SOLUZIONINNOVATIVE S.A.S.

INSTITUTO SUPERIOR TECNOLÓGICO MARIANO SAMANIEGO

Guía Fudamentos de Programación: Programación Estructurada

2024

Danny Jiménez

SOLUZIONINNOVATIVE S.A.S.

EDITORIAL

Guía Fundamentos de Programación: Programación Estructurada

ISBN: 978-9942-7294-0-8

Autor:
Danny Jiménez

SOLUZIONINNOVATIVE S.A.S.

EDITORIAL

Primera Edición, septiembre 2024

Guía de Fundamentos de Programación: Programación Estructurada

ISBN: 978-9942-7294-0-8

Editado por:

Sello editorial: ©Soluzioninnovative S.A.S. Editorial

No Radicación: 164767

Editorial: ©Soluzioninnovative S.A.S.

Editorial Los Andes y El Sufragio

Dirección de Publicaciones Científicas Soluzioninnovative S.A.S.

Editorial Riobamba, Chimborazo, Ecuador

Teléfono: +593967468602

Código Postal: 060108

https://orcid.org/0009-0000-2466-4133

https://doi.org/10.61396/editorialsoluzioninnovative.lib19

Índice general

Índice de tablas

Índice de figuras

Filosofía Institucional

La institución nació del antiguo Colegio Municipal Nocturno que cambia su denominación a colegio Mariano Samaniego en 1973 en el rectorado del Padre Isaac García Martín. Desde 1976 el Hermano Santiago Fernández García asume como rector y desarrolla la infraestructura y la gestión administrativa de la institución hasta su fallecimiento el 11 de noviembre de 1993.

Bajo el rectorado del Dr. Wilson Bravo Ludeña, el 13 de julio de 1998, la unidad educativa se eleva a la categoría de Instituto Técnico Superior Fisco misional Mariano Samaniego según acuerdo ministerial 36-32. Finalmente, mediante Acuerdo No. 173 de la Dirección Ejecutiva del Consejo Nacional de Educación Superior CONESUP, de fecha 22 de enero del 2004 se reconoce al Instituto Técnico Superior Mariano Samaniego, a la categoría de Instituto Tecnológico Mariano Samaniego con las especialidades de Tecnólogos en Análisis de Sistemas y Administración de Empresas.

Con esta introducción histórica actualmente la institución oferta la carrera desarrollo de software en modalidad hibridad y en línea, esta guía es un esfuerzo sincero dedicada a nuestros jóvenes que no poseen los recursos necesarios para trasladarse a otras provincias a desarrollar sus estudios de tercer nivel, declarando como nuestra, la filosofía del Hermano Santiago Fernández García: La educación como un beneficio social y como derecho humano.

Nuestra misión

Somos una Institución de educación superior, con perspectiva evangélica orientada a formar profesionales de nivel técnico y tecnológico, con carácter humanístico, investigativo, emprendedor e innovador; con principios y valores cristianos sólidos que permitan desarrollar en los estudiantes pensamiento crítico, analítico; fomentando la vinculación con la sociedad, el desarrollo sustentable y sostenido contribuyendo a la solución de problemas en los ámbitos: tecnológico, de salud, de educación, cultural, económico, político, social y medio ambiente.

Nuestra visión

Es Visión del Instituto Superior Tecnológico Mariano Samaniego ser la institución educativa reconocida por el liderazgo de sus profesionales en los sectores privado, público y social, por la investigación y desarrollo tecnológico que realiza para impulsar la economía basada en el conocimiento, generará modelos de gestión e incubación de microempresas, la salud, la educación y colaborará en el mejoramiento de la administración pública y sus políticas, creando modelos y sistemas innovadores para el desarrollo sostenible de la comunidad.

Introducción

Esta guía inicia con una breve historia de la computación, y las necesidades que motivaron el desarrollo del procesamiento automático. A continuación, se describe como la memoria es necesaria para el procesamiento y las definiciones de variables asociadas a la memoria. A continuación, se introducen las estructuras de pensamiento para el desarrollo de algoritmos descriptivos.

En el siguiente capítulo se aplican estas estructuras en diagramas de flujo y se detallan las estructuras necesarias para resolver problemas didácticos. Finalmente se desarrollar programación estructurada en el lenguaje de programación JAVA utilizando el IDE Eclipse.

Se plantea una serie de prácticas que incluyen todas las estructuras condicionales, de ciclos determinados e indeterminados y se finalizan utilizando estructuras de almacenamiento el vector y la matriz.

Orientaciones generales para el estudio

Lectura Comprensiva: Se recomienda leer atentamente el material, asegurándose de entender los conceptos fundamentales antes de avanzar.

Práctica Activa: La programación es una habilidad que se perfecciona con la práctica. Por lo tanto, es crucial realizar los ejercicios y actividades propuestas en la guía. Esto ayuda a reforzar los conceptos y a desarrollar la capacidad de resolver problemas de manera autónoma.

Reflexión Crítica: Se sugiere que los estudiantes reflexionen sobre lo aprendido, cuestionando cómo y por qué funcionan las técnicas de programación. Esta reflexión puede incluir la comparación de diferentes enfoques para resolver un problema.

Uso de Recursos Adicionales: En caso de dificultades, se alienta a los estudiantes a utilizar recursos adicionales como tutoriales, foros de discusión, y la tutoría, para aclarar dudas y profundizar en los temas más complejos.

Organización del Tiempo: Se recomienda planificar el tiempo de estudio, dividiendo el material en sesiones manejables y estableciendo metas claras para cada sesión. Esto ayuda a mantener un progreso constante y evita la acumulación de trabajo.

Evaluación y Retroalimentación: Finalmente, se sugiere que los estudiantes evalúen su propio progreso revisando los ejercicios resueltos y buscando retroalimentación, ya sea de la tutoría o de otros estudiantes, para identificar áreas de mejora.

Desarrollo de contenidos

CAPITULO I: Algoritmos

Al finalizar este capítulo el estudiante podrá describir las técnicas esenciales utilizadas para analizar, diseñar y elaborar un algoritmo descriptivo de acuerdo al problema planteado.

1. Generalidades

1.1. Introducción a la computación

A partir del aumento de la población en las sociedades modernas, todos los datos necesarios para la administración pública, así como también de los emprendimientos asociados al desarrollo de las masas humanas, demandan tratamiento de la información. El desarrollo de software es una necesidad imperativa en todas las actividades de la sociedad actual, nuestra era es considerada, la era de las tecnologías de la información. Se ha pasado del procesamiento manual de la información hacia la digitalización generalizada de prácticamente toda la gestión del conocimiento moderno.

La automatización del tratamiento de los datos con la finalidad de producir información fiable, que se encuentre disponible al instante mismo de su tratamiento, es la base del desarrollo de todos los emprendimientos, el tejido industrial y productivo del Ecuador. Los sistemas de información a la par con la evolución de las comunicaciones, ha cambiado nuestras formas de relacionarnos, el comportamiento de los consumidores, y como el comercio realiza sus transacciones.

1.2. Breve historia de la computación

Según (Barceló, 2008), en el año 1880 existía una necesidad creciente de procesamiento de los datos, por citar un ejemplo: después del crecimiento vertiginoso de población en EE. UU, al final del siglo XIX, grandes oleadas de inmigrantes ponían en riesgo de saturación el censo de EE.UU. de 1890, en 1887 aún se estaba procesando manualmente el censo de 1880, sin haberlo completado (History Channel, 2021), los funcionarios encargados de esta tarea necesitaban con gran urgencia una nueva forma de tabulación de datos.

Es entonces cuando aparece Herman Hollerith, quien introdujo una serie de máquinas que facilitarían el tabulado de la información, las personas estaban representadas por tarjetas perforadas en las cuales se representaban los datos como: Edad, sexo, raza y profesión (Barceló García, 2008, p. 30), estas tarjetas se introducían en una máquina que procesaba la información de forma eléctrica basado en la lógica de Boole, los resultados se registraban en manecillas que parecían relojes, estas máquinas aceleraban de forma dramática el procesamiento de la información, es así como en seis semanas se había hecho el recuento del censo de 1890 de 62’622.250 habitantes. (Louridas, 2023).

Este invento se comercializó, y se extendió hacia otros sectores que también necesitaban procesar su información, por ejemplo, los ferrocarriles, Herman Hollerith finalmente fundo una compañía que fue la base para el desarrollo de la IBM. La invención de Herman Hollerith es un hito Histórico del procesamiento analógico al paso del tratamiento digital. 50 años más tarde y motivados por una de las mayores barbaries de nuestra especie, la segunda guerra mundial, se desarrollan nuevas máquinas, enfocadas al procesamiento de la información con propósitos distintos (History Channel, 2021).

Al parecer de (Szymanczyk, 2011), Una de esas máquinas es propuesta por los británicos, entre ellos Tommy Flowers, con la finalidad de romper el código alemán denominado Enigma. Y de esa forma ayudar a ganar la segunda Guerra mundial. Esta máquina se le denomino Colossus, comparaba caracteres a un ritmo de 25.000 caracteres por segundo y con esta capacidad de procesamiento se descifraban mensajes.

EEUU que era el mayor proveedor de armas para la Europa en guerra, tenía escases de tablas de disparos para piezas de artillería, estas tablas eran un complemento de las armas de artillería y permitían calcular los disparos de acuerdo con varios parámetros, altitud, temperatura, velocidad del viento, … entre otros (History Channel, 2021).

El autor Vicente (Trigo Aranda, 2010), describe como John William Mauchly, un físico de la Universidad de Pensilvania desarrollo una maquina llamada ENIAC, que desarrollaba estas tablas de disparo en minutos. Lo mejor de esta máquina, es que eran de propósito general y se podría reconfigurar para otros tipos de procesamiento (History Channel, 2021).

Finalmente, John Presper Eckert y John William Mauchly fundan una compañía y desarrollan la primera máquina de propósito general con el fin de comercializarla, esta máquina se denominó UNIVAC, podría programarse para una serie de procesamientos de datos como: Nominas, inventarios y facturación (History Channel, 2021).

La entrada definitiva del procesamiento automático de la información se consolido, cuando se utilizó la UNIVAC para el pronóstico de las elecciones de EEUU de 1952, la CBS utilizo esta máquina para predecir los resultados, se definió como ganador a Dwight D. Eisenhower con una victoria arrolladora (History Channel, 2021).

Estas máquinas fueron evolucionadas, estimuladas por la guerra fría, por la conquista del espacio, y luego por la reducción de los tamaños gracias a los avances de los transistores, circuitos integrados y finalmente el microprocesador.

Los computadores finalmente se popularizaron, de 500.000 máquinas en 1981 a 7’000.000 para el final de la década.

En la década de los 90 la venta de los computadores se disparó, aparece internet y las tecnologías de las telecomunicaciones, que nos conectan al mundo.

Hoy en día existen computadoras en muchos dispositivos que se conectan al internet y que nos muestran el mundo desde otra perspectiva. La tendencia actual apunta hacia software conectado a la red, a sistemas basados en la nube, a nuevas formas de relacionarse en la red. A la disponibilidad de la información allí donde se tenga disponible una conexión global.

Esta reseña histórica describe la tendencia que marca la evolución de la tecnología, así como también de la ciencia que permite la captura, procesamiento y publicación de resultados.

Esta tendencia nos indica que el camino de las sociedades modernas es la capacitación de sus habitantes en estas tecnologías, que permite el procesamiento de los datos, con la finalidad de producir información veraz.

Nuestro país no ha escapado al desarrollo tecnológico y toda esta evolución nos afecta de forma directa, según las estadísticas del (INEC, 2024), el INSTITUTO NACIONAL DE ESTADISTICAS Y CENSOS, en el Ecuador en 2015 el 27,7% de la población cuenta con computadores de escritorio y el 24,8% tiene computadores portátiles. También, se detalla que en nuestro país en 2015 el 41% de la población urbana tiene acceso a internet, y el 13,7% de la población rural. El 57,6% de la población usa computadoras en sus actividades cotidianas. Y el 50,5% usa el internet.

1.3. Conceptos básicos y generaciones de las computadoras

1.3.1. Definición de computador

Existen muchos conceptos de ordenador o computador, enfocados desde distintos puntos de vista. Desde nuestra perspectiva, enfocada al desarrollo de software, para Joyanes Aguilar & Zahonero Martinez (2005), definimos que un computador es: Un conjunto de elementos eléctricos (hardware) y programas (software), que interactúan entre sí, con la finalidad de procesar datos y producir información. Las computadoras han evolucionado desde los motores analíticos de Charles Babbage a los dispositivos móviles actuales, que nos permiten nuevas posibilidades de procesamiento y comunicaciones. Estos avances tecnológicos se han marcado en generaciones que podemos identificar con algunas características bien definidas.

1.3.2. **Generaciones de computadoras**

Para Amaya Amaya (2009), las generaciones de los computadores se pueden resumir en:

Primera generación (1940-1952).

- Eran computadoras muy grandes, ocupaban habitaciones enteras, consumían mucha electricidad y generaban mucho calor.
- Se basaban en válvulas de vacío, que constituían los elementos de control para estas computadoras.
- Su uso fundamental fue en aplicaciones científicas y militares.
- Se empieza a usar el sistema binario para representar la información
- Para conservar la información se usaban las tarjetas perforadas.

Segunda generación (1952-1964).

- Se sustituye la válvula de vacío por el transistor. Los transistores eran más rápidos, pequeños y más confiables que los tubos al vacío.
- Las máquinas ganaron potencia y fiabilidad, disminuyendo tamaño, consumo y precio, haciéndose más prácticas y asequibles. Se expanden los campos de aplicación, al administrativo y de gestión.
- Comienza a utilizarse lenguajes de programación más desarrollados, que hacían más sencilla la programación.
- Se usa memoria externa, la cinta y los tambores magnéticos.

Tercera generación (1964-1971).

- En 1964 surge el circuito integrado (chip), que consistía en el encapsulamiento de gran cantidad de componentes electrónicos.
- Las computadoras pudieron hacerse más pequeñas, ligeras y eficientes. Consumían menos electricidad, por tanto, generaban menos calor.
- La miniaturización se extendió a todos los circuitos de la computadora.
- Hubo un gran desarrollo de los sistemas operativos, en los que se incluyó la multiprogramación, en tiempo real y el modo interactivo. Comienza a utilizarse los discos magnéticos.

Cuarta generación (1971-1981).

- En 1971 Aparece el microprocesador, que permite la integración de toda la unidad central de proceso de una computadora en un sólo circuito integrado.

- Esta tecnología permite la fabricación de microcomputadoras y computadoras personales.
- Se utiliza el almacenamiento externo se utiliza el disquete (floppy disk).
- Se desarrollan las supercomputadoras, aparecen nuevos lenguajes de programación de todo tipo y las redes de transmisión de datos (teleinformática).

Quinta generación (1981-1990).

- A partir de esta generación ya no hay diferencia en la tecnología que se utiliza para la creación de las máquinas, sino en la manera en que se emplea.
- La quinta generación se diferencia por la interconexión entre todo tipo de computadoras, dispositivos y redes (redes integradas).
- Comienzan a crearse esquemas de funcionamiento en paralelo.
- Desarrollos en base a inteligencia artificial, robótica y sistemas expertos.
- Utilización del lenguaje natural (lenguajes de quinta generación). Integración de datos, imágenes y voz (entornos multimedia).

Sexta generación (1990- ?)

- Se caracteriza por la evolución de las comunicaciones a la par de la tecnología.

1.4. Arquitectura de un computador

Los sistemas de cómputo tienen una finalidad muy puntual desde el punto de vista del desarrollo de aplicaciones, su objetivo principal es automatizar el procesamiento de datos con la finalidad de producir información valida y eficaz.

Para poder realizar esto es necesario que existan datos de entrada, luego un proceso que es el objeto de la aplicación, cuyo resultado es la información. Esto coincide con la arquitectura que describiéremos a continuación.

Este ha sido el problema clásico de la computación desde sus orígenes, Jhon Von Neumann, planteo la arquitectura de un computador que se sigue usando aun en las computadoras modernas.

Figura 1.- Diagrama simplificado de la arquitectura de un computador, elaboración propia

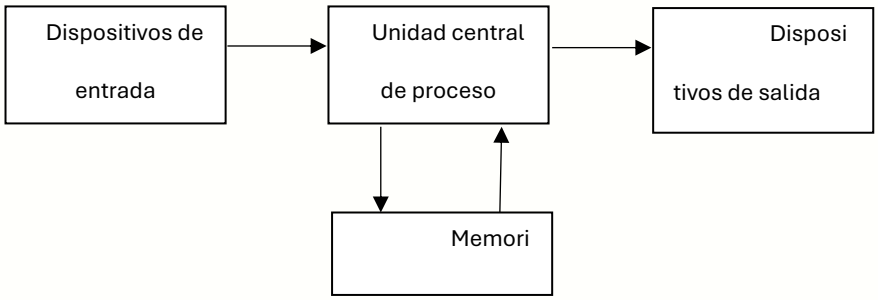
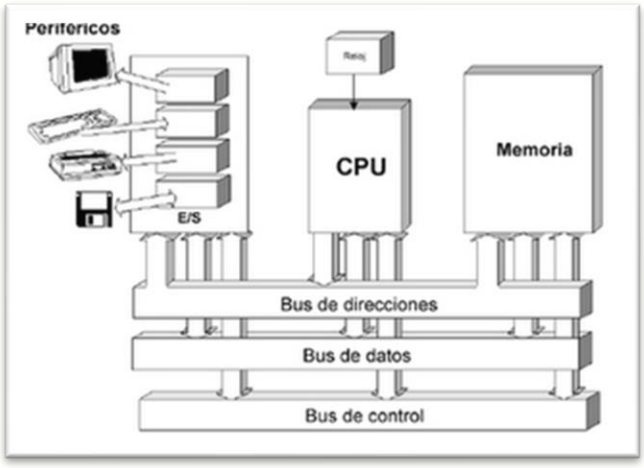


Figura 2.- Arquitectura de Von Neumann



1.5. Como se ejecuta un programa

Para explicar esto, partiremos de la analogía de como nosotros procesamos nuestros cálculos. Supongamos lo siguiente; Suponiendo que tu maestro de secundaria te realiza la siguiente pregunta: Cuanto es siete multiplicado por siete, seguramente la respuesta es inmediata, el estudiante responde es cuarenta y nueve.

Lo importante en este planteamiento no es la respuesta sino como se realizó el proceso. Podríamos detallar lo siguiente:

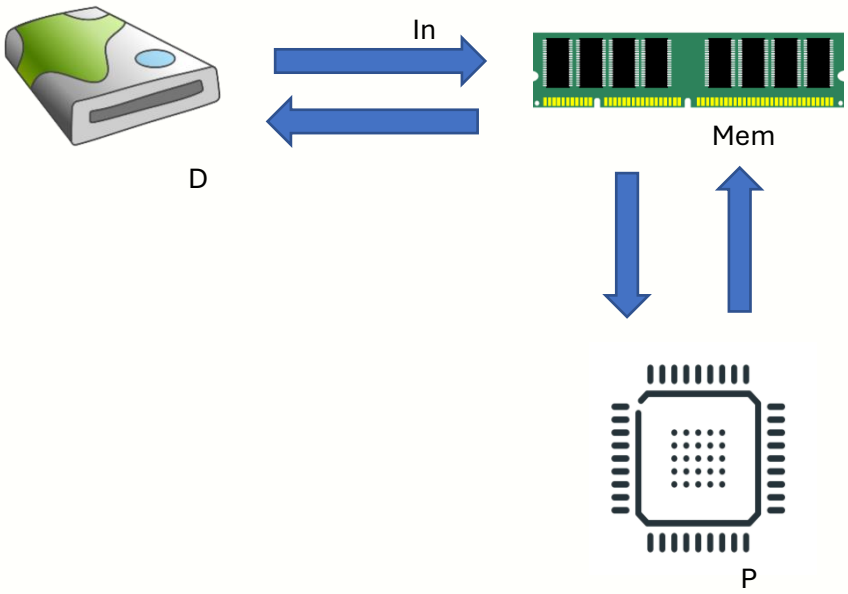
1. Mediante su voz (Unidad de salida) el docente realiza el cuestionamiento ¿Siete multiplicado por siete?
2. El estudiante mediante su oído (Unidad de entrada) capta el mensaje y lo almacena en su memoria, teniendo presente que los dos utilizan el mismo protocolo de comunicación “el castellano”.
3. A continuación, el estudiante, busca en sus registros almacenados en su *memoria* y determina el procedimiento más adecuado para realizar el cálculo; Tiene memorizada la tabla del siete, realiza aproximaciones, realiza sumas.
4. Calcula el resultado y lo almacena en su memoria.
5. Finalmente informa mediante su voz el resultado de la operación (Unida de salida).

Lo más importante, es que hemos identificado aspectos fundamentales del procesamiento, lo principal es que sin memoria no podría ejecutarse el proceso y que necesariamente deben también existir procedimientos que permitan realizar el procesamiento, además de unidades de entrada y salida, que coincide con el planteamiento de la arquitectura de un computador.

En un computador sucede algo similar, primero se lee las instrucciones necesarias del almacenamiento permanente (disco duro), a continuación estas instrucciones se cargan a la memoria

RAM seguidamente se ejecutan en el procesador, estas instrucciones a su vez pueden nuevamente leer o escribir en la memoria y nuevamente ejecutarse, además si es necesario solicitar datos desde las unidades de entrada (teclado), hasta que finalmente por el monitor del computador u otra unidad de salida se presentara el resultado.

Figura 3.- Como se ejecuta un programa, elaboración propia

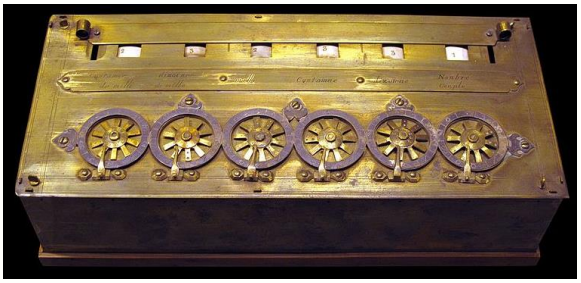


1.6. Representación de la información en un computador

Para que se pueda realizar el procesamiento, necesitamos almacenar nuestra información en memoria, y para poder almacenarla debemos representarla en un lenguaje que nos permita escribir y recuperar información.

Los primeros dispositivos que permitían realizar cálculos eran enteramente manuales, algunos como la "pascalina" permitía realizar cálculos a través de engranajes, se utilizaba la representación numérica en base decimal. Sin embargo, realizar el procesamiento en base decimal complicaba los procesos.

Figura 4.- Dispositivo la pascalina



Nota.- Licencia Creative Commons

El siguiente avance en el procesamiento de cálculos se da según (Martínez & García-Beltrán, 2000): “en 1939, el profesor John Vincent Atanasoff de la Universidad de Iowa junto con su compañero Clifford Berry, desarrollaron una máquina de calcular, conocida como ABC (AtanasoffBerry Computer), estaba basada en tubos de vacío y operaba en sistema binario”

La utilización del sistema numérico binario, fue una verdadera revolución, facilitaba las operaciones y el almacenamiento de la información.

El sistema binario también es la base de cálculo de instrumentos como el ábaco, que apareció hace aproximadamente 5000 años, en la cuna de la civilización el valle del “Tigris-Eufrates”, al sur este de Asia, el ábaco de polvo era una pequeña superficie cubierta de polvo en la que se realizaban cálculos.

En la América prehispánica, existía la Yupana (ábaco inca) (Constanza Mora & Valero, 2020), una herramienta de cálculo, también conocida como Taptana, se trata de tableros con escaques o casilleros encontrados en todo el Tawantinsuyo. Según el uso que se le da, toma las denominaciones de Yupana o Taptana; si se usa para hacer cálculos aritméticos a manera de ábaco, se le llama Yupana y si se utiliza como tablero de juego, se denomina Taptana (Rojas-Gamarra & Stepanova, 2015).

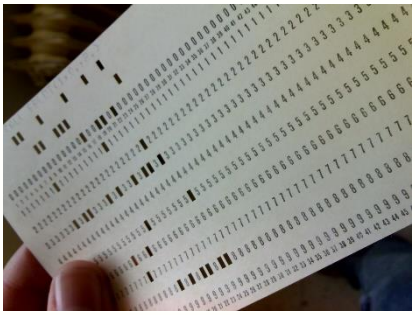
El khipu utilizado por las culturas americanas prehispánicas, consistía en nudos a lo largo de cuerdas con espacios simétricos, la ausencia del nudo representaba el cero, aun no se ha definido una teoría que interprete claramente su uso, pero el conceso es que representan números, existen muchos tipos de khipus; grupos de nudos de hasta nueve, de más de nueve, de cuerdas de colores, de tipos de nudos distintos, aparentemente su uso se extiende más allá de la contabilidad y uso estadístico, podrían representar información histórica como los árboles genealógicos del Ayllu, calendarios solares u otros.

La facilidad de utilizar dos símbolos (el cero y el uno) para la representación de la información, permite que se desarrollen las primeras formas de almacenamiento de información, como las tarjetas perforadas, que fueron utilizadas ampliamente en telares como programas de tejidos.

Con esta tecnología rudimentaria aparece la memoria, en la cual podemos guardar programas, para luego ejecutarlos, almacenar datos con la finalidad de procesarlos y generar resultados o información.

También estos dos símbolos, el cero puede representar FALSO y el uno VERDADERO, permitiendo que sean la base del procesamiento de un ordenador.

Figura 5.- Tarjeta perforada (Licencia Creative Commons)



2. Introducción a los algoritmos

La palabra algoritmo toma su nombre de Al-khôwarizmi un matemático y astrónomo del siglo IX quien, al escribir un tratado sobre manipulación de números y ecuaciones, el Kitab al-jabr w'almugabala, usó en gran medida la noción de lo que se conoce hoy como algoritmo.

Un algoritmo, desde nuestro punto de vista (la programación) se puede definir como: *"Conjunto ordenado de pasos, que se deben ejecutar con la finalidad de resolver un problema"* (Cairó Battistutti, Metodología de la Programación, 2008), existen distintos tipos de algoritmos:

2.1. Tipos de algoritmos

Oviedo Fadul (2004), describe dos tipos de algoritmos:

2.1.1. Cualitativos. - Son todos aquellos pasos o instrucciones descritos por medio de palabras que sirven para llegar a la obtención de una respuesta o solución de un problema.

2.1.2. Cuantitativos. - Son todos aquellos pasos o instrucciones que involucran cuantitativos cálculos numéricos para llegar a un resultado satisfactorio.

2.2. Características de un algoritmo

Para Cairó Battistutti (2008), las características de un algoritmo son:

- *Un algoritmo debe resolver el problema para el que fue formulado.*

Es preciso determinar que los algoritmos deben plantearse de acuerdo a un problema y el cual necesariamente deben resolver, no tendría sentido que el algoritmo no resuelva el problema para el que fue planteado.

- *Los algoritmos son independientes del computador*

Al tratarse de algoritmos, estos no están codificados en un lenguaje de programación, por tanto, pueden implementarse en cualquier sistema de cómputo. Si un algoritmo esta implementado en un lenguaje de programación se convierte en programa.

- *Los algoritmos tienen que ser precisos.*

AL tener como objetivo, resolver un problema específico, el algoritmo no puede ser ambiguo, y debe ser preciso en la resolución del problema.

- *Los algoritmos tienen que ser finitos.*

Un algoritmo debe tener inicio y un fin determinado por el propio procedimiento, se considera un error si un algoritmo se repite por infinito número de pasos. Un algoritmo es escrito con un objetivo: conseguir un resultado. El algoritmo no cumple su cometido si tenemos que espere infinitos ciclos de tiempo.

- *Los algoritmos deben de poder repetirse.*

Los algoritmos deben poder repetirse las veces que sean necesarias, si utilizamos la misma entrada debe producir la misma salida dentro de los parámetros determinados por el procedimiento definido en el mismo.

2.3. Elementos que conforman un algoritmo

Un algoritmo esquemáticamente se puede expresar en:

- *Entrada*. Los datos iniciales que posee el algoritmo antes de ejecutarse o que se solicitan en la ejecución.
- *Proceso*. Acciones que lleva a cabo el algoritmo y para el cual fue escrito.
- *Salida*. Información que obtiene finalmente el algoritmo.

Figura 6.- Esquema de un algoritmo

3. Algoritmos descriptivos

Un algoritmo descriptivo, describe por medio de palabras el procedimiento que debe seguirse para el resolver el problema planteado, es un algoritmo de alto nivel y pueden existir situaciones ambiguas, sin embargo, determina el procedimiento adecuado.

Existen situaciones en las que se asume que la solución es explicita y no necesita de un algoritmo específico para acciones determinadas.

Este tipo de algoritmos se presentan en esta guía como una forma de introducir los razonamientos necesarios para poder describir, utilizar y comprender las estructuras de programación que se presentarán más adelante.

Ejemplo 1 :

Desarrolle un algoritmo que permita a un estudiante asistir a clases en el instituto tecnológico superior Mariano Samaniego.

Un procedimiento posible es el siguiente:

1. Preparar los materiales
2. Ducharse
3. Vestirse
4. Merendar
5. Cepillarse los dientes
6. Salir de casa
7. Caminar al Instituto Tecnológico Superior Mariano Samaniego
8. Llegar al ITSMS
9. Buscar el aula
10. Ubicarse en un asiento
11. Atender a la clase

Se puede determinar que es un conjunto de instrucciones numeradas que indican su orden de ejecución, sin embargo, se pueden invertir algunas de ellas consiguiendo el mismo resultado. Se puede invertir el paso uno con el paso dos, es decir un estudiante puede ducharse primero y luego preparar los materiales o viceversa, sin que el resultado final se altere.

También debería ser explicito para un estudiante de la institución el paso siete, sin embargo, si es un estudiante de otra institución, nuestro algoritmo puede fallar porque dicho estudiante no conoce la dirección.

Se pude advertir que el paso siete y el ocho, no se pueden invertir porque la lógica del procedimiento fallaría.

De lo anterior podemos concluir que los algoritmos descriptivos pueden presentar situaciones en las que, si el contexto del algoritmo no es el adecuado, no será posible su ejecución o simplemente el resultado no será el esperado. También debemos precisar que para este tipo de algoritmos el contexto, el alcance y el dominio del problema a resolver deben estar claros y delimitado.

Ejemplo 2 :

Desarrollar un algoritmo descriptivo que permita cambiar la rueda de un vehículo.

1. Apagar el motor
2. Poner el freno de mano
3. Asegurarse que se encuentra en la primera marcha.
4. Asegurar que el vehículo no se desplace.
5. Buscar la gata.

6. Colocar la gata
7. Aflojar las tuercas
8. Levantar el vehículo con la gata
9. Terminar de sacar las tuercas
10. Quitar la rueda
11. Buscar la rueda de emergencia
12. Colocar la rueda
13. Apretar las tuercas
14. Bajar el vehículo
15. Verificar que las tuercas estén bien apretadas
16. Fin, Termino de cambiar la rueda

En este ejemplo podemos observar que en el contexto se debe asumir pasos que podrían ser un nuevo algoritmo, *Apagar el motor* puede llevar una serie de pasos que se debe asumir que quien realice la ejecución del algoritmo lo puede hacer.

Hasta ahora se han desarrollado dos ejemplos que permiten determinar que un algoritmo se debe implementar a través de pasos ordenados de forma lógica, sin embargo, los algoritmos finalmente deben implementarse en un ordenador que permita el tratamiento automático mediante un lenguaje de programación.

3.1. Variables

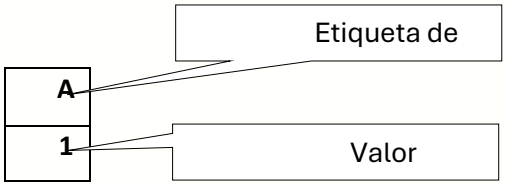
Si recordamos en el capítulo uno, determinamos que es necesario la existencia de memoria para que se pueda realizar un cálculo, es necesario poder almacenar cierta información y que la podamos referenciar, de la misma forma es necesario definir espacios para almacenar datos, a estos espacios los denominamos VARIABLES.

*Una variable es un espacio de memoria reservado para almacenar un valor que corresponde a un tipo de dato, u*na variable es representada y usada a través de una etiqueta (un nombre).

La podemos imaginar con un almacén que permite guardar datos, cambiarlos o actualizarlos.

La instrucción: A=1; la podemos ilustrar de la siguiente forma:

Figura 7.- Esquema de una variable



Análisis de instrucciones individuales.

1	Se reserva o define la variable A	A	*Se define un espacio de memoria vacío*
2	Se asigna A=1;	A 1	*Se almacena el uno en la variable*
3	Se asigna A=5;	A 5	*Se desecha el valor anterior y se almacena el cinco*

Análisis de instrucciones individuales.

1	Se define la variable A=2	A 2	*Se define un espacio de memoria A y se guarda el dos.*
2	Se asigna B=A;	A B 2 2	*Se define un nuevo espacio de memoria B y se guarda el contenido de A, utilizamos la etiqueta no confundir que A en este caso no es un valor sino más bien una etiqueta que permite usar su contenido.*
3	Se asigna A=A+1;	A B 3 2	Se actualiza el contenido de la memoria A con el contenido anterior que es dos y se le suma uno, el total que es tres que se guarda en la misma variable A.

Este último paso, el numeral tres, permite realizar una serie de procedimientos básicos y fundamentales en la programación, se le conoce como *variable contadora*.

3.2. Nombre de las variables

Es importante recordar que las variables son necesarias para poder realizar los procesos que son el objetivo de los algoritmos, los nombres que las definen son importantes para la comprensión de cada paso que realiza en el proceso, en este sentido planteamos las siguientes normas que rigen como etiquetarlas:

Utilizaremos letras o palabras.

- Los nombres de las variables no pueden empezar con un número
- Si el nombre de la variable está compuesto por dos palabras la siguiente palabra puede ser separada por un guion bajo, o la primera letra de la siguiente palabra debe iniciar con mayúscula, ejemplos: “Promedio_edades”, ”PromedioEdades”
- El nombre de la variable no puede incluir espacios en blanco
- Los nombres de las variables pueden contener números a partir del segundo carácter, ejemplo: “Numero1”.
- Los nombres de las variables deben tratar de explicitar la operación o proceso a representar, para almacenar el resultado de la suma de dos números podremos podríamos utilizar el nombre: “SUMA”.
- Los nombre de las variables no pueden contener caracteres especiales como: @, / ,\.

3.3. Lógica de los algoritmos descriptivos

Con los algoritmos descriptivos se pretender establecer un procedimiento claro y conciso acerca de la resolución de problemas específicos, sin embargo, en este capítulo nos orientamos a un contexto específico, con la finalidad de orientar y plantear la lógica básica de las estructuras de programación. Se busca establecer la narración de cómo se resolvería un problema dado, de una forma coherente, se establecerá una secuencia que puede ser interpretada en cada uno de sus pasos con la finalidad de llegar a un resulta claro y sin contradicciones.

Utilización de operaciones aritméticas

Para las operaciones se puede utilizar los símbolos convenciones:

Tabla 1.- Símbolos convencionales para las operaciones aritméticas

Nro.	Operación	Símbolo
1	Suma	+
2	Resta	-
3	Multiplicación	*
4	División	/
5	Resto	%

Ejemplo 3 : Desarrollar un algoritmo descriptivo que permita sumar los números 2 y 4.

1. Inicio
2. Inicializamos la variable SUMA=0;
3. Calculamos la siguiente operación: SUMA=2+4;
4. Imprimimos el contenido de SUMA
5. Fín

Del algoritmo anterior podemos concluir que nuevamente se hace necesaria la utilización de una variable en este caso SUMA, que le damos un valor inicial de cero.

Es evidente que no necesitamos de un algoritmo para calcular la suma de los números 2 y 4, sin embargo, esto nos permite establecer la lógica siguiente:

- Declaramos una variable SUMA, que la utilizaremos como un almacén, que inicialmente guarda el valor cero, a este procedimiento le llamaremos inicialización de la variable.
- El algoritmo también realiza el cálculo de la suma de los dos números y los guarda en la variable SUMA, esto nos permite imprimir su contenido como resultado final, esta lógica es necesaria, porque nos permite automatizar los cálculos y la impresión de los resultados.

Ejemplo 4 : Desarrollar un algoritmo descriptivo que solicite que se ingresen dos números, el algoritmo debe imprimir la suma y la multiplicación de estos dos números.

1. Inicio
2. Inicializamos las variables: SUMA=0, MUMLTIPLICACIÓN=0, N1=0, N2=0;
3. Solicitamos que se ingrese el primer número y lo guardamos en la variable N1.
4. Solicitamos que se ingrese el segundo número y lo guardamos en la variable N2.
5. Calculamos la suma: SUMA=N1+N2;
6. Calculamos la Multiplicación: MUMLTIPLICACIÓN =N1*N2;
7. Imprimimos el resultado con el siguiente mensaje: “La suma es:”+SUMA+” La multiplicación es: + MULTIPLICACIÓN”
8. Fin

De algoritmo podemos concluir lo siguiente:

- No podemos diseñar un algoritmo sin la comprensión completa, exacta y clara del problema a resolver, esta es la fase de *análisis.*
- Es necesario planear como resolver el problema, establecer que es necesario, esta es la fase de *diseño*.

- Entendido el problema y planeada su solución podemos escribir el algoritmo, esta fase se *desarrolló*.
- Del algoritmo anterior también podemos concluir que lo podemos repetir con números distintos, o con datos diferentes. Ampliando su aplicación.
- Nuevamente se evidencia que la utilización de las variables permite que los números ingresados puedan ser distintos, y la lógica del programa funciona adecuadamente, también es importante que los nombres de las variables sean adecuados, por ejemplo, el cálculo de la suma se almacena en la variable denominada "SUMA".
- Se diferencia claramente entre datos de entrada (los números ingresados), y la información resultante (La suma y la multiplicación)
- En la línea 7, podemos observar que se configura un mensaje, el texto escrito entre comillas ("La suma es:"), se desplegara textualmente, el signo + indica que se concatena el texto entre comillas y el contenido de la variable SUMA por ejemplo. El resultado de este mensaje será:

"La suma es"+ SUMA + "La multiplicación es:"+ MULTIPLICACIÓN

La suma es 10 La multiplicación es: 25

Ingreso o entrada de datos

Los algoritmos desde nuestra perspectiva siempre tendrán el objetivo de ser implementados en un computador, en este sentido es importante aclarar que los programas que se procesan mediante un computador la entrada de datos normalmente se realiza desde el teclado. Asumiremos que las entradas en los algoritmos se realizarán desde el teclado.

Ejemplo 5 : Desarrolle un algoritmo descriptivo que solicite un valor en una variable X, el algoritmo debe calcular e informar el resultado de la ecuación 2x+3.

1. Inicio
2. Inicializamos la variable X=0, RESULTADO=0
3. Solicitamos que se ingrese el valor X, con el mensaje "Ingrese el valor de X", ->X
4. Calculamos: RESULTADO=2*X+3
5. Imprimimos el resultado: "El resultado de 2x+3 es:"+ RESULTADO
6. Fin

De este algoritmo podemos concluir lo siguiente:

- En la línea 3 solicitamos el valor de X, y asumimos que lo que digite en el teclado, luego de presionar la tecla *enter*, se almacenara en X
- En la línea cuatro utilizamos el símbolo * que lo usamos como la operación de la multiplicación.

Estructuras lógicas

La capacidad de procesamiento de los algoritmos se basa en estructuras lógicas que estandarizan el procesamiento y la comprensión de la lógica de procesamiento.

Tabla 2.- Tipos de instrucciones

Nro.	Nombre de la estructura	Descripción
1	De salto	Permite saltar de una instrucción a otra
2	Condicional	Permite evaluar el contenido de una variable, y en decencia del resultado ejecutar un procedimiento u otro.
3	De Repetición	Permite repetir una parte del procedimiento de acuerdo a una condición.

Estas estructuras son la base del procesamiento estructurado, su combinación genera nuevas estructuras.

Estructuras de salto y condicional

La estructura *condicional* nos permite evaluar una condición, con dos posibles resultados: verdadero o falso, si el resultado es verdadero podemos definir un bloque de instrucciones, si el resultado es falso podemos definir un bloque distinto de instrucciones. Esto nos permite ejecutar código alternativo de acuerdo a la condición selectiva.

La estructura de *salto* permite *evaluar* una condición y saltar parte del código de acuerdo a un planteamiento lógico.

Ejemplo 6 : Realice un algoritmo descriptivo, que nos permita decidir si un estudiante se puede matricular a décimo año del bachillerato general unificado de la modalidad nocturna, tiene que tomar en cuenta que solo se pueden matricular estudiantes mayores de 15 años.

1. Inicio
2. Inicializamos las variables: EDAD=0, X=0;
3. Solicitamos que se ingrese la edad y la almacenamos en la variable EDAD

4. Preguntamos: ¿El contenido de la variable EDAD es mayor que 15?

4.1. Si la respuesta es que si

4.1.1. Imprimimos: “El estudiante SI se puede matricular a décimo año”

4.1.2. Saltamos al paso:5

4.2.Si la respuesta es que no

4.2.1. Imprimimos: “El estudiante NO se puede matricular a décimo año”

4.2.2. Saltamos al paso: 5

5. Fin

De este algoritmo podemos inferir:

- En el paso cuatro preguntamos si el contenido de la variable EDAD es mayor que quince, de la misma manera podríamos preguntar: *es diferente, menor, mayor igual y menor igual a.*
- En el paso 4.1 se ejecuta el bloque de código cuando EDAD es mayor que 15.
- En el paso 4.1.1 Se imprime el resultado.
- En al paso 4.1.2 se produce un salto al paso 5, que es el fin del algoritmo, es necesario este salto porque ya se ha cumplido el objetivo del algoritmo, determinar que si se puede matricular el estudiante.
- En el paso 4.2 se ejecuta el bloque de código cuando EDAD es menor que 15.

Ejemplo 7 : Realice una algoritmo descriptivo, que solicite que se ingrese un número, el algoritmo debe informar si el número es positivo o negativo.

1. Inicio
2. Inicializamos la variable N=0;
3. Preguntamos: ¿El contenido de la variable N es mayor que cero?

3.1.Si la respuesta es que SI

3.1.1. Imprimimos: “El número es positivo”

3.1.2. Saltamos al paso: 4

3.2.Si la respuesta es que NO

3.2.1. Imprimimos: “El número es negativo o cero”

3.2.2. Saltamos al paso: 4

4. Fin

Ejemplo 8 : Realice un algoritmo descriptivo que permita encontrar el resultado de la ecuación de 1º grado: 5x+6x+7, el valor de x debe ser solicitado en el algoritmo. Debe comprobar si el valor ingresado es válido, es decir que es un número.

1. Inicio
2. Inicializamos la variable X=0, RESULTADO=0
3. Solicitamos que se ingrese el valor X, con el mensaje “Ingrese el valor de X”, ->X
4. Preguntamos: ¿X es un número?
   - 4.1. Si la respuesta es SI
     - 4.1.1. Calculamos la operación: RESULTADO=5*X+6*X+7
     - 4.1.2. Imprimimos: “El resultado de la ecuación 5x+6x+7 es:”+ RESULTADO
     - 4.1.3. Saltamos al paso 5
   - 4.2. Si la respuesta es NO
     - 4.2.1. Imprimimos: “Error, El valor ingresado no es un número”
     - 4.2.2. Saltamos al paso 5
5. Fin

Ciclos, estructuras repetitivas

Estas estructuras permiten que un bloque de código se repita constantemente de acuerdo a una condición planteada por la lógica del algoritmo.

Ejemplo 9 : Realice un algoritmo descriptivo que permita contar las bolas de billar de una caja de madera.

1. Inicio
2. Inicializamos la variable C=0;
3. Preguntamos: ¿Existen bolas en la caja de madera?
   - 3.1. Si la respuesta es que SI
     - 3.1.1. Sacamos una bola de billar
     - 3.1.2. Actualizamos el valor de C, con la operación C=C+1;
     - 3.1.3. Saltamos al paso 3
   - 3.2. Si la respuesta es que NO
     - 3.2.1. Saltamos al paso 4
4. Imprimimos: “Existen” + C + “ Bolas de billar”
5. Fín

Para configurar un ciclo debemos considerar:

- Que el cuestionamiento sea adecuado; en este caso ¿Existen bolas en la caja de madera?, nos permitirá saber cuándo finalizar el ciclo (Cuando ya no existan bolas)

- Debemos considerar que el ciclo se repetirá mientras la respuesta al cuestionamiento sea afirmativa (Que si existan bolas en la caja).
- Debe existir dentro del ciclo una instrucción que permita que el ciclo termine, la instrucción del paso *3.1.1 Sacamos una bola de billar*, permitirá que la caja en algún momento se quede vacía, esto cambiara el resultado del cuestionamiento de la pregunta del paso *3. Preguntamos: ¿Existen bolas en la caja de madera?,* y esto terminara el ciclo.

Ejemplo 10 :Realice un algoritmo descriptivo para contar e informar las bolas de billar pares de una caja de madera.

1. Inicio
2. Inicializamos la variable C=0;
3. Preguntamos: ¿Existen bolas en la caja de madera?
   - 3.1.Si la respuesta es que SI
     - 3.1.1. Sacamos una bola de billar
     - 3.1.2. Preguntamos: ¿La bola de billar contiene un número par?
       - 3.1.2.1. Si la respuesta es que Si
         - 3.1.2.1.1. Actualizamos el valor de C, con la operación C=C+1;
         - 3.1.2.1.2. Saltamos al paso 3
       - 3.1.2.2. Si la respuesta es que NO
         - 3.1.2.2.1. Saltamos al paso 3
   - 3.2.Si la respuesta es que NO
     - 3.2.1. Saltamos al paso 4
4. Imprimimos: “Existen” + C + “Bolas de billar pares”
5. Fín

3.4. Seguimiento a la ejecución de un algoritmo

A medida que se utilizan todas las estructuras lógicas en un solo algoritmo, se complica el seguimiento a la ejecución del mismo, para ello existe una técnica que nos permite conocer el estado de las variables en cada paso de ejecución, se trata de registrar las entradas, los cambios que se produzcan en las variables y las salidas de nuestros algoritmos.

Para poder hacer esto debemos crear una tabla con las siguientes columnas.

Tabla 3.- Tabla de seguimiento de ejecución de un algoritmo descriptivo

Nro.	Variable: X	Variable: RESULTADO	Variable: …	Salidas o comentarios

Nro.- Indica un número secuencial que representa el orden de ejecución

Variable: X, Variable: RESULTADO, Variable: … .- Se debe crear una columna por cada variable de nuestro algoritmo.

Salidas o comentarios. - Aquí se registrarán las salidas de nuestro algoritmo y los comentarios que aclaren el paso de ejecución del algoritmo.

Ejemplo 11 : Realice un algoritmo descriptivo que permite imprimir los números pares pertenecientes a las bolas de billar pares de una caja de madera.

1. Inicio
2. Inicializamos la variable NUM=0;
3. Preguntamos: ¿Existen bolas en la caja de madera?
   - 3.1.Si la respuesta es que SI
     - 3.1.1. Sacamos una bola de billar
     - 3.1.2. Preguntamos: ¿La bola de billar contiene un número par?
       - 3.1.2.1. Si la respuesta es que Si
         - 3.1.2.1.1. Actualizamos NUM con el número de la bola;
         - 3.1.2.1.2. Imprimimos el contenido de la variable NUM;
         - 3.1.2.1.3. Saltamos al paso 3
       - 3.1.2.2. Si la respuesta es que NO
         - 3.1.2.2.1. Saltamos al paso 3
   - 3.2.Si la respuesta es que NO
     - 3.2.1. Saltamos al paso 4
4. Fín

Para poder realizar el seguimiento, asumiremos que existen 5 bolas en la caja y que el orden de salida delas bolas es el siguiente: 5, 4, 6, 7, 9.

Tabla 4.- Matriz de seguimiento ejercicio 11

Nro.	Variable: NUM	Salidas o comentarios
1		*Inicio*
2	0	En el paso 2 se inicializa NUM
3	4	
4		Imprime: 4
5	6	
6		Imprime: 6
7		*Fin*

- En el paso 1, se inicia en algoritmo y se registra en la matriz.
- En el paso 2 de inicializa la variable NUM con 0, este valor se registra en la columna de la variable NUM.
- La repuesta es si al paso 3, que pregunta si existen bolas de billar. Y se continua por la seria 3.1
- En el paso 3.1.1, se asume que se saca la bola 5, en el paso 3.1.2 la respuesta es que el número de la bola no es par, se continua por la serie 3.1.2.2
- En el paso 3.1.2.2.1 regresamos al paso 3.
- En el paso 3 la respuesta es que, si existen bolas en la caja, sacamos la bola con el número 4 en el paso 3.1.1, y la repuesta es si en el paso 3.1.2, continuamos con la serie 3.1.2.1
- En el paso 3.1.2.1.1 actualizamos la variable NUM con el número 4 que es par, esto se registra en el paso 3 de la matriz en la columna de la variable NUM.
- En el paso 3.1.2.1.2 se imprime el contenido de la variable NUM, esto se registra en el paso 4 de la matriz. A continuación, se regresa al paso 3 del algoritmo.
- De forma análoga, se ejecuta los mismos pasos con la bola que contiene el número 6
- Se continua con el algoritmo, pero como los números siguientes contienen números impares, se continua hasta agotar las bolas de la caja.
- Finalmente, en el paso 7 de la matriz se registra el fin.

Ejemplo 12 : Escribir un algoritmo descriptivo que solicite números enteros de forma recurrente o cíclica, el algoritmo debe sumar todos los números ingresados, también debe informar del resultado final, el algoritmo finaliza cuando se ingrese el cero.

1. Inicio
2. Inicializamos SUM=0, NUM
3. Imprimimos el mensaje: “Ingrese un número entero”, -> NUM
4. ¿Es NUM diferente de cero?
   - 4.1. Si la respuesta es SI
     - 4.1.1. Imprimimos el contenido de: NUM
     - 4.1.2. Calculamos la siguiente operación: SUM=SUM+NUM;
     - 4.1.3. Saltamos al paso 3
   - 4.2. Si la respuesta es NO
     - 4.2.1. Imprimimos el contenido de: SUM
     - 4.2.2. Saltamos al paso 5
5. Fin

Tabla 5.- Matriz de seguimiento ejercicio 12

Nro.	Variable: NUM	Variable: SUM	Salidas o comentarios
1			*Inicio*
2		0	En el paso 2 se inicializa SUM, no se define valor en NUM.
3			Mensaje: “Ingrese un número entero”
4	2		Asumimos que se ingresa el 2 y se almacena en NUM
5			Imprime el contenido de NUM: 2
6		2	SUM=0+2;
7			Mensaje: “Ingrese un número entero”
8	5		Asumimos que se ingresa el 5 y se almacena en NUM
9			Imprime el contenido de NUM: 5
10		*7*	*SUM=2+5;*
11			Mensaje: “Ingrese un número entero”
12			Asumimos que se ingresa el 0 y se almacena en NUM
13			*Se imprime el contenido de SUM: 7*
14			*Fin*

CAPITULO II: Diagramas de flujo

En el capítulo anterior describimos como mediante una serie de pasos descriptivos, podemos plantear como resolver un problema, los procedimientos que se definan, para resolver dicho problema depende de muchos factores, pero podemos estandarizar al menos tres etapas que permiten analizar, diseñar y desarrollar un algoritmo, diagrama de flujo incluso un programa.

Al finalizar la unidad el estudiante podrá describir y utilizar las técnicas esenciales utilizadas para construir un diagrama de flujo.

1. Fases para el desarrollo de un diagrama de flujo

Para Joyanes Aguilar & Zahonero Martinez (2005), para resolver un problema se determinar una secuencia de fases, a continuación, se describe hasta la tercera fase que es de nuestro interés para el desarrollo de un diagrama de flujo.

1.1. Análisis del problema

En esta etapa respondemos a la pregunta ***¿Qué queremos hacer?***, pero sin pensar en cómo hacerlo, la idea de esta etapa es lograr la comprensión completa y extensa de lo queremos resolver, sin pensar en cómo lo vamos a hacer.

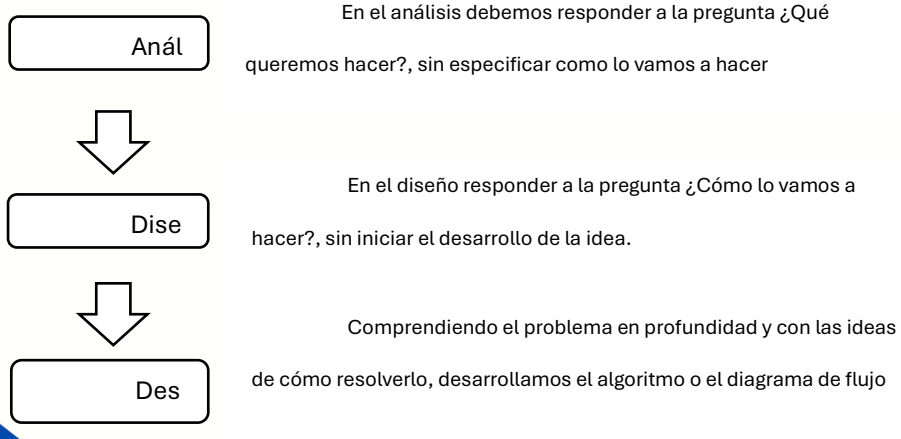
1.2. Diseño de la solución

Partiendo de la idea que tenemos claro lo queremos hacer o lo que queremos resolver, pasamos a la fase de diseño en la que respondemos a la pregunta ***¿Cómo lo vamos a hacer?***, pero sin empezar a resolverlo, en esta fase determinamos como vamos a resolver el problema, que operaciones utilizaremos, en que almacenaremos los datos, es decir diseñamos la solución.

1.3. Desarrollo del diagrama de flujo

Comprendiendo claramente lo que queremos hacer o lo que queremos resolver, con los límites establecidos (alcance), y con las ideas claras de solución procedemos a escribir el algoritmo o a dibujar el diagrama de flujo.

Figura 8.- Proceso para elaborar un diagrama de flujo



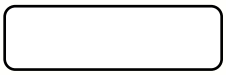
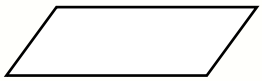
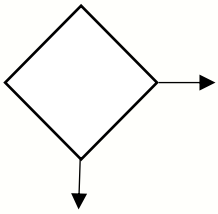
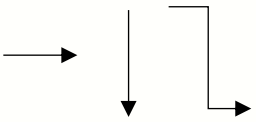
1. Diagrama de flujo

Un diagrama de flujo a diferencia de un algoritmo descriptivo permite normalizar los procedimientos, las entradas y las salidas, con símbolos específicos para cada operación o procedimiento, de tal forma que se elimina la ambigüedad que podría existir en los algoritmos descriptivos. Para Cairó Battistutti (2008), se pude definir un diagrama de flujo como una **representación gráfica de un algoritmo, con símbolos normalizados y estandarizados, que definen un flujo de ejecución inequívoco, cuyo objetivo es resolver un problema planteado, esta representación permite mejorar la comprensión y facilitar su seguimiento**.

1.1 Símbolos de un diagrama de flujo

Para Cairó Battistutti (2008) los símbolos a utilizar son los siguientes:

Tabla 6.- Simbología para un diagrama de flujo

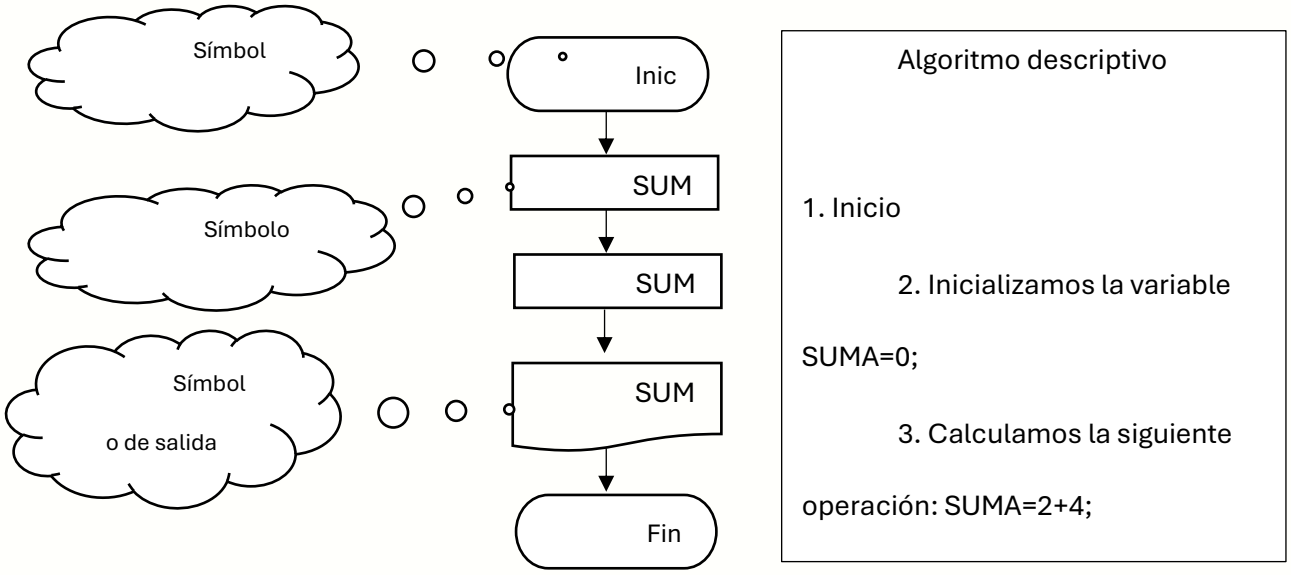
Símbolo	Descripción
	Símbolo utilizado para determinar el inicio y el fin del diagrama de flujo.
	Símbolo utilizado para la entrada de datos.
	Indica proceso, aquí se detallan todas las operaciones que incluyan por ejemplo operaciones matemáticas.
	Decisión, permite evaluar un cuestionamiento, con dos posibles resultados, SI o NO. Utilizamos normalmente los operadores lógicos y los conectores lógicos.
	Salida, indica que se presenta la información al usuario.
	Se utilizan para indicar la dirección del flujo de ejecución
	Conector, indica donde continua un flujo dentro de la misma página.
	Conector, indica donde continua un flujo en una página diferente.

Nota.- Esta simbología se detalla en (Cairó Battistutti, Metodología de la Programación, 2008)

Estos símbolos son los que utilizaremos en esta guía didáctica, nos permitirán resolver todos los ejercicios didácticos que se plantean. Los diagramas de flujo permiten una normalización de las instrucciones y son un paso más hacia la programación. Los algoritmos descriptivos, según el planteamiento de esta guía nos permiten la comprensión de la lógica utilizada en la programación, los diagramas de flujo consolidad esta lógica y nos permite implementar estas instrucciones en un programa.

Ejemplo 13 : Desarrollar un diagrama de flujo que permita sumar los números 2 y 4.

Figura 9.- Diagrama de flujo ejercicio 13



Operadores lógicos

Utilizaremos los siguientes.

Tabla 7.- Operadores lógicos

Nro.	Símbolo	Operador	Descripción
1	==	Igualdad	Permite comparar la igualdad
2	!=	Diferente	Permite comparar si dos elementos son diferentes
3	<>	Diferente	Permite compara si dos elementos son diferentes
	>	Mayor	Compara si un elemento es mayor
4	>=	Mayor o igual	Compara si un elemento es mayor o igual
5	<	Menor	Compara si un elemento es menor
6	<=	Menor que	Compara si un elemento es menor o igual

Para poder procesar, también es necesario la utilización de estos operadores lógicos, esto nos sirven, sobre todo para poder realizar las comparaciones, las preguntas en los algoritmos y son fundamentales en el desarrollo de los mismos, podemos preguntar por ejemplo si una variable es igual

a un número, con dos posibles resultados SI o No, podemos definir procedimientos para cada resultado. En este sentido se puede utilizar la lista descrita en la tabla anterior de acuerdo a las necesidades de los problemas planteados.

Conectores lógicos

Los operadores lógicos que se describieron en el apartado anterior nos permiten realizar cuestionamientos simples de una instrucción, los conectores lógicos nos permiten unir varios de este cuestionamiento, brindando la posibilidad de realizar decisiones más complejas, describimos los conectores lógicos a continuación.

Tabla 8.- Conectores lógicos

Nro.	Símbolo	Conector	Descripción
1	&&	Y (AND)	Es verdad si las dos proposiciones son verdaderas el resultado es verdadero
2	||	O (OR)	Si las proposiciones son falsas el resultado es falso
3	~	Negación	Cambia el valor de verdad de una proposición

1. Conector && (AND)

Este conector nos permite unir varias proposiciones o cuestionamientos y obedece a la siguiente tabla de verdad. Sea P la proposición uno, y Q la proposición dos, los valores de verdad para el conector && es el siguiente:

Tabla 9.- Tabla de verdad del conector AND (&&)

P	Q	Conector &&
V	V	V
V	F	F
F	V	F
F	F	F

Podemos resumir esta tabla así: El conector && es verdad cuando las dos proposiciones son verdaderas, en todos los demás casos son falsas. Ejemplo: La expresión ((x>1) && (x<10)), esta

expresión es verdadera siempre y cuando el valor contenido en la variable X es mayor que uno y además el valor también es menor que 10, un ejemplo de valor para x que cumple con esto X=5, sin embargo, para un valor de X=11 el resultado de la expresión seria falso porque 11 no es menor que 10.

2. Conector || (OR)

Este conector nos permite unir varias proposiciones o cuestionamientos y obedece a la siguiente tabla de verdad. Sea P la proposición uno, y Q la proposición dos, los valores de verdad para el conector || es el siguiente:

Tabla 10.- Tabla de verdad del conector lógico OR (||)

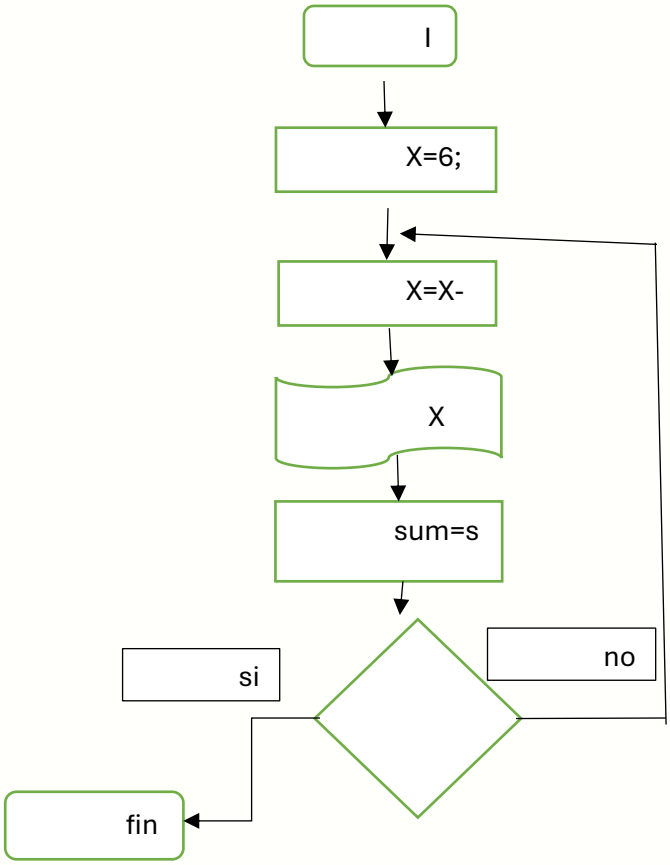
P	Q	Conector ||
V	V	V
V	F	V
F	V	V
F	F	F

Podemos resumir esta tabla así: El conector || es falso cuando las dos proposiciones son falsas, en todos los demás casos es verdadera. Ejemplo: La expresión ((x>1) || (y<10)), esta expresión será falsa siempre y cuando el valor contenido en la variable X es menor que uno (Esto hace falsa esta expresión) o igual que uno y además el valor contenido en la variable Y es también mayor que 10 o igual a 10, en todos los demás casos esta expresión es verdadera.

Ejercicios de diagramas de flujo

Ejercicio 1.- Imprimir de forma descendente

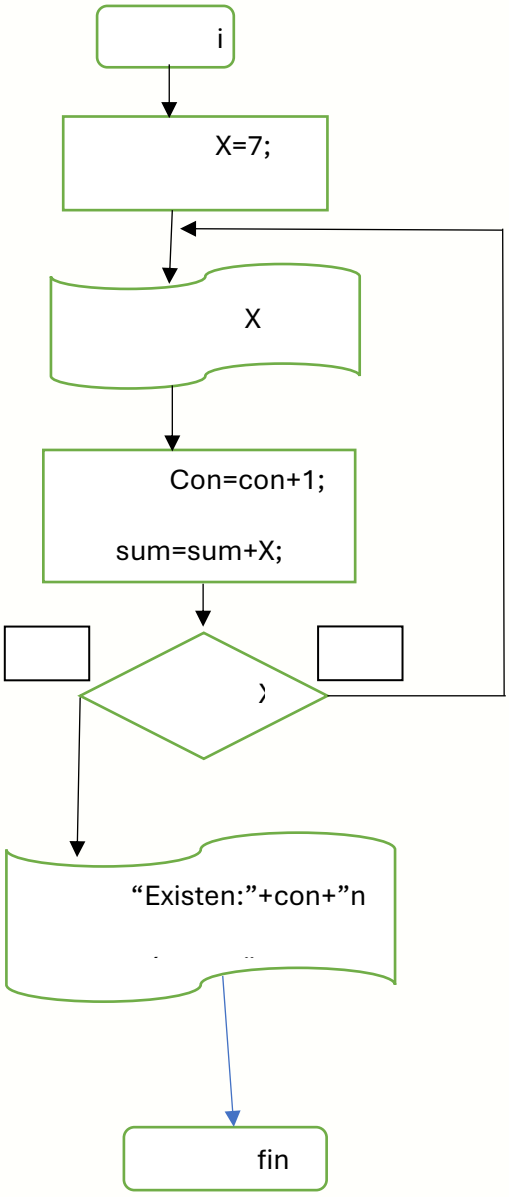
Desarrollar un diagrama de flujo que permita imprimir los números de forma descendente, iniciando en el 5 y finalizando en 1, el diagrama de calcular la suma de todos los números impresos.



Nro	X	sum	Salidas
1			Inicio
2	6	0	
3	5		
4			Imprime:5
5		5	
6	4		
7			Imprime:4
8		9	
9	3		
10		12	Imprime:3
11	2		
12			Imprime:2
13		14	
14	1		
15			Imprime:1
16		15	
17			Fin

Ejercicio 2.- Imprimir números múltiplos de siete

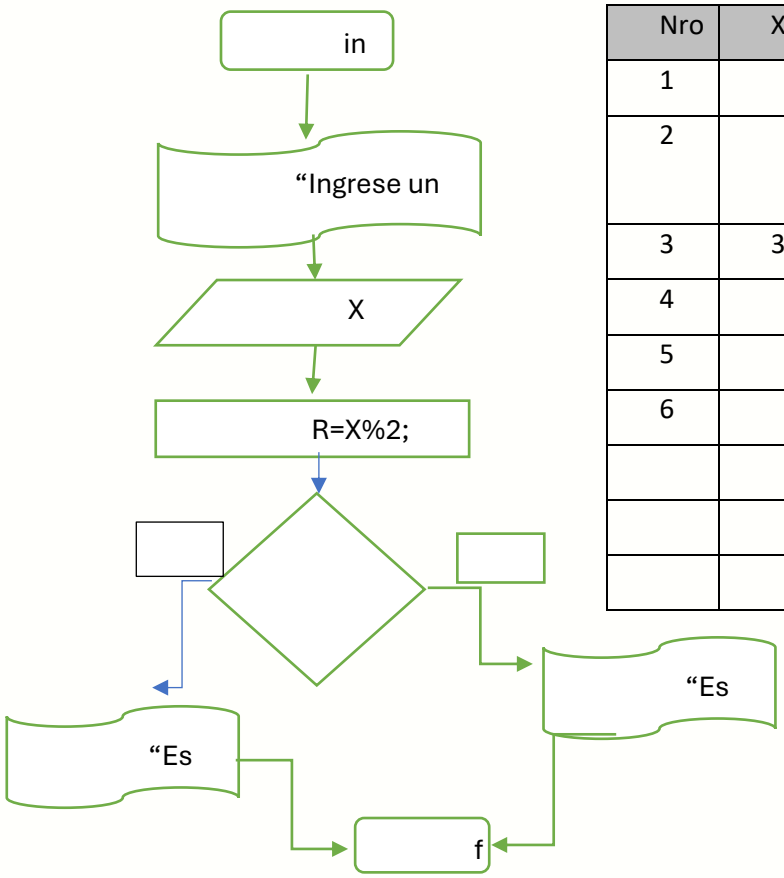
Desarrollar un diagrama de flujo que permita imprimir los números múltiplos de siete, iniciando en el 7 y finalizando en 21, el diagrama de flujo debe contar cuantos números existen y la suma de todos ellos.



Nro	X	CON	SUM	SALIDAS
1				Inicio
2	7	0	0	
3				7
4	14	1	7	
5				14
6	21	2	21	
7				21
8	28	3	42	
9				Existen: 3 números La suma es: 42
10				Fín

Ejercicio 3.- Identifica pares e impares

Desarrollar un diagrama de flujo que solicite que se ingrese un número, el diagrama de flujo deberá informar si el número es par o impar.



Nro	X	R	SALIDAS
1			Inicio
2			“Ingrese un número”
3	3		
4		1	
5			Es impar
6			fin

Ejercicios propuestos

Después de revisar y leer los contenidos del capítulo, es importante aplicar y poner en práctica lo aprendido resolviendo los ejercicios que se presentan a continuación. De tener dificultades, solicitar la ayuda de la tutoría.

Ejercicio 4.- Solicita clave

Desarrollar un diagrama de flujo que solicite que se ingrese una clave, si la clave es incorrecta el diagrama informa y solicita que se ingrese una nueva, además cuenta el intento. Si la clave es correcta el diagrama informa y finaliza.

Ejercicio 5.- Ciclo indeterminado

Dibujar un diagrama de flujo que solicite que se ingrese un número del 1 al 5 de forma cíclica, si el número se encuentra entre 1 y el 5 se informará que el número es válido, si el número se encuentra fuera del rango se informará que el número no es válido, el ciclo finaliza cuando se ingrese el cero.

Ejercicio 6.- Ciclo indeterminado

Dibujar un diagrama de flujo que solicite que se ingrese un número entero de forma cíclica, el diagrama debe identificar si el número ingresado es par o impar, el diagrama finaliza cuando se ingrese el cero, además informa cuantos números pares e impares se ingresaron.

Ejercicio 7.- Ciclo indeterminado

Dibujar un diagrama de flujo que solicite que se ingrese un número entero mayor que cero, el diagrama debe imprimir los números de forma secuencial desde el 0 hasta el número ingresado. Además, debe calcular la suma de todos ellos.

Ejercicio 8.- Ciclo indeterminado

Dibujar un diagrama de flujo que solicite de forma cíclica que se ingrese una nota, la nota debe encontrarse entre 1 y 20 caso contrario el programa informa que la nota no es válida, el programa termina cuando se ingrese el 0, y calcula el promedio de todas las notas válidas.

Ejercicio 9.- Ciclo indeterminado

Dibujar un diagrama de flujo que solicite que se ingresen dos números enteros. El diagrama debe Imprimir los números naturales que hay entre los dos, debe iniciar por el más pequeño, debe contar cuántos de ellos son pares y calcular la suma de los impares.

Ejercicio10.- Ciclo indeterminado

Dibujar un diagrama de flujo que imprima el número mayor de una serie de números que vamos introduciendo por teclado. El diagrama finaliza cuando se ingrese el cero.

Ejercicio11.- Ciclo indeterminado

Dibujar un diagrama de flujo que imprima el número menor de una serie de números que vamos introduciendo por teclado. El diagrama finaliza cuando se ingrese el cero.

CAPITULO III: Desarrollo de programas

Al finalizar la unidad el estudiante podrá utilizar los comandos adecuados para escribir un programa elemental de acuerdo al problema planteado.

1. El lenguaje de programación JAVA

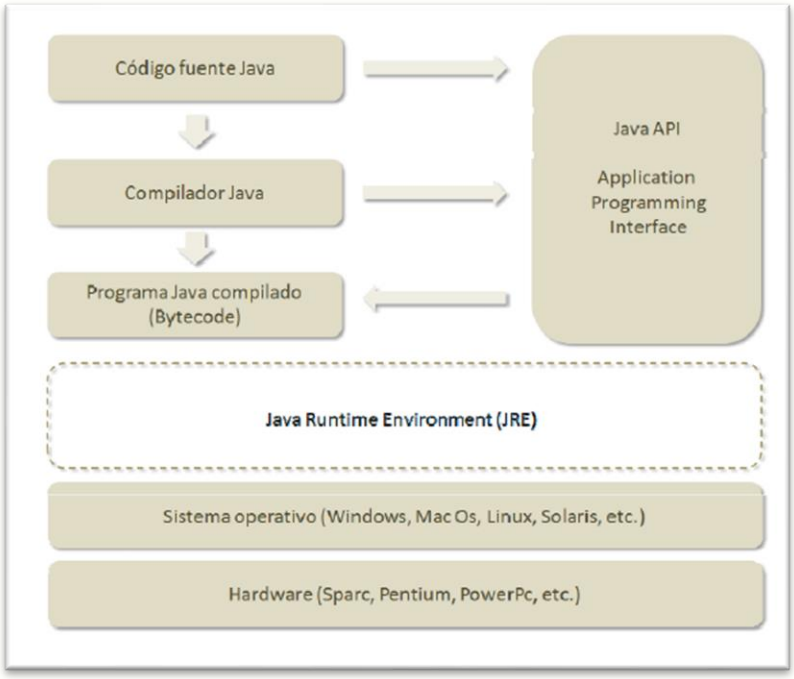
Java es un lenguaje de programación desarrollado por Sun Microsystems. Java fue presentado en la segunda mitad del año 1995 y desde entonces se ha convertido en un lenguaje de programación muy popular, Java es un lenguaje muy valorado porque los programas Java se pueden ejecutar en diversas plataformas con sistemas operativos como Windows, Mac OS, Linux o Solaris. (Java, 2024)

Para Martínez Ladrón de Guevara (2024), con el lenguaje Java se buscaba diseñar un lenguaje que permitiera programar una aplicación una sola vez que luego pudiera ejecutarse en distintas máquinas y sistemas operativos. Para conseguir la portabilidad de los programas Java se utiliza un entorno de ejecución para los programas compilados. Este entorno se denomina Java Runtime Environment (JRE). Es gratuito y está disponible para los principales sistemas operativos. Esto asegura que el mismo programa Java pueda ejecutarse en Windows, Mac OS, Linux o Solaris.

Figura 10 .- Entorno de ejecución Java (Obtenido de: https://acortar.link/G6uRCG)

Los programas Java se compilan a un lenguaje intermedio, denominado Bytecode, este código es interpretado por la máquina virtual de Java del entorno de ejecución (JRE), de esta forma se consigue la portabilidad en distintas plataformas. El JRE es una pieza intermedia entre el código Bytecode y los distintos sistemas operativos existentes en el mercado. En realidad, Java no solo es un lenguaje de programación. Java es un lenguaje, una plataforma de desarrollo, un entorno de ejecución y un conjunto de librerías para desarrollo.

Figura 11.- La plataforma de Java, fuente: https://www.ingenieriasystems.com/2015/12/la-plataforma-de-java.html#google_vignette



1. Entornos de desarrollo para JAVA

Existen distintos entornos de desarrollo de aplicaciones Java. Este tipo de productos ofrecen al programador un entorno de trabajo integrado, estos productos se denominan IDE (Integrated Development Environment), por sus siglas en ingles. Existen entornos de distribución libre como: Eclipse, además también existen productos comerciales están JBuilder.

Figura 12.- El proceso de edición y compilación de JAVA, obtenido de: https://www.ingenieriasystems.com/2015/12/el-proceso-de-edicion-y-compilacion-en.html

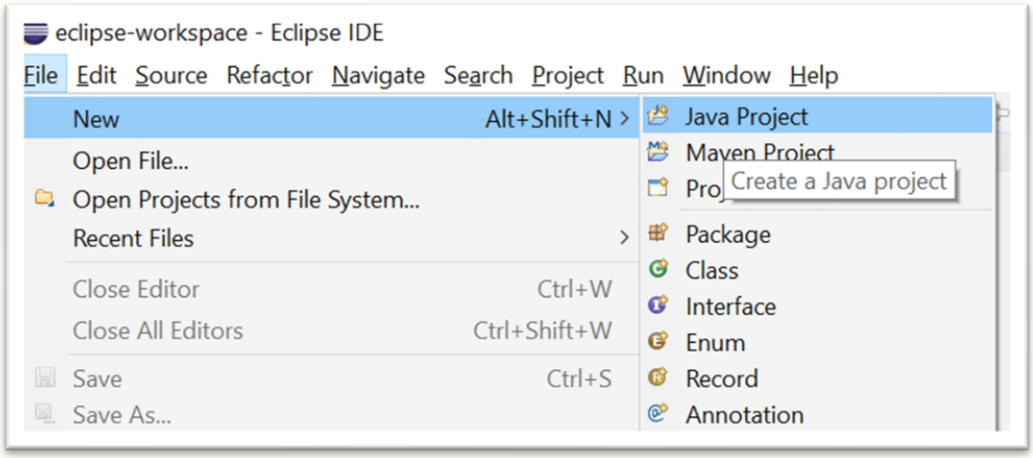
2. El entorno de desarrollo integrado Eclipse

Creación de un proyecto en el IDE Eclipse

Para crear un nuevo proyecto en eclipse debemos iniciar la aplicación, a continuación, realizar clic en el menú “File”, opción “New” y a continuación “Java Project”.

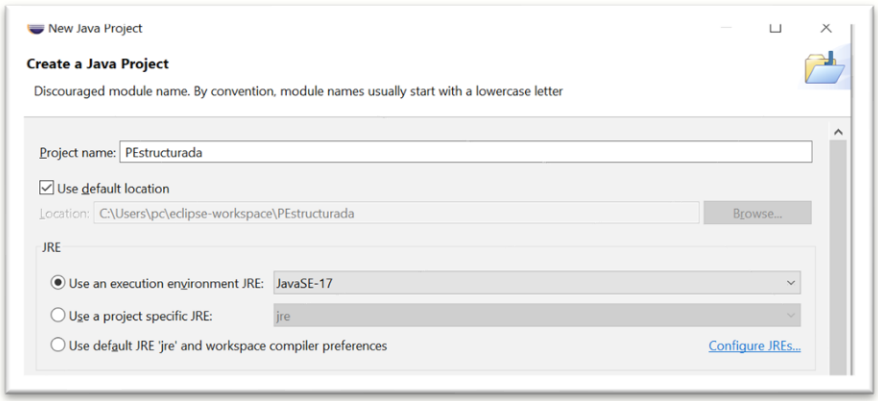
Figura 13.- Creación de un nuevo proyecto en el IDE Eclipse

Nota. Figura creada por el autor en el IDE Eclipse



A continuación, la aplicación presente la interfaz para crear el proyecto, en el nombre del proyecto (Project name), incluiremos el nombre de nuestro proyecto, para esta guía utilizaremos el nombre “PEstructurada”. Podemos también observar en la imagen que utilizaremos el entorno de ejecución: “JavaSe-17”, que es la versión de JRE que utilizaremos para nuestros programas.

Figura 14.- Creación del proyecto Java



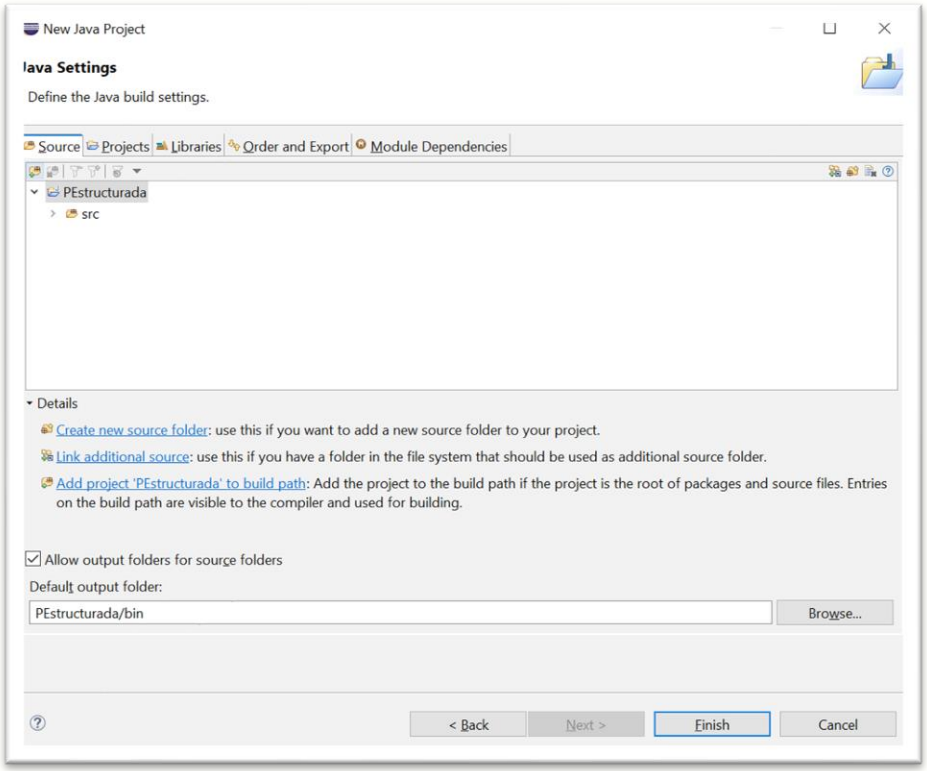
Nota. Figura creada por el autor en el IDE Eclipse

A continuación, realizamos clic en el botón “Next”, se nos presenta la interfaz donde podemos cambiar la configuración del proyecto, como por ejemplo que nuestros recursos se almacenen en la

carpeta “src”, comprobamos que se encuentre marcada la opción “Allow output folders for source folders” y realizamos clic en el botón “Finish” para finalizar la creación del proyecto.

Figura 15.- Configuración del proyecto

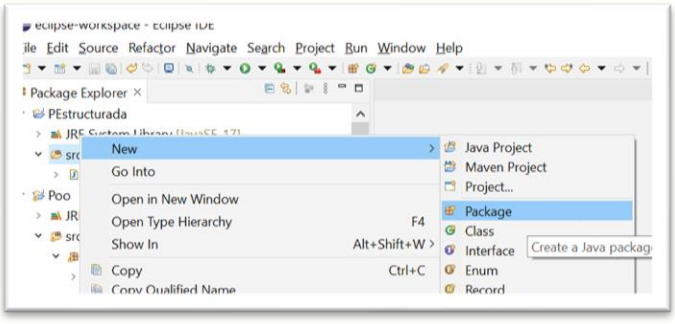
Nota. Figura creada por el autor en el IDE Eclipse



3. Creación del paquete de trabajo

JAVA gestiona sus recursos por paquetes, para nuestro proyecto es necesario crear un paquete de trabajo, para ello realizaremos clic derecho en la carpeta “src”, a continuación, seleccionamos la opción “Package”.

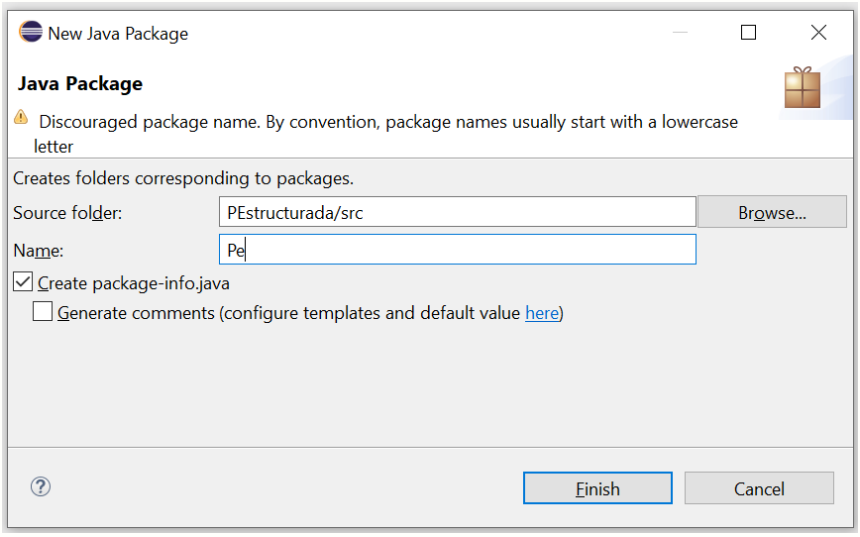
Figura 16.- Creación del paquete de trabajo



Nota. Figura creada por el autor en el IDE Eclipse

Aquí se nos presenta la interfaz, le indicamos que nuestro paquete se llamará “Pe”, nos aseguramos que se encuentre seleccionada la opción “Create package-info.java” y realizamos clic en el botón “Finish”.

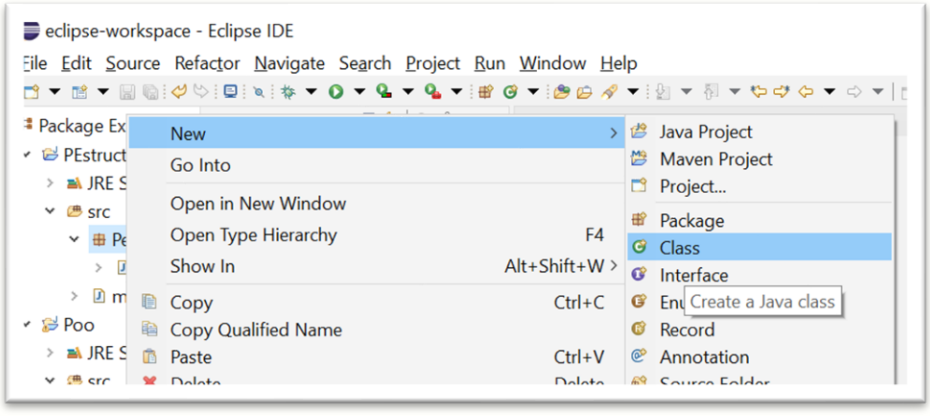
Figura 17.- Creación del paquete de trabajo (Figura creada por el autor en el IDE Eclipse)



4. Creación de una clase en JAVA.

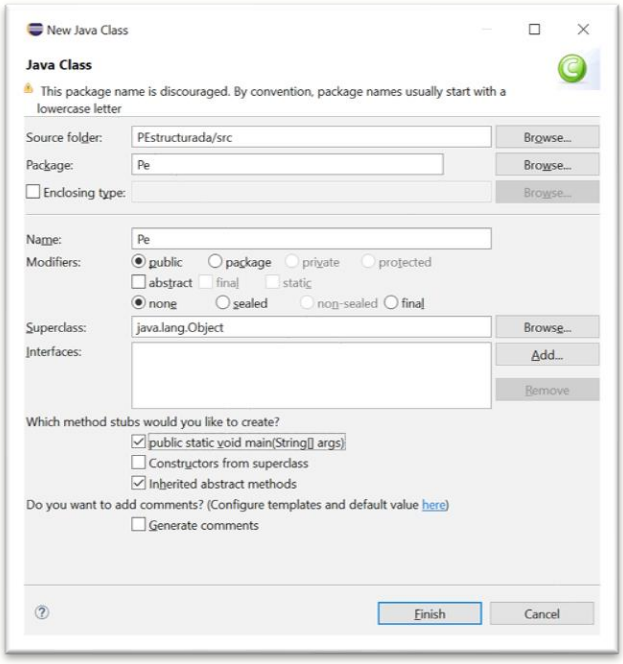
Finalmente, para poder desarrollar nuestros programas debemos crear la clase principal en la que desarrollaremos todos nuestros programas, para ello realizamos clic derecho sobre el paquete “Pe”, alli se nos presenta un menú del cual elegiremos la opción “New” y luego “Class”.

Figura 18.- Creación de la clase de trabajo (Figura creada por el autor en el IDE Eclipse)



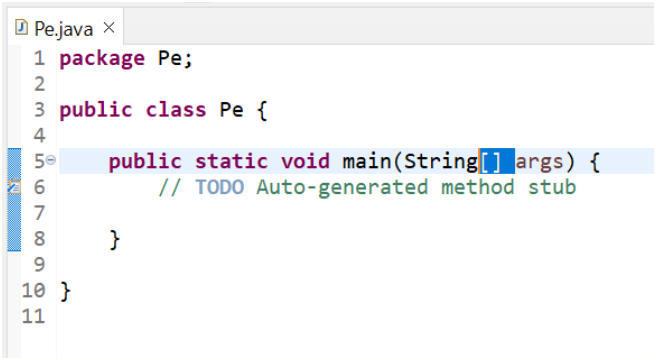
A continuación se presenta la interfaz para la creación de la clase, le llamaremos “Pe”, debemos asegurarnos que la opción “public static void main(String[] args)”, con esto crearemos la clase principal de nuestro proyecto, y la ejecución iniciará por el método denominado “main()”, aquí escribiremos todos nuestros programas.

Figura 19.- Creación de la clase principal (Figura creada por el autor en el IDE Eclipse)



A continuación, se nos presenta la clase recién creada, todos los programas de esta guía deben escribirse a partir de la línea 6, respetando el código existente, no debemos borrar el código generado por el IDE Eclipse.

Figura 20.- Estructura de una clase JAVA



Nota. Imagen generada en el IDE Eclipse.

5. Prácticas de laboratorio

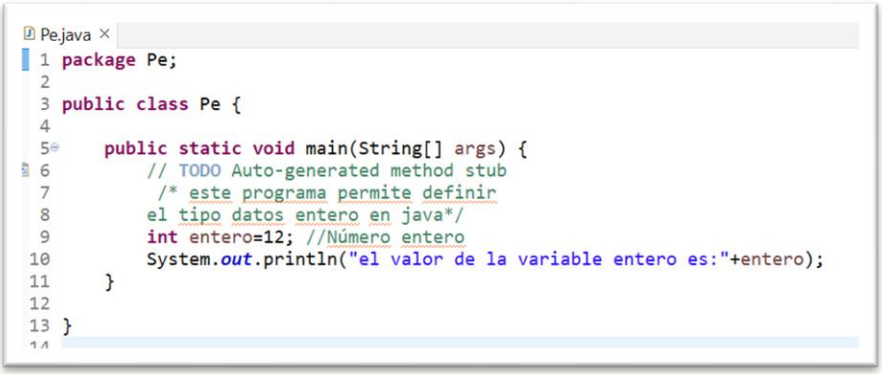
En esta sección la metodología propuesta es desarrollar programas da acuerdo a la guía de cada práctica, se debe revisar el objetivo de aprendizaje y el procedimiento propuesto. Al desarrollarse estas prácticas en el laboratorio de computación es recomendable recordar las medidas de seguridad.

Medidas de seguridad para el uso del laboratorio de computación

1. No se debe desconectar los conectores eléctricos de ningún equipo, de ser necesario solicitar el apoyo del docente tutor o el encargado del laboratorio.
2. No se quitar las cubiertas de los computadores, ni manipular los armarios de conectividad de red.
3. Si el estudiante trabajo en su equipo personal realizar la conexión eléctrica en los reguladores habilitados para este fin, informar al docente de clase antes de realizar el procedimiento.
4. En caso de evacuación por emergencia, desarrollar la evacuación a paso ligero por la derecha de acuerdo a los simulacros desarrollados por la institución.
5. Dirigirse al punto e seguridad más cercano al laboratorio.
6. En caso de movimiento telúrico inclinarse al costado de las mesas de computadores con la finalidad de aplicar el concepto de triangulo de vida.

Práctica 1: Los comentarios y la salida por consola

Figura 21.- Práctica 1: Primer programa java



Nota. En este programa se define una variable de tipo entero y se lo envía a la salida por consola

Proceso previo. – Para poder desarrollar esta práctica debemos tener instalado y creado el proyecto en el IDE Eclipse.

Fundamentos. – Interpretar los conceptos de variable, comentarios de bloque, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse.

Objetivo. – Describir y utilizar las variables de tipo entero en java, además de utilizar el método de salida por consola en el IDE Eclipse, procesos de depuración de código y ejecución del proyecto.

Horas de práctica. – 1 Hora

Procedimiento. – En el método principal main de la clase principal de nuestro proyecto, debemos incluir un comentario de bloque en el que se describe el objetivo del programa, además, debemos

definir una variable de tipo entero que incluya el nombre del estudiante, a continuación, debemos incluir el comando para desplegar (System.***out***.println()), el contenido de la variable por consola.

Evaluación del aprendizaje. – El estudiante presenta la evidencia de declaración de variables tipo entero que incluyan su nombre en el nombre de la variable y la evidencia de la consola de salida.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 2: Tipos de datos java

Figura 22.- Programa tipos de datos en java

*Pe.java

```
1  package Pe;
2
3  public class Pe {
4
5      public static void main(String[] args) {
6          /* Este programa permite definir
7          varios tipos de datos de java*/
8
9          byte bit=4;          //Número Entero con signo
10         short corto=0;       //Número entero corto
11         int entero=12;       //Número Entero
12         long largo=8;        //Número entero largo
13         float numeroFloat=12;    //Numérico con decimales
14         double numeroDouble=13.2;   //Numérico con decimales de doble precisión doble
15         boolean estado=true;         //Valores lógicos
16         char caracter='a';           //Caracteres
17         String cadena="Cadena de caracteres"; //Cadena de caracteres
18
19         System.out.println("el valor del byte:"+bit);
20         System.out.println("el valor del entero corto:"+corto);
21         System.out.println("el valor del entero:"+entero);
22         System.out.println("el valor del entero largo:"+largo);
23         System.out.println("el valor del numero decimal:"+numeroFloat);
24         System.out.println("el valor del numero decimal doble:"+numeroDouble);
25         System.out.println("el valor del caracter:"+caracter);
26         System.out.println("el valor del cadena:"+cadena);
27     }
28 }
```

Nota. En este programa se define los tipos de datos JAVA y se los envía a la salida por consola

Proceso previo. – Para poder desarrollar esta práctica debemos tener instalado y creado el proyecto en el IDE Eclipse, además de conocer el proceso de ejecución del proyecto.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación.

Objetivo. – Describir y utilizar los distintos tipos de datos; primitivos y clases de java, además de utilizar el método de salida por consola en el IDE Eclipse.

Horas de práctica. – 1 Hora

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir nombres de variables que incluyan el nombre del estudiante de los tipos primitivos de java: byte, short , int, long, float, double, boolean , char y de la clase String. A continuación, debemos imprimir por consola el contenido de todas las variables incluida una descripción.

Evaluación del aprendizaje. – El estudiante presenta la evidencia de declaración de las variables de los tipos de datos de java que incluyan su nombre en el nombre de la variable y la evidencia de la consola de salida.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 3: Captura desde teclado

Figura 23.- Programa de captura de datos desde el teclado

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3
4  public class Pe {
5
6      public static void main(String[] args) {
7
8          float numero;
9          System.out.println("ingerse un número");
10         Scanner scanner =new Scanner(System.in);//Se define un objeto scaner
11         numero=scanner.nextFloat();
12
13         System.out.println("El número: "+numero);
14     }
15 }
16
```

Nota. Este programa captura datos de tipo decimal y lo imprime por consola

Proceso previo. – Para poder desarrollar esta práctica debemos interpretar el concepto de variable y el tipo de datos, el proceso de ejecución del proyecto en el IDE Eclipse e interpretar el significado de entrada de datos al programa.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación y el significado de datos de entrada en una aplicación.

Objetivo. – Describir como un programa en JAVA captura datos desde el teclado y el código necesario para implementarlos.

Horas de práctica. – 1 Hora

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir el nombre de una variable que incluyan el nombre del estudiante, el tipo debe ser *float*, además se debe incluir la clase necesaria para la captura de datos con la instrucción: import java.util.Scanner; finalmente en la aplicación debemos utilizar el método nextFloat() que permite captura un número con decimales desde el teclado. Se debe también incluir la clase Scaner con la instrucción: **import** java.util.Scanner; en la línea número dos del programa.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan los tipos de datos de java, además la evidencia de la captura de la variable y su salida por consola.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 4: Operaciones aritméticas en java

Figura 24.- Programa con las operaciones aritméticas básicas de Java

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5
6          int dato1=3,dato2=4,resultado=0; //variables
7          resultado=dato1+dato2;
8          System.out.println("el resultado de la suma es:"+resultado);
9          resultado=dato1-dato2;
10         System.out.println("el resultado de la resta es:"+resultado);
11         resultado=dato1/dato2;
12         System.out.println("el resultado de la división es:"+resultado);
13         resultado=dato1*dato2;
14         System.out.println("el resultado de la multiplicación es:"+resultado);
15         resultado=dato1%dato2; //Operación resto
16         System.out.println("el resultado del resto es:"+resultado);
17     }
18 }
19
```

Nota. Este programa calcula las operaciones aritméticas en JAVA

Proceso previo. – Para poder desarrollar esta práctica debemos interpretar el concepto de variable y el tipo de datos, el proceso de ejecución del proyecto en el IDE Eclipse e interpretar la salida por consola.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, y describir los símbolos de las operaciones aritméticas en JAVA.

Objetivo. – Describir como realizar las operaciones fundamentales en JAVA, y como desplegar los resultados por la consola.

Horas de práctica. – 1 Hora

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir el nombre de al menos tres variables de tipo entero que incluyan el nombre del estudiante, el programa debe calcular las operaciones fundamentales (suma, resta, multiplicación, división y el resto) con dos variables y el resultado almacenarlo en una tercera variable, todos los resultados deben enviarse a la consola de salida.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la la salida del cálculo de las operaciones por consola.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 5: La estructura if – else

Figura 25.- Programa que utiliza la estructura IF y los conectores lógicos

*Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          float num1,num2;
6          System.out.println("ingrese el 1º número");
7          Scanner scanner =new Scanner(System.in);
8          num1=scanner.nextFloat();
9          System.out.println("ingrese el 2º número");
10         num2=scanner.nextFloat();
11
12         if(num1==num2){
13            System.out.println("Los números son iguales");
14         }
15
16         if(num1!=num2){
17            System.out.println("Los números NO son iguales");
18         }
19
20         if(num1<=num2){
21                 System.out.println("El número :"+num1+" <= que el :"+num2);
22         }
23         else{
24                 System.out.println("El número :"+num2+" < que el :"+num1);
25         }
26     }
27 }
```

Proceso previo. – Para poder desarrollar esta práctica debemos interpretar el concepto de variable y el tipo de datos, el proceso de ejecución del proyecto en el IDE Eclipse e interpretar la salida por consola, capturar datos desde el teclado.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, además el significado de los conectores lógicos mayor, menor, mayor igual, menor igual, igual, diferente.

Objetivo. – Utilizar e interpretar la estructura de programación IF y los conectores lógicos en un programa JAVA.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir el nombre de al menos dos variables de tipo entero, debemos capturar desde el teclado el contenido de las variables de tipo entero, a continuación, utilizando la estructura de programación IF y los conectores lógicos se debe informar si los números son: iguales, diferentes y cual es mayor y menor.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida de si son iguales, diferentes, el mayor y el menor.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 6: La operación del resto en JAVA

Figura 26.- Programa que utiliza la operación del resto

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5
6          float resto,numero;
7          System.out.println("ingerse un número");
8          Scanner scanner =new Scanner(System.in);
9          numero=scanner.nextFloat();
10
11         resto=numero%2;
12         if(resto==0){
13             System.out.println("El numero es Par");
14         }
15         if(resto>0){
16             System.out.println("El numero es Impar");
17         }
18     }
19 }
20
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder capturar números desde el teclado, importar la clase Scanner, y las operaciones aritméticas.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, además el significado de los conectores lógicos mayor, menor, mayor igual, menor igual, igual, diferente, y las operaciones aritméticas en JAVA.

Objetivo. – Utilizar e interpretar la operación del resto en un programa JAVA y su uso para calcular los números pares e impares.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir el nombre de al menos dos variables de tipo entero, en una de ellas capturaremos un número entero desde el teclado, a continuación, calcularemos la operación del resto de la división entera por dos, si el resto es cero el número ingresado es par caso contrario es impar.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida si el número ingresado es par o impar.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 7: Los operadores lógicos en JAVA

Figura 27.- Programa que utiliza los operadores lógicos de Java

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5
6          /*       Operadores lógicos
7           * Operador  Significado
8           *     &&       Y
9           *     ||       O
10          *     !        NO
11          */
12         String  caracter;
13         char chValor;
14
15         System.out.println("ingerse un carácter");
16         Scanner scanner =new Scanner(System.in);
17         caracter=scanner.next();
18         chValor=caracter.charAt(0);
19
20         System.out.println("-"+caracter+"-");
21         if((chValor=='a') || (chValor=='e') || (chValor=='i')|| (chValor=='o') ||   (chValor=='u'))
22         { System.out.println("El caracter es una vocal"); }
23         else
24         {System.out.println("El caracter NO es una vocal"); }
25     }
26 }
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder capturar desde el teclado, importar la clase Scanner, y utilizar la estructura IF.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, además el significado de los operadores lógicos (y &&, o || y la negación!).

Objetivo. – Utilizar los operadores lógicos en un programa JAVA.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir dos variables de tipo char y String, en una de ellas capturaremos un carácter, a continuación, utilizaremos la estructura if incluyendo el operador lógico o, el programa finalmente informará si el carácter ingresado es una vocal o no.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida si el carácter ingresado es una vocal o no.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 8: El operador lógico && en JAVA

Figura 28.- Programa que calcula el mayor de tres números ingresados desde el teclado utilizando el operador lógico && (y)

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5
6          int num1,num2,num3;
7          System.out.println("ingerse el 1º número");
8          Scanner scanner =new Scanner(System.in);
9          num1=scanner.nextInt();
10         System.out.println("ingerse el 2º número");
11         num2=scanner.nextInt();
12         System.out.println("ingerse el 3º número");
13         num3=scanner.nextInt();
14         if((num1>num2) && (num1>num3))
15         {System.out.println("El "+num1+" es el mayor");}
16
17         if((num2>num1) && (num2>num3))
18         {System.out.println("El "+num2+" es el mayor");}
19
20         if((num3>num1) && (num3>num2))
21         {System.out.println("El "+num3+" es el mayor");}
22     }
23 }
24
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder capturar desde el teclado, importar la clase Scanner, y utilizar la estructura IF.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, además el significado del operador lógico &&.

Objetivo. – Utilizar el operador lógico && en un programa JAVA para calcular el mayor de tres números ingresados desde el teclado.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir tres variables de tipo entero, las debemos capturar desde el teclado, a continuación, utilizaremos la estructura if incluyendo el operador lógico Y (&&), cuál de los tres es el número mayor.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que informa del número mayor de tres números.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 9: If anidados en JAVA

Figura 29.- Programa que utiliza if anidados para determinar el mayor de tres números

*Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          /* If anidados:comprueba el mayor de 3 números*/
6          int num1,num2,num3;
7          System.out.println("ingerse el 1º número");
8          Scanner scanner =new Scanner(System.in);
9          num1=scanner.nextInt();
10         System.out.println("ingerse el 2º número");
11         num2=scanner.nextInt();
12         System.out.println("ingerse el 3º número");
13         num3=scanner.nextInt();
14         if((num1>num2))
15         {
16             if((num1>num3))
17             {System.out.println("El "+num1+" es el mayor");}
18             else
19             {System.out.println("El "+num3+" es el mayor");}
20         }
21         else
22         {
23             if(num2>num3)
24             {System.out.println("El "+num2+" es el mayor");         }
25             else
26             {
27                 if(num3>num1)
28                 {System.out.println("El "+num3+" es el mayor");}
29             }
30         }
31     }
32 }
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder capturar desde el teclado, importar la clase Scanner, y utilizar la estructura IF.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar el concepto de "if anidado".

Objetivo. – Utilizar el if anidados en un programa JAVA para calcular el mayor de tres números ingresados desde el teclado.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir tres variables de tipo entero, las debemos capturar desde el teclado, a continuación, utilizaremos la estructura if anidada calculamos cuál de los tres es el número mayor.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que informa del número mayor de tres números.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 10: Estructura switch en JAVA

Figura 30.- Programa que utiliza la estructura switch para establecer un menú de opciones.

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          int opcion;
6          System.out.println("ingerse un número");
7          Scanner scanner =new Scanner(System.in);
8          opcion=scanner.nextInt();
9
10         switch(opcion){
11                 case 1:
12                         System.out.println("Caso 1");
13                         break;
14                 case 2:
15                         System.out.println("Caso 2");
16                         break;
17                 case 3:
18                         System.out.println("Salir");
19                         break;
20                 default:
21                         System.out.println("Caso no Existente");
22         }
23     }
24 }
25
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder capturar desde el teclado, importar la clase Scanner.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar la estructura switch.

Objetivo. – Utilizar la estructura switch en un programa JAVA para crear un menú de tres opciones.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir una variable de tipo entero, la debemos capturar desde el teclado, a continuación, utilizaremos la estructura switch e informamos la opción elegida.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que informa si el carácter ingresado es un digito.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 11: Estructura switch en JAVA

Figura 31.- Programa que utiliza la estructura switch para determinar si un número ingresado es digito o no.

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          String c;
6          System.out.println("Pulse una tecla");
7          Scanner scanner =new Scanner(System.in);
8          c=scanner.next();
9
10         switch(c){
11                 case "0":
12                 case "1":
13                 case "2":
14                 case "3":
15                 case "4":
16                 case "5":
17                 case "6":
18                 case "7":
19                 case "8":
20                 case "9":
21                         System.out.println("Dígito");
22                         break;
23                 default:
24                         System.out.println("No es un dígito");
25         }
26     }
27 }
28
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder capturar desde el teclado, importar la clase Scanner.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar la estructura switch.

Objetivo. – Utilizar la estructura switch en un programa JAVA para determinar si un carácter ingresado es un digito o no.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir una variable de tipo char, la debemos capturar desde el teclado, a continuación, utilizaremos la estructura switch verificamos si el número ingresado es un digito.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que informa si el carácter ingresado es un digito.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 12: Estructura for en JAVA

Figura 32.- Programa que calcula la suma de los números del uno al cien, utilizando la estructura for.

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          int indice,suma=0;
6
7          for(indice=1;indice<=100;indice++){
8             suma=suma+indice;
9          }
10         indice--;
11         System.out.println("Suma de los "+indice+" Números:"+suma);
12         System.out.println("Imprime de forma descendente:");
13         for(indice=1;indice<=100;indice++){
14            System.out.println("Numero:"+indice);
15            indice++;
16         }
17     }
18 }
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder imprimir en la consola, definir variables de tipo entero.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar que son los ciclos determinados, interpretar el concepto de contador y acumulador

Objetivo. – Utilizar la estructura for en un programa JAVA para calcular la suma de los cien primeros números enteros.

Horas de práctica. – 1 Hora

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir dos variables de tipo entero, a continuación, utilizaremos la estructura for para calcular la suma de todos ellos utilizando el concepto de contador y acumulador.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que informa la suma de los primeros cien números enteros.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 13: Estructura while en JAVA

Figura 33.- Programa que solicita una clave, utilizando la estructura while.

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          String clave; //Almacena clave tecleada
6          String clave1="123"; //Clave correcta
7          int result; //Alamacena resultado de comparación
8
9          System.out.println("ingrese un clave (123 correcta)");
10         Scanner scanner =new Scanner(System.in);
11         clave=scanner.next();
12         clave=clave.toString();
13         result=clave.compareTo(clave1); //CompareTo devuelve 0 si son iguales
14         while(result!=0)
15         {
16             System.out.println("Clave incorrecta teclee otra"+clave+"-");
17             clave=scanner.next();
18             result=clave.compareTo(clave1);
19
20         }
21         System.out.println("Clave correcta");
22     }
23 }
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder imprimir en la consola, definir variables de tipo entero, interpretar los ciclos indeterminados.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar que son los ciclos indeterminados.

Objetivo. – Utilizar la estructura while en un programa JAVA para solicitar una clave de forma cíclica.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir tres variables, dos de tipo String y una de tipo entero, a continuación, solicitamos que se ingrese la clave, utilizaremos la estructura while para el ciclo indeterminado, el programa termina cuando se ingrese la clave correcta.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que informa de la clave incorrecta y la clave correcta.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 14: Estructura do-while en JAVA

Figura 34.- Programa que solicita que se ingresen números enteros de forma cíclica, el programa calcula la suma de todos ellos, el programa finaliza cuando se ingresa el cero e informa de la suma de todos los números ingresados.

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          int numero,resultado;
6          Scanner scanner =new Scanner(System.in);
7          System.out.println("Ingerse un número");
8          resultado=0;
9          numero=0;
10         do{
11                 numero=scanner.nextInt();
12                 resultado=resultado+numero;
13                 System.out.println("La suma es:"+resultado+" ingerse otro número (0 para salir)");
14         }while(numero>0);
15     }
16 }
17
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder imprimir en la consola, definir variables de tipo entero, capturar datos desde el teclado, interpretar los ciclos indeterminados.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar que son los ciclos indeterminados.

Objetivo. – Utilizar la estructura do-while en un programa JAVA para solicitar que se ingresen números enteros en un ciclo indeterminado.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir dos variables de tipo entero, y crear un objeto de la clase Scanner para capturar datos desde el teclado. A continuación, solicitamos que se ingrese un número entero, utilizaremos la estructura do-while para el ciclo indeterminado, el programa debe acular la suma de todos los números ingresados en una variable de tipo acumulador termina cuando se ingrese el cero.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que informa la suma de todos los números ingresados.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 15: Vectores en JAVA

Figura 35.- Programa que define un vector de diez posiciones, se llena utilizando la estructura for y se imprime el contenido en la consola.

Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5
6          int[] arreglo = new int[10];
7          int indice;
8
9          for(indice=0;indice<10;indice++){
10             arreglo[indice]=indice+1;
11         }
12
13         for(indice=0;indice<10;indice++){
14             System.out.println("arreglo["+indice+"]="+arreglo[indice]); ;
15         }
16
17     }
18 }
19
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder imprimir en la consola, definir variables de tipo entero, interpretar los ciclos determinados.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar que son los ciclos determinados, interpretar la estructura de datos unidimensional o vectores.

Objetivo. – Utilizar la estructura datos unidimensional (vector) en un programa JAVA, llenar el vector utilizando la estructura for e imprimir el contenido por consola.

Horas de práctica. – 2 Horas

Procedimiento. – En el método main de la clase principal de nuestro proyecto, debemos definir una variable de tipo entero, y un vector de tipo entero de diez posiciones. A continuación, llenamos el vector utilizando la estructura for, finalmente imprimimos el contenido del vector por consola.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que imprime el contenido dl vector por consola.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Práctica 16: Matrices en JAVA

Figura 36.- Programa que define una matriz de diez por diez, se llena utilizando la estructura for anidada y se imprime el contenido de la matriz en la consola.

*Pe.java

```
1  package Pe;
2  import java.util.Scanner;
3  public class Pe {
4      public static void main(String[] args) {
5          int[][] Matriz = new int[10][10];
6          int indice1,indice2;
7          for(indice1=0;indice1<10;indice1++){
8              for(indice2=0;indice2<10;indice2++){
9                  Matriz[indice1][indice2]=indice2;
10             }
11         }
12         for(indice1=0;indice1<10;indice1++){
13             for(indice2=0;indice2<10;indice2++){
14                 System.out.println("Matriz ("+indice1+", "+indice2+") valor:"+Matriz[indice1][indice2]);
15             }
16             System.out.println("-");
17         }
18
19     }
20 }
21
```

Proceso previo. – Para poder desarrollar esta práctica el estudiante debe poder imprimir en la consola, definir variables de tipo entero, interpretar los ciclos determinados.

Fundamentos. – Interpretar los conceptos de variable, identificar la clase principal y el método main, identificar la consola de salida del IDE Eclipse, interpretar el concepto de concatenación, interpretar que son los ciclos determinados, interpretar la estructura de datos unidimensional o matrices.

Objetivo. – Utilizar la estructura datos bidimensional (matriz) en un programa JAVA, llenar la matriz utilizando la estructura *for* e imprimir el contenido por consola.

Horas de práctica. – 2 Horas

Procedimiento. – En el método *main* de la clase principal de nuestro proyecto, debemos definir dos variables de tipo entero que funcionarán como índices, y una matriz de tipo entero de diez por diez. A continuación, llenamos el vector utilizando *for* anidados, finalmente imprimimos el contenido del vector por consola.

Evaluación del aprendizaje. – El estudiante presenta la evidencia del programa con los nombres de las variables que incluyan el nombre del estudiante, además la evidencia de la salida que imprime el contenido de la matriz por consola.

Normas de seguridad. – El estudiante debe conocer las vías y procedimientos de evacuación de los laboratorios de computación de la institución.

Referencias

2. Amaya Amaya, J. (2009). *Sistemas de información Gerenciales.* Bogotá: ECOE EDICIONES.

3. Barceló, M. (2008). *Una historia de la informática.* Barcelona, España: UOC.

4. CACES. (2021). *MODELO DE EVALUACIÓN EXTERNA 2024 CON FINES DE ACREDITACIÓN PARA LOS INSTITUTOS SUPERIORES TÉCNICOS Y TECNOLÓGICOS.* Quito: Consejo de Aseguramiento de la Calidad de la Educación Superior.

5. Cairó Battistutti, O. (2008). *Metodología de la Programación* (Quinta ed.). México: Alfaomega.

6. Constanza Mora, L., & Valero, N. (2020). LA YUPANA COMO HERRAMIENTA PEDAGÓGICA EN LA PRIMARIA. *LA YUPANA COMO HERRAMIENTA PEDAGÓGICA EN LA PRIMARIA*. Florida, Estados Unidos, Colombia: Institute for Human and Machine Cognition. Obtenido de https://cmapspublic2.ihmc.us/rid=1J2NH8QTM-2912G6-PZ5/

7. History Channel. (2021). La Historia del Ordenador. *La Historia del Ordenador*. Recuperado el 2024, de https://www.youtube.com/watch?v=fJPE1uaFahk

8. INEC. (20 de 03 de 2024). *Censo Ecuador*. Obtenido de https://censoecuador.ecudatanalytics.com/

9. Java. (17 de 09 de 2024). *Java | Oracle*. Obtenido de Java | Oracle: https://www.java.com/

10. Joyanes Aguilar, L., & Zahonero Martinez, I. (2005). *Programación en C Metodología, algoritmos y estructura de datos.* Madrid: McGraw-Hill.

11. Louridas, P. (2023). *Algoritmos.* Massachusetts: Melusina.

12. Martínez Ladrón de Guevara, J. (2024). *Fundamentos de programación en JAVA.* Madrid: Facultad de Informática Universidad Complutense de Madrid.

13. Martínez, R., & García-Beltrán, A. (2000). *BREVE HISTORIA DE LA INFORMÁTICA.* Madrid, España: ETSI Industriales – Universidad Politécnica de Madrid. Obtenido de https://moodle.upm.es/en-abierto/pluginfile.php/354/mod_label/intro/brevehistoriainformatica.pdf

14. Oviedo Fadul, A. (2004). *Diseño Estructurado de Algoritmos.* Sincelejo: Imprenor.

15. Rojas-Gamarra, M., & Stepanova, M. (03 de 10 de 2015). Sistema de numeración Inka en la Yupana y el Khipu. (R. L. Etnomatemática, Ed.) *Revista Latinoamericana de Etnomatemática, 8*(3), 46-48. Obtenido de https://www.redalyc.org/pdf/2740/274041587004.pdf

16. Szymanczyk, O. (2011). *Historia de las telecomunicaciones en la República Argentina.* Dunken.

17. Trigo Aranda, V. (2010). *Del Ábaco a Internet.* España: Creaciones Copyright, S.L.

SOLUZIONINNOVATIVE S.A.S.
EDITORIAL

editorialsoluzioniinnovative@gmail.com
https://soluzioninnovativegroup.com/repositorio/

Danny Gino Jiménez Torres

Inició su formación en la Escuela Politécnica Nacional, en la que obtuvo el título de tecnólogo en Análisis de Sistemas, fue aceptado para realizar prácticas profesionales en la Unidad Mixta de Investigación, conformada por la Universidad de Zaragoza y el Hospital Lozano Blesa de España. En la universidad de Zaragoza homologó su título a ingeniero en informática de gestión, y cursó el Máster en Bases de Datos e Internet. En esta misma ciudad dirigió el departamento de Informática en la empresa PRAMES donde participó de varios proyectos como senderos de Aragón, pionero en rutas para senderismo que luego fue patrocinado por el Ministerio de Turismo Español.

En 2024 obtuvo el título de Licenciado en Educación Básica por la Universidad Espíritu Santo de la ciudad de Guayaquil. Posee más de diez años de experiencia en educación superior, ha participado en varios proyectos que actualmente se encuentran en producción y conforman la infraestructura tecnológica del Instituto Superior Tecnológico Mariano Samaniego. Está a cargo de la Coordinación de la Carrera Desarrollo de Software desde el año 2013, y ha participado en el desarrollo varios proyectos de investigación en educación en el 2024.

ISBN: 978-9942-7294-0-8