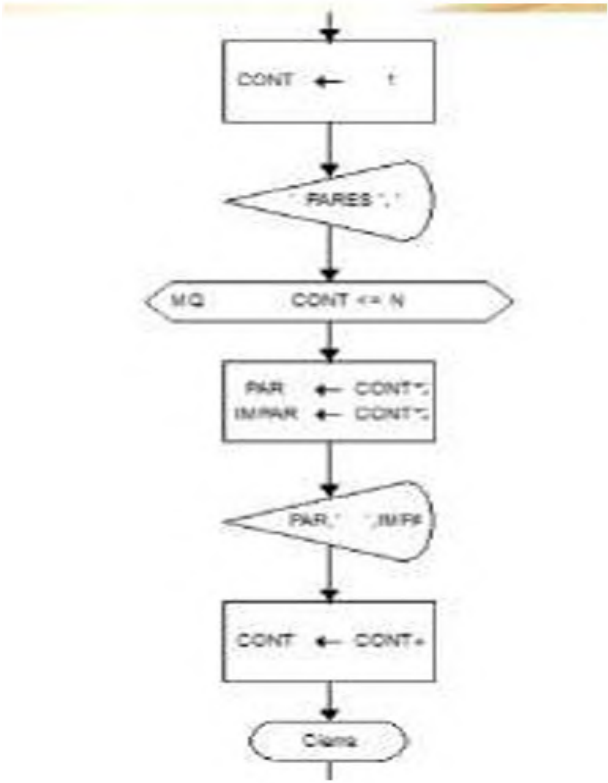


Figura 88. Estructura repetitiva mientras usando en DFD.

En la figura 88 se observa el objeto que cumple con la estructura repetitiva mientras y es MQ. Este objeto tenemos en la barra de herramientas de objetos del software DFD, como se observa en la figura 89.

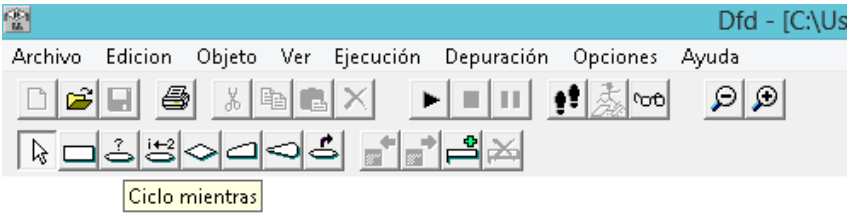


Figura 89. Objeto de la estructura repetitiva mientras

Ejemplo 1:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los 3 primeros números pares.

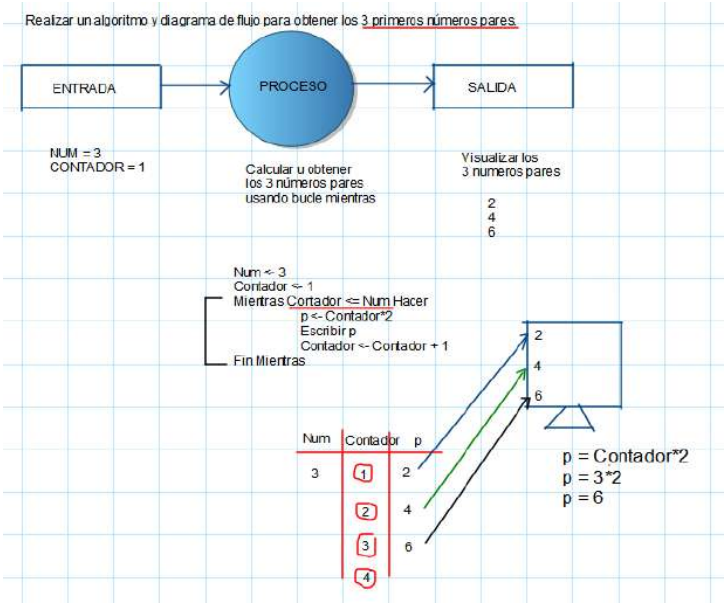


Figura 90. Metodología para resolver el ejemplo 1 con estructura repetitiva mientras.

Algoritmo tres_primeros_pares

Num = 3

Contador = 1

Mientras Contador <= Num Hacer

*p = Contador*2*

Escribir p

Contador = Contador + 1

FinMientras

FinAlgoritmo

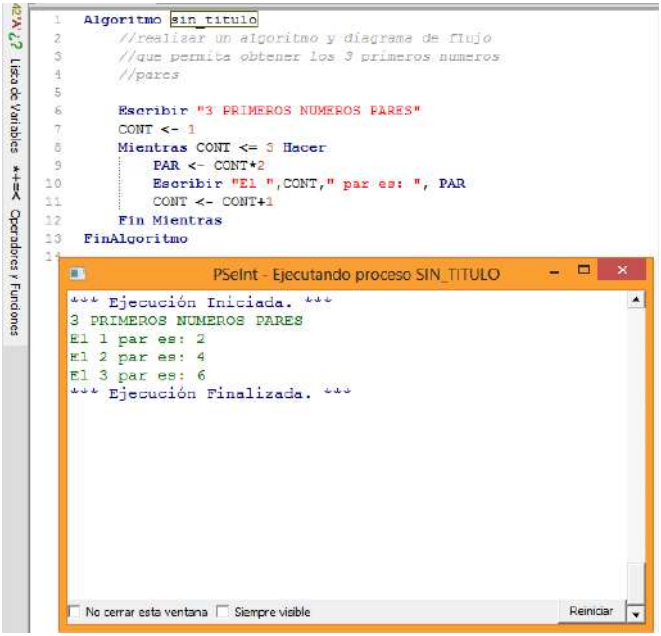


Figura 91. Algoritmo para resolver el ejemplo 1 con estructura repetitiva mientras

En la figura 91 se puede observar el algoritmo realizado en PSeInt usando la estructura repetitiva mientras para resolver el ejemplo 1.

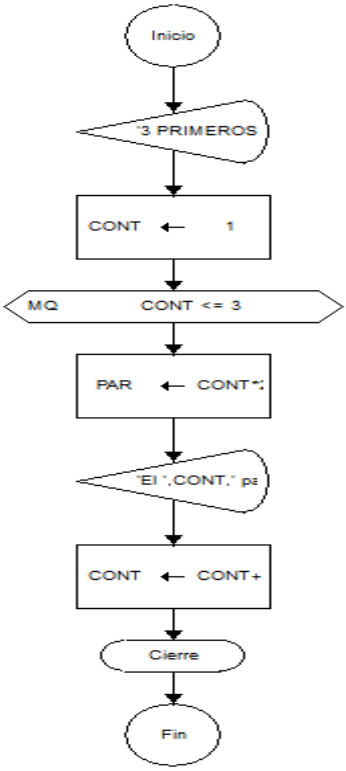


Figura 92. Diagrama de flujo usando la estructura repetitiva mientras en DFD para el ejemplo 1.

Otra forma de obtener los 3 primeros pares usando el residuo o MOD.

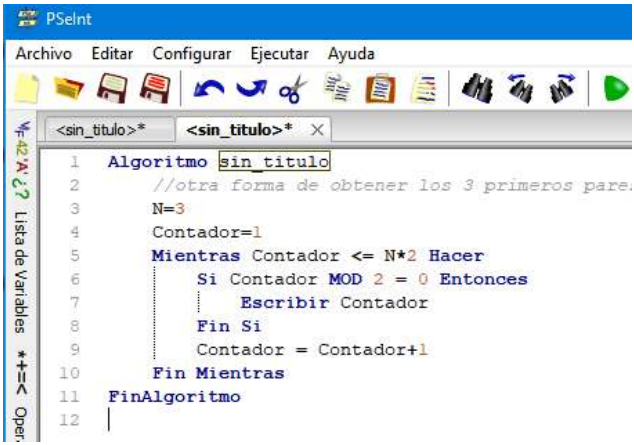


Figura 93. Uso de Mod para resolver el ejemplo 1.

Ejemplo 2:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los 3 primeros números impares.

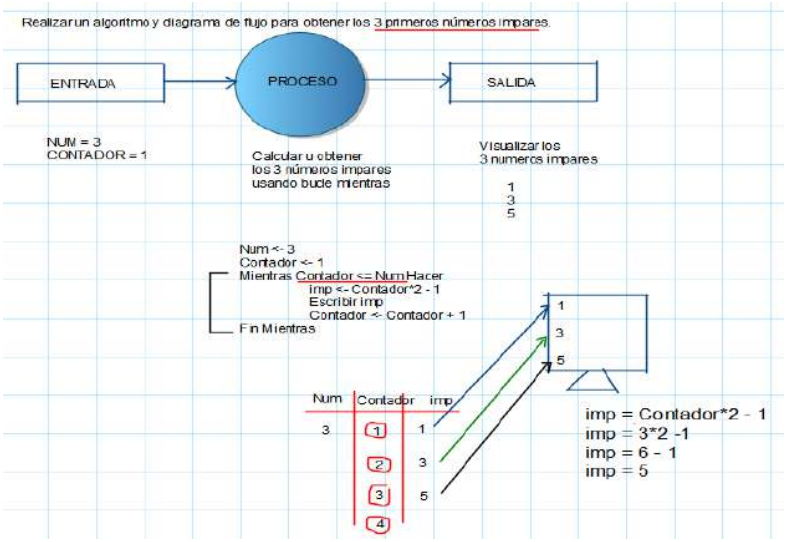


Figura 94. Metodología para resolver el ejemplo 2

Algoritmo tres_primeros_impares

Num=3

Contador = 1

Mientras Contador <= Num Hacer

*p = Contador*2 - 1*
Escribir p

Contador = Contador + 1

FinMientras

FinAlgoritmo

En la figura 95 se observa el algoritmo y diagrama de flujo para resolver los 3 primeros números impares.

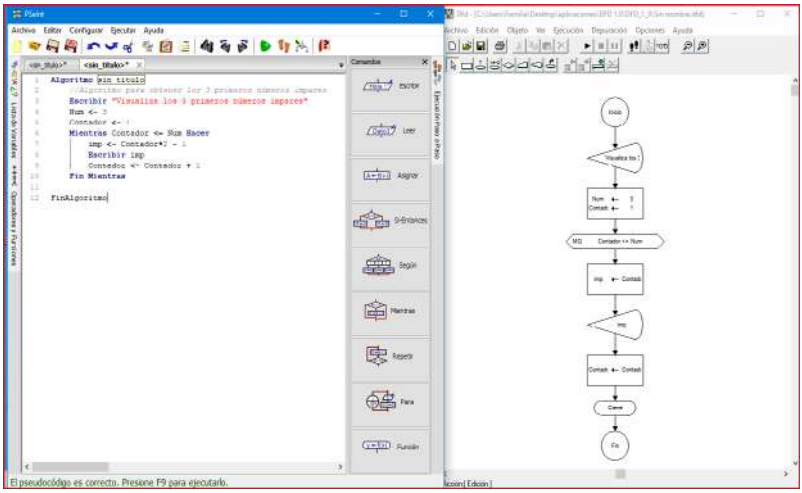


Figura 95. Algoritmo y diagrama de flujo para resolver el ejemplo 2.

Ejemplo 3:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los N primeros números pares

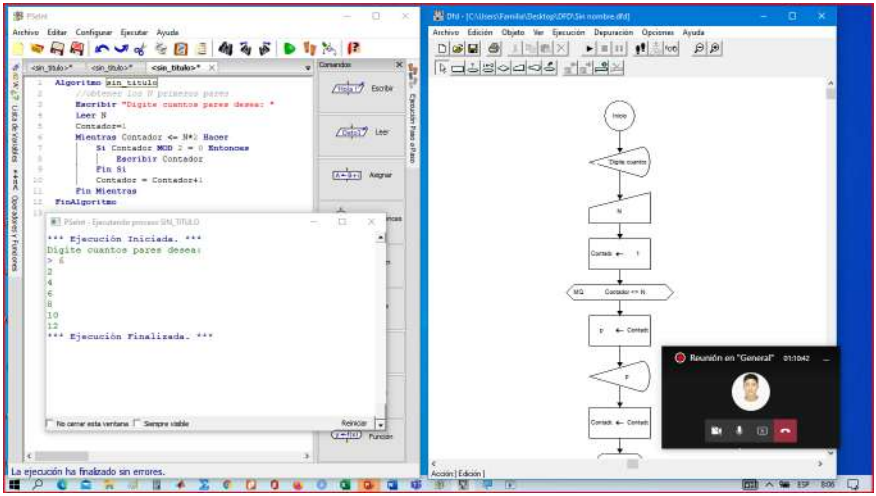


Figura 96. Algoritmo y diagrama de flujo para el ejemplo 3.

Ejemplo 4:

Realizar un diagrama de flujo que permita OBTENER los N primeros números pares y N primeros números impares, en columnas.

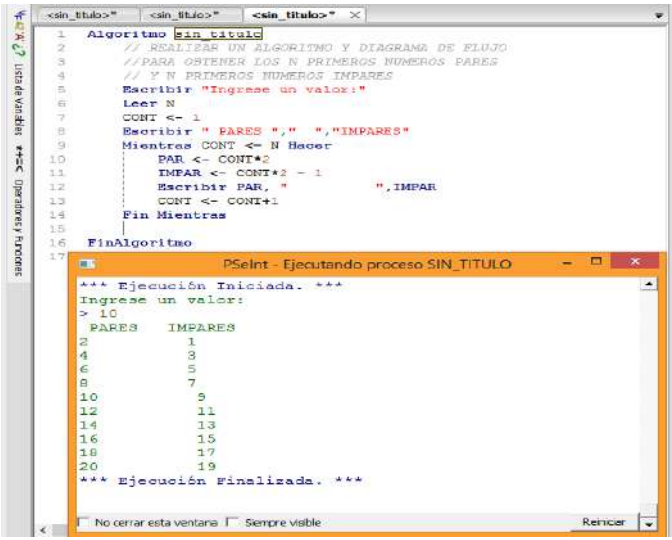


Figura 97. Algoritmo para el ejemplo 4 usando estructura repetitiva mientras.

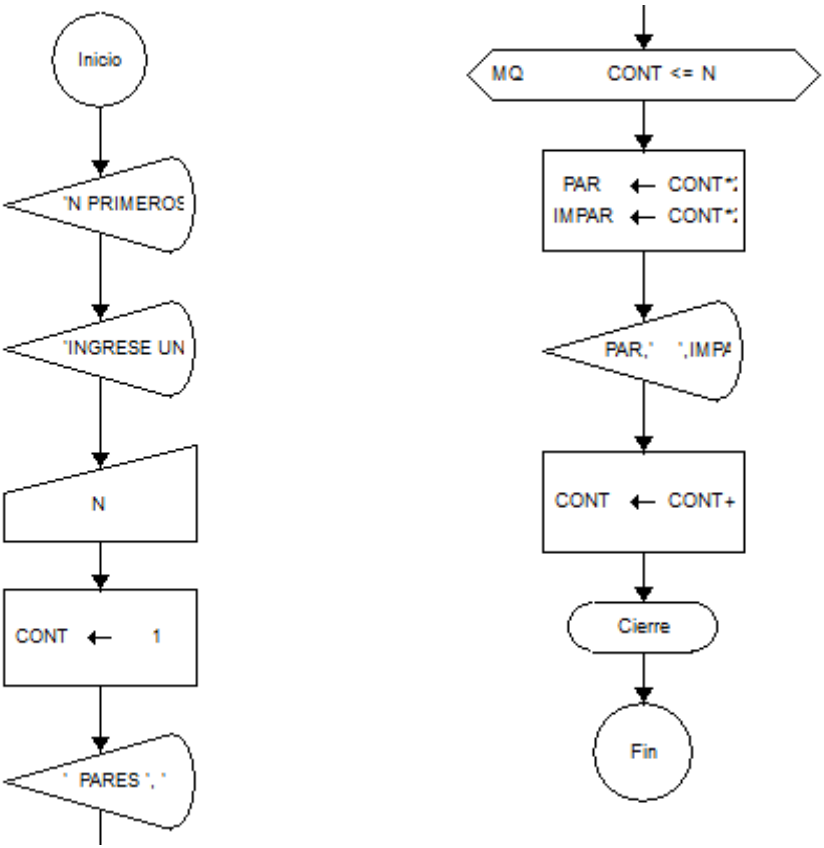


Figura 98. Diagrama de flujo para el ejemplo 4 usando estructura mientras.

Como se observa en la figura 97 y 98 tenemos el uso de la estructura repetitiva mientras para resolver el ejemplo 4.

Ejercicios propuestos de estructuras repetitivas mientras

Para los siguientes ejercicios realizar el algoritmo y diagrama de flujo:

1. Obtener el factorial de un número
2. Obtener los N primeros múltiplos de 5
3. Obtener los N primeros múltiplos de 7

4. Obtener los N primeros múltiplos de 9
5. Para saber si un número es primo o no
6. Para saber si un número es compuesto o no
7. Obtener los N primeros números primos
8. Obtener los N primeros números compuestos

Estructura repetitiva para o for

La estructura repetitiva Para o For es aquella en la que el número de iteraciones se conoce por anticipado, y por ello no se precisa poner ninguna condición de salida para detener el bucle. En su lugar un contador cuenta el número de iteraciones fijas y se termina cuando llega al valor final previamente definido.

Funcionamiento

Funcionamiento de la estructura PARA: En primer lugar se verifica el valor_inicial, luego el Valor_final y por último el Paso que significa incremento o decremento del contador, se repite la estructura con su correspondiente secuencia de acciones hasta que le valor_inicial llegue al valor_final.

Estructura repetitiva para – sintaxis en pseint

```
1  Algoritmo sin_titulo
2      Para variable_numerica<-valor_inicial Hasta valor_final Con Paso paso Hacer
3          secuencia_de_acciones
4      Fin Para
5  FinAlgoritmo
```

Figura 99. Sintaxis en PSeInt de la estructura repetitiva Para o For

Variable contador

Son variables enteras que se incrementan (+) o decrementan (-) con un valor constante, por ejemplo una variable CONT, cuyo valor se incrementa en uno; se conoce como variable contador. La técnica es:

Ejemplo:

Inicializar la variable CONT dentro del ciclo Para o For, decir hasta donde va ir valor_final.

e incrementar en 1a la variable inicial o CONT eso se hace en Paso.

Para CONTADOR<-1 Hasta 3 Con Paso 1 Hacer

SENTENCIAS

……..

Fin Para

Tener claro que Con Paso en la estructura Para quiere decir como en la estructura mientras CONT = CONT+1.

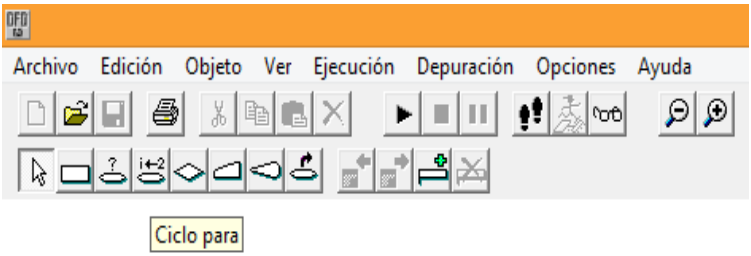


Figura 100. Ciclo Para en el software DFD

En la figura 100 se observa el objeto que se usa para el ciclo Para dentro del software DFD.

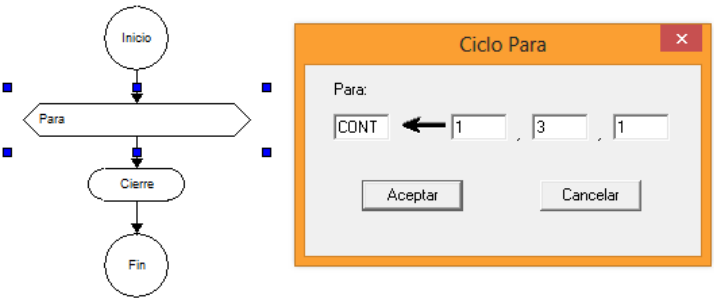


Figura 101. Ciclo para en el software DFD uso del objeto y llenado de valores.

En la figura 101 se observa el objeto Para ya insertado dentro del diagrama de flujo y su llenado de valores tal cual como se muestra.

Ejemplo 1:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los 3 primeros números pares.

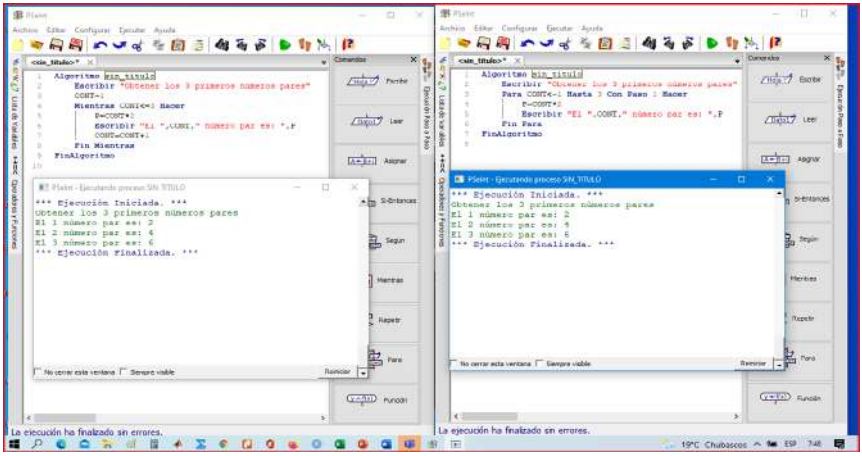


Figura 102. Algoritmo usando la estructura mientras y su correspondiente con la estructura Para en la resolución de los 3 primeros pares.

En la figura 102 se puede observar cómo se usa la estructura Para o For pasando desde la estructura mientras.

Nótese que la variable contador en la estructura Mientras o While se inicializa en 1 antes de que empiece la estructura y dentro de la misma se incrementa en 1, en cambio en la estructura Para o For donde se inicia la estructura se inserta la variable contador con su valor inicial en este caso igual a 1, luego hasta donde va que es 3 y finalmente en ConPaso se inserta el valor del incremento en este caso 1.

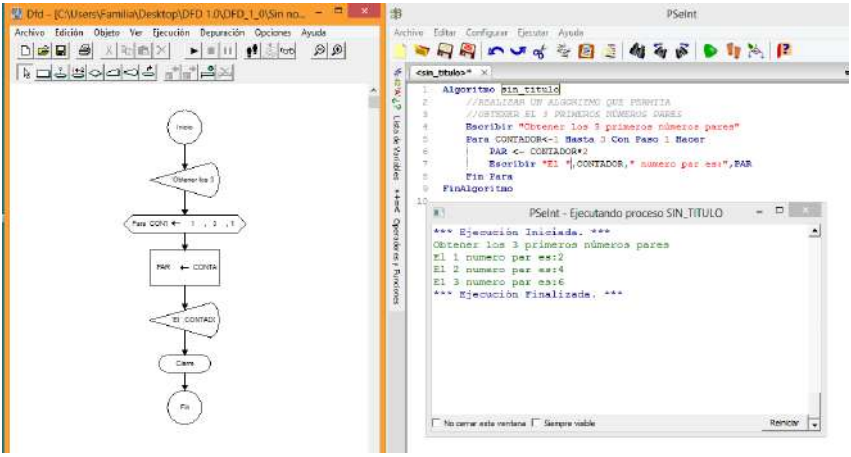


Figura 103. Diagrama de flujo y algoritmo para 3 pares usando la estructura Para o For

Ejemplo 2:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los 3 primeros números impares.

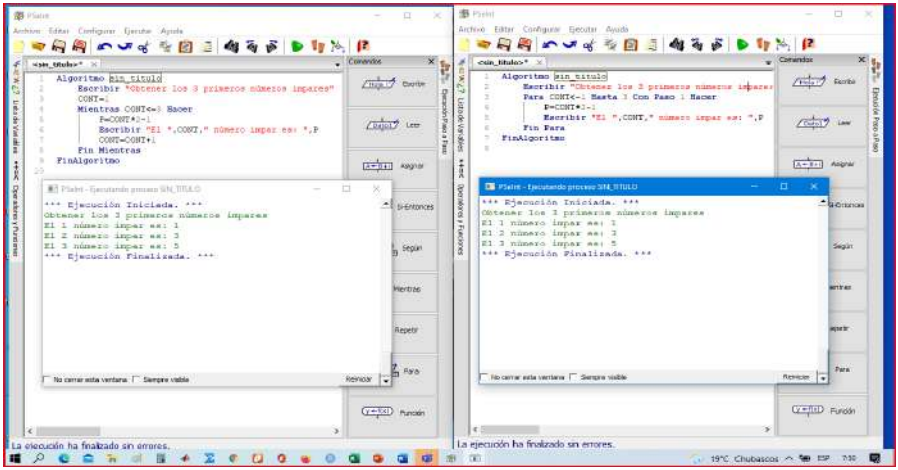


Figura 104. Algoritmo usando la estructura mientras y su correspondiente con la estructura Para en la resolución de los 3 primeros impares.

En la figura 104 se puede observar cómo se usa la estructura Para o For pasando desde la estructura mientras para resolver los 3 primeros impares.

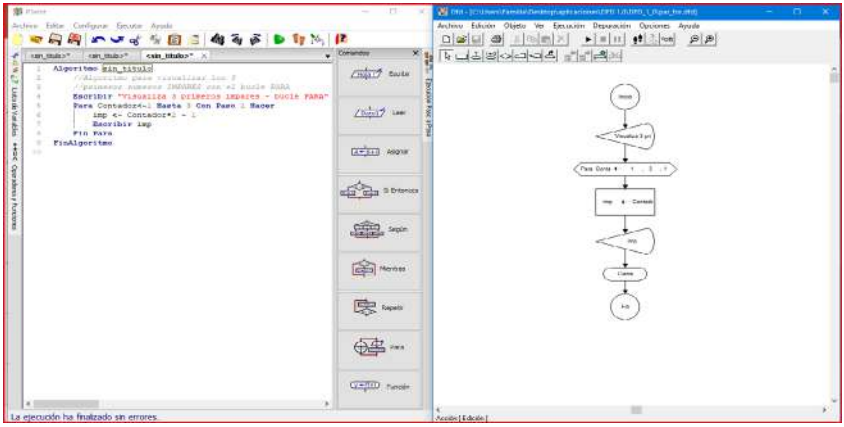


Figura 105. Algoritmo y diagrama de flujo para resolver los 3 primeros impares usando la estructura repetitiva Para o For.

Ejemplo 3:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los N primeros números pares e impares en columnas utilizando el bucle PARA.

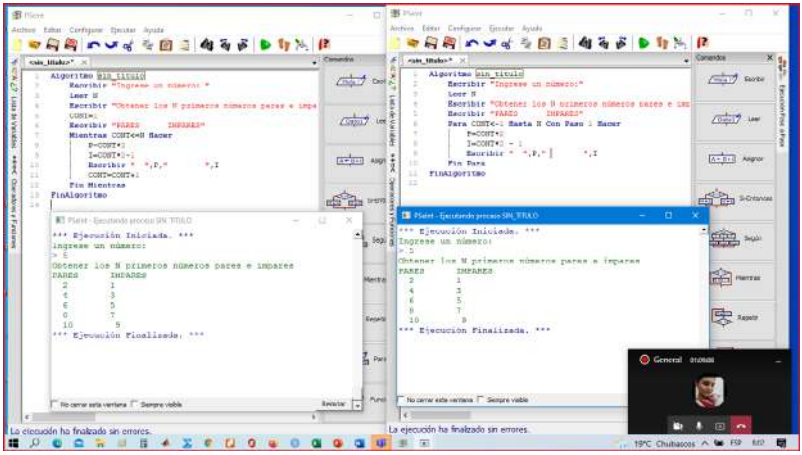


Figura 106. Algoritmo para el ejempo3 con estructura mientras y estructura para.

Ejemplo 4:

Realizar un algoritmo y diagrama de flujo que permita ingresar 5 números y solo sume los números pares.

Variable acumulador

Un acumulador es una variable numérica que permite ir acumulando operaciones. Me permite ir haciendo operaciones parciales. Un acumulador:

Se inicializa a un valor inicial según la operación que se va a acumular: a 0 si es una suma o a 1 si es un producto. Se acumula un valor variable.

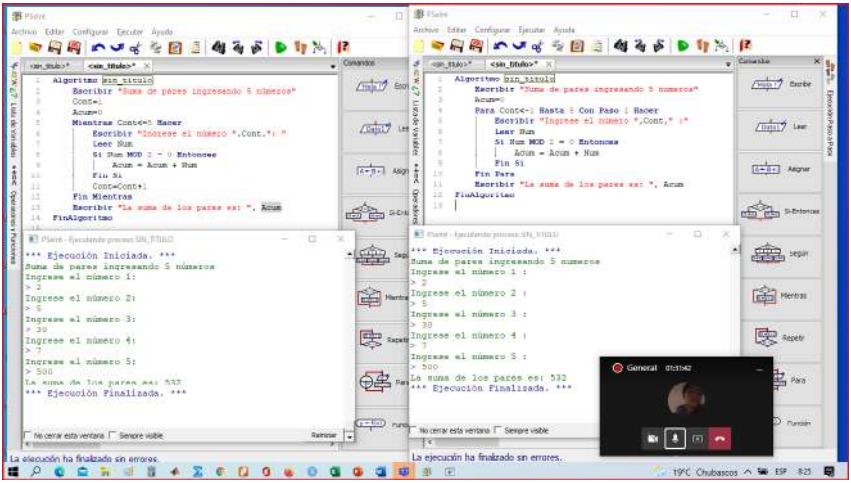


Figura 107. Algoritmo para resolver el ejemplo 4 usando estructura mientras y para.

Ejemplo 5:

Realizar un algoritmo y diagrama de flujo que permita OBTENER el factorial de un número.

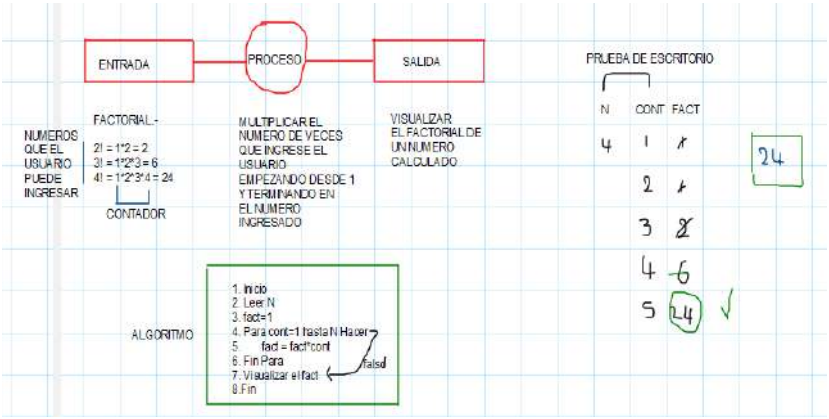


Figura 108. Metodología para resolver el ejemplo 5 usando estructura repetitiva Para.

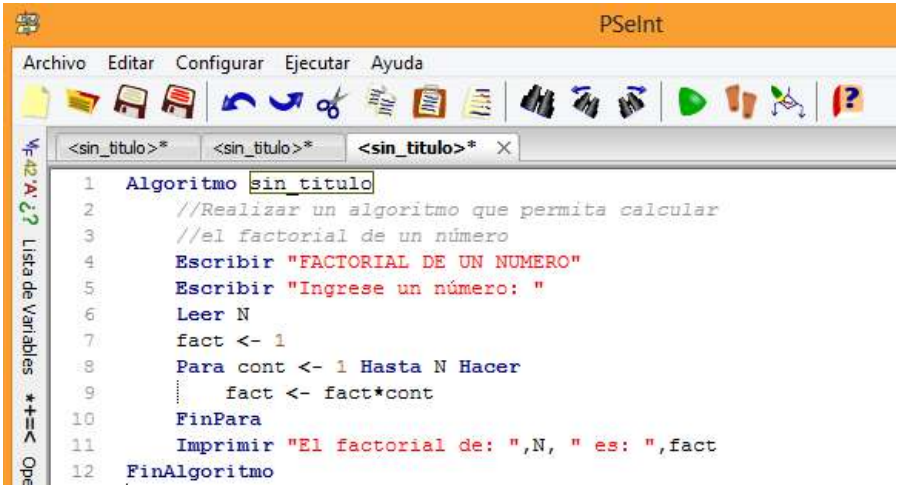


Figura 109. Algoritmo para resolver el factorial de un número usando estructura Para.

Ejercicios propuestos de estructura repetitiva para o for

Para los siguientes ejercicios realizar el algoritmo y diagrama de flujo:

a. Obtener el factorial de un número

b. Obtener los N primeros múltiplos de 5

c. Obtener los N primeros múltiplos de 7

d. Obtener los N primeros múltiplos de 9

e. Para saber si un número es primo o no

f. Para saber si un número es compuesto o no

g. Obtener los N primeros números primos

h. Obtener los N primeros números compuestos

Estructura repetir hasta que

La estructura repetitiva REPETIR HASTA de PSeInt, es una estructura de control repetitiva que realizará la repetición de un conjunto de instrucciones hasta que la condición sea FALSA, en caso contrario termina la ejecución, si la evaluación de la expresión relacional y/o lógica es VERDADERA. Esta estructura la cual ejecuta al menos una vez su bloque repetitivo, a diferencia del while o del for que podrían no ejecutar el bloque. Esta estructura repetitiva se utiliza cuando conocemos de antemano que por lo menos una vez se ejecutará el bloque repetitivo.

Funcionamiento

En primer lugar se realiza el conjunto de instrucciones, luego se verifica la condición. Si la misma resulta FALSA se ejecutan de nuevo las operaciones que indicamos entre la palabra Repetir y las palabras Hasta Que. En caso que la condición sea Verdadera se termina el bloque Repetir. El bloque se repite hasta que la condición sea Verdadera.

Importante: Si la condición siempre retorna falso estamos en presencia de un ciclo repetitivo infinito. Dicha situación es un error de programación, nunca finalizará el programa. Para ello se debe controlar con una variable de tipo CONTADOR.

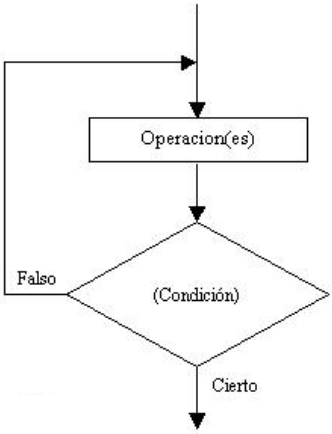


Figura 110. Diagrama de Flujo de la estructura Repetir - Hasta Que

En la figura 110. se muestra como es el ciclo de la estructura repetitiva Repetir-Hasta Que en un diagrama de flujo.

Estructura repetir – hasta que – sintaxis en pseint

```
Repetir
    secuencia_de_acciones
Hasta Que expresion_logica
```

Figura 111. Sintaxis PSeInt de estructura Repetir – Hasta Que.

En la figura 111. se muestra como es la sintaxis de la estructura repetitiva Repetir-Hasta Que, la misma existe en PSeInt.

Variable contador

Aquí la variable contador debe inicializarse antes de la estructura, por lo general con 1, depende mucho del proceso que se esté realizando, puede empezar también en cero.

EJEMPLO: Inicializar una variable a cero o uno antes del ciclo repetitivo. Dentro del ciclo repetitivo, incrementar en uno la variable.

CONT <- 1;
Repetir
secuencia_de_acciones
CONT=CONT+1 // No olvidar insertar esta línea por favor
Hasta Que (CONT>3) //Recuerde esta condición debe ser falsa para
//que se repita la estructura

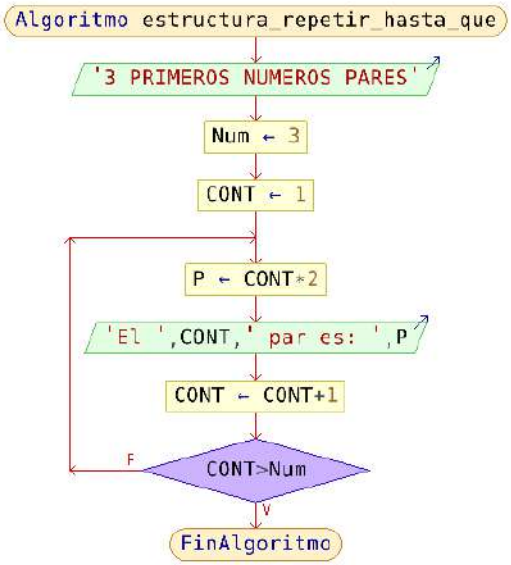


Figura 112. Diagrama de flujo para la estructura repetitiva Repetir-Hasta Que.

Como podemos observar en la figura 112 en el diagrama de flujo si la condición es falsa el ciclo continúa o se repite las veces que sea necesario hasta que la condición sea verdadera en cuyo caso sale de la estructura. También se puede observar que por lo menos ingresara a la estructura una vez.

En DFD como se observa en la figura 113 la estructura repetitiva Repetir-Hasta Que no existe. Este objeto tenemos solo en PSeInt y también en los lenguajes de programación como R, C++ y otros pero como bucle o estructura Do While.

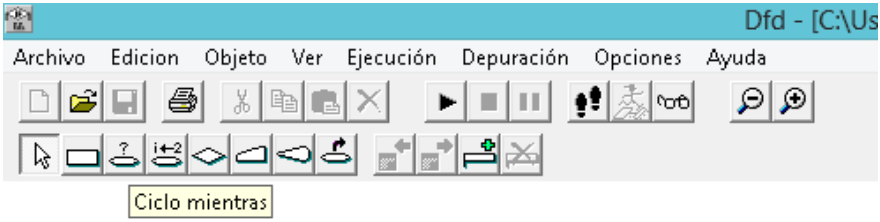


Figura 113. Objeto de la estructura repetitiva mientras

Ejemplo 1:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los 3 primeros números pares.

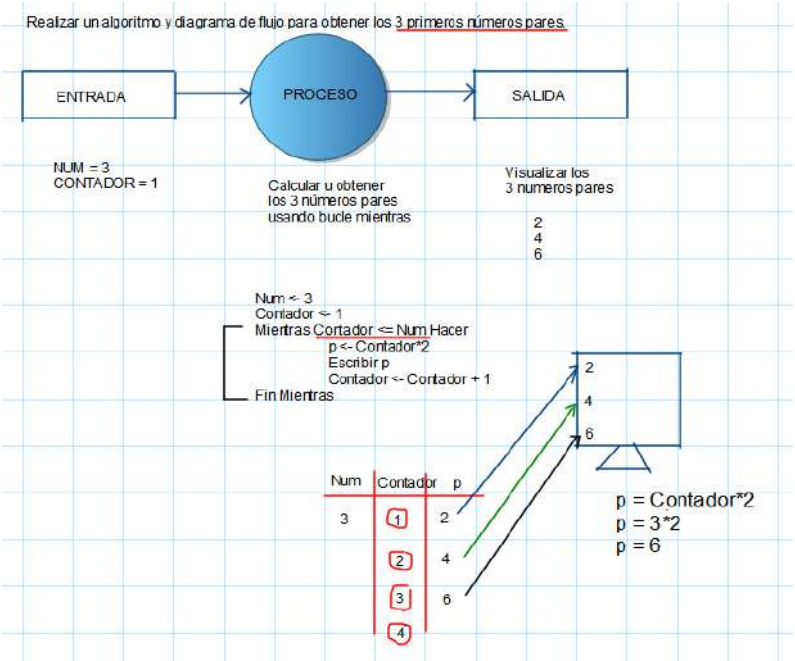


Figura 114. Metodología para resolver el ejemplo 1 con estructura Repetir-Hasta Que.

Algoritmo tres_primeros_pares

Num = 3

CONT<-1

Repetir

*p<-CONT*2*

Escribir P

CONT<-CONT+1

Hasta Que CONT > Num

FinAlgoritmo

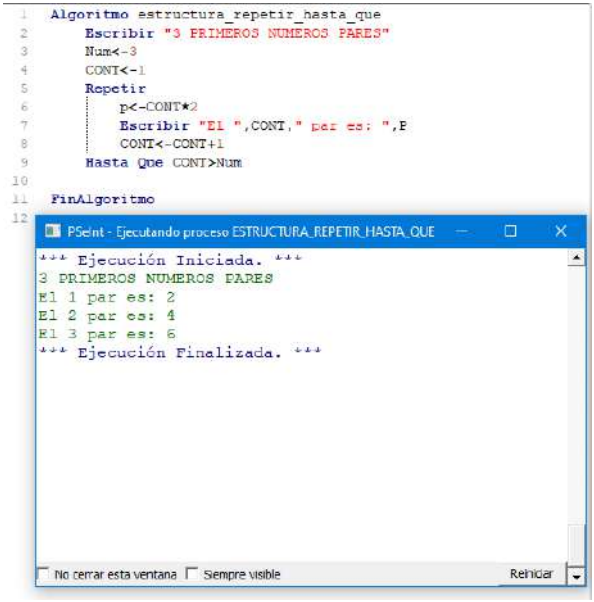


Figura 115. Algoritmo para el ejemplo1 con estructura repetir_hasta_que

En la figura 115 se puede observar el algoritmo realizado en PSeInt usando la estructura repetitiva “repetir hasta que” para resolver el ejemplo 1.

Otra forma de obtener los 3 primeros pares usando el residuo o MOD.

```
1   Algoritmo Uso_Mod_para_pares
2       Escribir "3 PRIMEROS NUMEROS PARES"
3       Num<-3
4       CONT<-1
5       K<-1
6       Repetir
7           Si CONT MOD 2 = 0 Entonces
8               Escribir "El ",K," par es: ",CONT
9               k<-k+1
10          Fin Si
11          CONT<-CONT+1
12      Hasta Que CONT>Num*2
13  FinAlgoritmo
14
```

Figura 116. Uso de Mod para resolver el ejemplo 1.

Como se puede observar en la figura 116 usamos la estructura repetitiva "repetir hasta que" con el MOD dentro de la misma, el contador K lo que hace es contar los pares que vamos obteniendo. Darse cuenta que en la condición insertamos Num*2 para que pueda ir el CONT hasta 6.

Ejemplo 2:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los 3 primeros números impares.

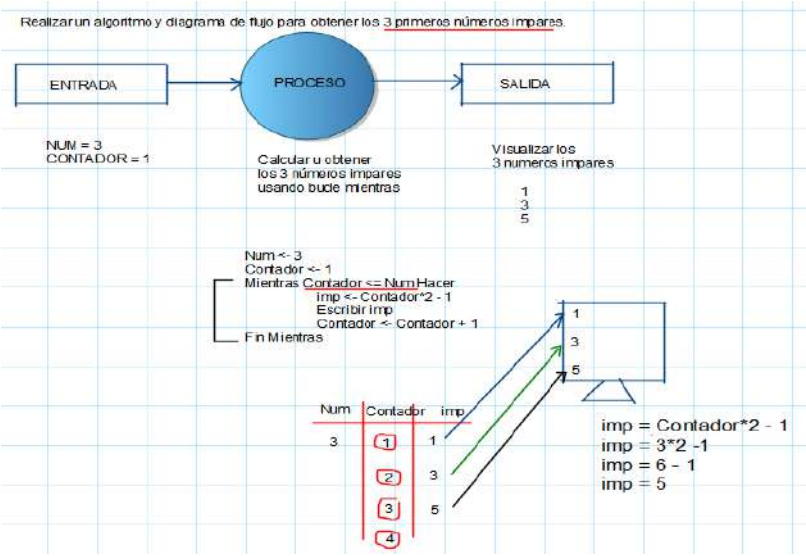


Figura 117. Metodología para resolver el ejemplo 2

Algoritmo tres_primeros_impares

Num = 3

CONT<-1

Repetir

*Imp<-CONT*2 - 1*

Escribir Imp

CONT<-CONT+1

Hasta Que CONT> Num

FinAlgoritmo

En la figura 117 se observa la metodología usada para construir el algoritmo para los 3 primeros números impares usando la estructura repetitiva “repetir hasta que”.

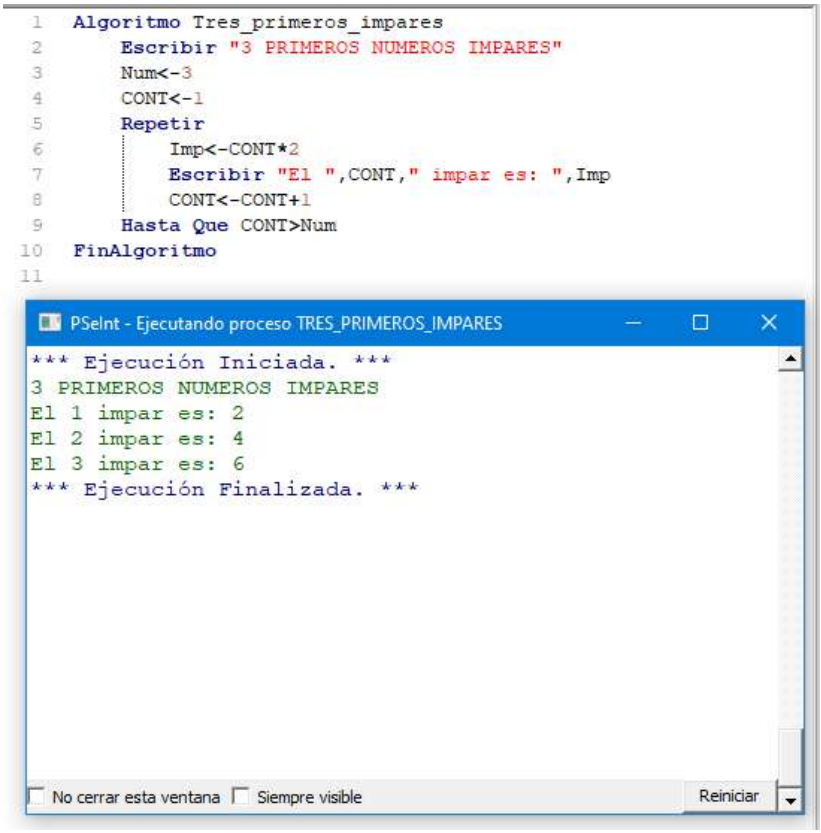


Figura 118. Algoritmo para resolver el ejemplo 2 con repetir hasta que.

Ejemplo 3:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los N primeros números pares

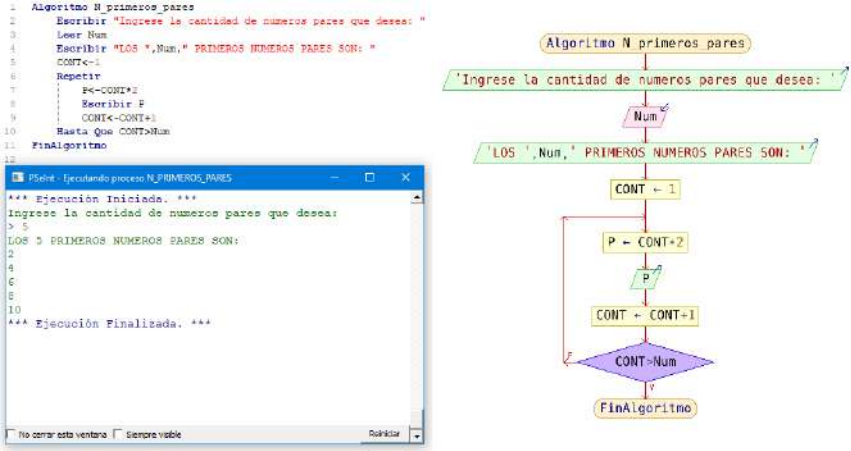


Figura 119. Algoritmo y diagrama de flujo para el ejemplo 3.

Ejemplo 4:

Realizar un algoritmo y diagrama de flujo que permita OBTENER los N primeros números pares y N primeros números impares, en columnas.

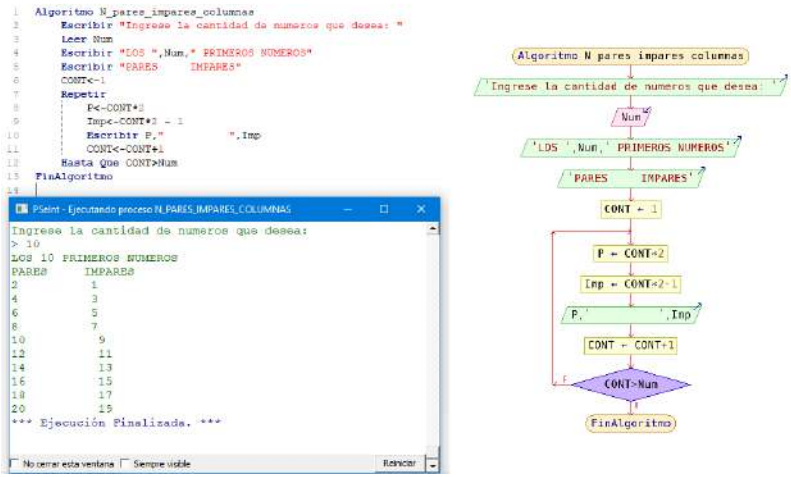


Figura 120. Algoritmo y DF para el ejemplo 4 usando estructura repetir hasta que.

Como se observa en la figura 119 y 120 tenemos el uso de la estructura repetitiva "repetir hasta que" para resolver el ejemplo 3 y el ejemplo 4 respectivamente.

Ejercicios propuestos

Para los siguientes ejercicios realizar el algoritmo y diagrama de flujo con estructura repetitiva "repetir hasta que":

1. Obtener el factorial de un número
2. Obtener los N primeros múltiplos de 5
3. Obtener los N primeros múltiplos de 7
4. Obtener los N primeros múltiplos de 9
5. Para saber si un número es primo o no
6. Para saber si un número es compuesto o no
7. Obtener los N primeros números primos
8. Obtener los N primeros números compuestos

AREGLOS O ARRAYS

Capítulo 4

ARREGLOS O ARRAYS

Un array (o arreglo) es una estructura de datos con elementos homogéneos, del mismo tipo, numérico o alfanumérico, reconocidos por un nombre en común. Para referirnos a cada elemento del array usaremos un índice (empezamos a contar por 0 o también por 1).

Dentro de los arreglos o arrays se tienen diferentes tipos, como:

- Arreglos unidimensionales o vectores
- Arreglos bidimensionales o matrices
- Arreglos de N dimensiones

Arreglos unidimensionales o vectores

Un arreglo es una secuencia de datos del mismo tipo que ocupan un lugar contiguo en memoria. Las posiciones consecutivas que ocupa el arreglo se denominan elementos del arreglo y se numeran sucesivamente 0, 1, 2, 3, etc.

El tipo de información que se almacena en un arreglo puede ser cualquiera de los tipos de dato básicos de PSEINT, es decir, CHAR, ENTERO, LOGICO, REAL, STRING.

Así por ejemplo, un arreglo puede contener, la edad de los alumnos de una clase, las temperaturas de cada día del mes en una ciudad determinada, o el número de personas que residen en cada una de las PROVINCIAS DEL ECUADOR. En la figura 4.1. se muestra los componentes de un vector:

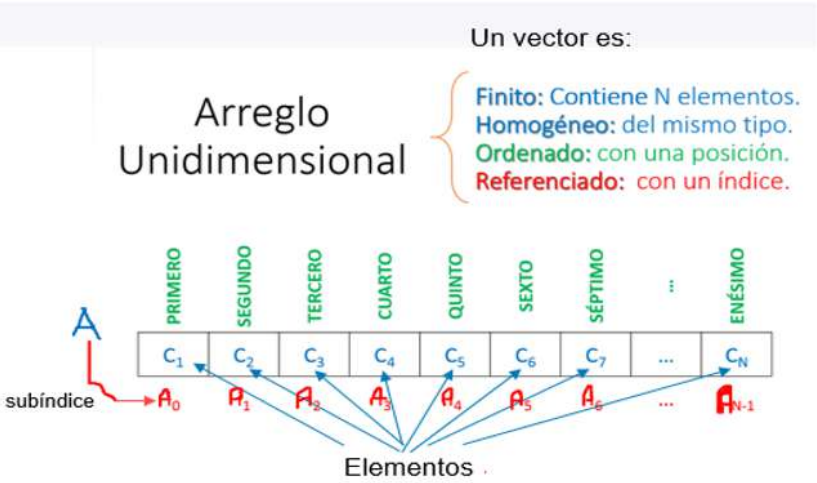


Figura 121. Componentes de un arreglo unidimensional o vector

Sintaxis para declarar un vector

Para declarar un vector tenemos que ejecutar dos instrucciones:

En primer lugar debemos declara el tipo de datos de la variable, con la palabra clave Definir.

Definir <nombre_vector> como <tipo_datos>

En nombre del vector se escribe una variable con el nombre que deseamos para nuestro vector y en el tipo de datos definimos que tipo va hacer: Entero, Real, Carácter o Lógico.

En segundo lugar, debemos indicar el número de elementos que va a tener el vector, para ello utilizamos la palabra clave Dimension:

Dimension <identificador> (<tamaño>);

Esta instrucción define un vector con el nombre indicado en <identificador>. El <tamaño> indica la dimensión del vector, este valor debe ser entero y debe ir entre paréntesis.

Nota.- Si está activada la opción controlar el uso de punto y coma en el menú Configurar siempre ubique un punto y coma al final del arreglo.

Ejemplo:

Por ejemplo definimos un array o vector llamado A de 10 elementos.

Definir A como Entero;
Dimension A(10);

Para asignar el valor de 10 a la posición 2 del vector A:
A(2)<-10;

Para mostrar el primer elemento del vector A:
Escribir A(1);

Para asignar a una variable R la posición 5 del vector A:
R<- A(5)

Almacenamiento de datos en arreglos

El almacenamiento de datos dentro de un arreglo consiste en permitir tal cual como se observa en la figura 122.

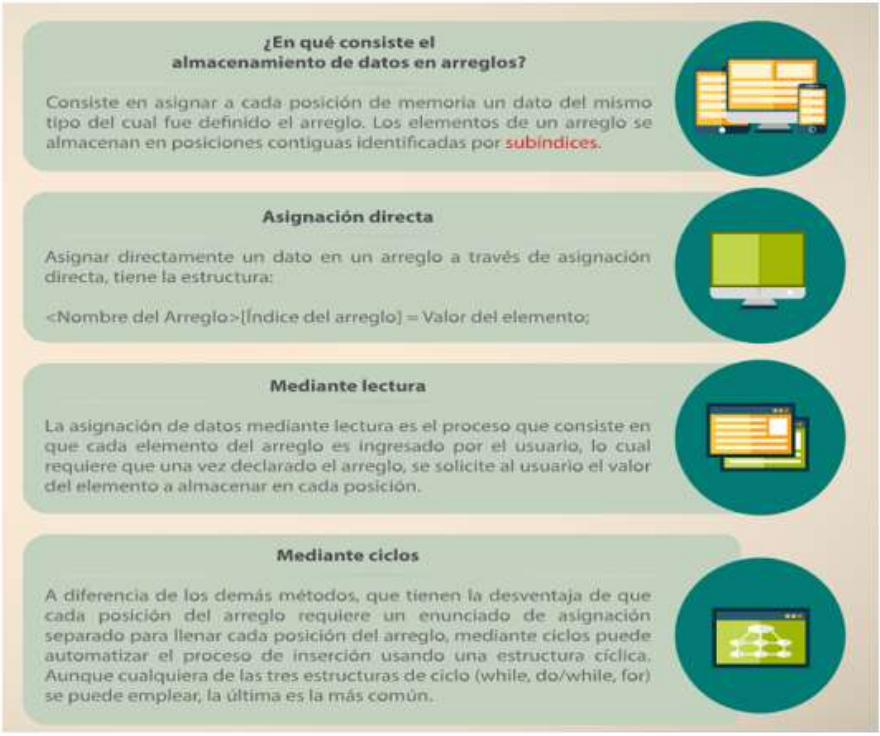


Figura 122. Almacenamiento de datos en arreglos.

En la figura 123 se muestra un EJEMPLO para almacenamiento de datos:

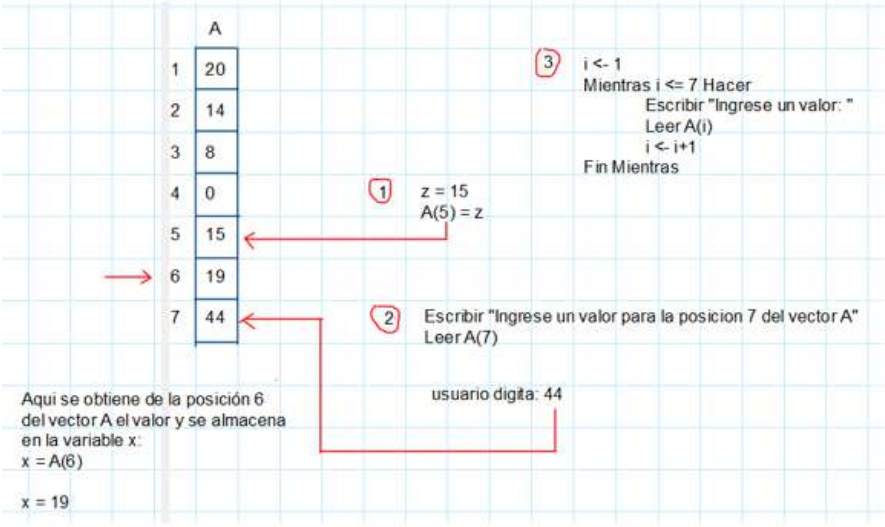


Figura 123. Ejemplo de almacenamiento de datos en arreglos unidimensionales

Ejemplo 1:

Realizar un algoritmo que permita ingresar y visualizar en un vector los números 20, 14, 8, 0, 5, 19 y 4.

```
1  Algoritmo ingreso_vector
2      Definir numeros como Entero
3      Dimension numeros(7)
4      numeros(1)<-20;
5      numeros(2)<-14;
6      numeros(3)<-8;
7      numeros(4)<-0;
8      numeros(5)<-5;
9      numeros(6)<-19;
10     numeros(7)<-4;
11     Para i<-1 Hasta 7 Con Paso 1 Hacer
12         Escribir "El elemento en la posición ",i," es: ",numeros(i)
13     Fin Para
14 FinAlgoritmo
15
```

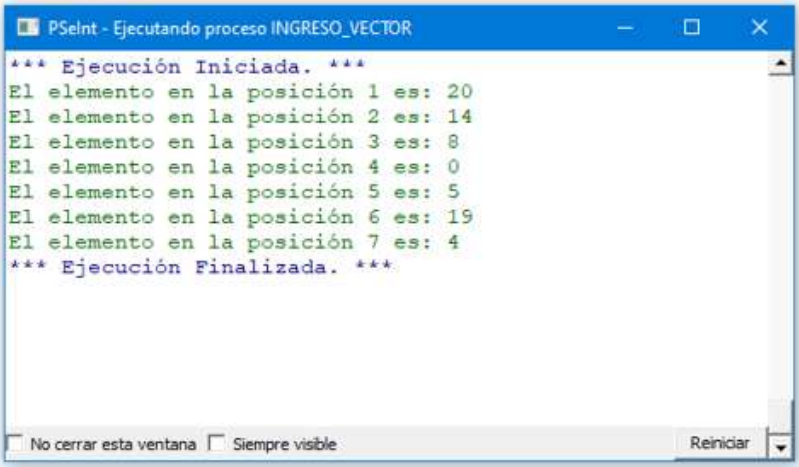


Figura 124. Algoritmo para ingresar valores a un vector y visualizar los mismos.

Usando la estructura para o for.

En la figura 124 se puede observar el algoritmo para asignar valores a un vector y su correspondiente estructura para o for para visualizar los datos del mismo vector.

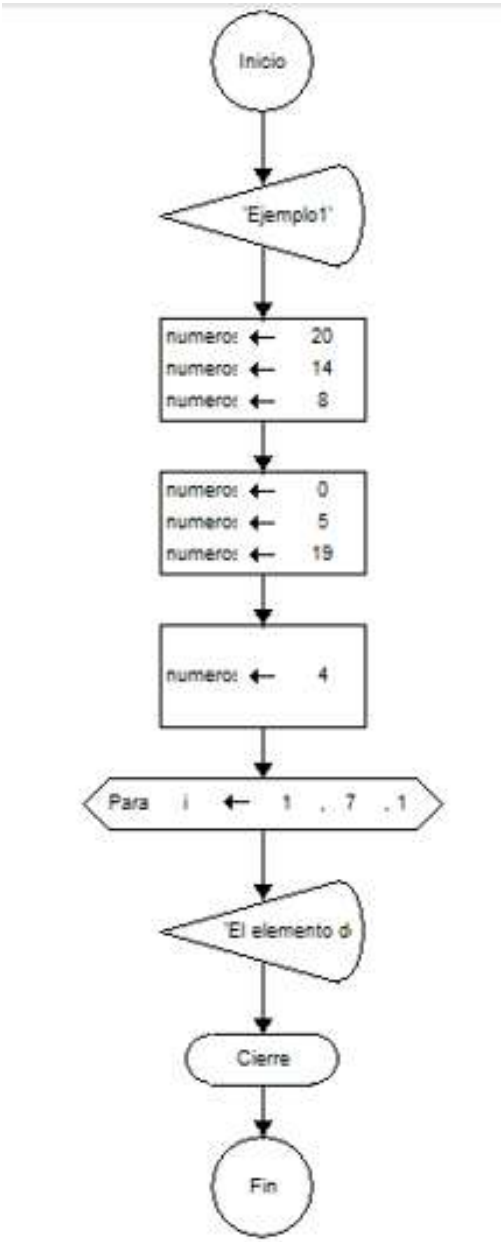


Figura 125. Diagrama de flujo para asignar valores a un vector.

Usando la estructura para o for.

En la figura 125 se muestra cómo se asignan valores a un vector usando el objeto rectángulo de asignación de datos, el objeto cono de visualización de datos y el objeto de la estructura para o for para seguir mostrando cada uno de los elementos del vector.

Ejemplo 2:

Realizar un algoritmo que permita ingresar 7 valores desde el teclado al vector.

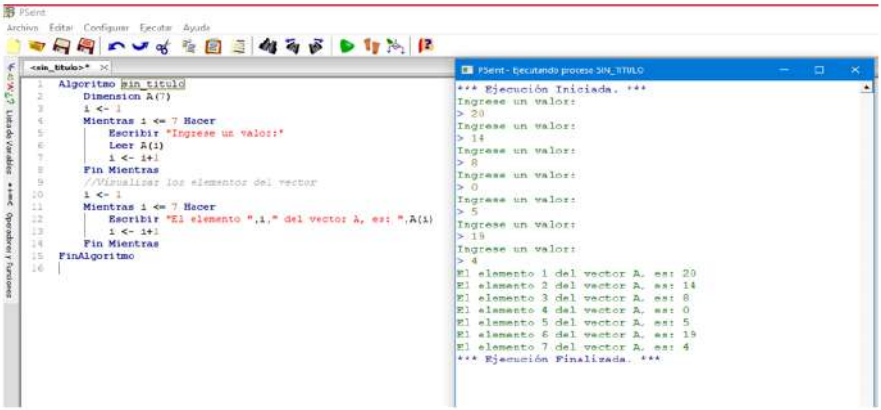


Figura 126. Algoritmo para el ingreso de 7 valores desde teclado al vector.

En la figura 126 se observa el algoritmo usando la estructura mientras para ingresar 7 valores desde el teclado al vector A y también para visualizar los mismos elementos del vector.

En la parte derecha de la figura 126 se observa la ejecución del algoritmo en PSeInt.

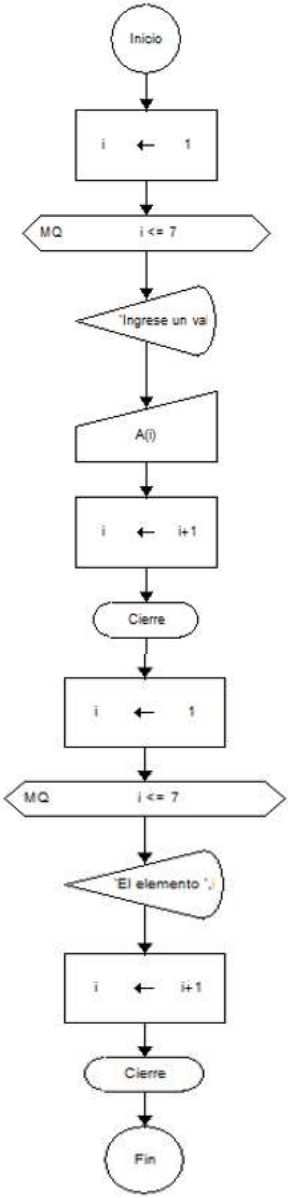


Figura 127. Diagrama de flujo para el ejemplo 2 de vectores.

En la figura 127 se observa cómo se usa el objeto de la estructura mientras para obtener los datos e ingresar al vector y de la misma manera para visualizar los mismos.

Ejemplo 3:

Realizar un algoritmo para ingresar 7 valores desde el teclado a un vector y visualizarlos de manera vertical y horizontal.

```
1   Algoritmo sin_titulo
2       Dimension B(7)
3       i = 1
4       //Ingreso de valores al vector B
5       //a traves del teclado
6       Mientras i <= 7 Hacer
7           Escribir "Ingrese el elemento [",i,"]:"
8           Leer B(i)
9           i = i+1
10      FinMientras
11      //Visualizar los elementos del vector
12      i=1
13      Escribir "Vector B"
14      Mientras i <= 7 Hacer
15          Escribir B(i)
16          i = i+1
17      FinMientras
18      //Visualizar de manera horizontal el vector
19      Escribir "Vector B"
20      i = 1
21      Mientras i <= 7 Hacer
22          Escribir B(i), "   " Sin Saltar
23          i = i+1
24      FinMientras
25      Escribir ""
26  FinAlgoritmo
27
```

Figura 128. Algoritmo para visualizar de manera vertical y horizontal 7 valores ingresados por teclado en un vector B.

En la figura 128 se observa el algoritmo que permite ingresar 7 valores desde el teclado y almacenarlos en un vector B, además se visualiza de manera vertical el vector y finalmente se visualizar de manera horizontal usando la instrucción “Sin Saltar”.

Ejercicio 1:

Realizar un algoritmo y diagrama de flujo que permita ingresar N números en un arreglo.

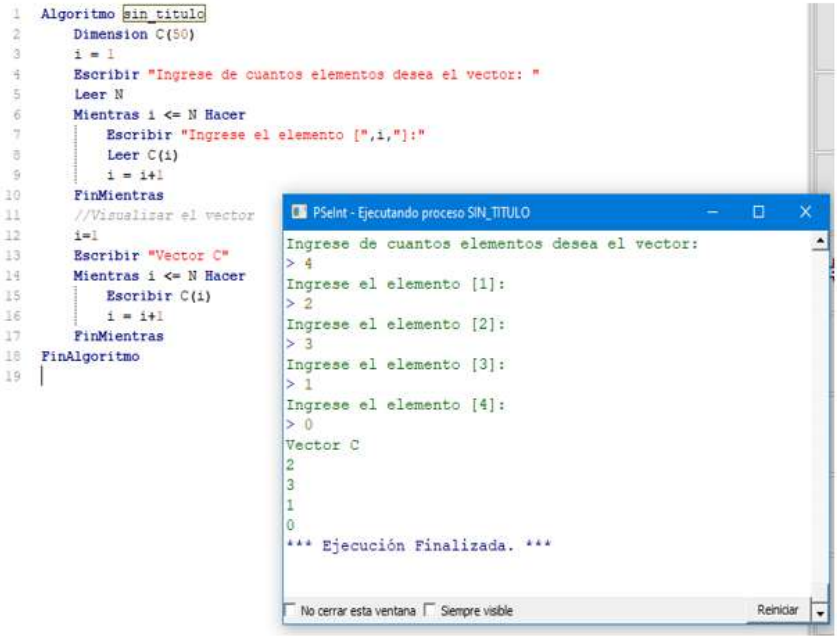


Figura 129. Algoritmo para el ejercicio 1 de vectores

En la figura 129 se observa el algoritmo que permite ingresar N valores a un vector C haciendo uso de la estructura repetitiva mientras.

Ejercicio 2:

Realizar un algoritmo y diagrama de flujo que permita ingresar los N primeros números pares en un arreglo.

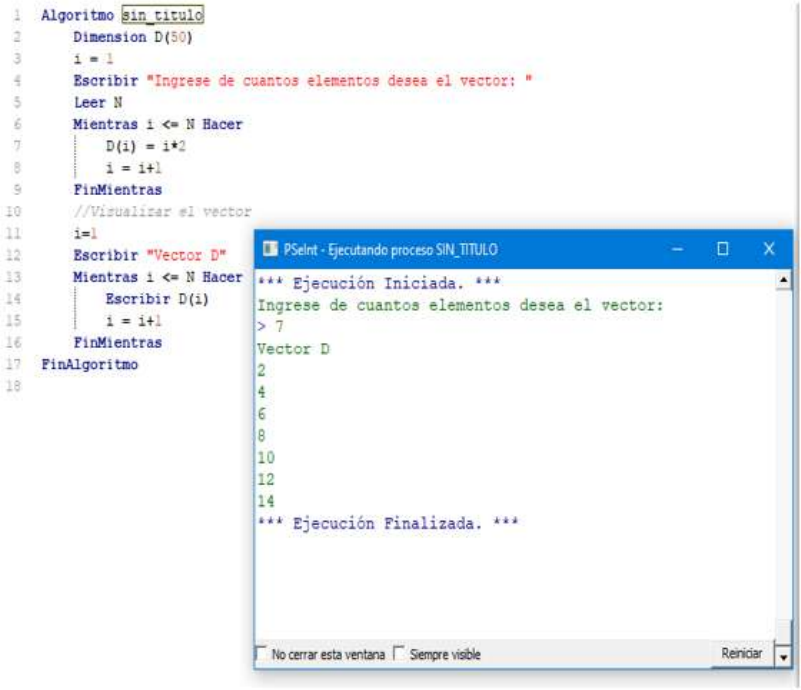


Figura 130. Algoritmo para el ejercicio 2 de vectores.

En la figura 130 se observa el ingreso de N números pares al vector D usando la estructura mientras o while.

Ejercicios propuestos de vectores

Para los siguientes ejercicios realizar un algoritmo y diagrama de flujo que permita:

1. Ingresar en un vector los N primeros números impares.

2. Ingresar en un vector los N primeros números múltiplos de 5.
3. Ingresar en un vector los N primeros números múltiplos de 7.
4. Ingresar en un vector los N primeros números primos.
5. Ingresar en un vector los N primeros números compuestos.

Operaciones con vectores

Para poder realizar las operaciones básicas como son suma, resta, multiplicación, división y otras como el residuo o módulo, potencia, etc., se debe tener vectores de la misma dimensión.

Suma de vectores

Ejemplo 1:

Realizar un algoritmo que permita sumar 2 vectores.

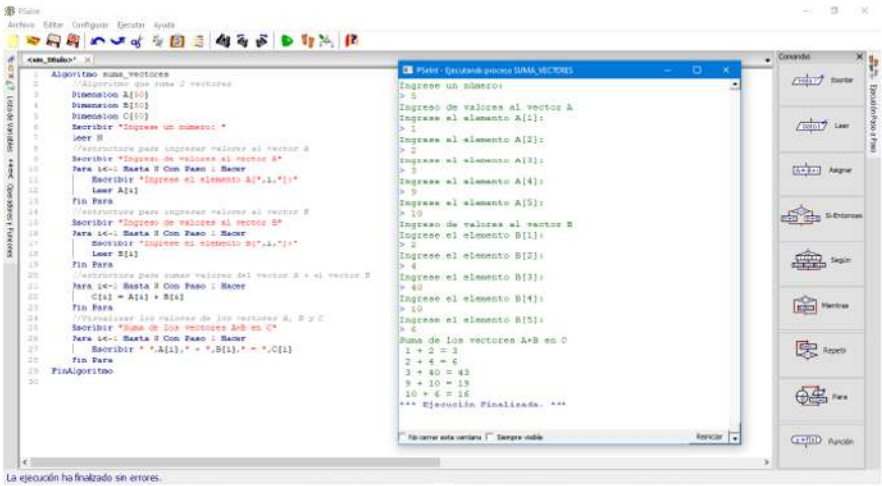


Figura .131. Algoritmo para sumar 2 vectores

Resta de vectores

Ejemplo 1:

Realizar un algoritmo que permita restar 2 vectores.

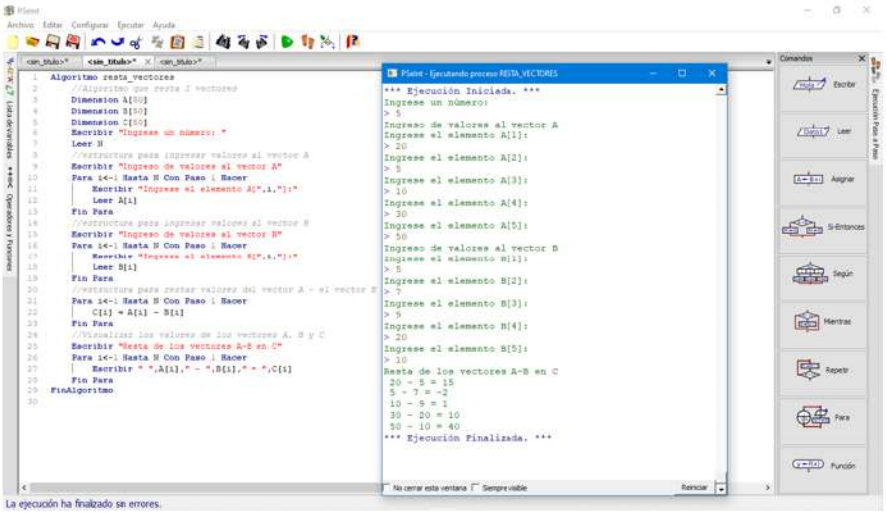


Figura 132. Algoritmo para restar 2 vectores

Multiplicación de vectores

Ejemplo 1:

Realizar un algoritmo que permita multiplicar 2 vectores.

División de vectores

Ejemplo 1:

Realizar un algoritmo que permita dividir 2 vectores.

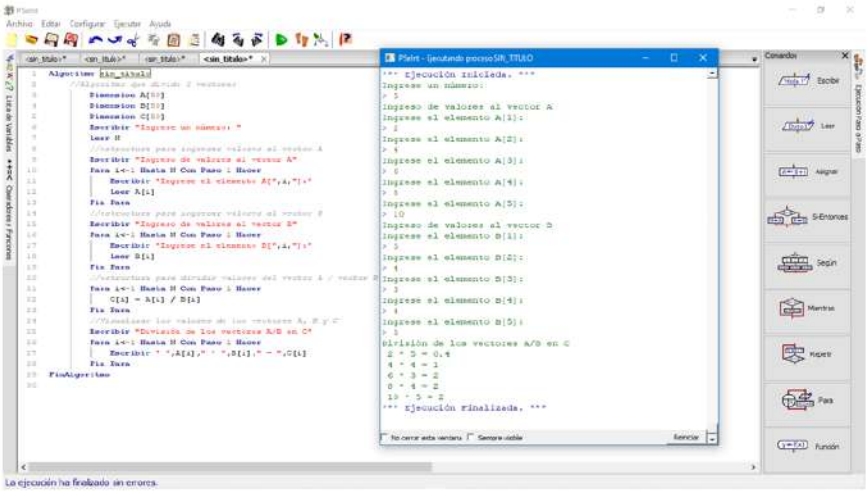


Figura 133. Algoritmo para dividir 2 vectores

Residuo de vectores

Ejemplo 1:

Realizar un algoritmo que permita obtener el MOD entre 2 vectores.

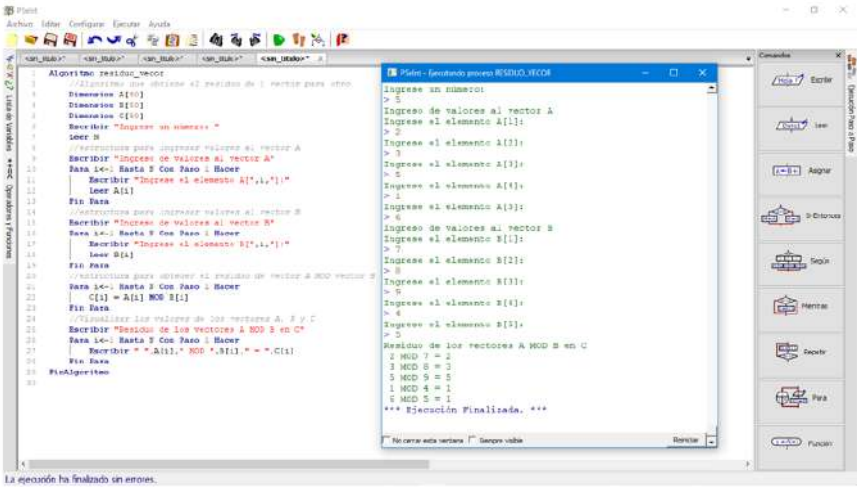


Figura 134. Algoritmo para obtener el residuo entre 1 vector con otro.

Potencia de vectores

Ejemplo 1:

Realizar un algoritmo que permita elevar un vector a la potencia del segundo vector.

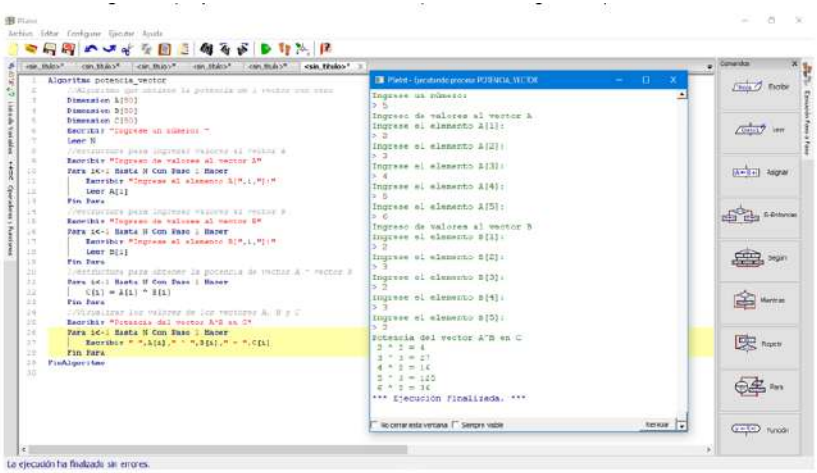


Figura 135. Algoritmo para obtener la potencia de 1 vector elevado al otro.

Búsqueda de elementos en un vector

Existen algunos métodos de búsqueda de los cuáles vamos a realizar el lineal y el binario.

Ejemplo 1:

Se quiere buscar un estudiante junto con sus notas a través de su código.

Para resolver este ejemplo se tienen que ingresar en un vector A los códigos, en el B los nombres, en el C la nota promedio de los parciales y en el D la nota del examen principal.

```
Algoritmo buscar_elemento
    //Algoritmo que obtiene la potencia de 1 vector con otro
    Dimension A[50]
    Dimension B[50]
    Dimension C[50]
    Dimension D[50]
    Escribir "Ingrese el número de estudiantes: "
    Leer N
    Escribir "Ingreso de información a los vectores"
    Para i<-1 Hasta N Con Paso 1 Hacer
        Escribir "Información Estudiante ",i,":"
        Escribir "Ingrese el código:"
        Leer A[i]
        Escribir "Ingrese el nombre: "
        Leer B[i]
        Escribir "Ingrese nota parciales: "
        Leer C[i]
        Escribir "Ingrese nota examen principal: "
        Leer D[i]
    Fin Para
    Escribir "Ingrese el código a buscar: "
    Leer dato
    ban=0
    Para i<-1 Hasta N Con Paso 1 Hacer
        Si dato = A[i] Entonces
            Escribir "El estudiante ",i," tiene la siguiente información:"
            Escribir "Codigo: ",A[i]
            Escribir "Nombres: ", B[i]
            Escribir "Nota Parciales: ", C[i]
            Escribir "Nota Examen Principal: ",D[i]
            ban=1
        Fin Si
    Fin Para
    Si ban=0 Entonces
       Escribir "Información no encontrada, el estudiante no se encuentra en la BD"
    SiNo
       Escribir "La información fue encontrada con éxito!"
    Fin Si
FinAlgoritmo
```

Figura 136. Algoritmo usando búsqueda lineal para buscar un dato dentro del vector.

Ordenar un vector

Para ordenar un vector existen varios métodos de ordenamiento, de los cuáles vamos hacer uso del método de ordenamiento por burbuja.

Método de ordenamiento por burbuja

El ordenamiento burbuja hace múltiples pasadas a lo largo de una lista. Compara los ítems adyacentes e intercambia los que no están en orden. Cada pasada a lo largo de la lista ubica el siguiente valor más grande en su lugar apropiado. En esencia, cada ítem "burbujea" hasta el lugar al que pertenece.

La figura 137 muestra la primera pasada de un ordenamiento burbuja. Los ítems sombreados se comparan para ver si no están en orden. Si hay n ítems en la lista, entonces hay n−1n−1 parejas de ítems que deben compararse en la primera pasada. Es importante tener en cuenta que, una vez que el valor más grande de la lista es parte de una pareja, este avanzará continuamente hasta que la pasada se complete.

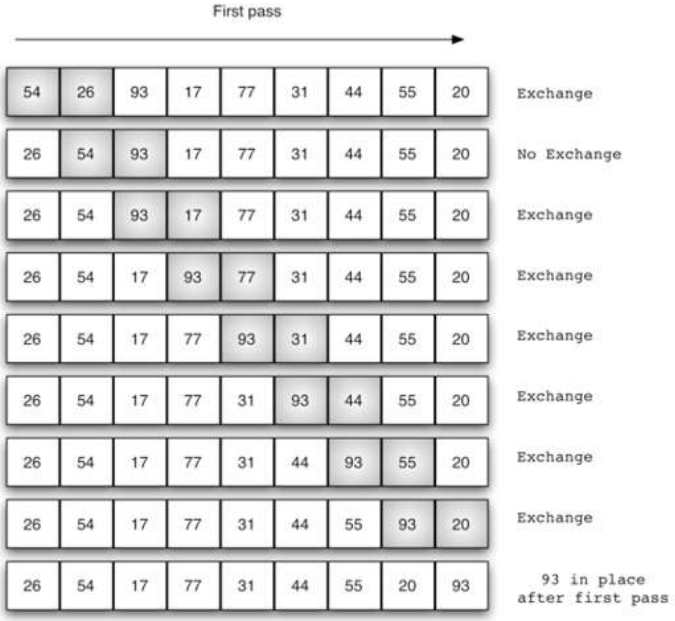


Figura 137. Primera pasada de un ordenamiento por burbuja

Ordenar de manera ascendente

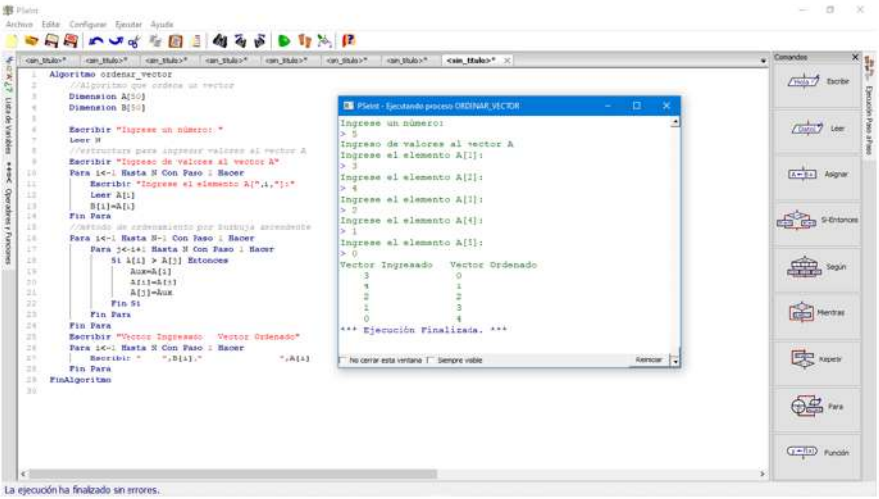


Figura 138. Vector ordenado ascendentemente usando el método de burbuja

Ordenar de manera descendente

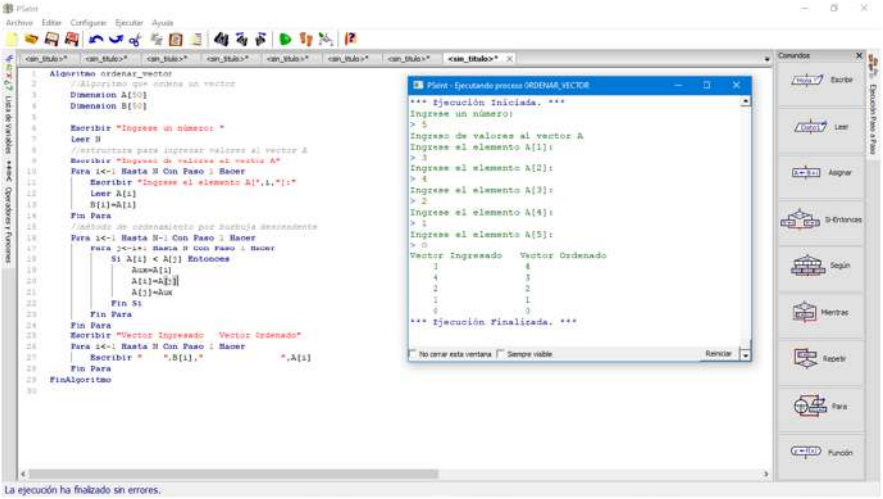


Figura 139. Vector ordenado descendentemente usando el método de burbuja

PROGRAMACIÓN EN R CON RSTUDIO

Capítulo 5

PROGRAMACIÓN EN R CON RSTUDIO

Lenguaje de programación

Es un lenguaje formal que mediante una serie de instrucciones le permite a un programador escribir un conjunto de órdenes acciones consecutivas.

Diferentes tipos de lenguajes de programación

Tabla 12. Diferentes lenguajes de programación

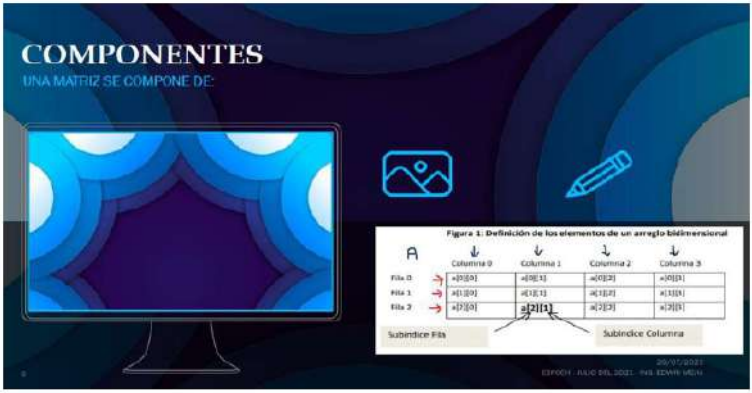
Lenguajes de programacion	**Observaciones**
C++	C++ es un lenguaje de programación multiparadigma puesto que permite programar de manera imperativa orientada a objetos o genérica.
R y studio	Rstudio es un entono de desarrollo integrado para el lenguaje de programación r dedicado a la computación estadística y gráficos. Incluye una consuela editor de sintaxis que apoya la ejecución de código así como herramientas para el trazado la depuración y la gestión del espacio de trabajo.
Visual studio code	Visual studio code es un editor de código fuente desarrollado por microsoft para windows linux y macos. Permite trabajar con diversos lenguajes de programación admite gestionar tus propios atajos de teclado y refactorizar el código.

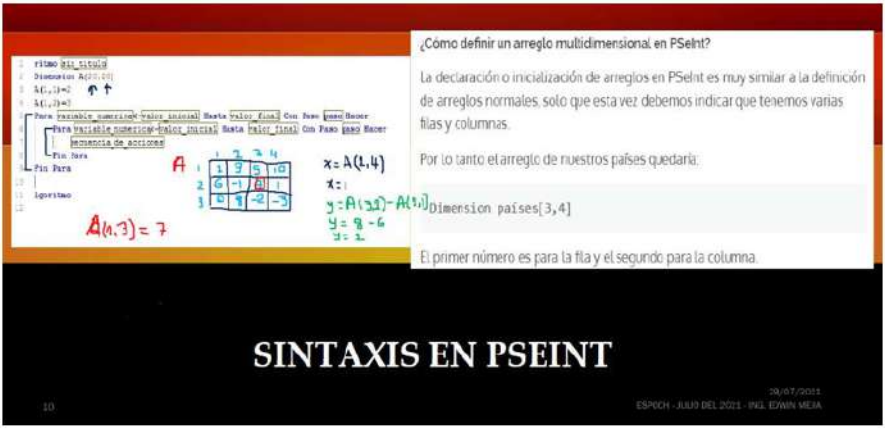
Java	Java es un lenguaje de programación y una plataforma informática comercial por primera vez en 1995 por sun microsystems.Java es rápido seguro y fiable desde portátiles hasta centros de datos. Desde consolas para juegos hasta super computadoras desde teléfonos móviles hasta la web. Java está en todas las partes

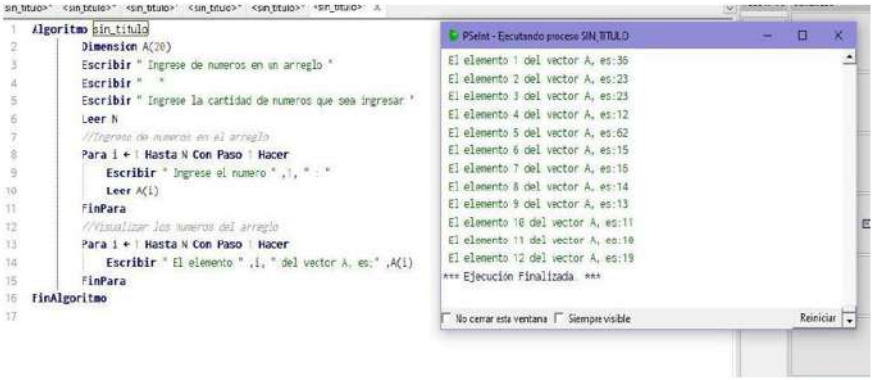
Arreglos bidimensionales

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.




¿Cómo definir un arreglo multidimensional en PSeInt?
La declaración o inicialización de arreglos en PSeInt es muy similar a la definición de arreglos normales, solo que esta vez debemos indicar que tenemos varias filas y columnas.
Por lo tanto el arreglo de nuestros países quedaría:
Dimension países[3,4]
El primer número es para la fila y el segundo para la columna.
SINTAXIS EN PSEINT
10
ESPOCH - JULIO DEL 2021 - ING. EDWIN MEJIA


Algoritmo sin_titulo
Dimension A(20)
Escribir " Ingrese de numeros en un arreglo "
Escribir " "
Escribir " Ingrese la cantidad de numeros que sea ingresar "
Leer N
//Ingreso de numeros en el arreglo
Para i ← 1 Hasta N Con Paso 1 Hacer
Escribir " Ingrese el numero " ,i, " : "
Leer A(i)
FinPara
//Visualizar los numeros del arreglo
Para i ← 1 Hasta N Con Paso 1 Hacer
Escribir " El elemento " ,i, " del vector A, es:" ,A(i)
FinPara
FinAlgoritmo
PSeInt - Ejecutando proceso SIN_TITULO
El elemento 1 del vector A, es:36
El elemento 2 del vector A, es:23
El elemento 3 del vector A, es:23
El elemento 4 del vector A, es:12
El elemento 5 del vector A, es:62
El elemento 6 del vector A, es:15
El elemento 7 del vector A, es:15
El elemento 8 del vector A, es:14
El elemento 9 del vector A, es:13
El elemento 10 del vector A, es:11
El elemento 11 del vector A, es:10
El elemento 12 del vector A, es:19
*** Ejecución Finalizada. ***
No cerrar esta ventana
Siempre visible
Reiniciar

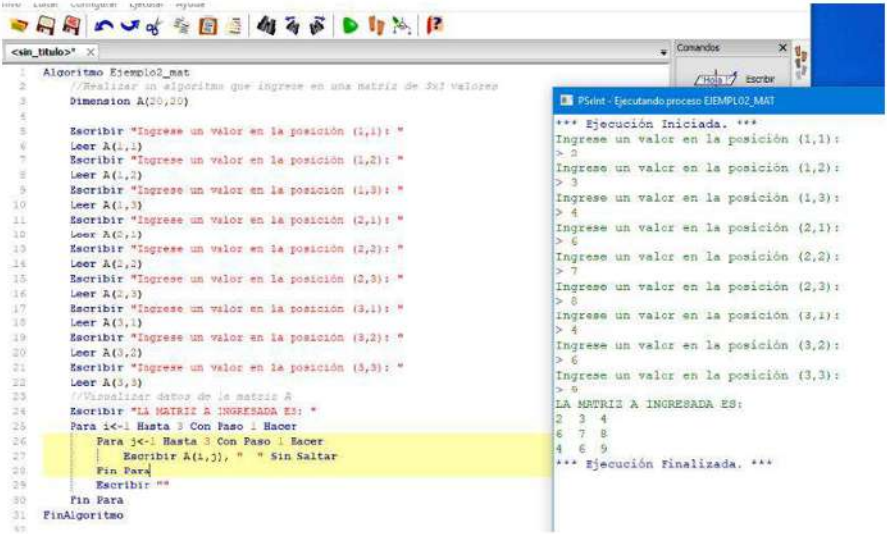
Asignar valores a las posiciones del arreglo multidimensional

Recordemos que debemos indicar en que fila y columna ubicaremos el dato en cuestión, así:

```
Algoritmo Arreglos_Bi
    Dimension paises[3,4]
    paises[1,1] <- "Mexico"
    paises[2,1] <- "Argentina"
    paises[3,1] <- "Colombia"

    paises[1,2] <- "Ciudad Juarés"
    paises[2,2] <- "Buenos Aires"
    paises[3,2] <- "Bogotá"

    paises[1,3] <- "Monterrey"
    paises[2,3] <- "Córdoba"
    paises[3,3] <- "Cali"

    paises[1,4] <- "Guadalajara"
    paises[2,4] <- "La plata"
    paises[3,4] <- "Barranquilla"

FinAlgoritmo
```

Mexico	Ciudad Juarez	Monterrey	Guadalajara
Argentina	Buenos Aires	Córdoba	La Plata
Colombia	Bogotá	Cali	Barranquilla

```
Para j <- 1 Hasta 4 Con paso 1 Hacer
    Escribir países(1,j)
Fin Para
```



Algoritmo matriz3x3

//Realizar un Algoritmo que permita ingresar en un matriz de 3x3 valores

Dimension A(3,3)

Escribir "INGRESE EN UNA MATRIZ DE 3X3 VALORES"
Para i<- 1 Hasta 3 Hacer
Para j <- 1 hasta 3 hacer

Escribir "Ingrese un valor" Leer A(i,j)
FinPara

Fin Para

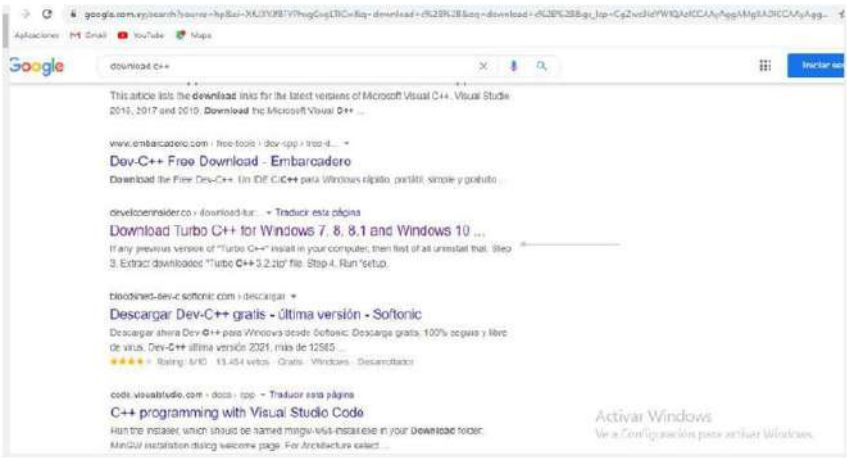
//Visulizar la matriz// Para i <- 1 Hasta 3
para j <- 1 hasta 3 hacer Escribir A(i,j) Sin Saltar Escribir " " Sin Saltar FinPara
Escribir "" Fin Para FinAlgoritmo

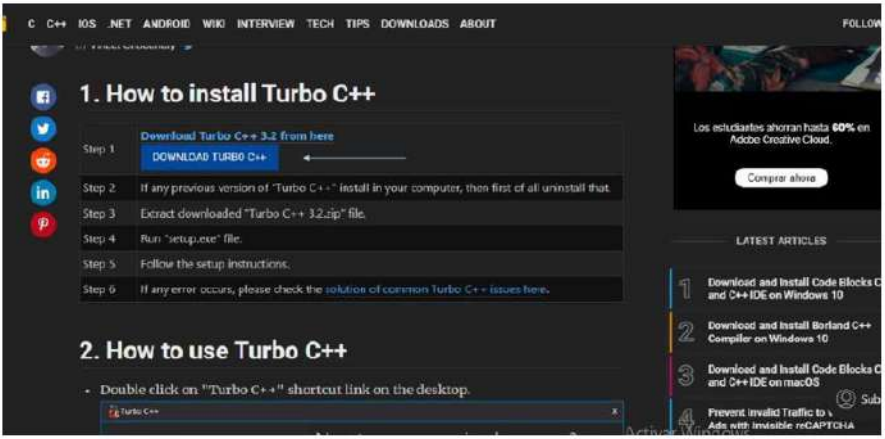
```
Ingrese una matriz de 3x3 valores.
Ingrese el valor
9
6
3
8
5
2
7
4
1
```

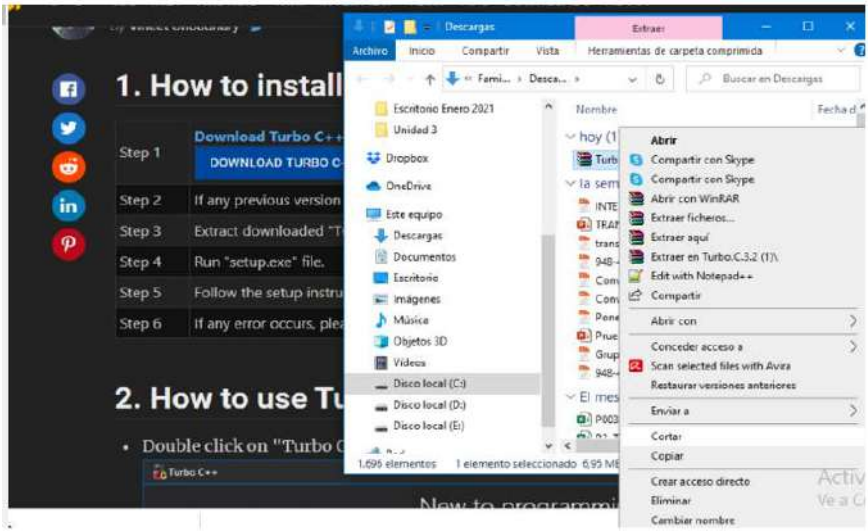
9	6	3
8	5	2
7	4	1

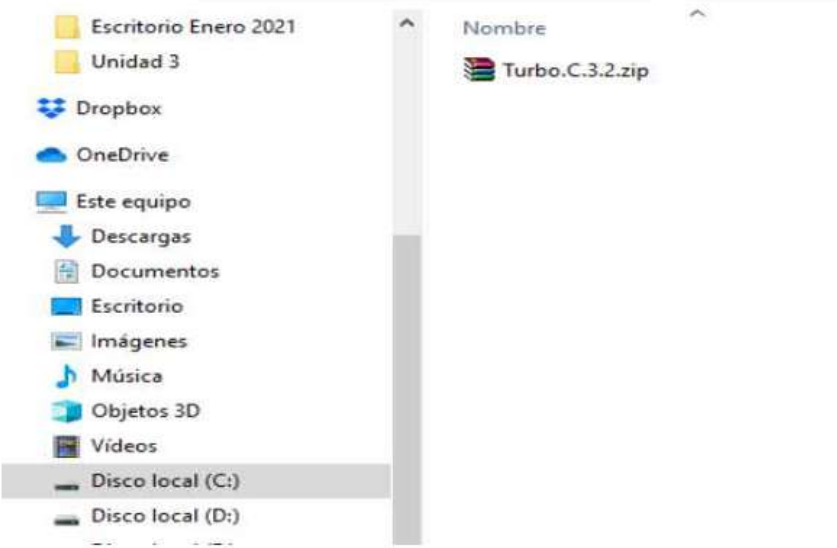
Instalación de C++

Este lenguaje de programación es uno de los principales lenguajes que se debería aprender dentro de cualquier carrera que tenga como base la programación.

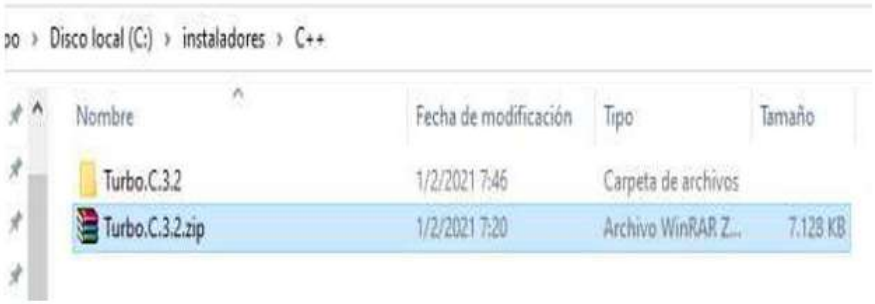


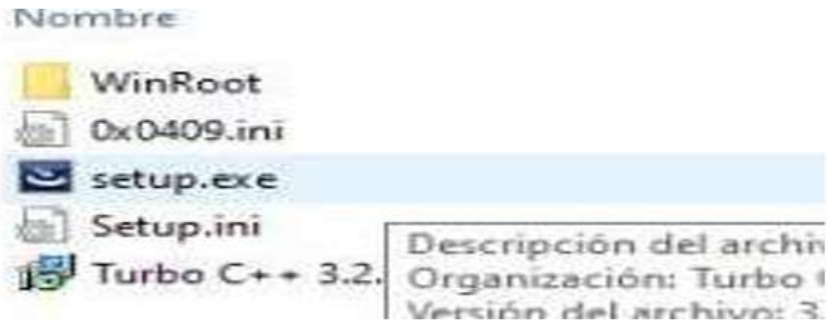

C C++ IOS .NET ANDROID WIKI INTERVIEW TECH TIPS DOWNLOADS ABOUT
FOLLOW
1. How to install Turbo C++
Step 1
Download Turbo C++ 3.2 from here
DOWNLOAD TURBO C++
Step 2
If any previous version of "Turbo C++" install in your computer, then first of all uninstall that.
Step 3
Extract downloaded "Turbo C++ 3.2.zip" file.
Step 4
Run "setup.exe" file.
Step 5
Follow the setup instructions.
Step 6
If any error occurs, please check the solution of common Turbo C++ issues here.
2. How to use Turbo C++
Double click on "Turbo C++" shortcut link on the desktop.
Turbo C++
Los estudiantes ahorran hasta 60% en Adobe Creative Cloud.
Comprar ahora
LATEST ARTICLES
1 Download and Install Code Blocks C and C++ IDE on Windows 10
2 Download and Install Borland C++ Compiler on Windows 10
3 Download and Install Code Blocks C and C++ IDE on macOS
4 Prevent Invalid Traffic to Ads with Invisible reCAPTCHA

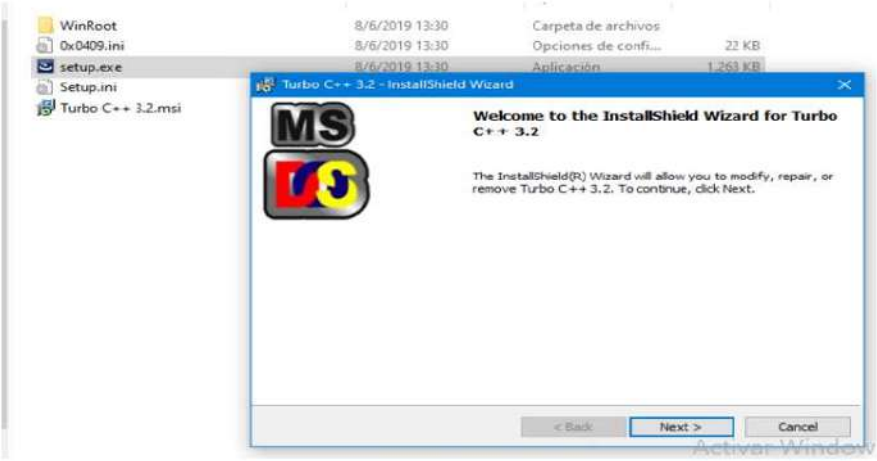

1. How to install
Step 1
Download Turbo C++
DOWNLOAD TURBO C
Step 2
If any previous version
Step 3
Extract downloaded "T
Step 4
Run "setup.exe" file.
Step 5
Follow the setup instru
Step 6
If any error occurs, ple
2. How to use Tu
Double click on "Turbo C
Turbo C++
Descargas
Extraer
Archivo
Inicio
Compartir
Vista
Herramientas de carpeta comprimida
Buscar en Descargas
Escritorio Enero 2021
Unidad 3
Dropbox
OneDrive
Este equipo
Descargas
Documentos
Escritorio
Imágenes
Música
Objetos 3D
Vídeos
Disco local (C:)
Disco local (D:)
Disco local (E:)
Nombre
Abrir
Compartir con Skype
Compartir con Skype
Abrir con WinRAR
Extraer ficheros...
Extraer aquí
Extraer en Turbo.C.3.2 (1)\
Edit with Notepad++
Compartir
Abrir con
Conceder acceso a
Scan selected files with Avira
Restaurar versiones anteriores
Enviar a
Cortar
Copiar
Crear acceso directo
Eliminar
Cambiar nombre
1.695 elementos
1 elemento seleccionado 6,95 MB


Escritorio Enero 2021
Unidad 3
Dropbox
OneDrive
Este equipo
Descargas
Documentos
Escritorio
Imágenes
Música
Objetos 3D
Vídeos
Disco local (C:)
Disco local (D:)
Nombre
Turbo.C.3.2.zip


Vista
Herramientas de carpeta comprimida
Disco local (C:)
instaladores
C++
Nombre
Fecha de modificación
Abrir
Compartir con Skype
Compartir con Skype
Abrir con WinRAR
Extraer ficheros...
Extraer aquí
Extraer en Turbo.C.3.2\
Edit with Notepad++
Compartir


bo › Disco local (C:) › instaladores › C++
Nombre
Fecha de modificación
Tipo
Tamaño
Turbo.C.3.2
1/2/2021 7:46
Carpeta de archivos
Turbo.C.3.2.zip
1/2/2021 7:20
Archivo WinRAR Z...
7.128 KB

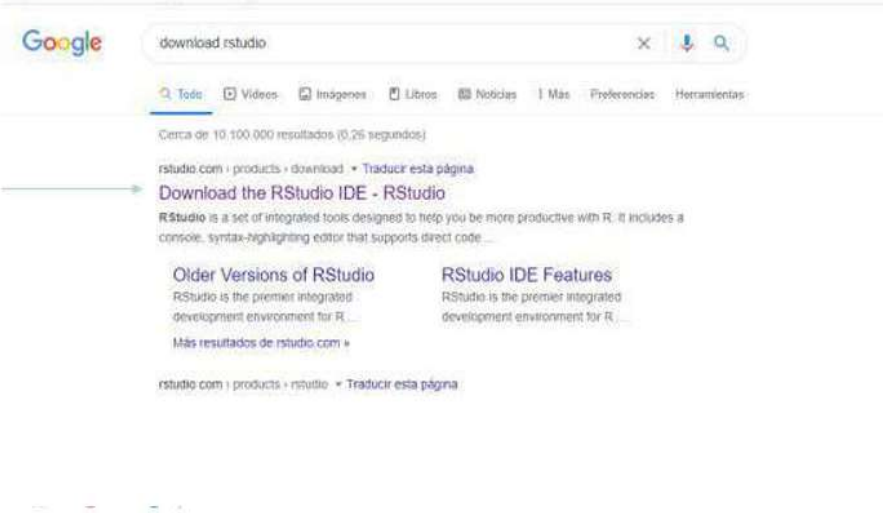

Nombre
WinRoot
0x0409.ini
setup.exe
Setup.ini
Turbo C++ 3.2.
Descripción del archi
Organización: Turbo
Versión del archivo: 3


WinRoot 8/6/2019 13:30 Carpeta de archivos
0x0409.ini 8/6/2019 13:30 Opciones de confi... 22 KB
setup.exe 8/6/2019 13:30 Aplicación 1.263 KB
Setup.ini
Turbo C++ 3.2.msi
Turbo C++ 3.2 - InstallShield Wizard
Welcome to the InstallShield Wizard for Turbo C++ 3.2
The InstallShield(R) Wizard will allow you to modify, repair, or remove Turbo C++ 3.2. To continue, click Next.
< Back
Next >
Cancel
Activar Windows

Instalación de R

Lenguaje de programación que está en auge hoy mismo diciembre del 2021 que estamos desarrollando estos diferentes algoritmos, diagramas de flujo y programas en R.

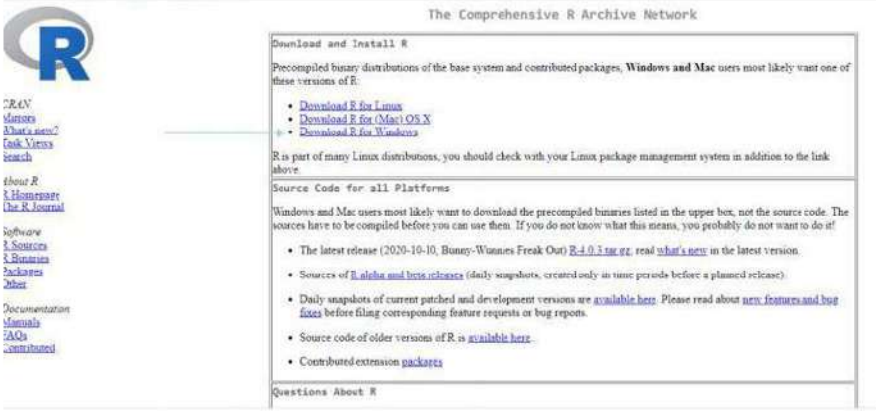
Instalando R



Todos los instaladores

Los usuarios de Linux pueden necesitar importar la clave pública de firma de código de RStudio antes de la instalación, según la política de seguridad del sistema operativo.

RStudio requiere un sistema operativo de 64 bits. Si está en un sistema de 32 bits, puede usar una versión anterior de RStudio.




Clic aqui
R-4.0.3 for Windows (32/64 bit)
Download R 4.0.3 for Windows (85 megabytes, 32/64 bit)
Installation and other instructions
New features in this version
If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.
Frequently asked questions
Does R run under my version of Windows?
How do I update packages in my previous version of R?
Should I run 32-bit or 64-bit R?
Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.
Other builds
Patches to this release are incorporated in the r-patched snapshot build.
A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
Previous releases
Note to webmasters: A stable link which will redirect to the current Windows binary release is
<CRAN MIRROR>/bin/windows/base/release.html
Last change: 2020-10-10

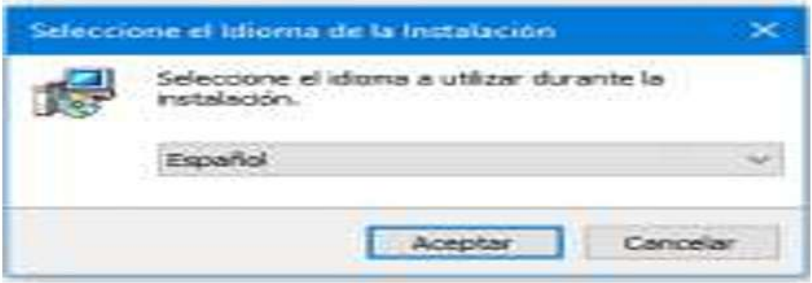

R-4.0.3 for Windows (32/64 bit)
Download R 4.0.3 for Windows (85 megabytes, 32/64 bit)
Installation and other instructions
New features in this version
If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.
Frequently asked questions
Does R run under my version of Windows?
How do I update packages in my previous version of R?
Should I run 32-bit or 64-bit R?
Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.
Other builds
Patches to this release are incorporated in the r-patched snapshot build.
A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
Previous releases
Note to webmasters: A stable link which will redirect to the current Windows binary release is
<CRAN MIRROR>/bin/windows/base/release.html
Last change: 2020-10-10
CRAN
Mirrors
What's new?
Task Views
Search
About R
R Homepage
The R Journal
Software
R Sources
R Binaries
Packages
Other
Documentation
Manuals
FAQs
Contributed
Esperamos que se descargue

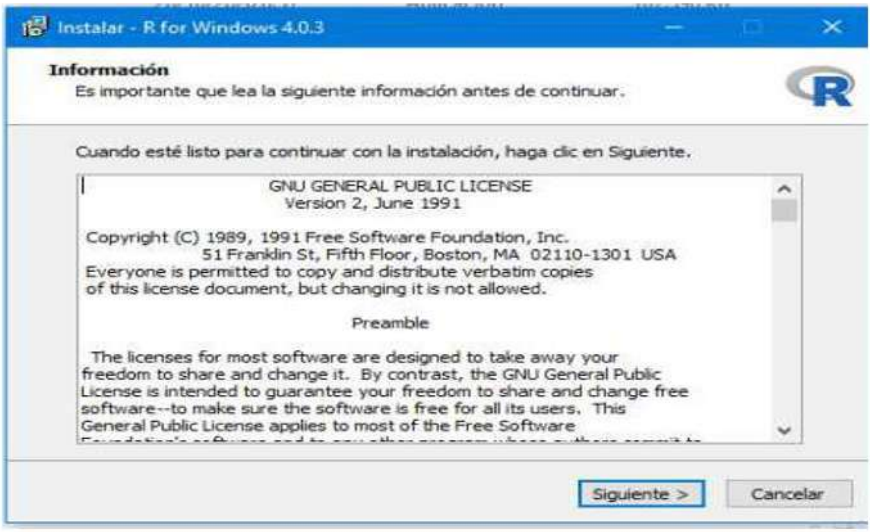
Instalación de R estudio

Todos los instaladores

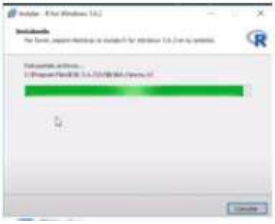
Los usuarios de Linux pueden necesitar importar la clave pública de firma de código de RStudio antes de la instalación, según la política de seguridad del sistema operativo.

RStudio requiere un sistema operativo de 64 bits. Si está en un sistema de 32 bits, puede usar una versión anterior de RStudio.



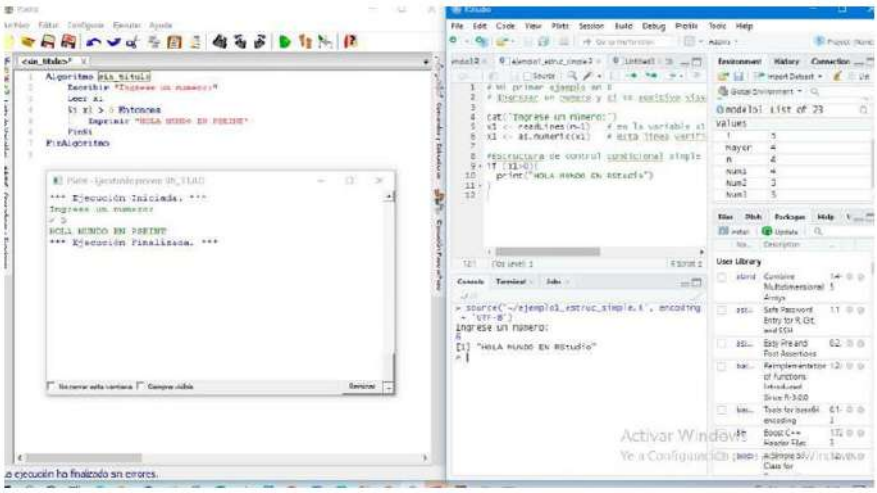

Instalar - R for Windows 4.0.3
Información
Es importante que lea la siguiente información antes de continuar.
Cuando esté listo para continuar con la instalación, haga clic en Siguiente.
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software
Siguiente >
Cancelar


Instalar - R for Windows 4.0.3
Seleccione la Carpeta de Destino
¿Dónde debe instalarse R for Windows 4.0.3?
El programa instalará R for Windows 4.0.3 en la siguiente carpeta.
Para continuar, haga clic en Siguiente. Si desea seleccionar una carpeta diferente,
haga clic en Examinar.
C:\Program Files\R\R-4.0.3
Examinar...
Se requieren al menos 2,5 MB de espacio libre en el disco.
< Atrás
Siguiente >
Cancelar



Ejemplo

Hacer un programa en psint y R Estudio que nos permita visualizar ‘Hola mundo en Psint’ y ‘Hola mundo en R Estudio’.



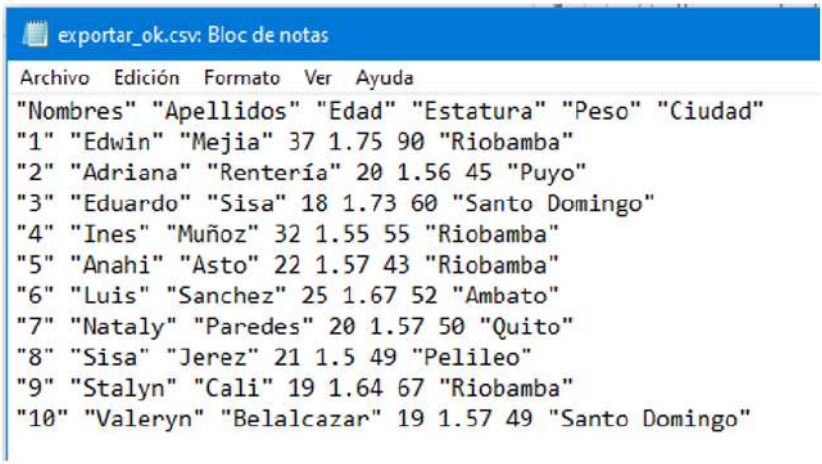
Importar, exportar y generar datos – vectores

Importar datos desde un archivo de Excel

- install.packages(“readxl”)
- library(readxl)
- file.choose()
- ej = read_excel(ruta1,sheet = ‘Hoja2’, range = ‘h7:k17’)

Importar datos desde un archivo CSV

- Csv viene de Coma Separated Values
- Es un archivo de texto
- Columnas separadas por comas, por puntos y coma o espacios.
- Una fila en cada línea



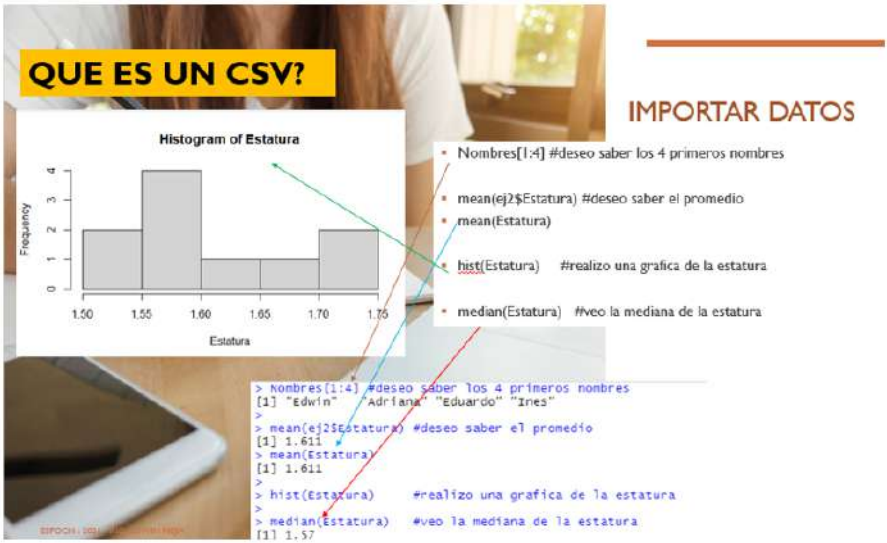
Que necesitamos para importar ficheros de csv a Rstudio.

- Instalar paquete readr, es un paquete muy sencillo.
- install.packages(“readr”)

- Cargar el paquete
- library(readr)
- Se necesita conocer la ruta del archivo csv
- file.choose()
- Copiar ruta de la consola al script
- ruta2 = “C:\\Users\\Familia\\Desktop\\clase8_r\\exportar_ok.csv“
- Utilizamos la función read.csv()
- ej1 = read.csv(ruta2)
- head(ej1)

- #cargar el archivo de csv a R
- #metodo1 - cargar csv con titulos
- ej1 = read.csv(ruta1)
- head(ej1)
- #Cuando el archivo csv tiene separador de espacios en blanco
- ej2 = read.csv(ruta1,sep = “”)
- head(ej2)


QUE ES UN CSV?
IMPORTAR DATOS
head(ej2) #retorna los registros de nuestro dataframe
attach(ej2) #con esto puedo acceder a las columnas de mi dataframe
ej2$Nombres #esto sin el uso de attach
Nombres #esto haciendo uso del attach
names(ej2) #para saber que columnas tengo en mi dataframe
> head(ej2) #retorna los registros de nuestro dataframe
Nombres Apellidos Edad Estatura Peso Ciudad
1 Edwin Nejia 37 1.75 90 Riobamba
2 Adriana Renteria 20 1.56 45 Puyo
3 Eduardo Sisa 18 1.73 60 Santo Domingo
4 Ines Nuñez 32 1.55 55 Riobamba
5 Anahi Asto 22 1.57 43 Riobamba
6 Luis Sanchez 25 1.67 52 Ambato
> ej2$Nombres #esto sin el uso de attach
[1] "Edwin" "Adriana" "Eduardo" "Ines" "Anahi" "Luis" "Nataly"
[8] "Sisa" "Stalyn" "Valeryn"
> Nombres #esto haciendo uso del attach
[1] "Edwin" "Adriana" "Eduardo" "Ines" "Anahi" "Luis" "Nataly"
[8] "Sisa" "Stalyn" "Valeryn"
> names(ej2)
[1] "Nombres" "Apellidos" "Edad" "Estatura" "Peso" "Ciudad"


QUE ES UN CSV?
IMPORTAR DATOS
Histogram of Estatura
Frequency
Estatura
1.50
1.55
1.60
1.65
1.70
1.75
Nombres[1:4] #deseo saber los 4 primeros nombres
mean(ej2$Estatura) #deseo saber el promedio
mean(Estatura)
hist(Estatura) #realizo una grafica de la estatura
median(Estatura) #veo la mediana de la estatura
> Nombres[1:4] #deseo saber los 4 primeros nombres
[1] "Edwin" "Adriana" "Eduardo" "Ines"
> mean(ej2$Estatura) #deseo saber el promedio
[1] 1.611
> mean(Estatura)
[1] 1.611
> hist(Estatura) #realizo una grafica de la estatura
> median(Estatura) #veo la mediana de la estatura
[1] 1.57

Vectores en R

Los vectores en R se crean de esta manera: A = c()

O también: A = vector(mode="integer", length = length(n)) con n=4

```
########################################
########    VECTORES      ##############
########################################
######## ING. EDWIN MEJÍA ##############
########################################

vector1 <- c(1,5,6,8,9,12)
vector1

vector2 <- c("Edwin","Rosa","Luisa","Gerardo")
vector2

vector3 <- -6:6
vector3

vector4 <- rep(vector1,3)
vector4

vector5 <- sin(seq(0,20,0.5))
vector5
```

```
Console   Terminal ×   Jobs ×
R 4.1.1 · ~/
> vector1 <- c(1,5,6,8,9,12)
> vector1
[1]  1  5  6  8  9 12
>
> vector2 <- c("Edwin","Rosa","Luisa","Gerardo")
> vector2
[1] "Edwin"   "Rosa"    "Luisa"   "Gerardo"
>
> vector3 <- -6:6
> vector3
 [1] -6 -5 -4 -3 -2 -1  0  1  2  3  4  5  6
>
> vector4 <- rep(vector1,3)
> vector4
 [1]  1  5  6  8  9 12  1  5  6  8  9 12  1  5  6  8  9 12
>
> vector5 <- sin(seq(0,20,0.5))
> vector5
 [1]  0.00000000  0.47942554  0.84147098  0.99749499  0.90929743  0.59847214
 [7]  0.14112001 -0.35078323 -0.75680250 -0.97753012 -0.95892427 -0.70554033
[13] -0.27941550  0.21511999  0.65698660  0.93799998  0.98935825  0.79848711
[19]  0.41211849 -0.07515112 -0.54402111 -0.87969576 -0.99999021 -0.87545217
[25] -0.53657292 -0.06632190  0.42016704  0.80378443  0.99060736  0.93489506
[31]  0.65028784  0.20646748 -0.28790332 -0.71178534 -0.96139749 -0.97562601
[37] -0.75098725 -0.34248062  0.14987721  0.60553987  0.91294525
>
```

Otros ejercicios de vectores

```
22  vector5[1:5]   #deseo los 5 primeros
23
24  vector5[(length(vector5)-4):length(vector5)] #los 5 ultimos numeros
25
26  vector5[c(1,3,12)]  #solo los elementos 1,3,12
27
28  vector5[-c(1,3,12)]  #todos menos el 1,3,12
29
30  vector5[vector5>0.98] #mayores a 0.98
31
32  which(vector5>0.98)  #que posiciones ocupan los mayores a 0.98
33
34  vector6=vector5
35  vector6[vector5>=0.7] = 1 #poner a 1 todos los mayores que 0.7
36  vector6
37
38  #todos los negativos poner a cero
39
```

```
Console   Terminal   Jobs
R 4.1.1 · ~/
> vector5[1:5]   #deseo los 5 primeros
[1] 0.0000000 0.4794255 0.8414710 0.9974950 0.9092974
>
> vector5[(length(vector5)-4):length(vector5)] #los 5 ultimos numeros
[1] -0.7509872 -0.3424806  0.1498772  0.6055399  0.9129453
>
> vector5[c(1,3,12)]  #solo los elementos 1,3,12
[1]  0.0000000  0.8414710 -0.7055403
>
> vector5[-c(1,3,12)]  #todos menos el 1,3,12
 [1]  0.47942554  0.99749499  0.90929743  0.59847214  0.14112001 -0.35078323
 [7] -0.75680250 -0.97753012 -0.95892427 -0.27941550  0.21511999  0.65698660
[13]  0.93799998  0.98935825  0.79848711  0.41211849 -0.07515112 -0.54402111
[19] -0.87969576 -0.99999021 -0.87545217 -0.53657292 -0.06632190  0.42016704
[25]  0.80378443  0.99060736  0.93489506  0.65028784  0.20646748 -0.28790332
[31] -0.71178534 -0.96139749 -0.97562601 -0.75098725 -0.34248062  0.14987721
[37]  0.60553987  0.91294525
>
> vector5[vector5>0.98] #mayores a 0.98
[1] 0.9974950 0.9893582 0.9906074
> which(vector5>0.98)  #que posiciones ocupan los mayores a 0.98
[1]  4 17 29
>
> vector6=vector5
> vector6[vector5>=0.7] = 1 #poner a 1 todos los mayores que 0.7
> vector6
 [1]  0.00000000  0.47942554  1.00000000  1.00000000  1.00000000  0.59847214
 [7]  0.14112001 -0.35078323 -0.75680250 -0.97753012 -0.95892427 -0.70554033
[13] -0.27941550  0.21511999  0.65698660  1.00000000  1.00000000  1.00000000
[19]  0.41211849 -0.07515112 -0.54402111 -0.87969576 -0.99999021 -0.87545217
[25] -0.53657292 -0.06632190  0.42016704  1.00000000  1.00000000  1.00000000
[31]  0.65028784  0.20646748 -0.28790332 -0.71178534 -0.96139749 -0.97562601
[37] -0.75098725 -0.34248062  0.14987721  0.60553987  1.00000000
>
```

Crear data.Frames o tablas

Crear un DATAFRAME en R que contenga los siguientes datos.

pais	pbi	ipc	desempleo
Alemania	1.0	1.1	6.0
Francia	1.4	1.2	8.6
Reino Unido	1.1	1.5	3.0
España	2.0	0.4	14.2
Eurozona	1.2	1.0	7.5

Código Rstudio para crear el dataframe con vectores.

```
# Creando variables

pais <- c("Alemania","Francia","Reino Unido","España","Eurozona")
pbi <- c(1.0,1.4,1.1,2.0,1.2)
ipc <- c(1.1,1.2,1.5,0.4,1.0)
desempleo <- c(6.0,8.6,3.0,14.2,7.5)
```

```
# Creando una base de datos (data frame)

datanew <- data.frame(pais,pbi,ipc,desempleo)
```

```
#Construcción de tabla de resultados con formato

kable(datanew, caption = "Base de datos de países",
      align = 'c', digits = round(1))
```

En datanew tenemos nuestro dataframe. Ahora exporte a txt, csv y Excel el mismo.

setwd("C:\\Users\\Familia\\Desktop\\clase8_r") #aquí debe estar la dirección donde almacenar los archivos

Write.table(datanew,file="exportar_datos1.txt") #almacena en .txt

setwd("C:\\Users\\Familia\\Desktop\\clase8_r")

write.table(datanew,file = "exportar_datos2.csv") #almacena en .csv

Antes de utilizar esta función de xlsx instalar java para 32 bits y java para 64 bits en windows

Instalar el paquete en Rstudio

install.packages("rJava")

library(rJava)

install.packages("xlsx")

library(xlsx)

write.xlsx(datanew,"Ejemplo1.xlsx") #almacena en Excel

Así deben asomar los archivos en la carpeta:

Ejercicios propuestos

Realizar con este data.frame lo siguiente:

pais	pbi	ipc	desempleo
Alemania	1.0	1.1	6.0
Francia	1.4	1.2	8.6
Reino Unido	1.1	1.5	3.0
España	2.0	0.4	14.2
Eurozona	1.2	1.0	7.5

a. Visualizar solo la primera fila

b. Visualizar la 1,3 y 5 filas

c. Visualizar la 2 y 4 filas

d. Visualizar con una formula las ultimas 3 filas

e. Visualizar los pbi mayores a 1.2

f. Visualizar los ipc menores a 1

g. Visualizar el desempleo mayor a 7

h. Visualizar todos, menos el 1, 4 y 5

i. Poner a 1 todos los mayores que 1.4 en pbi

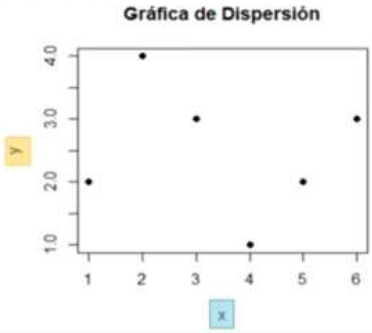
j. Poner a 0 todos los menores a 1 en ipc

Matrices en R

¿Por qué es importante aprender matrices?

- Para poder graficar en R!
  - Una matriz de dos columnas y un gráfica de dispersión son prácticamente lo mismo

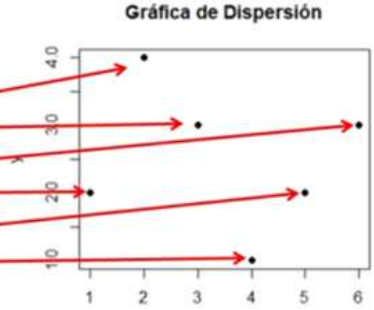
```
> matriz
     x y
[1,] 2 4
[2,] 3 3
[3,] 6 3
[4,] 1 2
[5,] 5 2
[6,] 4 1
```



¿Por qué es importante aprender matrices?

- Para poder graficar en R!
  - Una matriz de dos columnas y un gráfica de dispersión son prácticamente lo mismo

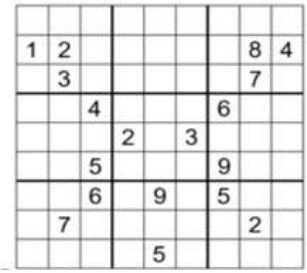
```
> matriz
     x y
[1,] 2 4
[2,] 3 3
[3,] 6 3
[4,] 1 2
[5,] 5 2
[6,] 4 1
```



Por tanto, si NO aprende matrices va hacer bien difícil que aprenda gráficas en R.

¿Qué es una Matriz?

- Es una "forma" de acomodar los datos que tiene renglones/filas y columnas.



¿Cómo hacer una matriz en R?

- Una forma es combinando varios vectores.
- Utilizar la función matrix(...)

```
matrix(data, ← Contenido
       nrow = 1, ← Filas
       ncol = 1) ← Columnas
```

- Cambiar nombres con colname y rowname

La función matrix

Se pueden crear mediante la función **matrix()**

Parámetros básicos:

- **byrow:** indica si se organizan por filas **(TRUE)**, o **por columnas (FALSE**, valor por defecto)
- **nrow** o **ncol:** número de filas o de columnas

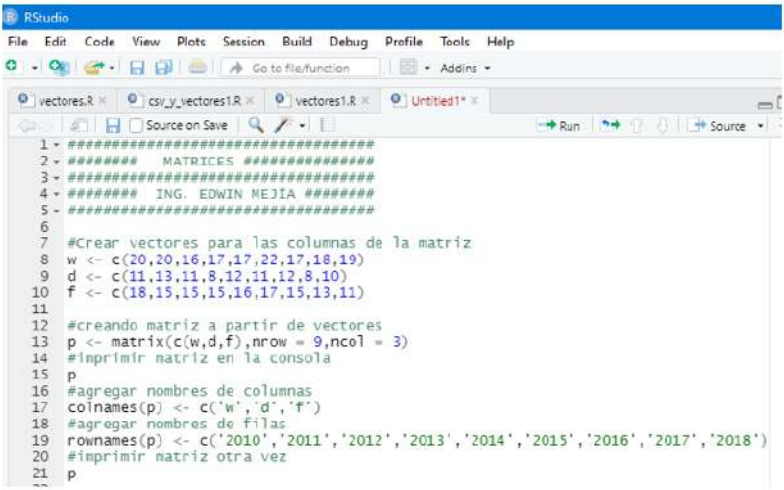
x=1:12

matrix(x, nrow=3)

matrix(x, nrow=3, byrow=TRUE)

matrix(x, ncol=4, byrow=TRUE)

matrix(x, ncol=5, nrow=5, byrow=TRUE)



¿Cómo operar con matrices?

- Sirve para modificar una matriz
- Las operaciones son la suma, resta, multiplicación y división
- Las operaciones básicas entre matrices son elemento a elemento

```r
###################################
### OPERACIONES CON MATRICES ######
###################################

#Restar 5 a la matriz
p-5
#Sumar matriz consigo misma
p + p
#Multiplicar consigo misma
p * p
p
```

Selección de elementos en una matriz?

- Selección de elementos
- Selección de filas/reglones
- Selección de columnas

```r
###################################
### SELECCIÓN DE ELEMENTOS ########
###       DE UNA MATRIZ     ########
###################################

#SELECCIONAR UN ELEMENTO DE LA MATRIZ
p[3,2] #Fila 3 y columna 2
p['2012','d']

#SELECCIONAR MAS DE UN ELEMENTO DE LA MATRIZ
p[c(3,4),c(2,3)]  #fila 3 y 4, columna 2 y 3
p[c('2012','2013'),c('d','f')]

#SELECCIONAR UNA FILA O RENGLON
p[3,] #Selecciono la fila 3
p['2012',]

#SELECCIONAR UNA COLUMNA
p[,2] #Seleccionar la columna 2
p[,'d']
```

```
> #SELECCIONAR UN ELEMENTO DE LA MATRIZ
> p[3,2]
[1] 11
> p['2012','d']
[1] 11
```

```
> #SELECCIONAR MAS DE UN ELEMENTO DE LA MATRIZ
> p[c(3,4),c(2,3)]
      d  f
2012 11 15
2013  8 15
```

```
> p[c('2012','2013'),c('d','f')]
      d  f
2012 11 15
2013  8 15
```

```
> #SELECCIONAR UNA FILA O RENGLON
> p[3,]
 w  d  f
16 11 15
> p['2012',]
 w  d  f
16 11 15
```

ACERCA DE LOS AUTORES

EDWIN FERNANDO MEJÍA PEÑAFIEL

Ingeniero en Sistemas de la ESPOCH. Magíster en Informática Aplicada. Docente Investigador UTB. Docente Investigador de la ESPOCH. Parte del Grupo de Investigación SEGINTE y del grupo GEAA-FIE Grupo de Energias Alternativas y Ambiente de la FIE. Presidente, Director y Miembro de proyectos de maestría de la IPEC-ESPOCH. Director de proyectos de pregrado. Miembro de la primera comisión de facultad de la FIE-ESPOCH. Miembro de la comisión de carrera de Ingeniería Electrónica y Estadística. Miembro de la comisión de evaluación de Ingeniería Industrial. Docente de las asignaturas de Programación, Inteligencia Artificial, Sistemas Expertos, Base de Conocimiento y otras relacionadas con el campo de la Informática y Computación. Director de la Academia Microsoft de la ESPOCH. Director del Club de Robótica. Artículos científicos indexados en diferentes revistas en el campo de la programación e inteligencia artificial.

JOHANNA ENITH AGUILAR REYES

Estudiante graduada con méritos mejor graduada de la promoción en la carrera de Ingeniería en Estadística Informática. Maestría en Gestión y Liderazgo Educacional. Cuenta con certificación de formador de formadores. Docente en la Escuela Superior Politécnica de Chimborazo por 10 años. Directora de planificación zonal 7 en el Ministerio de Educación. Organizadora por varios años de congreso internacional de Estadística. Parte de la comisión de carrera de Estadística. Parte de la comisión de Evaluación y Acreditación de la carrera. Parte de la comisión de Integración Curricular, cuenta con varias publicaciones regionales, investigadora del grupo de investigación Moodeling.

NANCY ELIZABETH CHARIGUAMÁN MAURISACA

Ingeniera en Estadística Informática (ESPOCH). Magíster en Matemática Básica (ESPOCH). Docente – Investigador en la Escuela Superior Politécnica de Chimborazo (ESPOCH) hasta la presente. Actualmente INVESTIGADORA – COORDINADORA del grupo de investigación ESTADISMÁTICA – ESPOCH.

RODRIGO RIGOBERTO MORENO PALLARES

Ingeniero Industrial. Maestría en Ingeniería Industrial y Productividad. Diplomado en Sistemas Integrados de Gestión. Cursos enfocados a la enseñanza. Catedrático en asignaturas como: Control de Calidad y Estadística en la Facultad de Mecánica de la ESPOCH, Matemática Básica, Investigación Operativa, Proyectos, Matemática Financiera en la Facultad de Administración de Empresas y en la Unidad a Distancia de la ESPOCH, Álgebra Superior en la Unidad de Admisión y Nivelación. Trabajos técnicos como Supervisor SSA y Control Mecánico en diferentes campos Área Petrolera, Levantamiento y Mejoramiento de Procesos en General. Consultor en Áreas de Ingeniería.

REFERENCIAS BIBLIOGRÁFICAS

- Carrasco, T. (2016). Novel Netware - Sistema Operativo basado en Sistemas Distribuidos. Consultado el 14 de agosto de 2020. https://sistemasdistribuidos.foroactivo.com/t60-novel-netware-sistema-operativo-basado-en-sistemas-distribuidos
- CGA. (2016). Otros periféricos. Consultado el 20 de mayo de 2019. https://www.juntadeandalucia.es/educacion/cga/mediawiki/index.php/Otros_Perifericos#Grabadoras-Lectoras
- Editorial Etecé. (2021). Algoritmo de Informática. Consultado el 15 de octubre de 2021. https://concepto.de/algoritmo-en-informatica/
- Fernández, Y. (2021). HDD vs SSD: diferencias y ventajas de ambos tipos de disco duro. Consultado el 28 de julio de 2021. https://www.xataka.com/basics/hdd-vs-ssd
- Goodwill Community Foundation. (2021). Qué es MacOS. Consultado el 18 de septiembre de 2020. https://edu.gcfglobal.org/es/curso-de-mac-os/que-es-macos/1/
- Guille, V. (2021). Tipos de memoria de una computadora. Consultado el 22 de agosto de 2021. https://www.tecnologia-informatica.com/tipos-memorias-computadora/
- Llamas, J. (2021). Tipos de monitor. Consultado el 16 de septiembre de 2020. https://economipedia.com/definiciones/tipos-de-monitor.html
- Mastoner. (2021). Que es un plotter y cómo funciona. Consultado el 26 de junio de 2021. https://www.mastoner.com/blog/

post/%C2%BFqu%C3%A9-es-un-plotter-y-c%C3%B3mo-funciona/

- Tecnología Inclusiva. (2021). Que es un Sistema Oeprativo. Consultado el 20 de enero de 2020. https://desarrollarinclusion.cilsa.org/tecnologia-inclusiva/que-es-un-sistema-operativo/
- Yanez, D. (2020). Los 25 Sistemas Operativos Libres Más Relevantes. Lifeder. Consultado el 28 de octubre de 2020. https://www.lifeder.com/sistemas-operativos-libres/.

Centro de Investigación y Desarrollo Profesional
Generador de Conocimientos...

ISBN: 978-9942-607-11-9

CIDE Editorial