GUÍA PARA EL EXAMEN EXTRAORDINARIO DE CIBERNÉTICA Y COMPUTACIÓN I

CCH ORIENTE

- ÁVILA NICOLÁS MARÍA DEL SOCORRO
- GARCÍA VÁZQUEZ EDGAR OMAR
- LÓPEZ LÓPEZ ERIKA
- LÓPEZ VARGAS GABRIELA
- TEOYOTL CALDERÓN YAZMIN

COORDINACIÓN:

LÓPEZ LÓPEZ ERIKA

Introducción

La presente guía tiene como objetivo ser un apoyo para el alumnado en su preparación al extraordinario de la asignatura de Cibernética y Computación I, que se imparte en el quinto semestre del Colegio de Ciencias y Humanidades. Ha sido elaborada en base al programa actualizado 2016 de la asignatura, y contempla los temas principales de la misma, es importante recalcar que el estudiantado deberá de revisar el Programa de Estudios de la asignatura para asegurar su preparación de acuerdo con el contenido temático de esta.

La guía consta de tres unidades y cada una de ellas esta compuesta por diferentes secciones, actividades de aprendizaje, una autoevaluación y la solución tanto para las actividades de aprendizaje como para las de autoevaluación (excepto aquellas que implique la reflexión del alumnado). Al final de la guía se presentan la bibliografía y fuentes de información utilizadas para su desarrollo.

INSTRUCCIONES

La guía de Cibernética y Computación I ha sido desarrollada para que la utilices como apoyo en la preparación de tu examen extraordinario, para ello te realizamos las siguientes recomendaciones:

- Revisa la información teórica – practica que se presenta en la sección de tu interés.

- Revisa los ejemplos proporcionados en la sección de tu interés antes de realizar las actividades de aprendizaje correspondientes.

- Desarrolla las actividades de aprendizaje y una vez que hayas concluido, verifica si tus resultados son los mismos a los proporcionados en la sección de respuestas, en caso de que hayas contestado erróneamente, refuerza tu conocimiento revisando nuevamente la información de la sección.

- Una vez que hayas reforzado tus conocimientos con las actividades de aprendizaje, realiza la actividad de autoevaluación correspondiente a la unidad que estes estudiando. Revisa tus resultados en la sección de respuestas y verifica si tus resultados son los mismos, en caso de que hayas contestado erróneamente, refuerza tu conocimiento revisando nuevamente la información de la unidad.

ÍNDICE

Unidad I. La Cibernética

Propósito:

Al finalizar la unidad el alumno:
Modelará un sistema relacionado con un tema de alguna disciplina de su interés, analizando el concepto de cibernética para interrelacionarlo con otras ciencias y los elementos que conforman un sistema.

Contenido

- Sección 1.1 Antecedentes de la Cibernética
- Sección 1.2 Sistemas
- Sección 1.3 Modelos

Cibernética: Es la ciencia que se ocupa de los procesos de dirección en los sistemas dinámicos complejos, teniendo por fundamento teórico las matemáticas y la lógica, así como el empleo de la automática.

Sistema: Es un grupo de componentes interrelacionados, que trabajan juntos hacia un objetivo común, al recibir entradas y producir salidas en un proceso organizado de transformación.

Ambiente: Es el medio que envuelve externamente el sistema

Sistemas abiertos: Presentan intercambio con el ambiente, a través de entradas y salidas.

Sistemas cerrados. No presentan intercambio con el medio ambiente que los rodea, son herméticos a cualquier influencia ambiental.

Sistemas físicos o concretos. Compuestos por equipos, maquinaria, objetos y cosas reales.

Sistemas abstractos. Compuestos por conceptos, planes, hipótesis e ideas.

Sistema aislado. En este tipo de sistemas no ocurre ningún tipo de intercambio, ni de energía ni de materia, con el exterior.

Sistemas de control: Son sistemas dinámicos y el control en ellos se realiza según el principio de realimentación.

Sistemas de control de lazo abierto: Es aquel cuya acción de control es independiente de la salida del sistema.

Sistemas de control de lazo cerrado: Es aquel cuya acción de control depende de la salida real.

Retroalimentación: Es un mecanismo de control de los sistemas dinámicos por el cual una cierta proporción de la señal de salida se redirige a la entrada, y así regula su comportamiento

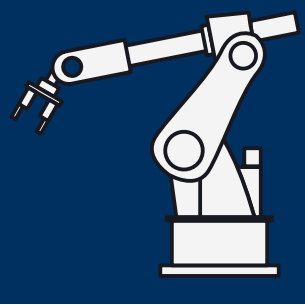
Sección 1.1 Antecedentes de la Cibernética

Aprendizaje:

1. Comprende la influencia de la cibernética en el desarrollo de la ciencia.
2. Describe el trabajo científico sobre la cibernética de Norbert Wiener, Arturo Rosenblueth, Claude Shannon, entre otros.

Temática

- **Definición del concepto de cibernética.**
- **Antecedentes de la cibernética.**
- **Relación de la cibernética con otras ciencias.**
- **Aplicaciones de la cibernética en la actualidad.**
- **Obra de distintos autores en trabajos científicos sobre la cibernética:**
  - Norbert Wiener.
  - Arturo Rosenblueth.
  - Claude Shannon.
  - Otros.



DEFINICIÓN DEL CONCEPTO DE CIBERNÉTICA

La cibernética (del griego χυβερνήτης, arte de dirigir) es la ciencia que se ocupa de los procesos de dirección en los sistemas dinámicos complejos y que tienen por fundamento teórico las matemáticas y la lógica, así como el empleo de la automática, especialmente de calculadoras electrónicas y de máquinas de control y lógico-informativas. (Bravo, 1968, pág. 9)

Norbert Wiener, acuño el término de ***cibernética***, al campo de la teoría de control y la comunicación en máquinas y animales, vocablo formado a partir del término χυβερνήτης (Kybernetes) o timonel. (Wiener, 1985, pág. 35)

El académico soviético A. N. Kolmogórov, en su prefacio al libro de W. Ross Ashby *Introducción a la cibernética*, define del siguiente modo el contenido de esta disciplina científica: "La cibernética se ocupa de estudiar los sistemas de cualquier naturaleza capaces de percibir, conservar y transformar información y utilizarla para la dirección y la regulación". (Bravo, 1968, pág. 59)

Otros autores han redefinido la cibernética, como W. Ross Ashby, llegó a determinar que *"La cibernética es la ciencia de control y de la comunicación con especial referencia a los sistemas adaptables o autocontrolados".* Jagjit A. Sing indica que *"La cibernética es la inquisición interdisciplinaria hacia la naturaleza y base física de la inteligencia humana, con el propósito de reproducirla en forma sintética"*, mientras que para Neville Moray, menciona que *"La cibernética es la ciencia que relaciona las entradas y salidas de un sistema, sus inputs y outputs".* Entre todas estas definiciones quedan implícitos dos conceptos: la comunicación y el sistema. (Ríos Estavillo, 1997, pág. 37)

ANTECEDENTES DE LA CIBERNÉTICA

La historia de la cibernética se puede dividir en dos periodos, el primer período comprendido entre los tiempos antiguos y el siglo XVII, el cual de acuerdo a Jramoi, hay que considerarlo como la prehistoria de los sistemas auto dirigidos, este período se caracteriza por la creación de mecanismos automáticos, que imitaban las propiedades externas de los animales y las personas (movimientos, gestos, sonidos); y el segundo periodo el cual comienza a partir del siglo XVII, que se caracteriza en fisiología por el descubrimiento de W. Harvey y en la técnica por la creación de mecanismos capaces de reproducir las facultades mentales del hombre y la de retroacción.

En el siglo IV se realizaron intentos de construir sistemas automáticos que reprodujesen los movimientos de los seres vivos. **Arquitas de Tarento** filósofo, matemático y político contemporáneo de Platón, es considerado el padre de la ingeniería mecánica y precursor occidental de la robótica, inventó el tornillo y la polea, entre otros muchos dispositivos. Fabricó el primer cohete autopropulsado de la historia, que usó con fines militares. Hacia el año 400 a. C. construyó un autómata consistente en una paloma de madera que rotaba por si sola gracias a un surtidor de agua o vapor y simulaba el vuelo.

Demetrio de Faleira (siglos IV – III) construyo un caracol que se arrastraba. El androide de Ptolomeo Filadelfo, mecanismo que imitaba los movimientos humanos (siglo III) y los actores automáticos que representaban en el teatro de Herón de Alejandría una obra en 5 actos y 8 cuadros sobre el regreso a la patria de los héroes de la guerra de Troya (siglo I).

Durante la Edad Media se tuvo una tendencia a reproducir los movimientos de los organismos vivos, tal es el caso del reloj de **Gaaz** (siglo V), que poseía un juego de figuras, las cuales, cada hora, salían de sus nichos y daban el correspondiente número de golpes, según una señal de la figura central; las figuras aulladoras de grifos y leones,

y también los pájaros cantores, a los lados del trono de oro del emperador bizantino Teófilo, obra del mecánico León el filósofo; El autómata de **R. Bacon y Alberto Magno** (siglo XII), mecanismo en forma de figura humana, que en respuesta a las llamadas a la puerta, la abría y saludaba al recién llegado con una inclinación de cabeza.

Durante el renacimiento aumentó el interés por la creación de autómatas que imitaran los movimientos de los animales y del hombre. Así **J. Müller** (1436 – 1476) – astrónomo, matemático y constructor alemán – creó una serie de autómatas, entre los cuales figuraba una mosca que corría alrededor de la mesa y un águila que fue colocada en las puertas de Nuremberg, para que desde ahí saludara agitando las alas y moviendo la cabeza al emperador Maximiliano, cuando éste hiciera su entrada en la ciudad.

Leonardo da Vinci (1452 – 1519) – pintor, escultor, narrador, músico, científico, matemático, arquitecto e ingeniero – construyó un mecanismo automático en forma de León, que en Milán, durante la ceremonia de recepción de Luis XII, se movía él solo por el salón del trono. Después de detenerse a los pies del rey, el león automático descubría con sus patas el pecho, del cual comenzaban a desprenderse flores de lis blancas, emblema de los monarcas de Francia. Además diseñó dos máquinas autómatas más, una para desbastar y pulir cilindros huecos y la otra para afilar agujas para la producción en mesa.

Juanelo Turriano, conocido matemático y mecánico del siglo XVI, preparo para Carlos V numerosos juguetes automáticos, entre los que figuraban soldados armados marchando, tocando el tambor y la corneta, pájaros voladores, etc. Entre estos destaca el autómata musical de una dama de la corte española con laúd. Emperadores, reyes y nobles rivalizaban por tener la colección más variada y completa de estos modelos mecánicos a escala que reproducían los movimientos humanos y qué servían de entretenimiento para los invitados.

RELACIÓN DE LA CIBERNÉTICA CON OTRAS CIENCIAS

La importancia de la cibernética para el progreso científico – técnico la determinan la exactitud y rapidez, cada vez mayores en la actualidad, la cibernética colabora con diversas ramas de la ciencia, algunas de éstas se me mencionan a continuación:

- **Biónica**: Se especializa en la producción de herramientas tecnológicas que simulen el funcionamiento o modelo de los seres vivos. Su objetivo es la combinación de sistemas electrónicos y biológicos, potenciando sus ventajas y características.

- **Neurofisiología**: Permite abordar y explorar la contribución del sistema nervioso central al control del movimiento. Con el desarrollo tecnológico se ha avanzado en técnicas de registro neurofisiológico, muchas de estas técnicas permiten conocer cambios funcionales y estructurales en el cerebro.

- **Computación**: Es la disciplina que, basada en la Electrónica, las Matemáticas y la Física, permite procesar de manera automatizada grandes volúmenes de información mediante la realización de todo tipo de cálculos numéricos.

- **Medicina**: Es la ciencia que tiene por objeto la conservación y el restablecimiento de la salud.

- **Física**: Estudia propiedades y transformaciones de la materia, fenómenos de todo tipo, e interroga experimental y teóricamente a la naturaleza, desde las partículas elementales hasta las galaxias.

- **Matemáticas**: Ciencia que estudia las relaciones entre cantidades, magnitudes y propiedades, y las operaciones lógicas mediante las cuales se pueden deducir cantidades, magnitudes y propiedades desconocidas.

APLICACIONES DE LA CIBERNÉTICA EN LA ACTUALIDAD

La cibernética ha evolucionado a lo largo de estas décadas y aún en la actualidad sigue vigente como en sus inicios, alguna de sus aplicaciones se menciona a continuación:

- **Inteligencia Artificial (IA)**. También llamada Inteligencia Sintética, Sistemas Inteligentes, Razonamiento Computacional, Inteligencia Computacional. Su visión es proveer de inteligencia a las máquinas a través de una programación principalmente algorítmica, apartada del conocimiento de los procesos de la inteligencia animal o humana.

- **Minería de Datos.** Surge como disciplina a partir de la convergencia de disciplinas teóricas -la estadística clásica, la inteligencia artificial y el aprendizaje computacional- con el crecimiento exponencial de las posibilidades técnicas de almacenamiento y procesamiento de información. Cada vez más se va haciendo factible predecir o anticipar comportamientos y patrones inmersos en las cada vez (Cebral Loureda, 2019)más grandes bases de datos de las que se disponen.

- **Redes Neuronales Artificiales.** Son algoritmos que pueden utilizarse tanto para problemas de clasificación como de regresión. Están construidos en base al comportamiento observado en los axones de las neuronas de los cerebros biológicos: es por esto que están especialmente vinculados a la cibernética.

- **Robótica**. Se ocupa del diseño, construcción, operación, fabricación y aplicación de los robots. Aspira a encontrar formas eficaces de colaboración entre robots y personas. Algunas de sus aplicaciones son en la medicina, al apoyar a mejorar los procedimientos quirúrgicos; en la industria automotriz, en donde realizan diversas actividades para el ensamblado de un auto; en la exploración espacial, en los transbordadores espaciales, en los teleoperadores, entre otros.

OBRA DE DISTINTOS AUTORES EN TRABAJOS CIENTÍFICOS SOBRE LA CIBERNÉTICA.

El siglo XVII es el punto de partida de la verdadera historia de los componentes tanto fisiológicos como técnicos de la cibernética. En 1615, el médico inglés **W. Harvey** descubrió el sistema de la circulación de la sangre, mostrando que dicha circulación es un sistema autorregido, en el que el corazón desempeña el papel de centro rector. W. Harvey fue educado en Cambridge y en 1597 se traslada a Padua, en donde frecuento inicialmente algunos cursos de Galileo, entonces catedrático de aquella universidad. De los programas académicos que se han conservado, puede inferirse que el estudiante asistió a las lecciones sobre “La esfera” y “La geometría de Euclides”, sustentadas en el periodo 1599 – 1600.

Blaise Pascal (1623-1662), eminente matemático, físico y filósofo francés construyó, a la edad de 19 años, la primera sumadora automática la “pascalina”. Su funcionamiento se basa en engranajes parecidos a los de un reloj. Fue construida por Pascal para ayudar a su padre, Etienne Pascal, un recaudador de impuestos, con la actividad tediosa de sumar y restar grandes secuencias de números.

Gotfried Wilhem Leibniz (1646-1716) matemático y filósofo alemán, empezó a trabajar en su invento, la máquina de calcular, un artificio diseñado para sumar, restar, multiplicar y dividir por repetición mecánica. Leibniz justificaba la construcción de la máquina para ahorrar tiempo en hacer los cálculos. El 1º de febrero de 1673 la máquina fue mostrada en la Royal Society durante su visita a Londres.

B. Pascal y Leibnitz abrieron una nueva página en la historia de los mecanismos automáticos, ya que fueron los primeros en tratar de reproducir con ayuda de medios mecánicos una de las facultades mentales del hombre.

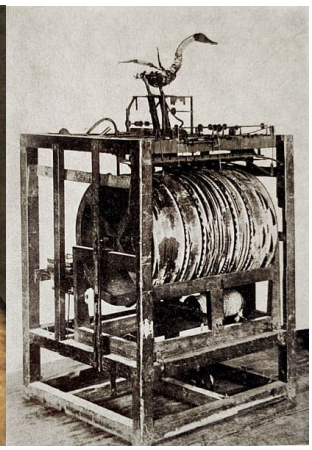
Christiaan Huygens (1629 – 1695) matemático y astrónomo holandés, fue el primero en usar el péndulo como regulador para grandes relojes, en 1657, en vez del balancín o foliot, lo que los dotaba de gran precisión. Los relojes empezaban a ser fiables medidores del tiempo. Huygens introdujo en la técnica un nexo entre el órgano rector y el regido, que aseguraba la retrasmisión al primero de la acción alterna del segundo. Este tipo de conexión fue denominado en 1906 por E. Rumer "retroacción". El sistema de retroacción se ha convertido en sinónimo de sistema automático, autorregido, y el concepto de retroacción se ha transmitido posteriormente a la fisiología.

René Descartes (1596 – 1650) nació el 31 de marzo de 1596 en el pequeño pueblo de La Haye, en la Turena francesa, en 1662 se publicó el "Tratado del hombre", fecha en la que Descartes ya había muerto, en el Tratado del hombre, establece una hipótesis acerca del hombre que rompe con la dominaba hasta entonces: las aristotélica, que tomaba como modelo al organismo animal y por eso definía al hombre como animal racional. Descartes abandona el modelo del organismo y lo sustituye por el modelo de máquina. Descartes piensa que el hombre, compuesto de cuerpo y mente, es una máquina sintiente, esto es, que entre sus múltiples funciones tiene también la de sentir. La descripción que hace del cuerpo humano en el Tratado del hombre explica las distintas funciones de éste a partir de los movimientos que genera el corazón, que para él es "principio de la vida"; mientras que el cerebro tiene que ver con la capacidad de sentir de la máquina que es el cuerpo humano, además está dotada de dos tipos de movimientos: unos físicos y otros sintientes, que convergen en la glándula pineal que en la concepción de Descartes es el centro (una especie de transductor) desde el que se

coordinan estos dos movimientos. Todas las funciones propias de la maquina viviente: la digestión el latido del corazón y de las arterias, la alimentación, el crecimiento y la respiración, son consecuencias naturales de la distribución de los órganos de la mencionada máquina, lo mismo que los movimientos de un reloj o de un autómata son resultados de la acción de contrapeso o ruedas.

En el Tratado del hombre Descartes estableció los principios de la doctrina de los reflejos, es decir, de los movimientos del organismo en respuesta a las excitaciones. Cuando se habla de movimientos, es decir, del paso de un estado a otro, es completamente natural plantearse la cuestión del tiempo mínimo necesario para semejante transición. Esta tarea se la impusieron los matemáticos que fundaron el cálculo de las variaciones, la cual desempeña un importante papel en la construcción de los sistemas cibernéticos actuales.

J. Vaucanson (1709 – 1782), mecánico francés de gran talento, constructor de una serie de telares y devanadoras de seda automáticos, fue conocido entre sus contemporáneos como creador de aparatos automáticos que imitaban los movimientos de los animales y del hombre. Hizo un pato de cobre que picoteaba el grano, bebía agua, graznaba e invitaba el proceso de la digestión, así como la figura de un flautista, del tamaño de una persona que ejecutaba diversas aristas.

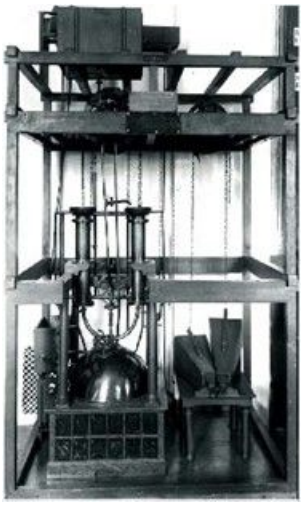
El telar automático de Vaucanson consistía en un sistema de mecanismos realizadores unidos entre sí y que funcionaban todos a partir de un mismo árbol distribuidor, al que hacía girar un motor. El mecanismo de la máquina tenía como base un cilindro perforado,

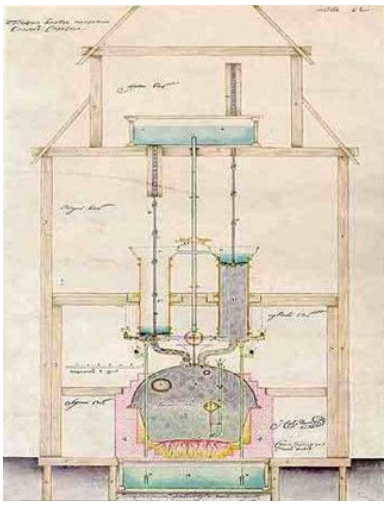
que, siguiendo un programa, efectuaba la selección de las agujas enhebradas a los hilos de la base.

Partiendo del telar de Vaucanson, el inventor francés **J. M. Jacquard** (1752 – 1834) en 1801 construyó su telar, el cual empleaba un sistema de tarjetas perforadas que determinaba la posición (atrás/adelante) del hilo de trama con respecto a la urdimbre. El funcionamiento se basaba en una serie de tarjetas – una por cada pasada de trama – perforadas y acomodadas de acuerdo con el patrón de diseño.

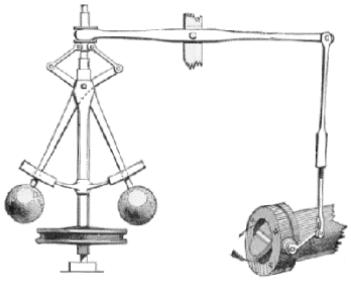
Estando en posición, permitían el paso de las agujas conectadas a los hilos de urdimbre correspondientes a las perforaciones, elevándolos para permitir el paso de la lanzadera. Una vez completado el movimiento se utilizaba la siguiente tarjeta y así sucesivamente. Al terminarse éstas, la secuencia comenzaba nuevamente, con lo cual se lograba un proceso continuo de diseños exactos. Esta técnica fue tan exitosa que, para 1812, el dispositivo fue incorporado a más de 18,000 telares en Francia, considerándose un cambio tecnológico muy importante.

Mikhail Vasilievich Lomonósov nace en 1711, en el pequeño pueblo ruso Mishaniskaia, creó una serie de mecanismos registradores: en 1748, un anemómetro, y en 1759 una brújula, ambos de registro automático. **Iván Polzunov** (1728 – 1766) construyó en 1765 el primer regulador automático e industrial del nivel de agua a base de flotador, en la caldera de la máquina de vapor creada por él. El flotador detecta el nivel de agua y controla la válvula que tapa la entrada de la caldera.

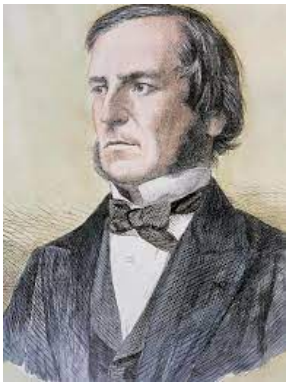




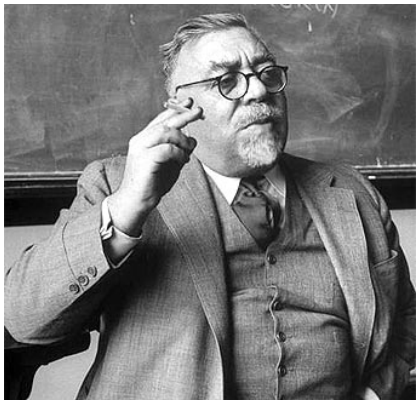
James Watt inventor escocés nacido en Greenock en 1736, obtuvo en 1784 una patente de perfeccionamiento de regulador centrífugo de la velocidad de rotación de la máquina de vapor, el cual desempeñó un gran papel en el desarrollo de la automática. El objetivo del regulador centrífugo era mantener una velocidad constante de la máquina independiente de las condiciones de carga.



George Boole (1815 – 1864) matemático inglés, considerado uno de los padres de las ciencias computacionales, en 1854 propone investigar las leyes fundamentales de las operaciones de la mente por las que se razona, para darles expresión en el lenguaje simbólico del cálculo, así crea los fundamentos de la lógica proposicional, un lenguaje formal para hacer inferencias lógicas, el Algebra de Boole. El legado de Boole se basa en una teoría matemática que simplifica los enunciados que tenían por respuesta «sí» o «no», usando para ello la aritmética binaria.



Norbert Wiener (1894 – 1964) ingeniero y matemático, considerado el ***"padre de la cibernética"***, realizó aportaciones importantes en varias áreas de las matemáticas, principalmente en análisis, probabilidad y teoría de control. Propuso emplear el vocablo "cibernética" para denominar la rama de la ciencia encargada de efectuar la dirección y la comunicación en los organismos vivos y en las máquinas. Su primer libro de cibernética, publicado en 1948, estaba dedicado a exponer los fundamentos generales de la ciencia sobre los mecanismos y sistemas autodirigidos, independientemente de que debieran su creación a la naturaleza o al individuo.

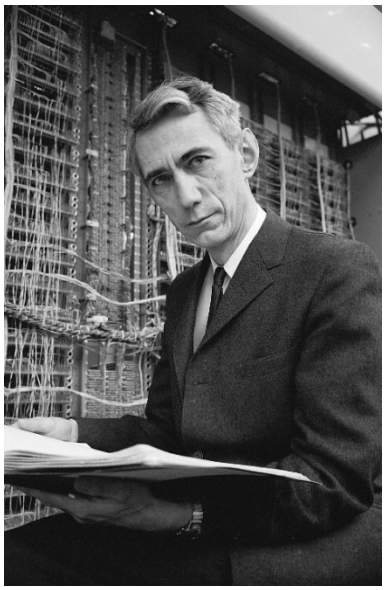


Wiener estudió sobre todo las nuevas máquinas de comunicación, de cómputo, con sistemas de información, de comportamientos esperados y controlados, de atención y memoria, de información y de monitoreos, todos tan parecidos al "individuo humano" y con posibilidades de desarrollo tan grandes, que no pudo menos de vincular el destino del hombre y de las máquinas para reflexionar y monitorear el proyecto humanista. Así decidió escribir un libro sobre cibernética y sociedad al que tituló *El Uso Humano de los Seres Humanos*, firmado el 12 de octubre de 1949 en el Instituto Nacional de Cardiología de México. Su libro no sólo constituye hasta hoy un alegato admirable contra el uso inhumano de los seres humanos, sino un serio llamado de alerta destinado a detener los peligros de entropía o destrucción que amenazan la existencia misma de la humanidad. (González Casanova, 2017, pág. 56)

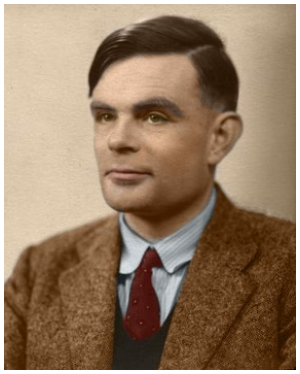
Arturo Rosenblueth Stearns (1900 – 1970) fisiólogo, filósofo, matemático y precursor de la Cibernética, nació en Ciudad Guerrero, se dedicó al estudio de la neurofisiología, y al hacerlo descubrió —junto con su maestro Walter B. Cannon— que la transmisión nerviosa tiene un carácter químico, justo cuando se creía que ésta sólo era un impulso eléctrico. Rosenblueth y un grupo colectivo de científicos de diversas disciplinas —y que se daría a sí mismo el nombre de "Grupo cibernética"— discutirían por varios años acerca

de los mecanismos que giran en torno a la relación del hombre y su entorno, del sistema nervioso y el cuerpo, y particularmente acerca de las semejanzas entre los animales y las máquinas, entre el sistema nervioso y la después llamada inteligencia artificial. Fue en el también llamado “Grupo Macy” donde Rosenblueth expone por primera vez las ideas que fundamentan la concepción y diseño de los servomecanismos, antecedente de las “máquinas de computar” que serían publicadas por el propio Arturo Rosenblueth, Norbert Wiener y Julian Bigelow en “Behaviour, Purposeful and Teleology”, en la revista Philosophy of Science de 1943, el cual dio vida a la ciencia cibernética.

Claude Elwood Shannon (1916 – 2001) ingeniero estadounidense, en 1948 desarrolló una teoría que después dio lugar a una nueva disciplina: Teoría de la Información.



Shannon concibe a la información como unidad cuantificable que no tiene en cuenta el contenido del mensaje. La comunicación es vista como un proceso secuencial que comienza desde la fuente, que elige un mensaje, que es transmitido, en forma de señal y mediante un canal de comunicación, hacia un receptor, que vuelve a convertir la señal en un mensaje dirigido a un destino. La teoría de Shannon permite, estudiar la cantidad de información de un mensaje en función de la capacidad del sistema para transmitir y/o almacenar. Esta capacidad se mide según el sistema binario (dos posibilidades I / O) en **bits** (binary digits) asociados a la velocidad de transmisión del mensaje, pudiendo esta velocidad ser disminuida por el ruido. Claude Shannon dice que el tiempo necesario para transmitir información es proporcional a la cantidad de información transmitida, es decir, si se transmite más información, será necesario mayor tiempo.



Alan Mathison Turing (1912 – 1954), matemático inglés, estudia en Cambridge (Inglaterra), a los 24 años proyecta su sistema científico teniendo como horizonte la posibilidad de la reconstrucción mental y física del hombre, donde muestra su interés interdisciplinario entrelazando las matemáticas y lógica con la biología.

Sus principales obras son:

- *"On computable numbers with an application to the Entscheidungsproblem".* 1936: En esta obra funda la teoría lógica de la calculabilidad que se basa en el concepto de "máquina de Turing". El escrito plantea las bases teóricas de la estructuración de la calculabilidad, que lleva a la constitución de la Informática teórica y la realización del computador.

- *"The chemical basis of morphogenesis*". 1952: En esta obra plantea las bases de una teoría general de la morfogénesis que tiene como objetivo presentar las diferentes formas existentes en la organización de los seres vivos. Esta teoría estudia las reacciones químicas en el seno del organismo por medio de una modelización matemática seguida de una simulación informática.

- *"Computing Machinery and Intelligence"*. 1950. Establece la relación entre dos temas de investigación: lógica y biología, propone el estudio de los procesos cognitivos y del lenguaje por simulación informática, base de lo que se llamaría, Inteligencia Artificial.

ACTIVIDADES DE APRENDIZAJE
SECCIÓN 1.1

1. Selecciona la respuesta correcta.

I. ¿Cuál es la característica principal del primer período de la cibernética?
a) Uso de computadoras
b) Creación de sistemas auto dirigidos
c) Descubrimiento de la electricidad
d) Desarrollo de la inteligencia artificial

II. ¿Qué dispositivo inventó Arquitas de Tarento?
a) Paloma mecánica
b) Autómata musical
c) Cohete autopropulsado
d) Androide de Ptolomeo

III. ¿Qué personaje construyó un caracol que se arrastraba?
a) W. Harvey
b) Arquitas de Tarento
c) Demetrio de Faleira
d) Leonardo da Vinci

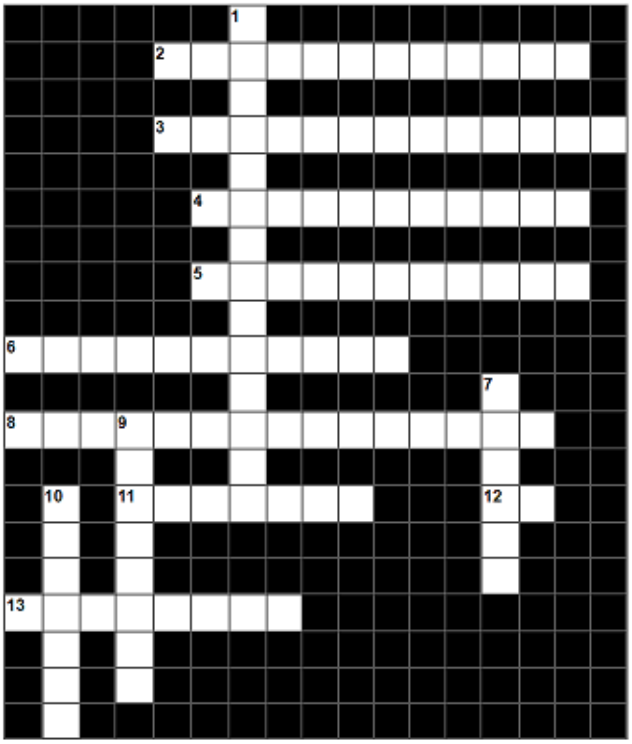
IV. ¿Qué figura histórica creó un autómata musical de una dama de la corte española?
a) Juanelo Turriano
b) Leonardo da Vinci
c) J. Müller
d) Demetrio de Faleira

V. ¿Qué matemático y mecánico preparó juguetes automáticos para Carlos V?
a) Arquitas de Tarento
b) Juanelo Turriano
c) Leonardo da Vinci
d) J. Müller

VI. ¿Qué descubrimiento hizo W. Harvey en fisiología?
a) Creación de autómatas
b) Descubrimiento de la electricidad
c) Funcionamiento del sistema nervioso
d) Descubrimiento de la circulación sanguínea

VII. ¿Qué pintor, escultor y científico creó un mecanismo automático en forma de León?
a) W. Harvey
b) Arquitas de Tarento
c) Demetrio de Faleira
d) Leonardo da Vinci

2. Contesta el siguiente crucigrama con palabras clave de la relación de la cibernética con otras ciencias y aplicaciones de la cibernética en la actualidad.



Horizontales

2 Disciplina que estudia el comportamiento y las funciones cerebrales.
3 Rama de la medicina que utiliza tecnología para el diagnóstico.
4 Estudio de las relaciones entre cantidades y magnitudes.
5 Disciplina que explora la comunicación y el control en sistemas.
6 Procesamiento automatizado de información.
8 Ciencia que se centra en el sistema nervioso y el movimiento.
11 Disciplina que simula el funcionamiento de seres vivos en tecnología.
12 Técnica que busca imitar la inteligencia humana en máquinas.
13 Área que se enfoca en la conservación y restauración de la salud.

Verticales

1 Uso de la tecnología para mejorar procesos industriales.
7 Ciencia que estudia las propiedades de la materia y la energía.
9 Diseño y construcción de robots
10 Aplicación de algoritmos para descubrir patrones en datos.

3. Relaciona las siguientes columnas

A. Descubrió el sistema de la circulación de la sangre y su autorregulación.	( )	René Descartes
B. Construyó la primera sumadora automática conocida como la "pascalina".	( )	W. Harvey
C. Trabajó en la invención de la máquina de calcular y su justificación para ahorrar tiempo en cálculos.	( )	J. Vaucanson
D. Introdujo el péndulo como regulador para relojes, aumentando su precisión.	( )	Norbert Wiener
E. Cambió el modelo de organismo al de máquina en su "Tratado del hombre".	( )	Arturo Rosenblueth Stearns
F. Fue conocido por construir aparatos automáticos que imitaban movimientos humanos y animales.	( )	Christiaan Huygens
G. Es considerado uno de los padres de las ciencias computacionales y creó el Álgebra de Boole.	( )	Blaise Pascal
H. Conocido como el "padre de la cibernética", propuso el término "cibernética" y estudió sistemas autodirigidos.	( )	Claude Elwood Shannon
I. Precursor de la cibernética, estudió la relación entre humanos y máquinas y fundamentó los servomecanismos.	( )	George Boole
J. Desarrolló la teoría de la información y la comunicación, vista como un proceso secuencial.	( )	Alan Mathison Turing
K. Fundó la teoría lógica de la calculabilidad y planteó las bases de la morfogénesis y la inteligencia artificial.	( )	Gotfried Wilhem Leibniz

Sección 1.2 Sistemas

Aprendizaje:

3. Comprende los componentes de un sistema.
4. Comprende los componentes esenciales de un sistema de control

Temática

Sistemas:

- Concepto.
- Elementos.
- Ambiente.
- Clasificación.

Sistemas de control:

- Lazo abierto.
- Lazo cerrado.
- Retroalimentación.

SISTEMAS

De acuerdo con O'Brien y Marakas, un sistema es un grupo de componentes interrelacionados, con un límite definido con claridad, que trabajan juntos hacia un objetivo común, al recibir entradas y producir salidas en un proceso organizado de transformación (O'Brien & Marakas, 2006, pág. 24).

Tiene tres componentes o funciones básicas interactivas:

- **Entrada (input)**. Son los elementos que ingresan al sistema para ser procesados.
- **Procesamiento**. Son los procesos de transformación que convierten las entradas en salidas.
- **Salida (output)**. Incluye la transferencia de los elementos que se han producido en un proceso de transformación hasta su destino final.

Ejemplo 1: Un sistema de producción de papas fritas, acepta materias primas como entradas y produce bienes terminados como salidas.

AMBIENTE

Es el medio que envuelve externamente el sistema. Está en constante interacción con el sistema, ya que éste recibe entradas, las procesa y efectúa salidas. La supervivencia de un sistema depende de su capacidad de adaptarse, cambiar y responder a las exigencias y demandas del ambiente externo. Aunque el ambiente puede ser un recurso para el sistema, también puede ser una amenaza.

CLASIFICACIÓN

- **Sistemas abiertos**. Presentan intercambio con el ambiente, a través de entradas y salidas. Intercambian energía y materia con el ambiente. Son adaptativos para sobrevivir. Su estructura es óptima cuando el conjunto de elementos del sistema se organiza, aproximándose a una operación adaptativa. La adaptabilidad es un continuo proceso de aprendizaje y de auto-organización.

- **Sistemas cerrados**. No presentan intercambio con el medio ambiente que los rodea, son herméticos a cualquier influencia ambiental. No reciben ningún recurso externo. Se da el nombre de sistema cerrado a aquellos sistemas cuyo comportamiento es determinístico y programado y que opera con muy pequeño intercambio de energía y materia con el ambiente.

- **Sistemas físicos o concretos.** Compuestos por equipos, maquinaria, objetos y cosas reales. El hardware.

- **Sistemas abstractos**. Compuestos por conceptos, planes, hipótesis e ideas. Muchas veces solo existen en el pensamiento de las personas. Es el software.

- **Sistema aislado**. En este tipo de sistemas no ocurre ningún tipo de intercambio, ni de energía ni de materia, con el exterior.

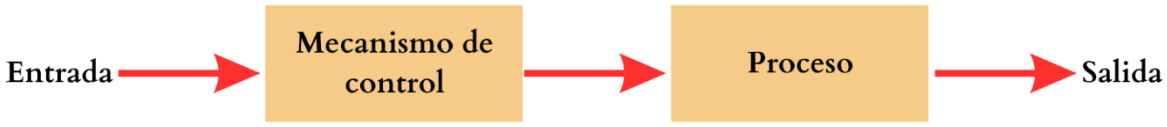
SISTEMAS DE CONTROL

Los sistemas de control son considerados como conjunto de subsistemas de control interactuantes para cada uno de los que puede ser definida la finalidad u objetivo de su funcionamiento, fines particulares que se le subordinan al objetivo general de trabajo del sistema en su conjunto.

Los sistemas de control son sistemas dinámicos, son sistemas de información, y el control en ellos se realiza según el principio de realimentación.

Un sistema de control puede disponer de procesos cíclicos capaces de detectar desequilibrios internos, mediante el análisis detallado de la complejidad de las relaciones dentro del sistema que puedan restablecer el equilibrio.

Lazo abierto. Es aquel cuya acción de control es independiente de la salida del sistema, es decir, en estos sistemas no se tiene en cuenta la salida para su realimentación y comparación con la entrada.



La exactitud de estos sistemas depende de su calibración, por lo que, en presencia de perturbaciones, dejan de cumplir o cumplen mal la función de control asignada.

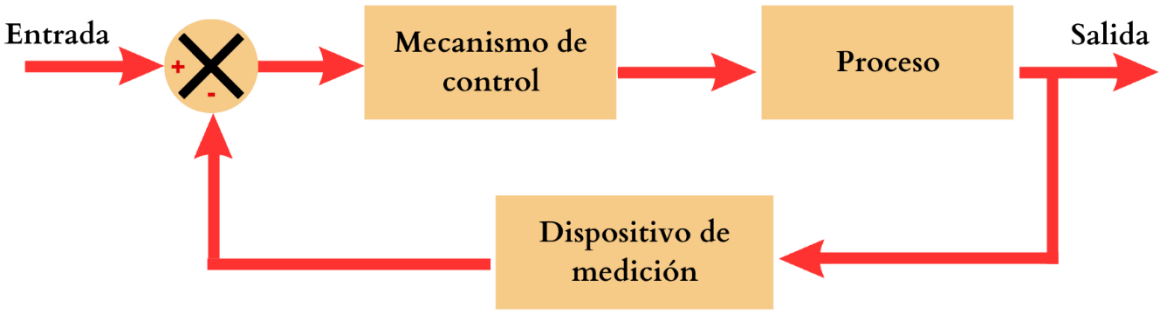
Ejemplo 2: Un ejemplo característico es la lavadora.

- Antes de ponerlo en funcionamiento, se determina tiempo de lavado y la fuerza de lavado.
- Al ponerlo en marcha, la lavadora funciona con la intensidad y tiempo seleccionado.
- Aunque la salida que produzca no se ajuste a lo esperado, el sistema no actuará de modo diferente.

Estos sistemas llevan en algún dispositivo de control de tiempo, denominado cronométrico, para regular las paradas.

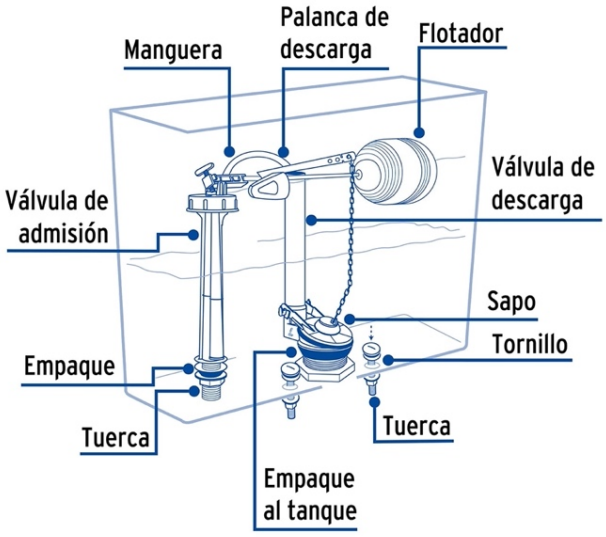
Lazo cerrado

Es aquel cuya acción de control depende de alguna manera, de la salida real, es decir, existe un dispositivo de medición, capaz de regular el mecanismo de control en función de la respuesta del sistema.



Los sistemas en lazo cerrado se llaman también sistemas en realimentación.

Ejemplo 3: La cisterna de agua del inodoro. Funciona del siguiente modo:



- Al accionar la palanca de descarga, se levanta el tapón inferior (sapo) y se produce el vaciado del depósito.

- El flotador cae hasta la parte inferior, la válvula de admisión se separa de la boca del llenado y permite la entrada de agua.

- Se inicia el proceso de llenado, a medida que el nivel del agua sube, el flotador también sube.

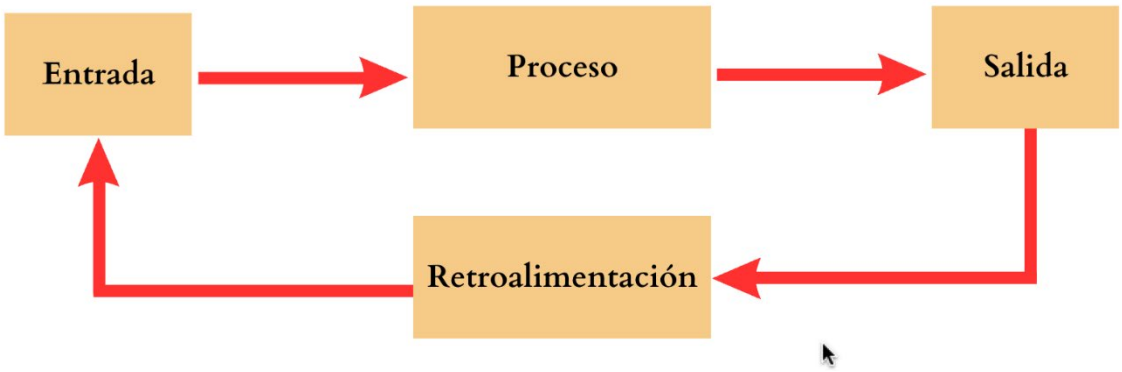
- Al llegar el flotador a la altura máxima, el flotador presiona la goma de la válvula de admisión y corta el paso del agua. De este modo, el tanque del inodoro se llena sin necesidad de abrir o cerrar manualmente el grifo de entrada de agua.

Retroalimentación

En la teoría de sistemas, en cibernética y en la teoría de control, entre otras disciplinas, la retroalimentación, cuyo término correcto es realimentación (en inglés feedback), es un mecanismo de control de los sistemas dinámicos por el cual una cierta proporción de la señal de salida se redirige a la entrada, y así regula su comportamiento.

Es la función del sistema que tiende a comparar la salida con un criterio o estándar previamente establecido. Tiene como objetivo controlar, es decir, mantener un constante monitoreo de los resultados, comparándolos con lo esperado.

También se considera la retroalimentación como un proceso cuya señal se mueve dentro de un sistema (entrada-proceso-salida-retroalimentación-entrada), formando un ciclo.

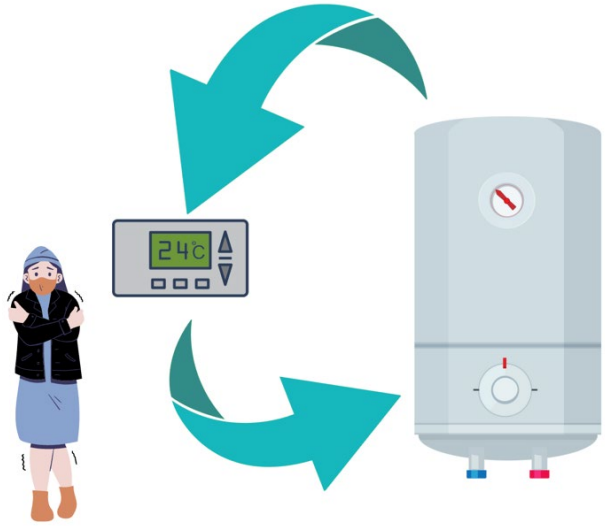


La retroalimentación puede ser negativa o positiva.

Retroalimentación negativa. Ocurre cuando el sistema se opone al estímulo inicial, de manera que aquello que ha variado retome su valor o posición determinada, conservando así la homeostasis (equilibrio dinámico del sistema). Se dice retroalimentación negativa

porque la respuesta del sistema de control es opuesta al estímulo y se caracteriza porque busca mantener los objetivos propuestos.

Ejemplo 4. El termostato es un dispositivo tecnológico que opera bajo el principio de la retroalimentación negativa. Éste actúa a partir de una temperatura deseada, indicada en el tablero. Si la temperatura del cuarto está por debajo de lo establecido en el indicador, se activa el elemento que disminuye el aire frío para que la temperatura suba hasta el nivel deseado de manera lenta; si, por el contrario, la temperatura está por arriba de la deseada, el regulador aumenta la cantidad de aire frío o se interrumpe el flujo de aire caliente. De este modo, la temperatura oscila alrededor del valor indicado.



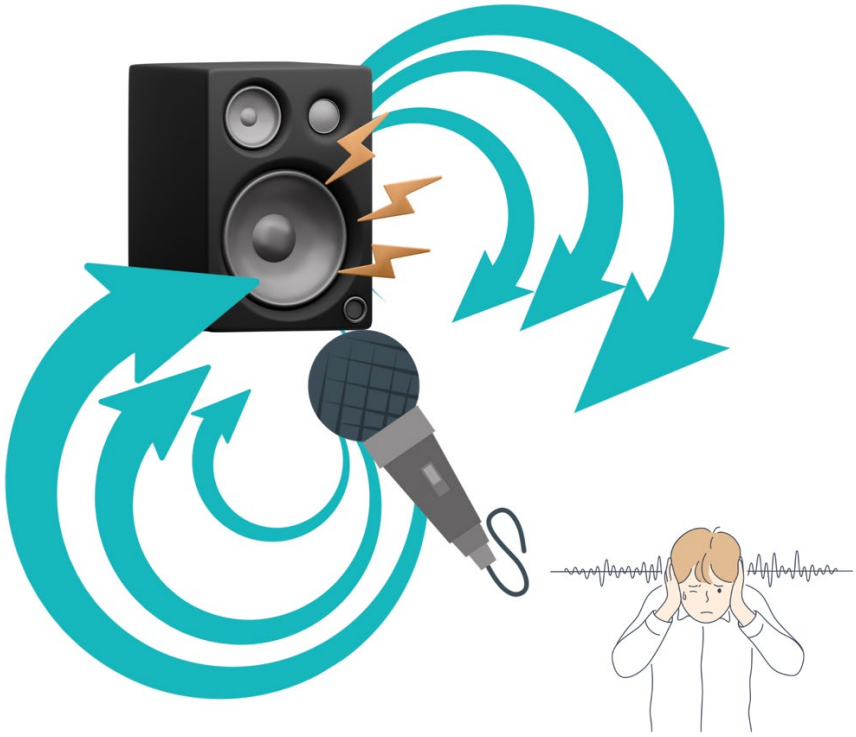
Retroalimentación positiva. Ocurre cuando al aplicar una perturbación a un sistema, desencadena una serie de variaciones que se propaga en otros componentes del sistema aumentando su inestabilidad. La retroalimentación positiva está asociada a los fenómenos de crecimiento y diferenciación, en donde se mantiene un sistema y se modifican sus metas y/o fines.

En la retroalimentación positiva, parte de la señal de salida se suma (incrementa) a la señal de entrada. Los bucles de realimentación positivos tienden a no mantener el

equilibrio, al contrario, transportan al sistema hacia nuevos estados. Son mecanismos de realimentación reforzadora, de amplificación de las desviaciones.

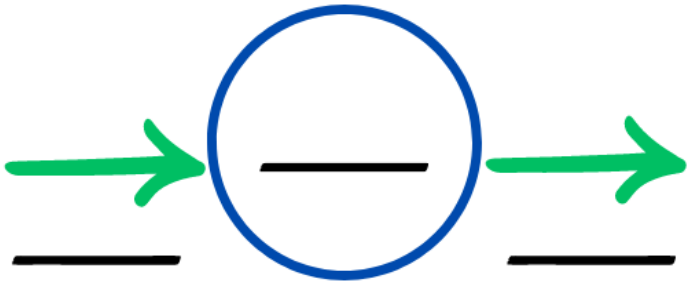
Ejemplo 5. El acople de un sonido.

La salida de audio que tiene lugar por el altavoz es captada por el micrófono que la vuelve a introducir, vuelve a salir el sonido amplificado que vuelve a entrar con mayor intensidad a través del micro, y en escasas décimas de segundo se produce el molesto sonido que nos es muy conocido. Los bucles de realimentación positivos representan un “efecto bola de nieve”, un círculo virtuoso.



ACTIVIDADES DE APRENDIZAJE
SECCIÓN 1.2

1. Escribe en el siguiente diagrama los tres componentes de un sistema



2. Relaciona las siguientes columnas

A. Presentan intercambio con el ambiente.	( ) Sistema aislado
B. No presentan intercambio con el medio ambiente que los rodea.	( ) Sistemas físicos o concretos
C. Compuestos por equipos, maquinaria, objetos y cosas reales.	( ) Sistemas cerrados
D. Compuestos por conceptos, planes, hipótesis e ideas.	( ) Sistemas abstractos
E. No ocurre intercambio de energía ni materia con el exterior.	( ) Sistemas abiertos

3. Define qué es un Sistema de Control:

_____

_____

_____

_____

4. Proporciona un ejemplo de **lazo abierto** e indica su funcionamiento:

5. Proporciona un ejemplo de **lazo cerrado** e indica su funcionamiento:

6. Selecciona las respuestas correctas:

I. ¿Cuál es el término correcto para referirse al mecanismo de control de los sistemas dinámicos que redirige parte de la señal de salida a la entrada para regular su comportamiento?
a) Reforzamiento
b) Regulación
c) Retroalimentación
d) Retrospección

II. ¿Cuál es el objetivo principal de la retroalimentación en un sistema?

a) Crear variaciones en la señal de salida
b) Mantener un equilibrio constante en el sistema
c) Aumentar la inestabilidad del sistema
d) Desencadenar perturbaciones en la entrada

III. ¿Qué tipo de retroalimentación busca mantener los objetivos propuestos y se opone al estímulo inicial?

a) Retroalimentación reforzadora
b) Retroalimentación positiva
c) Retroalimentación contraproducente
d) Retroalimentación negativa

IV. ¿En qué fenómenos está asociada principalmente la retroalimentación positiva?

a) Homeostasis y equilibrio
b) Mantenimiento del sistema
c) Crecimiento y diferenciación
d) Estabilidad y control

V. ¿Cómo se diferencia la retroalimentación positiva de la negativa en términos de su efecto en el sistema?
a) La retroalimentación positiva incrementa las desviaciones y puede llevar a nuevos estados
b) La retroalimentación positiva busca mantener el equilibrio.
c) La retroalimentación negativa está asociada al crecimiento.
d) La retroalimentación negativa no modifica los objetivos del sistema.

Sección 1.3 Modelos

Aprendizaje:

5. Comprende el concepto y la importancia del modelo.
6. Desarrolla el modelo de un sistema.
7. Explica cómo construyó el modelo del sistema, las partes que lo conforman y su funcionamiento.

Temática

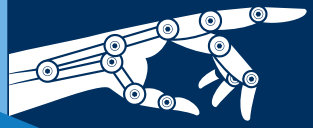
Modelos:

- Concepto.
- Tipos.
  - Naturales y artificiales.
  - Analógicos y digitales.
  - Matemáticos.
  - Conceptuales.
- Relación.

Elementos para modelar un sistema:

- Entrada y salida.
- Proceso.

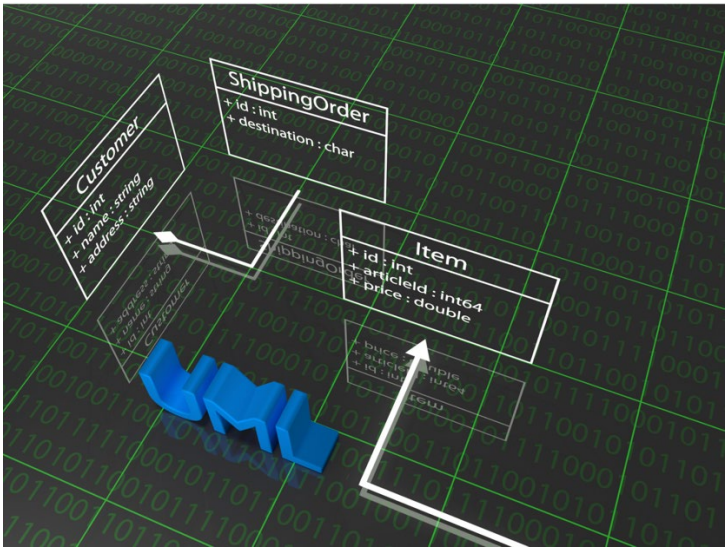
Desarrollo del proyecto.



MODELOS

Un modelo es una versión simplificada de un sistema complejo, se utilizan para tratar de conservar los aspectos pertinentes del sistema que están siendo modelados. Los modelos son aceptados por su utilidad, y no pueden juzgarse como ciertos o falsos. Los modelos son creados empleando herramientas de modelado.

Ejemplo 1. En informática, es muy utilizado el Lenguaje de Modelado Unificado (UML) utilizado para la especificación, visualización, construcción y documentación de una estructura o proceso y su comportamiento.

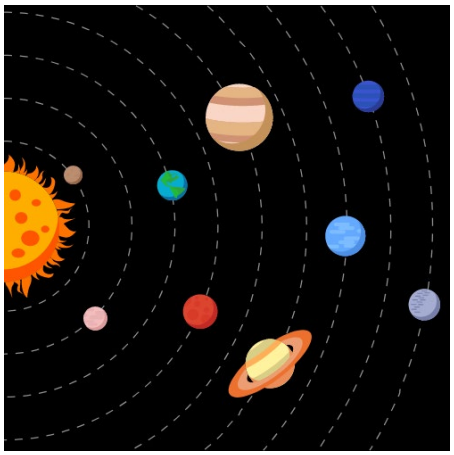


TIPOS DE MODELOS

Modelo Natural

Un sistema es natural cuando existe naturalmente sin que en su origen intervenga la mano del hombre.

Ejemplo 2. Sistema solar



Modelo Artificial

Un sistema es artificial cuando existe por la intervención del hombre.

Ejemplo 3. Una televisión, es un sistema artificial que inventando el hombre como medio de comunicación, información y distracción.

Modelo analógico

Es una representación material de un objeto o un proceso para entender, mejor su origen, formación o funcionamiento. Se constituyen mediante un conjunto de convenciones que sintetizan y codifican propiedades del objeto real para facilitar la lectura o interpretación de estas.

Los modelos análogos son representaciones a escala que constituyen la mejor alternativa cuando, para explicar un fenómeno físico, el análisis matemático es insuficiente y por algún motivo la experimentación directa no es viable.

Ejemplo 4. El modelado analógico como método para las enseñanzas de las estructuras geológicas

La utilización de modelos analógicos como método para las enseñanzas de las estructuras geológicas permite romper la artificialidad de los esquemas habituales realizados en la pizarra: condensan un proceso que dura millones de años en unos pocos segundos, y reducen las cadenas de montañas o los océanos a unos pocos centímetros.

Estos modelos permiten disponer de un material didáctico en el aula o laboratorio para trabajar con el alumnado los conceptos y metodologías propios de los estudios geológicos (Murcia López & Crespo-Blanc, 2008).

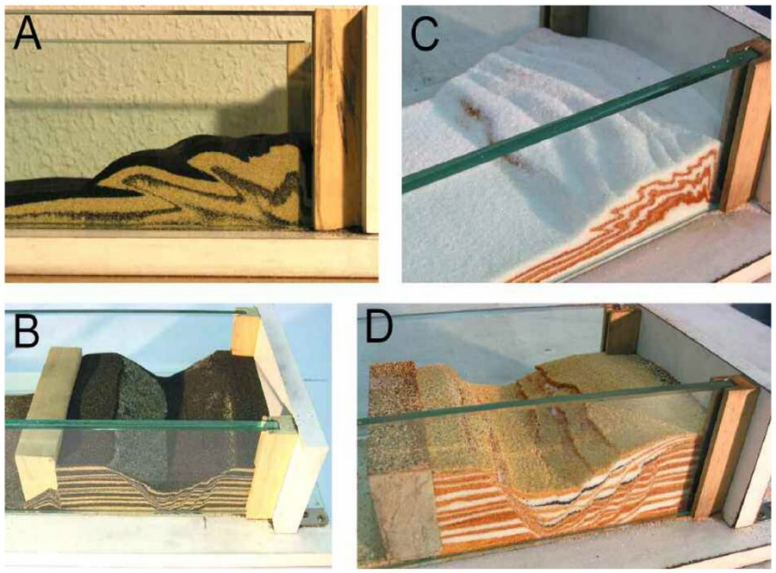


Fig. 1. Sistemas de cabalgamientos (arriba) y sistemas de fallas normales (abajo) realizados con el "mini-Laboratorio", con arena (A y B) o azúcar (C y D). En el modelo en extensión realizado con azúcar, la capa muy oscura que aparece es café que no se ha utilizado en el modelo en compresión (C).

Modelo digital

Los modelos digitales son estructuras de información que permiten generar, evaluar y representar un diseño, además de fabricarlo. La información que contienen se procesa con diversas herramientas según los objetivos y las acciones en cada fase del diseño.

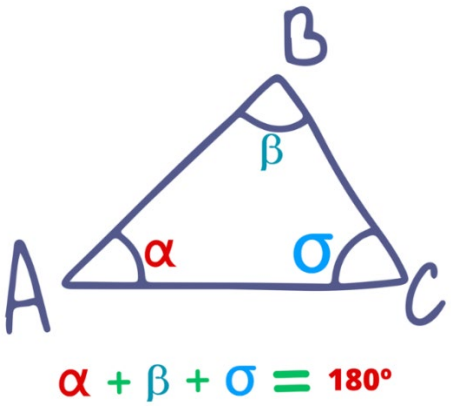
Un modelo digital puede utilizarse para generar y evaluar la forma y comportamiento de un proyecto, para optimizar su construcción y para controlar el funcionamiento del edificio una vez construido.

Ejemplo 5. Modelado digital 3D, estos modelos concretizan conceptos complejos o muy abstractos, y otorgan un referente visual que favorezca su comprensión. De este modo el estudiante puede explorar dicho modelo desde todos sus ángulos.

Modelo matemático

Un modelo matemático es la descripción de la realidad para poder aplicar herramientas matemáticas, sus técnicas y teorías y poder traducir los resultados obtenidos en pronósticos u operaciones, en el mundo real.

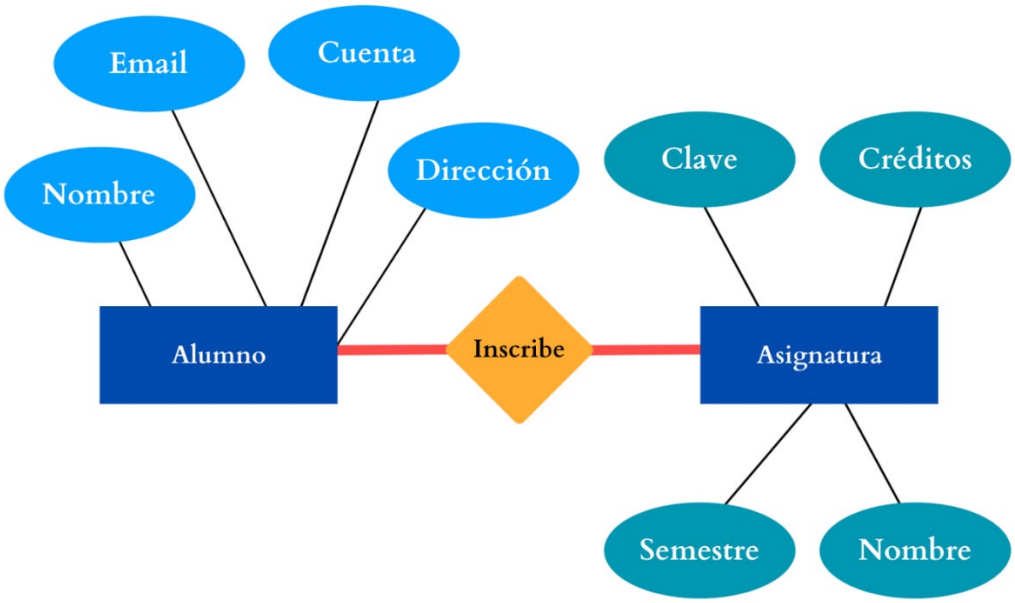
Ejemplo 6. un modelo matemático bien conocido es que la suma de los ángulos interiores en un triángulo plano es de 180º



Modelo conceptual

El modelo conceptual brinda una abstracción de un sistema real, permite visualizar en condiciones controladas, los componentes que lo conforman, la estructura como se pueden organizar, y la manera como se articulan e integran para soportar el desarrollo de las actividades de los procesos del sistema.

Ejemplo 7. Modelo conceptual de una base de datos, como puede ser el modelo Entidad Relación (E/R), el cual está compuesto por entidades, atributos y relaciones, y la función de este modelo es representar gráficamente la interacción entre estos elementos.



Relación

Las relaciones internas y externas de los sistemas han tomado diversas denominaciones, también se les conoce como: efectos recíprocos, interrelaciones, organización, comunicaciones, flujos, prestaciones, asociaciones, intercambios, interdependencias, coherencias, en otros.

Las relaciones entre los elementos de un sistema y su ambiente son de vital importancia para la comprensión del comportamiento del sistema. Las relaciones pueden ser recíprocas (circularidad) o unidireccionales. Presentadas en un momento del sistema, las relaciones pueden ser observadas como una red estructurada bajo el esquema input/output.

ELEMENTOS PARA MODELAR UN SISTEMA

El modelado de sistemas es el proceso para desarrollar modelos abstractos de un sistema, donde cada modelo presenta una visión o perspectiva diferente de dicho sistema. El modelado de sistemas se ha convertido en un medio para representar el sistema usando algún tipo de notación gráfica.

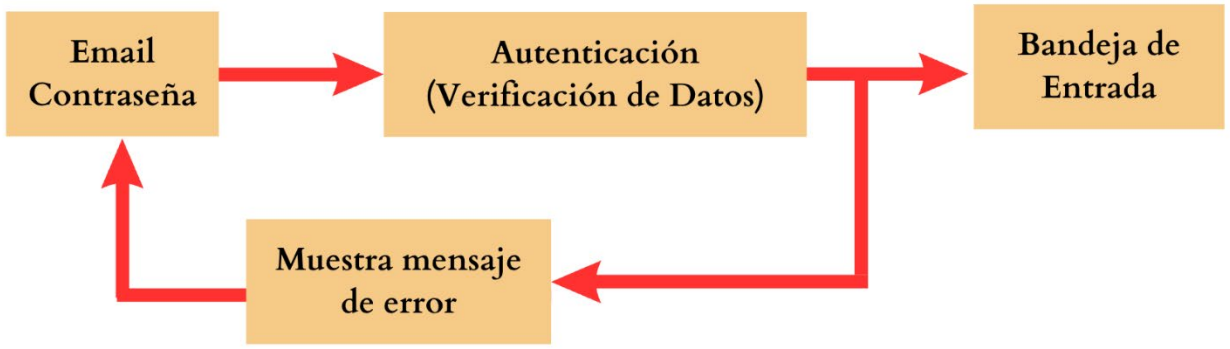
Los modelos se usan durante el proceso de ingeniería de requerimientos para ayudar a derivar los requerimientos de un sistema, durante el proceso de diseño para describir el sistema a los ingenieros que implementan el sistema, y después de la implementación para documentar la estructura y la operación del sistema.

Los elementos principales de un sistema son:

- **Entrada (input)**. Son los elementos que ingresan al sistema para ser procesados.
- **Procesamiento**. Son los procesos de transformación que convierten las entradas en salidas.
- **Salida (output)**. Incluye la transferencia de los elementos que se han producido en un proceso de transformación hasta su destino final.

Ejemplo 8. Login a correo electrónico

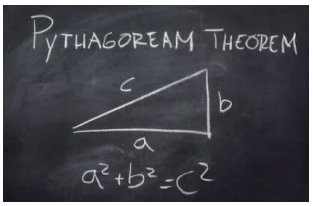
Entrada	• Correo electrónico (email) • Contraseña
Proceso	El servicio de correo electrónico (Gmail, Hotmail, Yahoo, etc.) verifica que los datos proporcionados sean correctos, si son correctos muestra la bandeja de entrada, en caso contrario envía un mensaje de error.
Salida	• Bandeja de entrada • Mensaje de error



1. **Relaciona las siguientes columnas de acuerdo con el tipo de modelo que representa cada imagen.**

A. 

( ) Modelo natural

B. 

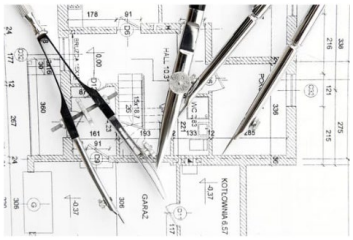
( ) Modelo digital

C. 

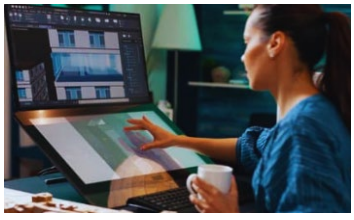
( ) Modelo analógico

D. 

( ) Modelo matemático

E. 

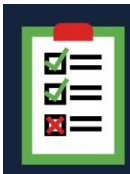
( ) Modelo artificial

F. 

( ) Modelo conceptual

2. Desarrolla un modelo para realizar la división de dos números, si el divisor es cero deberá de enviar un mensaje de error, en caso contrario mostrará el resultado. Identifica entrada(s), proceso y salida(s).

Entrada	
Proceso	
Salida	



AUTOEVALUACIÓN UNIDAD I

1. Define el concepto de cibernética

2. Resume brevemente los antecedentes históricos de la cibernética

3. Describe la relación de la cibernética con otras ciencias mencionadas en el texto

_____

_____

_____

_____

_____

_____

_____

_____

4. Menciona algunas aplicaciones actuales de la cibernética

_____

_____

_____

_____

_____

5. ¿Cuáles son los componentes básicos de un sistema?

_____

_____

6. Define qué es un sistema abierto y un sistema cerrado

_____

_____

_____

7. Explica el concepto de retroalimentación

_____

_____

_____

8. Explica qué es un modelo y cuál es su propósito

_____

_____

_____

_____

9. Describe brevemente los tipos de modelos mencionados en el texto

_____

_____

_____

_____

_____

10. Explica por qué es importante el modelado de sistemas

_____

_____

_____

_____

_____

SOLUCIONES UNIDAD I

SECCIÓN 1.1

1. I b); II c); III c); IV a); V b); VI d); VII d)

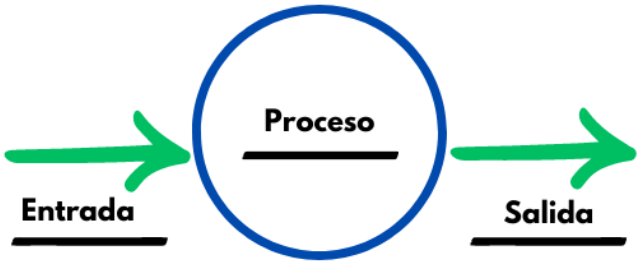
2.

						[1]A							
			[2]N	E	U	R	O	C	I	E	N	C	I	A
					T									
			[3]B	I	O	I	N	G	E	N	I	E	R	Í	A
					M									
				[4]M	A	T	E	M	Á	T	I	C	A	S
					T									
				[5]C	I	B	E	R	N	É	T	I	C	A
					Z									
[6]C	O	M	P	U	T	A	C	I	Ó	N				
						C						[7]F		
[8]N	E	U	[9]R	O	F	I	S	I	O	L	O	G	Í	A
			O			Ó							S	
	[10]M		[11]B	I	Ó	N	I	C	A				[12]I	A
	I		Ó										C	
	N		T										A	
[13]M	E	D	I	C	I	N	A							
	R		C											
	Í		A											
	A													

3.
A. W. Harvey
B. Blaise Pascal
C. Gotfried Wilhem Leibniz
D. Christiaan Huygens
E. René Descartes
F. J. Vaucanson
G. George Boole
H. Norbert Wiener
I. Arturo Rosenblueth Stearns
J. Claude Elwood Shannon
K. Alan Mathison Turing

SECCIÓN 1.2

1.



2.
A. Sistemas abiertos
B. Sistemas cerrados
C. Sistemas físicos o concretos
D. Sistemas abstractos
E. Sistema aislado

6. I c) ; II b) ; III d); IV c); V a)

SECCIÓN 1.3

1.
A. Modelo artificial
B. Modelo matemático
C. Modelo conceptual
D. Modelo natural
E. Modelo analógico
F. Modelo digital

2.

Entrada	• Dividendo • Divisor
Proceso	Se verifica si el divisor es diferente de cero, si es cero se envía un mensaje de error, en caso contrario se realiza la división
Salida	• Resultado de la división • Mensaje de error

Unidad II. Circuitos Lógicos

Propósito:

Al finalizar la unidad el alumno:
Utilizará el álgebra de Boole y el sistema de numeración binario para diseñar, construir o simular circuitos lógicos utilizando un protoboard o un simulador.

Contenido

- Sección 2.1 Sistemas de Numeración
- Sección 2.2 Aritmética del Sistema de Numeración Binario
- Sección 2.3 Funciones Booleanas
- Sección 2.4 Simplificación de Funciones
- Sección 2.5 Circuitos Lógicos

Sistema de Numeración: Conjunto de símbolos y reglas que permiten representar datos numéricos.

Sistema Decimal: Se compone de diez símbolos o dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9). El valor de cada dígito está asociado al de una potencia de base 10.

Sistema Binario: Utiliza sólo dos dígitos, el cero (0) y el uno (1). El valor de cada posición es el de una potencia de base 2.

Sistema Octal: Se representan mediante la combinación de ocho dígitos diferentes: 0, 1, 2, 3, 4, 5, 6 y 7. El valor de cada posición es el de una potencia de base 8.

Sistema Hexadecimal: Se representan con dieciséis dígitos: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F. Se utilizan los caracteres A, B, C, D, E y F representando las cantidades decimales 10, 11, 12, 13, 14 y 15 respectivamente. El valor de cada posición es el de una potencia de base 16.

Álgebra de Boole: Es una rama especial del álgebra que se usa principalmente en electrónica digital, sirve para simplificar expresiones o circuitos lógicos.

Tabla de Verdad: Es una tabla que muestra el valor de una proposición compuesta para cada composición de verdad que se pueda asignar.

Función Lógica (Booleana): Es toda variable binaria cuyo valor depende de una expresión formada por otras variables binarias relacionadas mediante los signos **+** y *****.

Circuito Lógico: Es aquel que maneja la información binaria en forma de "**1**" y "**0**", dos niveles lógicos de voltaje fijos. "1" nivel alto o "high" y "0" nivel bajo o "low".

Compuertas Lógicas: Son circuitos electrónicos diseñados para obtener resultados booleanos (0,1), los cuales se obtienen de operaciones lógicas binarias (suma, multiplicación). Dichas compuertas son AND, OR, NOT, NAND, NOR, XOR, XNOR.

Sección 2.1 Sistemas de Numeración

Aprendizaje:

1. Convierte números entre los sistemas de numeración binario, octal, decimal y hexadecimal.

Temática

Sistemas de numeración.

- Binario, octal, decimal y hexadecimal.
- Conversiones numéricas entre los sistemas: binario, octal, decimal y hexadecimal.

SISTEMA DE NUMERACIÓN

Un sistema de numeración es un conjunto de símbolos y reglas que permiten representar datos numéricos.

SISTEMA DECIMAL

El sistema de numeración que utilizamos es el **decimal,** que se compone de diez símbolos o dígitos (0, 1, 2, 3, 4, 5, 6, 7, 8 y 9) que tienen un valor dependiendo de la posición que ocupen en la cifra: unidades, decenas, centenas, millares, etc.

El valor de cada dígito está asociado al de una potencia de base 10, número que coincide con la cantidad de dígitos que integran el sistema decimal, y un exponente igual a la posición que ocupa el dígito menos uno, contando desde la derecha.

Ejemplo 1. En el sistema decimal el número **931**, significa:

9 centenas + 3 decenas + 1 unidad, es decir: 900 + 30 + 1 = 931

SISTEMA BINARIO

El sistema de numeración binario utiliza sólo **dos dígitos**, el **cero** (**0**) y el **uno** (**1**).

Por lo que los números de este sistema serán una combinación de estos dos dígitos, en una cifra binaria, cada dígito tiene distinto valor dependiendo de la posición que ocupe. El valor de cada posición es el de una potencia de **base 2**, por ser solo dos los dígitos que integran este sistema.

CONVERSIÓN DEL SISTEMA DECIMAL AL SISTEMA BINARIO

Para convertir al sistema binario un número del sistema decimal, realizaremos una serie de divisiones que tendrán por cociente al número **2** (que es la base del sistema binario, al que vamos a convertir el número), terminando cuando obtengamos por cociente el número 0. El número binario se obtendrá tomando los residuos en orden inverso

Ejemplo 2. Convertir al sistema binario el número $\mathbf{77_{10}}$

77 / 2 = 38 Residuo: **1**

38 / 2 = 19 Residuo: **0**

19 / 2 = 9 Residuo: **1**

9 / 2 = 4 Residuo: **1**

4 / 2 = 2 Residuo: **0**

2 / 2 = 1 Residuo: **0**

1 / 2 = 0 Residuo: **1**

orden inverso

el número binario se obtendrá tomando los residuos en orden inverso, por lo que obtenemos la cifra binaria: $\mathbf{77_{10} = 1001101_2}$

CONVERSIÓN DEL SISTEMA BINARIO AL SISTEMA DECIMAL

Supongamos que tenemos el número binario $\mathbf{1011_2}$ y queremos conocer el valor decimal, que se calcula realizando una sumatoria de los resultados de cada multiplicación de cada uno de los dígitos binarios por una potencia con base 2 y cuyo exponente dependerá del orden de cada digito, ya que se acomodaran en secuencia empezando por el 0 y terminando de acuerdo con la posición del último digito, así:

1	0	1	1
2^3	2^2	2^1	2^0
8	4	2	1

es decir: $\mathbf{1*2^3 + 0*2^2 + 1*2^1 + 1*2^0 =}$

$\mathbf{8 + 0 + 2 + 1 = 11}$

y para expresar que ambas cifras describen la misma cantidad lo escribimos así:

$$\mathbf{1011_2 = 11_{10}}$$

SISTEMA OCTAL

En el sistema de numeración octal, los números se representan mediante la combinación de ocho dígitos diferentes: **0**, **1**, **2**, **3**, **4**, **5**, **6** y **7**.

Cada dígito tiene, naturalmente, un valor distinto dependiendo del lugar que ocupen.

El valor de cada una de las posiciones viene determinado por cada una de las potencias de **base 8**.

CONVERSIÓN DEL SISTEMA DECIMAL AL SISTEMA OCTAL

La conversión de un número decimal a octal se hace con la misma técnica que ya hemos utilizado en la conversión a binario, mediante divisiones sucesivas cuyo divisor será 8 y colocando los residuos obtenidos en orden inverso.

Ejemplo 3. Escribir en octal el número decimal $\mathbf{122_{10}}$

Primero se deben hacer las siguientes divisiones:

122 / 8 = 15 Residuo: **2**

1 / 8 = 1 Residuo: **7**

1 / 8 = 0 Residuo: **1**

orden inverso

Tomando los residuos obtenidos en orden inverso tendremos la cifra octal:

$$\mathbf{122_{10} = 172_8}$$

CONVERSIÓN DEL SISTEMA OCTAL AL SISTEMA DECIMAL

La conversión de un número octal a decimal es igualmente sencilla, conociendo el peso de cada posición en una cifra octal.

Ejemplo 4. Convertir el número 237_8 a decimal basta con desarrollar el valor de cada dígito, multiplicándolo por su potencia correspondiente y al final sumando los resultados de dichas operaciones.

2	3	7
8^2	8^1	8^0
64	8	1

$$2*8^2 + 3*8^1 + 7*8^0 =$$

$$128 + 24 + 7 = 159_{10}$$

y para expresar que ambas cifras describen la misma cantidad lo escribimos así:

$$237_8 = 159_{10}$$

SISTEMA HEXADECIMAL

En el sistema hexadecimal los números se representan con dieciséis dígitos: **0**, **1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9**, **A**, **B**, **C**, **D**, **E** y **F**. Se utilizan los caracteres **A**, **B**, **C**, **D**, **E** y **F** representando las cantidades decimales **10**, **11**, **12**, **13**, **14** y **15** respectivamente.

Dígitos Hexadecimales																
Dígito	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Valor	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

El valor de cada uno de estos símbolos depende, como es lógico, de su posición, que se calcula mediante potencias de **base 16**.

CONVERSIÓN DEL SISTEMA DECIMAL AL SISTEMA HEXADECIMAL

La conversión de un número decimal a hexadecimal lleva el mismo procedimiento en donde se realizan divisiones cuyo divisor será 8 y retomando los residuos para formar el número hexadecimal.

Ejemplo 5. Convertir a hexadecimal del número $\mathbf{1735_{10}}$

Primero será necesario hacer las siguientes divisiones:

1735 / 16 = 108 Residuo: **7**

108 / 16 = 6 Residuo: **12** **lo cambiamos por el dígito hexadecimal equivalente C**

6 / 16 = 0 Residuo: **6**

orden inverso

De ahí que, tomando los restos en orden inverso, obtenemos el número en hexadecimal:

$$\mathbf{1735_{10} = 6C7_{16}}$$

CONVERSIÓN DEL SISTEMA HEXADECIMAL AL SISTEMA DECIMAL

Para realizar esta conversión retomaremos el proceso de la sumatoria de los resultados de la multiplicación por potencias en base 16.

Ejemplo 6. Convertir a decimal el siguiente número hexadecimal $\mathbf{1A3F_{16}}$:

1	A (10)	3	F(15)
16^3	16^2	16^1	16^0
4,096	256	16	1

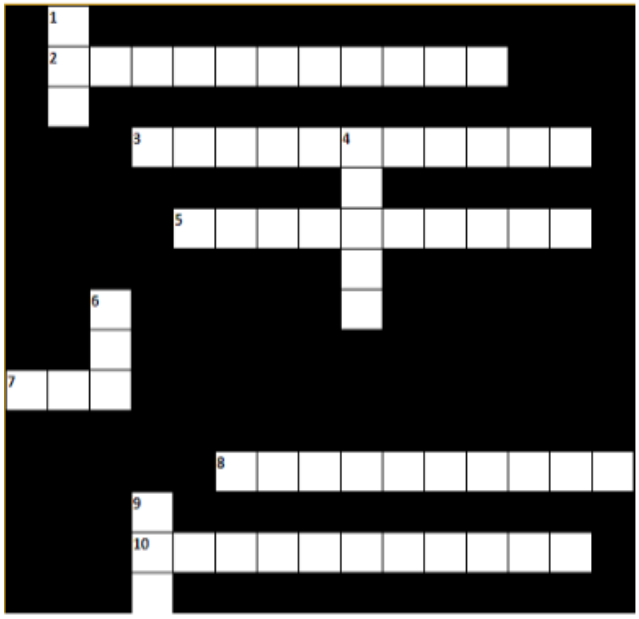
$$\mathbf{1A3F_{16} = 1*16^3 + A(10)*16^2 + 3*16^1 + F(15)*16^0}$$

$$\mathbf{1*4096 + 10*256 + 3*16 + 15*1 = 6719}$$

$$\mathbf{1A3F_{16} = 6719_{10}}$$

ACTIVIDADES DE APRENDIZAJE
SECCIÓN 2.1

1. Realiza los siguientes ejercicios, resolviendo las conversiones del sistema binario para contestar el crucigrama.



VERTICALES

1. Resultado de la conversión del número 1000000000 a decimal
4. Resultado de la conversión del número 10011100101000 a decimal
6. Resultado de la conversión del número 1101000111 a decimal
9. Resultado de la conversión del número 1001100110 a decimal

HORIZONTALES

2. Resultado de la conversión del número 1462 a binario
3. Resultado de la conversión del número 1256 a binario
5. Resultado de la conversión del número 980 a binario
7. Resultado de la conversión del número 1100000001 a decimal
8. Resultado de la conversión del número 756 a binario
10. Resultado de la conversión del número 1313 a binario

2. **Resuelve los ejercicios de conversiones en el sistema octal.**

   a) 2847_{10} a ?$_{8}$

   b) 159_{10} a ?$_{8}$

   c) 75_{10} a ?$_{8}$

   d) 104_{8} a ?$_{10}$

   e) 7412_{8} a ?$_{10}$

   f) 6352_{8} a ?$_{10}$

3. **Resuelve los ejercicios de conversiones en el sistema hexadecimal.**

   a) 9582_{10} a ?$_{16}$

   b) 784_{10} a ?$_{16}$

   c) 35_{10} a ?$_{16}$

   d) $1B4_{16}$ a ?$_{10}$

   e) $4A1F_{16}$ a ?$_{10}$

   f) $735C_{16}$ a ?$_{10}$

Sección 2.2 Aritmética del Sistema de Numeración Binario

Aprendizaje:

2. Realiza operaciones aritméticas con el sistema de numeración binario.

Temática

Aritmética del sistema de numeración binario.

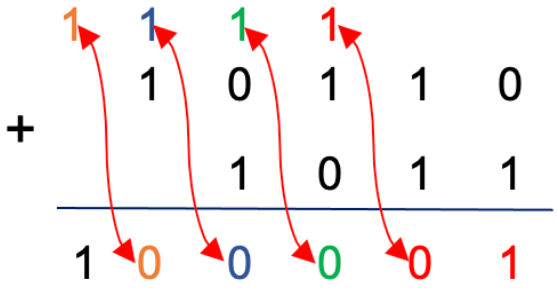
- Operaciones de adición, sustracción, multiplicación y división.

SUMA DE NÚMEROS BINARIOS

* Reglas *
0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 1 0

Para resolver la suma de números binarios, se realiza el mismo procedimiento que en la suma decimal, considerando para ello las reglas de suma entre binarios. Se suma columna a columna, y en caso de aplicarse la suma 1 + 1 = 10 anotaremos el 0 y colocamos el 1 en la siguiente columna para sumarse con los siguientes dígitos.

Ejemplo 1: 10110 + 1011

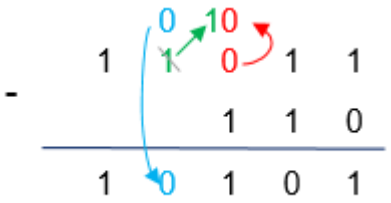


RESTA DE NÚMEROS BINARIOS

* Reglas *
0 - 0 = 0
0 - 1 = acarreo
1 - 0 = 1
1 - 1 = 0
10 - 1 = 1

Para resolver la resta de números binarios, se realiza el mismo procedimiento que en la resta decimal, considerando para ello las reglas de resta entre binarios. Se resta columna a columna, y en caso de necesitarse se pide prestado al siguiente digito.

Ejemplo 2. 11011 – 110

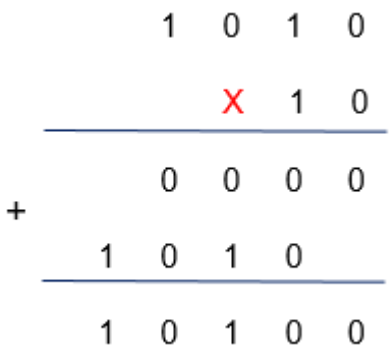


MULTIPLICACIÓN DE NÚMEROS BINARIOS

* Reglas *
0 * 0 = 0
0 * 1 = 0
1 * 0 = 0
1 * 1 = 1

Para resolver la multiplicación de números binarios, se realiza el mismo procedimiento que en la multiplicación decimal, considerando para ello las reglas de multiplicación entre binarios y posteriormente las de la suma binaria.

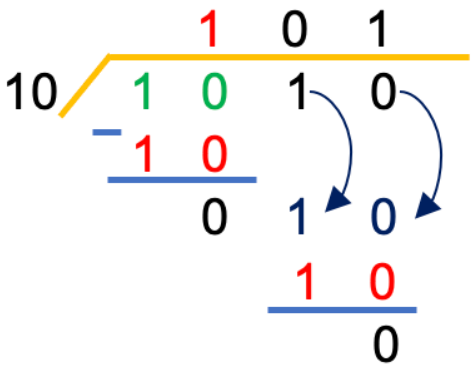
Ejemplo 3. 1010 X 10



DIVISIÓN DE NÚMEROS BINARIOS

La división binaria es similar a la división de números decimales, la única diferencia es que, al hacer las restas, dentro de la división estas se hacen en sistema binario.

Ejemplo 4. $1010_2 / 10_2$



- Se multiplica **1** * **10** = **10** se coloca debajo del **dividendo** y se hace la resta, se baja el siguiente digito **1, 1** es menor que **10**, se baja entonces el siguiente digito **0.**
- Se multiplica **1** * **10** = **10** se coloca debajo del **dividendo** y se hace la resta.
- En este caso el residuo es 0.

Por lo que el resultado de $1010_2 / 10_2$ es **101**

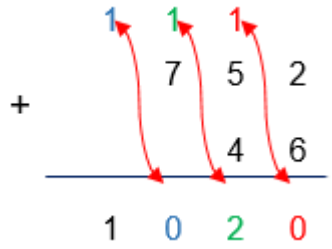
ARITMÉTICA OCTAL

SUMA DE NÚMEROS OCTALES

+	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	1	2	3	4	5	6	7	10
2	2	3	4	5	6	7	10	11
3	3	4	5	6	7	10	11	12
4	4	5	6	7	10	11	12	13
5	5	6	7	10	11	12	13	14
6	6	7	10	11	12	13	14	15
7	7	10	11	12	13	14	15	16

Para resolver la suma de números octales, se realiza el mismo procedimiento que en la suma octal, considerando para ello la tabla de suma entre octales. Se suma columna a columna, y en caso de obtener un número de 2 dígitos se coloca el primero y el segundo se agrega a la siguiente columna.

Ejemplo 5. $752_8 + 46_8$

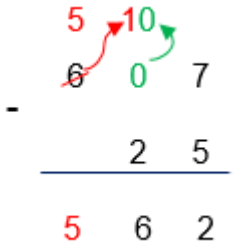


RESTA DE NÚMEROS OCTALES

Para la realización de la resta, se resta columna por columna de derecha a izquierda, considerando el acarreo negativo (cuando se pide prestado), si es necesario.

Recuerda que $10_8 = 8_{10}$ por lo tanto $10_8 - 1_8 = 7_8$

Ejemplo 6 . $607_8 - 25_8$

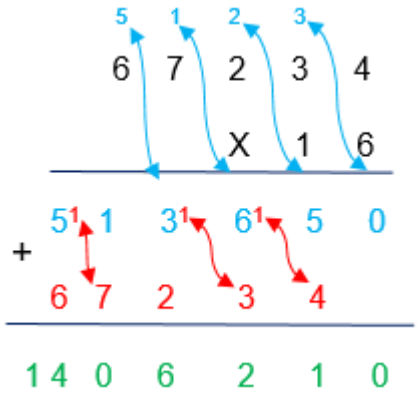


MULTIPLICACIÓN DE NÚMEROS OCTALES

*	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6	7
2	0	2	4	6	10	12	14	16
3	0	3	6	11	14	17	22	25
4	0	4	10	14	20	24	30	34
5	0	5	12	17	24	31	36	43
6	0	6	14	22	30	36	44	52
7	0	7	16	25	34	43	52	61

Para realizar la multiplicación octal, sola basta tomar en cuenta la tabla de multiplicación de valores octales y posteriormente la de la suma de valores octales.

Ejemplo 7. $67234_8 * 16_8$



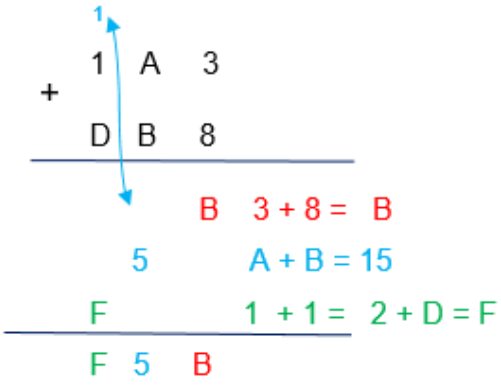
ARITMÉTICA HEXADECIMAL

SUMA DE NÚMEROS HEXADECIMALES

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14
6	6	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15
7	7	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16
8	8	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17
9	9	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18
A	A	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19
B	B	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A
C	C	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B
D	D	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C
E	E	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D
F	F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E

Para realizar la suma hexadecimal, sola basta tomar en cuenta la tabla de valores de suma hexadecimal.

Ejemplo 8. $1A3_{16} + DB8_{16}$

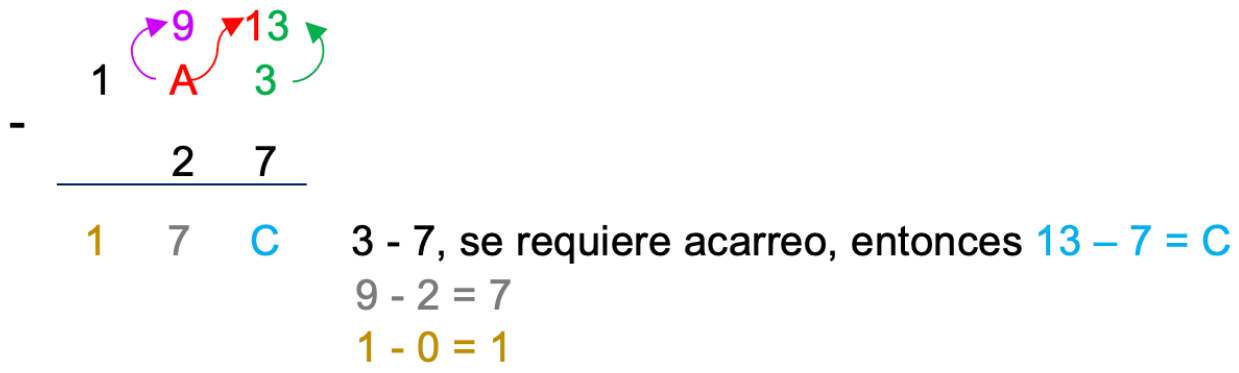


RESTA DE NÚMEROS HEXADECIMALES

Para realizar la resta de números hexadecimales, se debe restar columna por columna considerando el acarreo negativo, es decir, cuando se pide prestado, eso sí es necesario.

Recuerda que $10_{16} = 16_{10}$ por tanto $10_{16} – 1_{16} = F_{16}$

Ejemplo 9. $1A3_{16} - 27_{16}$



ACTIVIDADES DE APRENDIZAJE
SECCIÓN 2.2

1. Realiza los siguientes ejercicios

a) $10101_2 + 10111_2$

b) $100011_2 - 1100_2$

c) $101011_2 * 10_2$

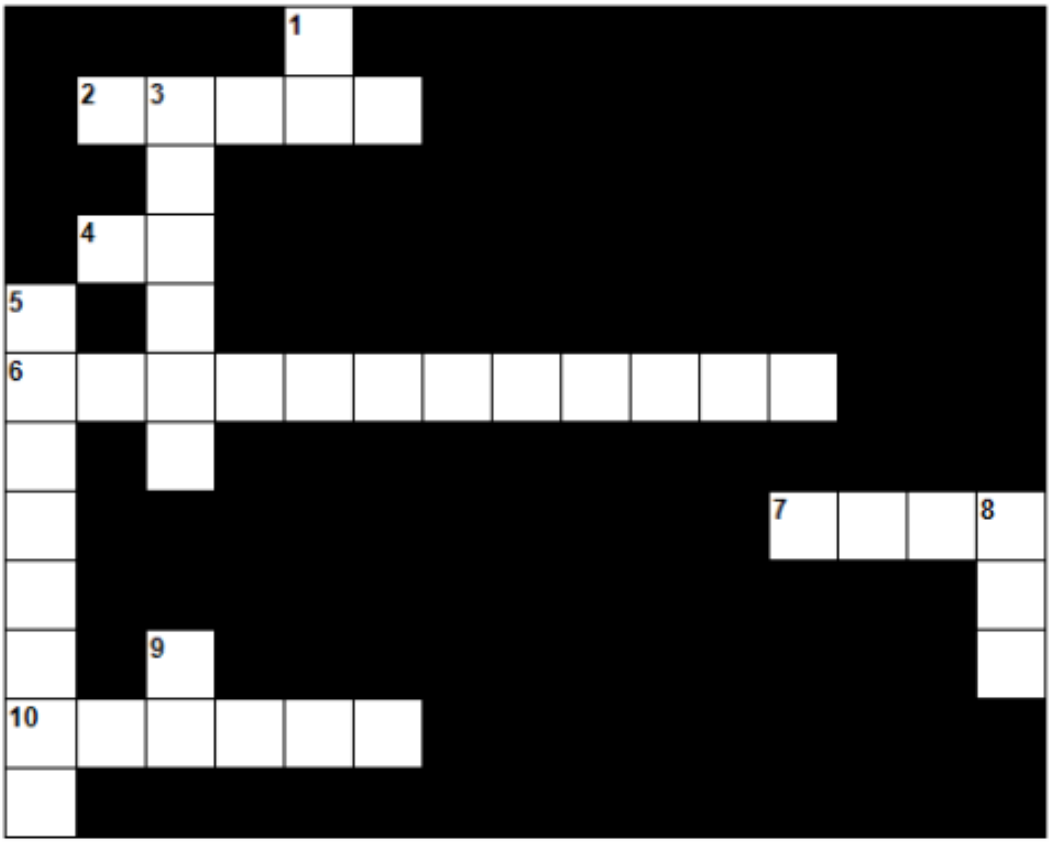
d) $10100_2 \ / \ 11_2$

e) $136_8 + 675_8$

f) $567_8 * 23_8$

g) $A6C_{16} + 125_{16}$

2. Realiza los siguientes ejercicios, resolviendo las conversiones y contesta el crucigrama.



VERTICALES:

1. Convierte el siguiente número binario 110101 a hexadecimal.
3. Convierte los siguientes números decimales a binario 25 y 13; a continuación realiza su adición y expresa el resultado en binario.
5. Realiza la siguiente multiplicación de números binarios 1111 x 1101.
8. Realiza la conversión del siguiente número decimal a hexadecimal 1691.
9. Es el resultado de la conversión del número binario 10101 a sistema decimal.

HORIZONTALES:

2. Resultado de la conversión del número A37E (que esta expresado en base 16) a número decimal.
4. Resultado de la multiplicación 8 * 7, expresado en sistema octal.
6. Si convertimos el número 5371 que se encuentra en sistema octal, que número nos da expresado en sistema binario.
7. Es el resultado de la conversión del número 37246 en sistema octal; a hexadecimal.
10. Resultado que se obtiene dela conversión del numero 42 decimal a número binario.

Sección 2.3 Funciones Booleanas

Aprendizaje:

3. Construye tablas de verdad de funciones booleanas.

Temática

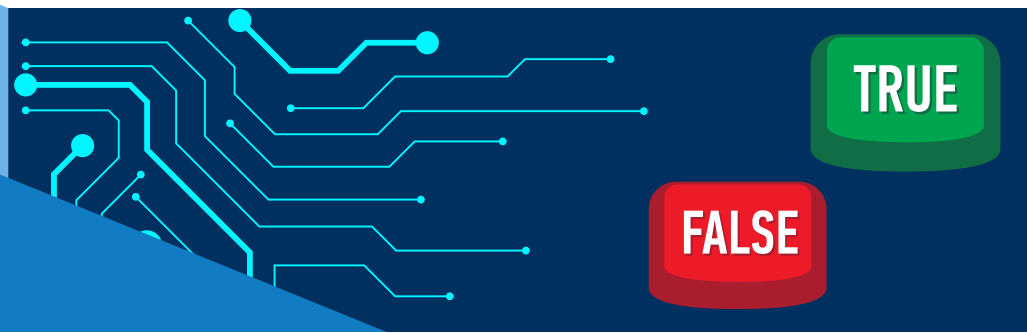
Elementos del álgebra de Boole:

- Variable booleana.
- Operaciones básicas: conjunción, disyunción y negación.

Función booleana:

- Concepto.
- Expresiones booleanas.

Tablas de verdad.



ALGEBRA DE BOOLE

El **álgebra de Boole**, es una rama especial del álgebra que se usa principalmente en electrónica digital; y una de sus aplicaciones es que sirve de método para simplificar expresiones o circuitos lógicos. El ÁLGEBRA BOOLEANA, LLAMADA ASÍ EN HONOR DE SU CREADOR, fue inventada en el año 1854 por el matemático inglés George Boole.

La lógica booleana solo permite dos estados del circuito, como **True** y **False**. Estos dos estados están representados por **1** y **0**, donde 1 representa el estado "**Verdadero**" y 0 representa el estado "**Falso**".

OPERACIONES BÁSICAS

El álgebra de Boole está definida por 3 operaciones básicas: la **suma** (**OR**), el **producto** (**AND**) y el **complemento** (**NOT**).

- La operación **Suma** u **OR** se representa como: F= A+B

A	B	F=A+B
0	0	0
0	1	1
1	0	1
1	1	1

- La operación **Producto** o **AND** se representa como: F= A * B

A	B	F=A*B
0	0	0
0	1	0
1	0	0
1	1	1

- La operación **Complemento** o **NOT**, que significa negación se representa como: F= A'

A	F=A'
0	1
1	0

TABLAS DE VERDAD

Una **tabla de verdad** es una tabla que muestra el valor de una proposición compuesta para cada composición de verdad que se pueda asignar.

Fue desarrollada por Charles Sanders Peirce por el año de 1880, pero el formato más utilizado lo introdujo Ludwing Wittgenstein.

El valor **Verdadero** se representa con **V** y la notación numérica **1**, mientras que el valor **Falso** se representa con **F** y notación numérica **0**.

Para desarrollar una tabla de verdad se utiliza la siguiente formula: $\mathbf{Nc = 2^n}$ donde **Nc** son el número de combinaciones posibles mientras que **n** son el número de variables.

	X_3	X_2	X_1	X_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1

	X_3	X_2	X_1	X_0
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

De acuerdo con lo anterior si tenemos 3 variables, A, B y C construyamos la tabla de verdad.

Nc = 2^3 por lo que Nc = 8 (combinaciones posibles)

C	B	A
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

FUNCIONES LÓGICAS

Se define Función Lógica (Booleana) a toda variable binaria cuyo valor depende de una expresión formada por otras variables binarias relacionadas mediante los signos **+** y *****.

Funciones básicas:

- Unión (**OR**), es decir **A + B**
- Intersección (**AND**), es decir **A * B**
- Negación (**NOT**), es decir **A'**

Ejemplo 1. Función booleana equivalente a la tabla:

A	B	C	F	
0	0	0	0	
0	**0**	**1**	**1**	← A'B'C
0	1	0	0	
0	1	1	0	
1	**0**	**0**	**1**	← AB'C'
1	0	1	0	
1	1	0	0	
1	**1**	**1**	**1**	← ABC

F= A'B'C + AB'C' + ABC

REPRESENTACIÓN DE FUNCIONES LÓGICAS

- **Minitérminos (Suma de productos)**

Para obtener la función de la tabla de verdad expresada en **minitérminos** se deberán colocar **sumas de los productos** de las entradas, aquellas en donde las salidas sean valores Verdaderos.

C	B	A	F(C,B,A)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

F (C,B,A) = C'B'A' + C'BA' + C'BA + CBA

⦿ Maxitérminos (Productos de sumas)

Para obtener la función de la tabla de verdad expresada en **maxitérminos** se deberán colocar **productos de sumas** de las entradas, aquellas en donde las salidas sean valores Verdaderos.

C	B	A	F(C,B,A)
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

F (C,B,A) = (C'+B'+A') * (C'+B+A') * (C'+B+A) * (C+B+A)

Ejemplo 2. Dada la siguiente función **F= (X + Y) (X Y'+Z')** obtener la tabla de verdad correspondiente

X	Y	Z	X+Y	Y'	X Y'	Z'	X Y'+Z'	F= (X + Y) (X Y'+Z')
0	0	0	0	1	0	1	1	**0**
0	0	1	0	1	0	0	0	**0**
0	1	0	1	0	0	1	1	**1**
0	1	1	1	0	0	0	0	**0**
1	0	0	1	1	1	1	1	**1**
1	0	1	1	1	1	0	1	**1**
1	1	0	1	0	0	1	1	**1**
1	1	1	1	0	0	0	0	**0**

1. Obtener la tabla de verdad de las siguientes expresiones

a) F = (AB'+C) (ABCD)

b) F = (AB) + (AB'C) (A') + (C') (A)

c) F = (XYZ) + (XY) + (Y'Z') (X'Y)

2. Dada la tabla de verdad, obtener su función expresada en minitérminos.

a)

X	Y	Z	F(XYZ)
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

b)

A	B	C	F(ABC)
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**1**
1	0	1	**1**
1	1	0	**1**
1	1	1	**1**

Sección 2.4 Simplificación de Funciones

Aprendizaje:

4. Simplifica funciones booleanas utilizando postulados y teoremas básicos.

Temática

Simplificación de funciones booleanas.

- Postulados.
- Teoremas básicos.

SIMPLIFICACIÓN DE FUNCIONES

Al formular expresiones matemáticas para circuitos lógicos es importante tener conocimiento del álgebra booleana, que define las reglas para expresar y simplificar enunciados lógicos binarios.

POSTULADOS		
Postulado 1	El elemento identidad de la suma es el "0".	(A + 0 = A)
Postulado 2	El elemento identidad de la suma es el "0".	(A · 1 = A)
Postulado 3	La suma es conmutativa	A + B = B + A
Postulado 4	El producto es conmutativo	A · B = B · A
Postulado 5	La suma es asociativa	(A + B) + C = A + (B + C)
Postulado 6	El producto es asociativo	(A · B) · C = A · (B · C)
Postulado 7	El producto es distributivo respecto de la suma	A · (B + C) = (A · B) + (A · C)
Postulado 8	La suma es distributiva respecto del producto	A + (B · C) = (A + B) · (A + C)
Postulado 9	Para cada valor A existe un valor $\bar{A}$	A· $\bar{A}$ = 0 A + $\bar{A}$ = 1
Postulado 10	El álgebra de Boole es cerrada bajo las operaciones suma, producto y negación.	

TEOREMAS	
Teorema 1	A + A = A
Teorema 2	A * A = A
Teorema 3	A * 0 = 0
Teorema 4	A + 1 = 1
Teorema 5	A + A * B = A
Teorema 6	A + A' * B = A + B
Teorema 7	A * (A + B) = A
Teorema 8	A * (A' + B) = A * B
Teorema 9	A' * (A + B') = A' * B'
Teorema 10	(A' + B') * (A' + B) = A'

LEYES DE DE MORGAN	
Ley 1	$\overline{A + B} = \overline{A} \cdot \overline{B}$
Ley 2	$\overline{A \cdot B} = \overline{A} + \overline{B}$

⇒ Al usar los teoremas y leyes booleanas, podemos simplificar las expresiones booleanas, mediante las cuales podemos reducir el número requerido de compuertas lógicas a implementar.

Ejemplo 1. Simplifica la siguiente expresión aplicando las leyes e identidades booleanas mencionadas:

F = (X · Y · Z) + (Y · Z) +(X · Y)

1. Aplicando la ley asociativa y la ley fundamental de que A · 1 = A

F = X · (Y · Z) + 1 · (Y · Z) + (X · Y)

2. Ahora es posible factorizar el término (Y · Z):

F = (X + 1) · (Y · Z) + (X · Y)

3. Dado que A + 1 = 1, por lo tanto, X + 1 = 1:

F = 1 · (Y · Z) + (X · Y)

4. Al realizar la operación tendremos ya simplificada la expresión:

F = (Y · Z) + (X · Y)

5. Aún podemos simplificar la expresión al factorizar Y:

F = Y · (Z + X)

1. Obtener la simplificación de funciones de las siguientes expresiones.

a) F = A’B’C’ + A B’C’ + ABC

b) F = CD + AB’C + ABC’ + BCD

c) F = A’B’C’ + A’BC’ + ABC’ + A B’C’

Sección 2.5 Circuitos Lógicos

Aprendizaje:

5. Describe los conceptos de interruptor, circuito eléctrico, compuerta lógica y circuito lógico.
6. Aprende a utilizar el protoboard o un simulador.
7. Construye la función booleana a partir de la tabla de verdad, empleando suma de productos.
8. Construye un semisumador
9. Construye un sumador completo
10. Diseña circuitos lógicos a partir de un problema cotidiano usando la metodología aprendida.

Temática

Interruptor y circuito eléctrico.
Compuertas y circuitos lógicos:

- Compuerta: And, Or y Not.

Uso del protoboard o un simulador.
Obtención de la función booleana

- Suma de productos

Semisumador

- Planteamiento del problema (caja negra)
- Obtención de la tabla de verdad
- Obtención de la función booleana
- Simplificación
- Construcción o simulación del circuito lógico

Sumador completo

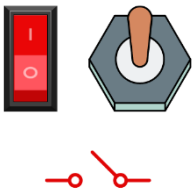
- Planteamiento del problema (caja negra)
- Obtención de la tabla de verdad
- Obtención de la función booleana
- Simplificación
- Diagrama del diseño lógico
- Construcción o simulación del circuito lógico

Construcción del circuito lógico

CIRCUITOS LÓGICOS

Un Circuito Lógico es aquel que maneja la información binaria en forma de "**1**" y "**0**", dos niveles lógicos de voltaje fijos. "1" nivel alto o "high" y "0" nivel bajo o "low".

Un **interruptor** es un dispositivo eléctrico que nos permite realizar una función de encendido y apagado. Su funcionamiento consiste en dejar pasar o no la corriente en un circuito eléctrico.



COMPUERTAS LÓGICAS

Las compuertas lógicas son circuitos electrónicos diseñados para obtener resultados booleanos (0,1), los cuales se obtienen de operaciones lógicas binarias (suma, multiplicación). Dichas compuertas son AND, OR, NOT, NAND, NOR, XOR, XNOR. Este tipo de dispositivos lógicos se encuentran implementados con transistores y diodos en un semiconductor y actualmente podemos encontrarlas en formas de circuitos integrados lógicos.

COMPUERTA AND

Para la compuerta AND, la salida estará en estado alto de tal manera que solo si las dos entradas se encuentran en estado alto. Por esta razón podemos considerar que es una multiplicación binaria.

- **Operación**

  Q = A * B

- **Tabla de verdad y símbolo**

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

COMPUERTA OR

La compuerta OR, la salida estará en estado alto cuando cualquier entrada o ambas estén en estado alto. De tal manera que sea una suma lógica.

- **Operación**

$$Q = A + B$$

- **Tabla de verdad y símbolo**

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

COMPUERTA NOT

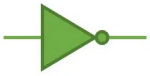
En la compuerta NOT, el estado de la salida es inversa a la entrada. Evidentemente, una negación.

- **Representación**

$$Q = Q'$$

- **Tabla de verdad y símbolo**

Q	Q´
0	1
1	0



IMPLEMENTACIÓN DE FUNCIONES

- Una vez teniendo una función, pasamos a la implementación utilizando las compuertas lógicas, por lo que vamos a cambiar las expresiones por compuertas.

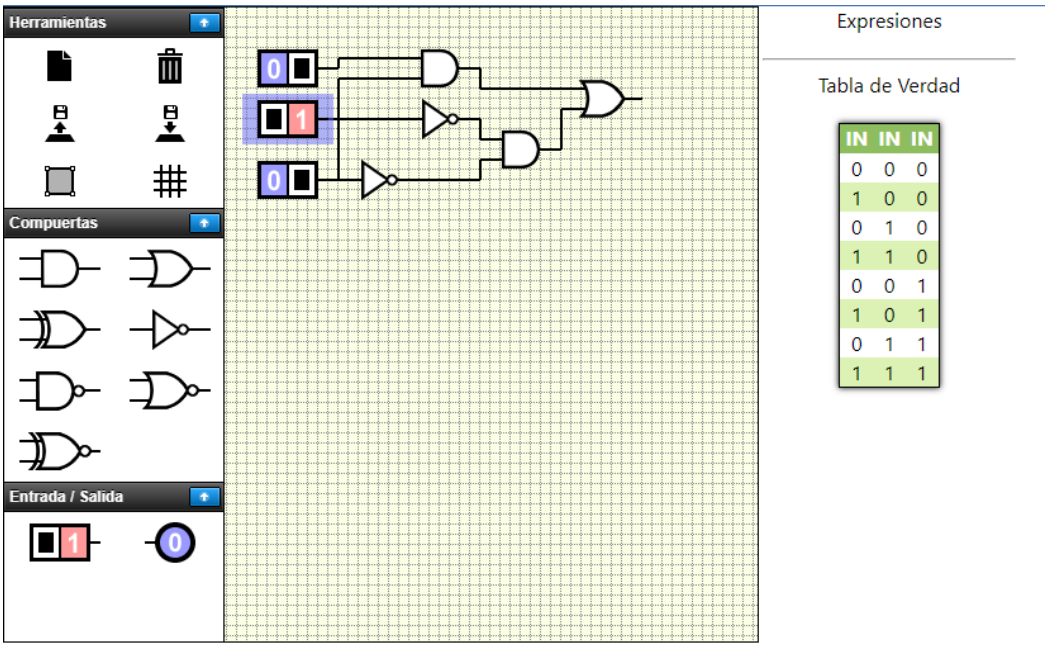
- Una herramienta fácil de utilizar es un simulador, por ejemplo, un simulador en línea:

  http://163.10.22.82/OAS/compuertas_logicas/Simulacion/editor_simple.html

Ejemplo 1. Dada la función B'C' + AC realizar su implementación mediante compuertas.

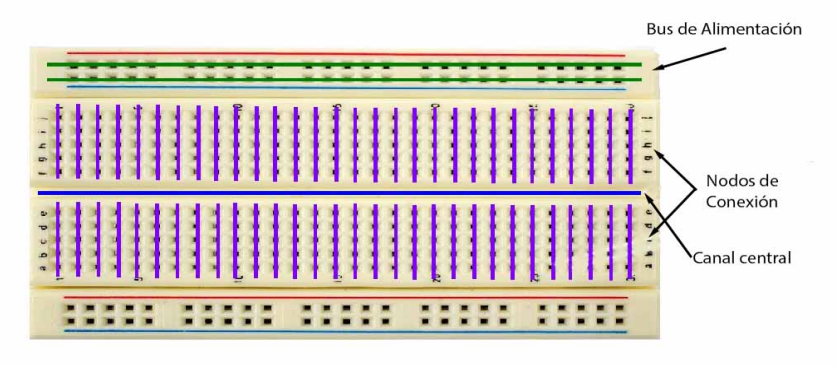
Analizaremos la función para ver como la implementaremos.

⇒ 3 variables A, B, C
⇒ B' y C' por lo que pasan por la compuerta NOT
⇒ B'C' representan una multiplicación, compuerta AND
⇒ AC representan una multiplicación, compuerta AND
⇒ Y el resultado de ambas multiplicaciones se suman, compuerta OR



USO DE EL PROTOBOARD

El Protoboard es una placa de pruebas para circuitos electrónicos. En ella se pueden armar circuitos de prueba previo al montado en PCB.



Hoy en día también se puede utilizar un simulador en línea para implementar circuitos en una protoboard, por ejemplo: https://www.circuito.io/ o https://www.tinkercad.com/

SEMISUMADOR (MEDIO SUMADOR)

Diseñar un circuito sumador de dos números binarios de un solo digito.

PLANTEAMIENTO DEL PROBLEMA

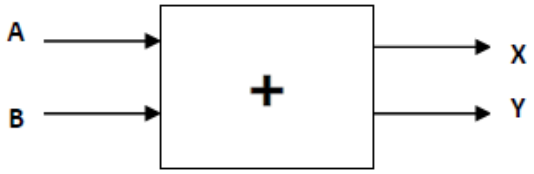
- La suma de dos números binarios de un digito se puede expresar así:

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 10

- Para plasmar los resultados en dos dígitos, agregamos un 0 a la izquierda de los primeros 3 resultados, que no afecta el valor.

0 + 0 = 00
0 + 1 = 01
1 + 0 = 01
1 + 1 = 10

PLANTEAMIENTO DEL PROBLEMA COMO UNA CAJA NEGRA



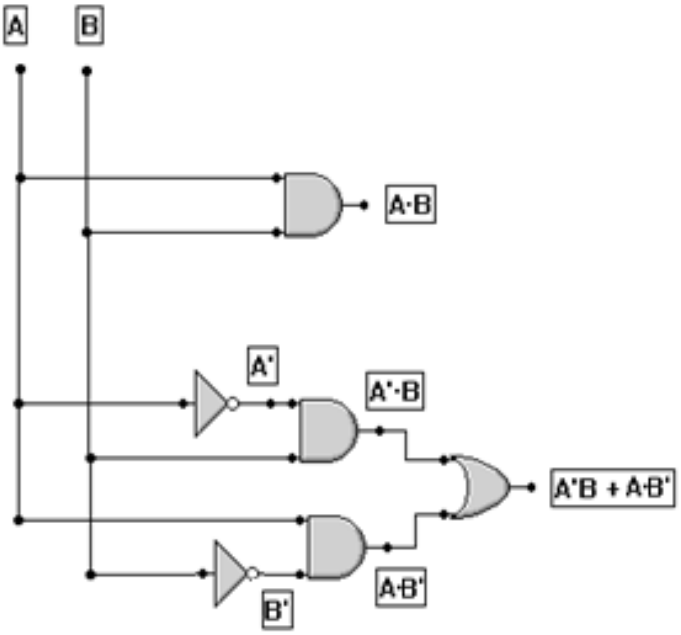
- Obtenemos la tabla de verdad

A	B	X	Y
0	**0**	**0**	**0**
0	**1**	**0**	**1**
1	**0**	**0**	**1**
1	**1**	**1**	**0**

- Obtenemos la función de la tabla de verdad, para X y para Y

  X = AB
  Y = A'B + AB'

- Implementamos las funciones con compuertas



SUMADOR COMPLETO

Diseñar un circuito sumador de tres números binarios.

PLANTEAMIENTO DEL PROBLEMA

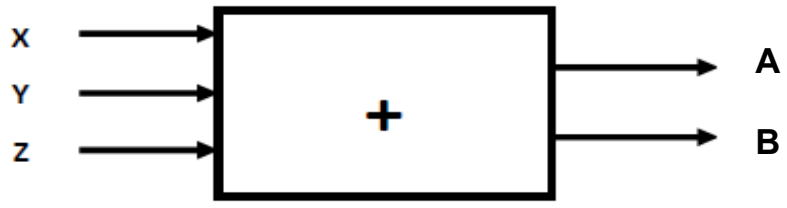
- La suma de tres números binarios se puede expresar así:

0 + 0 + 0 =	0
0 + 0 + 1 =	1
0 + 1 + 0 =	1
0 + 1 + 1 =	10
1 + 0 + 0 =	1
1 + 0 + 1 =	10
1 + 1 + 0 =	10
1 + 1 + 1 =	11

- Para plasmar los resultados en dos dígitos, agregamos un 0 a la izquierda, que no afecta el valor.

0 + 0 + 0 =	00
0 + 0 + 1 =	01
0 + 1 + 0 =	01
0 + 1 + 1 =	10
1 + 0 + 0 =	01
1 + 0 + 1 =	10
1 + 1 + 0 =	10
1 + 1 + 1 =	11

PLANTEAMIENTO DEL PROBLEMA COMO UNA CAJA NEGRA



- Obtenemos la tabla de verdad

X	Y	Z	A	B
0	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**
0	**1**	**0**	**0**	**1**
0	**1**	**1**	**1**	**0**
1	**0**	**0**	**0**	**1**
1	**0**	**1**	**1**	**0**
1	**1**	**0**	**1**	**0**
1	**1**	**1**	**1**	**1**

- Obtenemos la función de la tabla de verdad, para A y para B

A = X'YZ + XY'Z + XYZ' + XYZ
B = X'Y'Z + X'YZ' + XY'Z' + XYZ

- Simplificamos las funciones

A = X'YZ + XY'Z + XYZ' + XYZ

X'YZ + X(Y'Z+YZ' + YZ)

X'YZ + X(Y'Z + Y(Z' + Z))

X'YZ + X(Y'Z + Y(1))

X'YZ + X(Y'Z + Y)

X'YZ + X(Y + Z)

X'YZ + XY + XZ

Y(X + Z) + XZ

YX + YZ + XZ

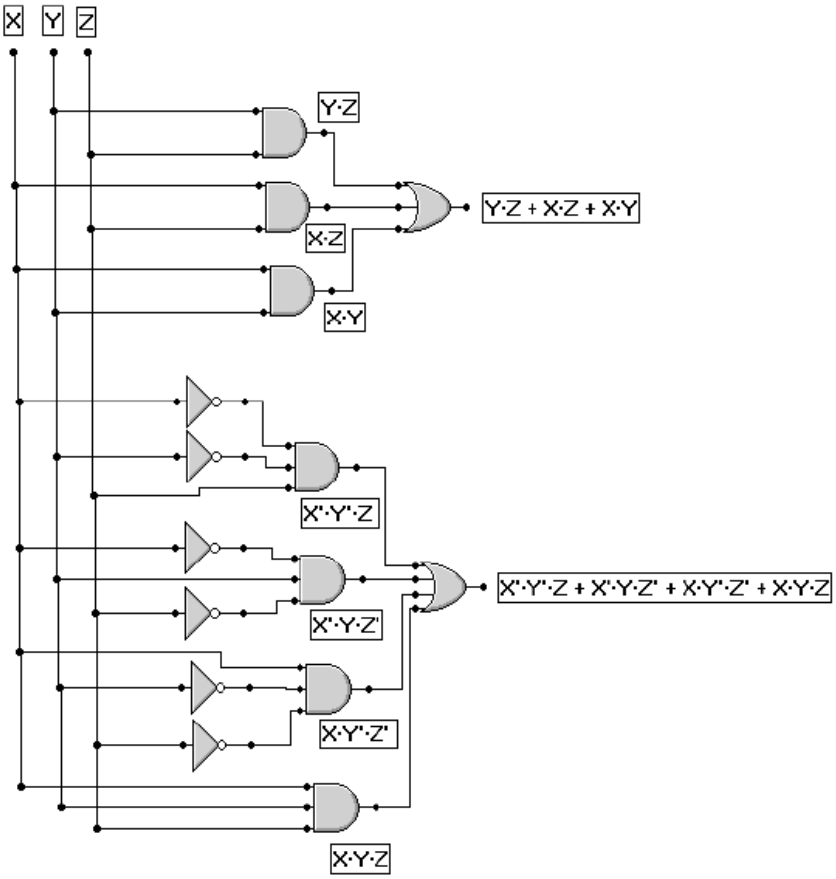
A = YX + YZ + XZ

B = X'Y'Z + X'YZ' + XY'Z' + XYZ

Esta expresión no se puede simplificar por el álgebra de Boole, por tanto, se queda igual.

B = X'Y'Z + X'YZ' + XY'Z' + XYZ

- Implementamos las funciones con compuertas



Ejemplo 2. Resolver el siguiente problema.

Diseñar el circuito lógico del problema, y empleando compuertas lógicas realiza la implementación de la siguiente maquina: Se sabe que una maquina mezcladora de leche, que utiliza 3 insumos “Leche en polvo”, “Agua” y “Grasa vegetal”.

La máquina produce leche, bajo los siguientes supuestos:

- En las entradas están presentes todos los insumos
- En la salida deben de estar presentes “Leche en polvo” y “Agua”

Planteamiento del problema

ENTRADAS	SALIDAS
L = Leche en polvo	
A = Agua	ML = Mezcla de leche
G = Grasa vegetal	

Planteamiento del problema como una caja negra

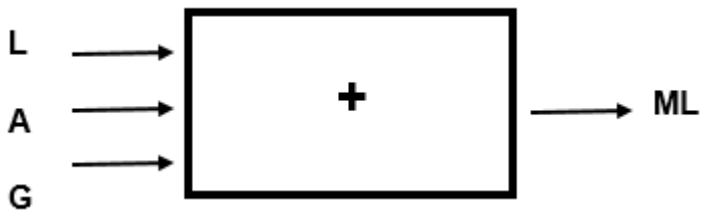


Tabla de verdad y función

L	A	G	ML
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

ML = LG'A + LGA

Implementación con compuertas

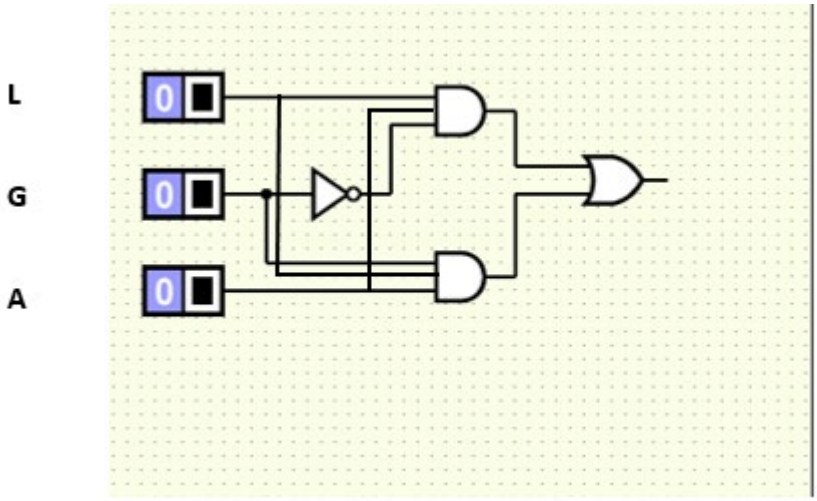


Tabla de Verdad

IN	IN	IN
0	0	0
1	0	0
0	1	0
1	1	0
0	0	1
1	0	1
0	1	1
1	1	1

1. Realiza la implementación con compuertas de las siguientes funciones. (Puedes emplear el simulador en línea).

a) F = A'B'C' D+ A'C' + ACD'

b) F = CD + AB'C + A'BC' + BCD'

c) F = A'B'C' + A'BC' + ABC' + A B'C'

2. Realice los siguientes diseños de circuitos:

a) Diseñe un circuito con tres entradas, que indique cuando el número total de unos (1) a la entrada es un número par.

b) Diseñar un circuito de una máquina de helados, con 4 entradas: Leche, Agua, Hielo y Saborizante.

La máquina funciona cuando tiene los siguientes insumos:

- Hielo y saborizante
- Leche, hielo y saborizante
- Agua, hielo y saborizante
- Leche, agua, hielo y saborizante

1. Realiza la conversión del número 385_{10} a número binario

2. Realiza la conversión del número 10101000011_2 a número decimal

3. Realiza la suma de: 1011110 + 111010 + 1100111

4. Resuelve la siguiente multiplicación 1100111 * 11110

5. Convierte del sistema octal al sistema decimal 52_8

6. Convierte del sistema hexadecimal a sistema decimal $AB52F_{16}$

7. Convierte del sistema decimal a sistema hexadecimal 291031_{10}

8. Realiza la tabla de verdad de la siguiente función:

$$F= [(XYZ) + (YZ' + [(XY') ']] + [(XY'Z)']$$

9. Reduce la siguiente expresión:

$$F= AB'C + A'BC + A'B'C + A'B'C' + AB'C'$$

10. Realiza su implementación con compuertas ABC + A'C' + AB'C'

SECCIÓN 2.1

1. Crucigrama

1. 5121
2. 10110110110
3. 10011101000
4. 10024
5. 1111010100
6. 839
7. 769
8. 1011110100
9. 614
10. 10100100001

2.

a) 5437_8
b) 237_8
c) 113_8
d) 68_{10}
e) 3850_{10}
f) 3306_{10}

3.

a) $256E_{16}$
b) 310_{16}
c) 23_{16}
d) 436_{10}
e) 18975_{10}
f) $735C_{10}$

SECCIÓN 2.2

1.

a) Realiza los siguientes ejercicios
b) 10 1100
c) 1 0111
d) 101 0110
e) 110
f) 1033_8
g) $15\ 725_8$
h) $B91_{16}$

2.

1. 35_{16}
2. 41854_{10}
3. 100110_2
4. 70_8
5. 11000011_2
6. 101011111001_2
7. $69B_{16}$
8. $3EA6_{16}$
9. 21_{10}
10. 101010_2

SECCIÓN 2.3

1.

a) F = (AB'+C) (ABCD)

A	B	C	D	F
0	0	0	0	**0**
0	0	0	1	**0**
0	0	1	0	**0**
0	0	1	1	**0**
0	1	0	0	**0**
0	1	0	1	**0**
0	1	1	0	**0**
0	1	1	1	**0**
1	0	0	0	**0**
1	0	0	1	**0**
1	0	1	0	**0**
1	0	1	1	**0**
1	1	0	0	**0**
1	1	0	1	**0**
1	1	1	0	**0**
1	1	1	1	**1**

b) F = (AB) + (AB'C) (A') + (C') (A)

A	B	C	F
0	0	0	**0**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**1**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

c) F = (XYZ) + (XY) + (Y'Z') (X'Y)

X	Y	Z	F
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

2.

a) F (XYZ)= X'Y'Z' + XY'Z' + XY'Z + XYZ
b) F (ABC)= A'BC' + A'BC + A'BC + AB'C' + AB'C + ABC' + ABC

SECCIÓN 2.4

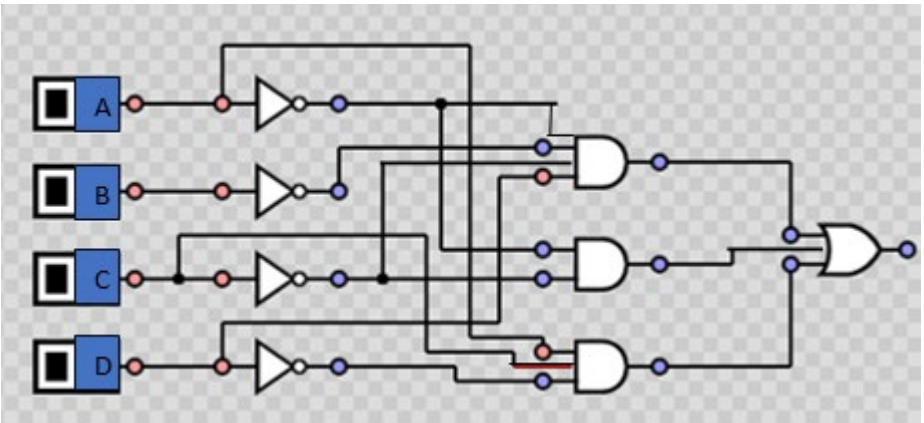
1.

a) F = B'C' + ABC
b) F = CD + A
c) F = B'C' + BC'

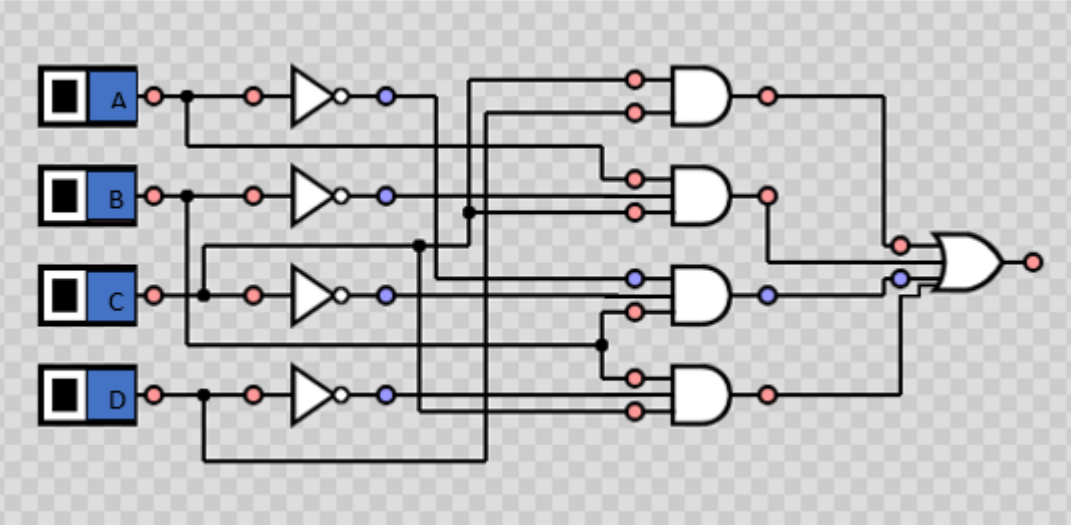
SECCIÓN 2.5

1.

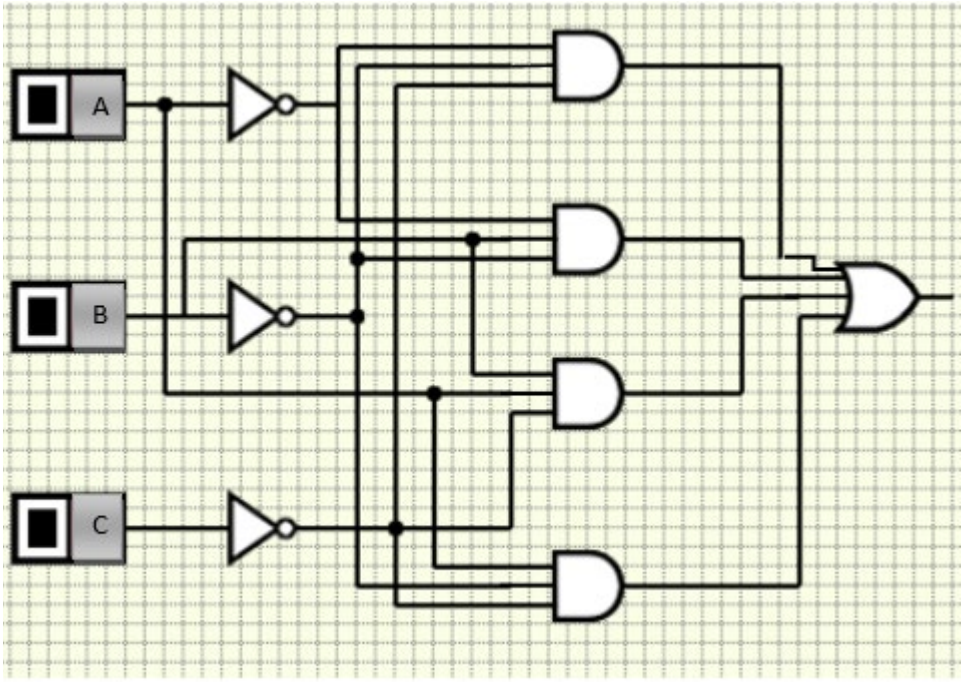
a) **F = A'B'C' D+ A'C' + ACD'**



b) **F = CD + AB'C + A'BC' + BCD'**



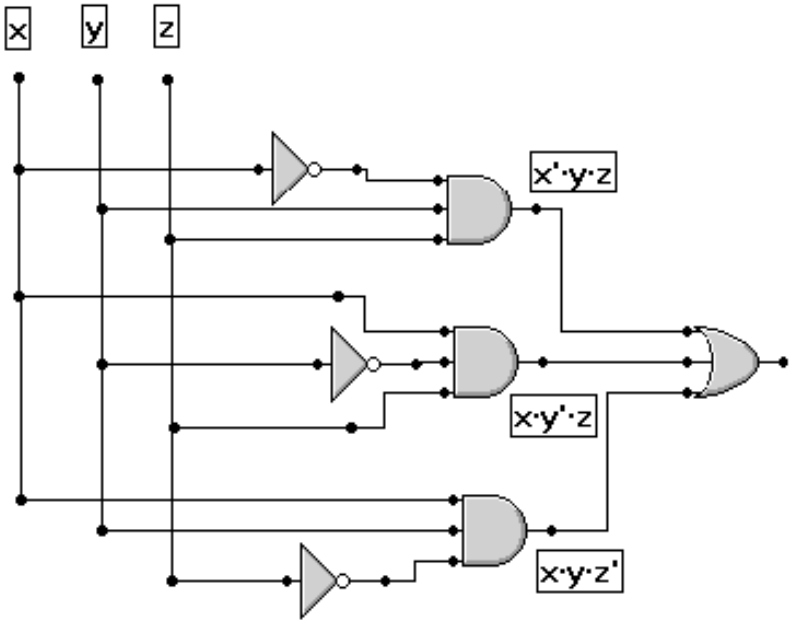
c) F = A'B'C' + A'BC' + ABC' + A B'C'



2.

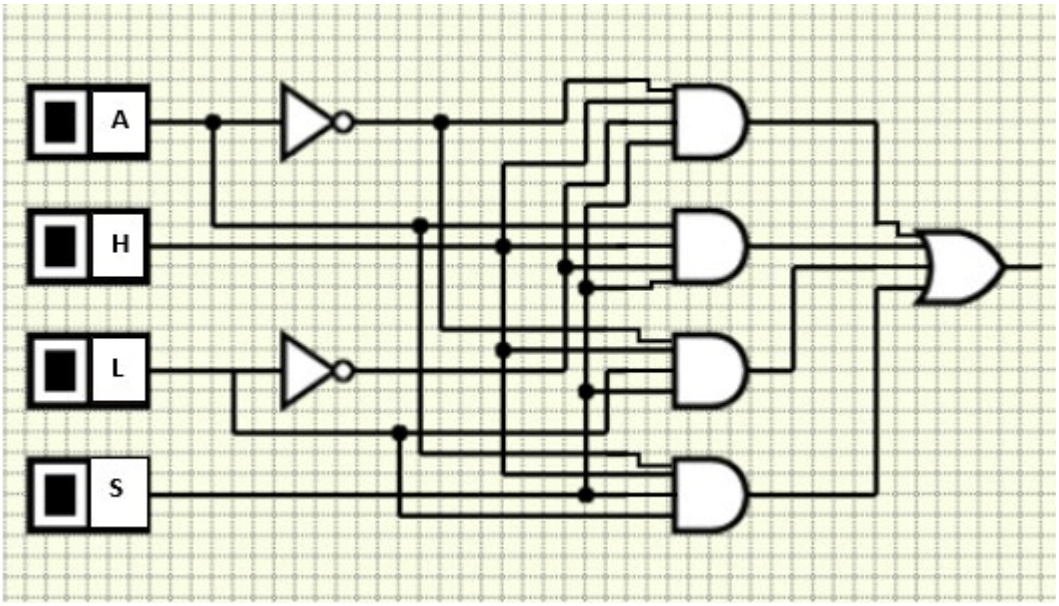
a) Circuito con tres entradas, que indique cuando el número total de unos (1) a la entrada es un número par.

F= X'YZ + XY'Z + XYZ'



b) Diseñar un circuito de una máquina de helados, con 4 entradas: Leche, Agua, Hielo y Saborizante.

F=A'HL'S + AHL'S + A'HLS + AHLS



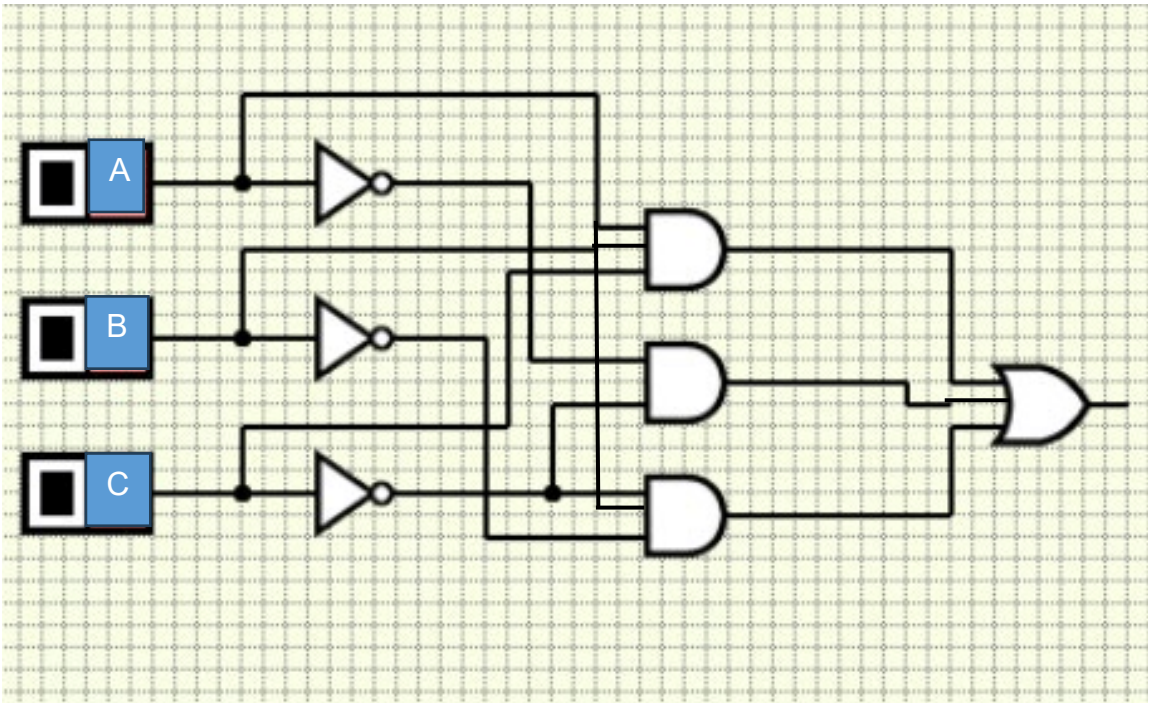
AUTOEVALUACIÓN UNIDAD II

1. 110000001_2
2. 1347_{10}
3. 11111111
4. 1000 0101 0010
5. 42_{10}
6. 701743_{10}
7. $470D7_{16}$
8.

X	Y	Z	F
0	0	0	**1**
0	0	1	**1**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**1**
1	1	1	**1**

9. F= B + AC’

10. Realiza su implementación con compuertas ABC + A’C’ + AB’C’



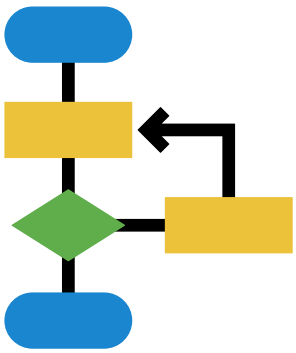
Unidad III. Metodología de solución de problemas e Introducción al lenguaje de programación Java

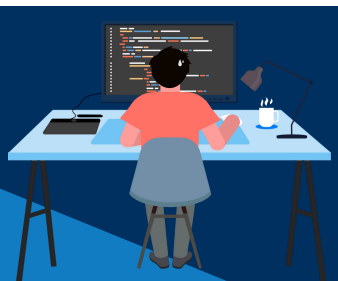
Propósito:

Al finalizar la unidad el alumno:
Aplicará la metodología de solución de problemas mediante la construcción de algoritmos y la codificación en el lenguaje de programación Java para tener una visión integral del proceso de solución.

Contenido

- Sección 3.1 Definiciones y conceptos generales de un problema
- Sección 3.2 Expresiones y tipos de operadores
- Sección 3.3 Algoritmo, Diagrama de Flujo y Pseudocódigo
- Sección 3.4 Lenguaje de Programación Java
- Sección 3.5 Implementación de un Programa con el Lenguaje de Programación en Java
- Sección 3.6 Sentencia Condicional if y switch
- Sección 3.7 Estructuras Ciclícas





Problema: Es un determinado asunto o una cuestión que requiere de una solución.

Problema Determinístico: El resultado será determinado por los valores de entrada y será siempre el mismo.

Problema Probabilístico: Tienen una solución probable porque una variable de entrada será tomada al azar y definirá que camino se seguirá y de este dependerá el resultado.

Problema Secuencial: Se resuelven utilizando secuencias de instrucciones.

Problema Condicionales: La solución va a depender del valor verdadero o falso que tome una condición.

Operadores aritméticos: Permiten realizar operaciones matemáticas con tipos de datos enteros o reales.

Operadores Relacionales: Se utilizan para establecer una relación entre dos valores, luego compara estos valores entre sí y produce un resultado verdadero o falso.

Operadores Lógicos: Se utilizan para establecer relaciones entre valores lógicos: verdadero (V) y falso (F).

Algoritmo: Es una secuencia ordenada y cronológica de pasos que llevan a la solución de un problema o a la ejecución de una actividad.

Diagrama de Flujo: Es la representación esquemática de la secuencia de instrucciones de un algoritmo.

Pseudocódigo: Representar la solución a un algoritmo de la forma más detallada posible y lo más parecida al lenguaje de programación que posteriormente se utilizará.

Java: Es un lenguaje de programación de propósito general utilizado en el desarrollo de programas de software.

Sentencias Condicionales: Se utilizan para la toma de decisiones, las cuales consideran la evaluación de una comparación o expresión lógica, en base a su resultado se realizarán ciertas acciones.

Sentencias Cíclicas: Permiten ejecutar un bloque de instrucciones de forma repetitiva, una cantidad de veces cualquiera o una serie de instrucciones de acuerdo con la evaluación de una condición.

Sección 3.1 Definiciones y conceptos generales de un problema

Aprendizaje:

1. Define el concepto de problema.
2. Identifica los elementos de un problema.
3. Describe la diferencia entre problemas determinísticos, probabilísticos, secuenciales y condicionales.
4. Conoce las etapas de la metodología de solución de problemas.

Temática

Definiciones y conceptos generales de un problema.

Elementos y relaciones del problema:

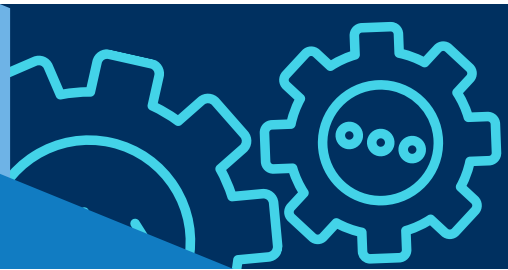
- Entrada.
- Proceso.
- Salida.

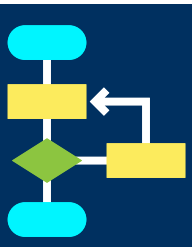
Tipos de problema:

- Determinísticos.
- Probabilísticos.
- Secuenciales.
- Condicionales.

Etapas de la metodología de solución de problemas:

- Planteamiento del problema.
- Análisis del problema.
- Diseño de la solución del problema:
  - Elaboración de algoritmos.
  - Representación del algoritmo a través de pseudocódigo y diagrama de flujo.
  - Prueba de escritorio.





DEFINICIÓN DE PROBLEMA

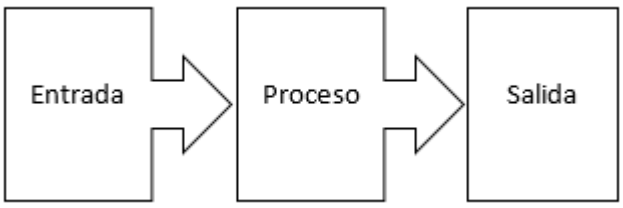
Un problema es un determinado asunto o una cuestión que requiere de una solución, esto es resolver una dificultad o una duda. A nivel social se trata de alguna situación que al solucionar aporta beneficios a la sociedad. Existen muchos y variados tipos de problemas que son específicos de diversas ciencias donde se desarrollan, siendo los más frecuentes con campos como la Filosofía, las Matemáticas, la Medicina, entre otros.

Un problema matemático es una incógnita acerca de una cierta entidad matemática que debe resolverse a partir de otra entidad que hay que descubrir. En otras palabras, un problema matemático plantea una pregunta y fija ciertas condiciones, tras lo cual se debe hallar un número que resuelva la incógnita.

ELEMENTOS DE UN PROBLEMA

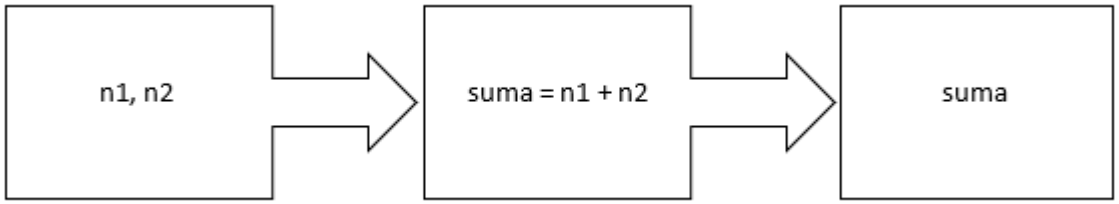
Para solucionar un problema es necesario ubicar los tres elementos esenciales que lo componen: datos o causas, operaciones o hechos y resultados o consecuencias, dependiendo del tipo de problema que se esté analizando. Los diagramas EPS (por sus siglas Entrada, Proceso, Salida) son la representación gráfica de la solución a un problema. Este tipo de diagrama está integrado por los tres elementos del problema:

- **Entrada**: son los datos con los que cuenta el problema.
- **Proceso**: es la forma con la que se va a obtener la solución al problema, normalmente representado por una fórmula matemática.
- **Salida**: es el resultado o la solución al problema, lo que se está buscando.



Ejemplo 1: Resolver la suma de dos números n1 y n2.

Para ello identificamos los datos de entrada, la operación es una suma y esta es el resultado que se desea obtener, así se representa en un diagrama EPS.

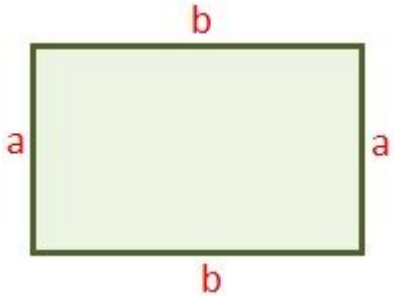


TIPOS DE PROBLEMAS

Un criterio para clasificar los tipos de problemas es la relación de los datos de entrada y los de salida, con ello están los problemas determinísticos y probabilísticos.

- **Determinístico**: en este tipo de problemas el resultado será determinado por los valores de entrada y será siempre el mismo; son problemas donde sabemos con certeza cuáles son los datos de entrada y aplicando una fórmula matemática siempre dará el mismo resultado.

Ejemplo 2: Calcular el perímetro de un rectángulo de 8 unidades de base por 4 unidades de altura.



Solución: La fórmula para calcular el perímetro de un rectángulo es sumar dos veces la base **b** más dos veces la altura **a**, así:

P = 2*base + 2*altura = 2*8 + 2*4 = 24 unidades

- **Probabilístico**: son los problemas que tienen una solución probable porque una variable de entrada será tomada al azar y definirá que camino se seguirá y de este dependerá el resultado. Si el experimento puede tener “n” resultados igualmente probables y mutuamente excluyentes, “r” número de los cuales corresponden a la

ocurrencia de algún evento A, entonces la probabilidad de que el evento A ocurra, o P(A), se define de la siguiente manera:

$$P(A) = \frac{r}{n} = \frac{Resultados\ favorables}{Total\ de\ Resultados}$$

Ejemplo 3: Encontrar la probabilidad de sacar una cruz al lanzar una moneda.

Solución: Una moneda tiene dos resultados totales, al lanzarla podemos obtener sol o águila, y la probabilidad de obtener un resultado favorable para cruz se calcula con la fórmula:

$$P(A) = \frac{r}{n} = \frac{1}{2} = 0.5$$

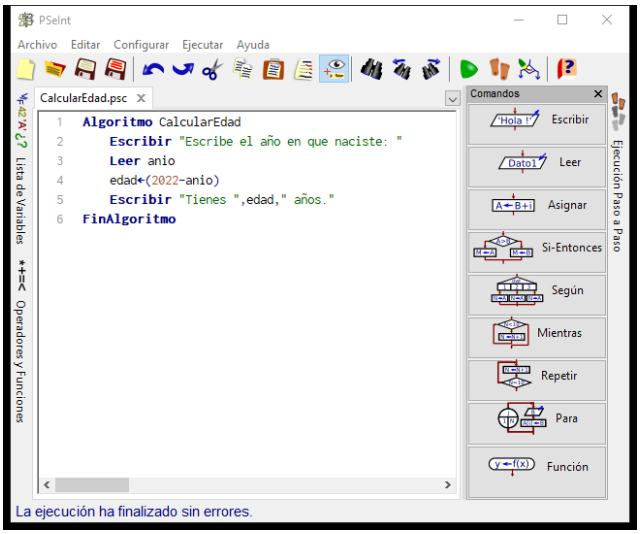
Para calcular la probabilidad de dos o más eventos se necesitan seguir otras reglas.

Otro criterio para clasificar los problemas es el tipo de proceso que se aplica a los datos para encontrar el resultado, en este caso se tienen problemas secuenciales y condicionales

- **Secuencial**: se basa en la estructura del algoritmo para su solución; se resuelven utilizando secuencias de instrucciones, esto es los pasos se aplican uno después de otro en un orden consecutivo.

Ejemplo 4: Calcular la edad a partir del año de nacimiento.

Solución: Utilizando el programa PSeInt se puede hacer un algoritmo para la solución del problema.

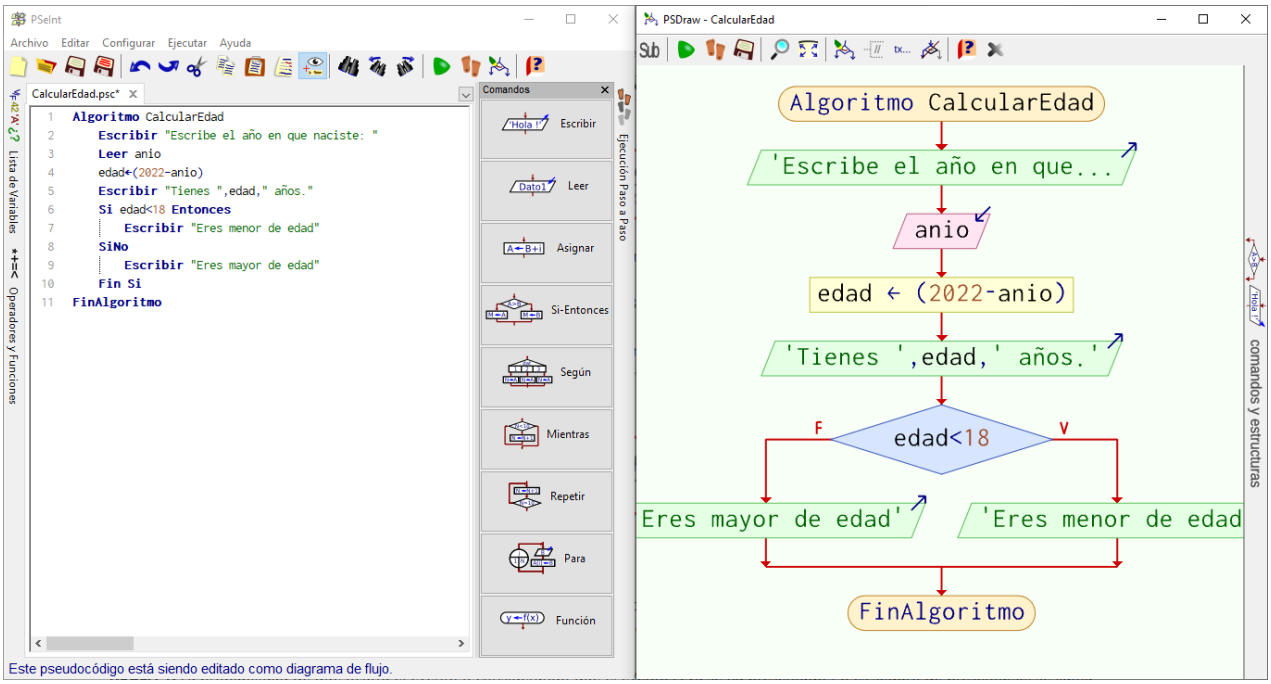


- **Condicionales**: en estos problemas la solución va a depender del valor verdadero o falso que tome una condición, esto es,

se puede tomar un camino u otro y esto llevará a la ejecución de ciertas acciones u otras.

Ejemplo 5: Solicitar el año de nacimiento e indicar si es mayor o menor de edad.

Solución: Utilizando el programa PSeInt se puede diseñar un algoritmo y diagrama de flujo para la solución del problema. La condición es que la edad sea menor de 18 envía el mensaje “Eres menor de edad” sino entonces envía “Eres mayor de edad”.



ETAPAS DE LA METODOLOGÍA DE SOLUCIÓN DE PROBLEMAS

Para solucionar un problema con un programa de computadora se requiere seguir el proceso de programación, el cual consiste en una serie de pasos. A continuación, se describe cada uno de los pasos que se deben seguir:

1. **Definición del problema**: se debe identificar el problema y comprender la utilidad de la solución que se alcance; se debe tener una visión general del problema estableciendo las condiciones iniciales, los puntos de partida y los límites.

2. **Análisis del problema**: es necesario entender en detalle el problema para obtener los datos disponibles como materia prima, definir el proceso necesario para convertir los datos en la información de salida o solución del problema.

3. **Diseño**: se procede a diseñar la lógica para la solución al problema a través de dos procesos:
   - ***Elaboración del algoritmo***: se estructura la secuencia lógica de pasos que la computadora deberá seguir para solucionar el problema utilizando alguna técnica de diseño de algoritmos como seudocódigo o diagramas de flujo.
   - ***Prueba de escritorio***: se simula el funcionamiento del algoritmo con datos propios respecto al problema y se comprueban los resultados a fin de validar la correcta operación del algoritmo o se deberá modificar para probarlo hasta que esté correcto.

4. **Codificación del programa**: en este paso se procede a codificar el programa en el lenguaje de programación que se vaya a utilizar. Este proceso es sumamente sencillo porque ya se tiene el programa en seudocódigo, solo se convierten las acciones del algoritmo en instrucciones de computadora. El programa codificado debe editarse, compilarse, probarse y depurarse; es decir, se ejecuta para verificar su buen funcionamiento y se hacen las correcciones o los ajustes pertinentes hasta que quede correcto.

5. **Implantación del programa**: el programa se instala y se pone a funcionar, entrando en operación dentro de la situación específica para la que se desarrolló.

6. **Mantenimiento del programa**: el programa que está en operación puede presentar errores, los cuales deben corregirse o podría requerir cambios o ajustes en sus datos, proceso o información, por lo que surge la necesidad de darle mantenimiento y se tendrá que regresar a algún paso precedente: al 4, al 3, al 2 o al 1.

1. **Con la ayuda de los cuadros elabora el diagrama entrada, proceso, salida (EPS) de cada problema.**

a) Realizar la multiplicación de tres números a, b, c

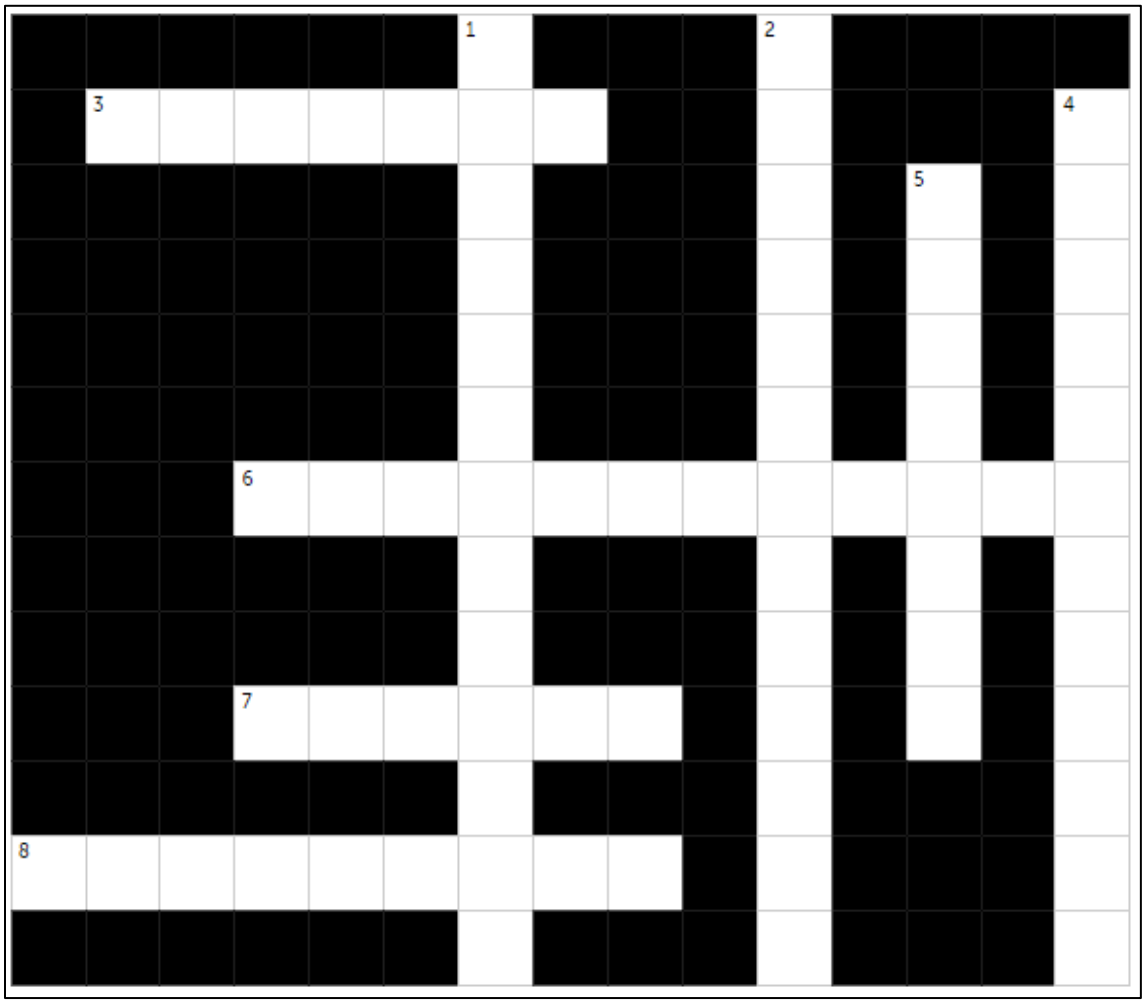
b) Obtener el valor de la hipotenusa (h) de un triángulo rectángulo conociendo sus catetos (ca, co).

c) Calcular el índice de masa corporal de un adulto dividiendo el peso entre el cuadrado de la estatura.

2. Relaciona las columnas concepto y descripción anotando entre los paréntesis la letra.

Concepto	Letra	Descripción
1. Problema probabilístico.	( )	**A.** Se lanzan dos dados, cual es la probabilidad de obtener 5 y 2.
2. Ejemplo de problema determinístico…….	( )	**B.** Se resuelve utilizando los pasos, uno después de otro en orden consecutivo.
3. Definición de problema…	( )	**C.** Una fórmula matemática para obtener el valor de una incógnita.
4. Problema secuencial……	( )	**D.** Una persona tiene sobrepeso cuando el IMC es mayor o igual de 30.
5. Entrada de un problema..	( )	**E.** Resolver una dificultad, una cuestión o una duda.
6. Problema condicional…...	( )	**F.** Es la solución del problema.
7. Ejemplo de problema probabilístico……..	( )	**G.** Su solución depende del valor verdadero o falso de una condición.
8. Salida de un problema….	( )	**H.** Obtener el perímetro de un círculo.
9. Ejemplo de problema condicional……..	( )	**I.** Son los datos con los que cuenta el problema para solucionarlo.
10. Proceso de un problema…….	( )	**J.** En este problema una variable será tomada al azar.

3. Contesta el siguiente crucigrama con palabras clave de la metodología de solución de problemas.



HORIZONTAL

3. Validar la correcta operación del algoritmo.
6. Se convierten las acciones del algoritmo en instrucciones de computadora.
7. Tiene dos procesos para construir la lógica de la solución.
8. Secuencia lógica de pasos para solucionar un problema.

VERTICAL

1. Consiste en corregir o ajustar datos, proceso o información.
2. Obtener una visión general del problema estableciendo las condiciones y límites.
4. El programa entra en operación.
5. Identificar los datos disponibles y el proceso para obtener la solución.

Sección 3.2 Expresiones y Tipos de Operadores

Aprendizaje:

5. Analiza el resultado de expresiones aritméticas utilizando la jerarquía de las operaciones.
6. Construye expresiones lógicas utilizando operadores relacionales y lógicos.

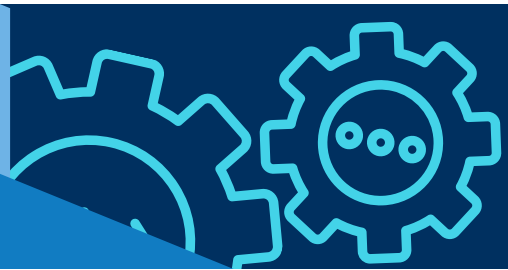
Temática

Expresiones y operadores aritméticos:

- Asignación.
- Operadores aritméticos.
- Jerarquía de operadores aritméticos.
- Evaluación de expresiones aritméticas.

Expresiones y operadores relacionales y lógicos:

- Operadores relacionales.
- Operadores lógicos.
- Jerarquía de operadores lógicos.
- Evaluación de expresiones lógicas.



EXPRESIONES Y OPERADORES ARITMÉTICOS

Los **operadores** son elementos que relacionan de forma diferente, los valores con los que trabajamos en la programación se usan para manipular los valores y transformarlos, con el objetivo de alcanzar la finalidad de los programas.

Los operandos son los valores que se utilizan para alimentar los operadores y pueden venir almacenados en variables o constantes, pero también pueden ser valores.

Los **operadores aritméticos** permiten realizar operaciones matemáticas con tipos de datos enteros o reales. Los siguientes son los operadores aritméticos que se usan con los símbolos y un ejemplo:

Símbolo	Nombre	Expresión	Resultado
+	Suma	7+2	9
-	Resta	7-2	5
*	Multiplicación	7*2	14
/	División	7/2	3.5
mod	Módulo	12 mod 7	5
^	Potencia	2^3	8

Cuando hay varios operadores en una misma expresión los lenguajes de programación tendrán que evaluar en un orden determinado las **reglas de precedencia de operadores o jerarquía de operadores aritméticos**.

Dentro de una misma expresión los operadores se evalúan en el siguiente orden:

1. Exponenciación
2. Multiplicación, división, módulo
3. Suma y resta

En el caso de que en una misma expresión se asocien operadores con igual nivel de prioridad, estos se evalúan de izquierda a derecha.

Las expresiones pueden contener paréntesis que rompen las reglas de precedencia de los operadores, pues con ellos se define que operadores se van a relacionar con qué operandos, independientemente de las reglas mencionadas anteriormente. Las reglas de los paréntesis son las siguientes:

- Todas las expresiones entre paréntesis se evalúan primero.
- Las expresiones con paréntesis anidados se evalúan de dentro a fuera.
- El paréntesis más interno se evalúa primero.

Ejemplo 1. Operadores aritméticos.

Atendiendo la precedencia de operadores	Atendiendo los paréntesis
4 + 2 * 5 Primero se multiplica * 4 + 10 Después se suma + 14	(4 + 2) * 5 Primero se suma + 6 * 5 Después se multiplica * 30
2.1 * 1.5 + 12.3 Primero se multiplica * 3.15 + 12.3 Después se suma + 15.45	2.1 * (1.5 + 12.3) Primero se suma + 2.1 * 13.8 Después se multiplica * 28.98

EXPRESIONES Y OPERADORES RELACIONALES Y LÓGICOS.

Los **operadores relacionales** se utilizan para establecer una relación entre dos valores, luego compara estos valores entre sí y produce un resultado verdadero o falso. Los siguientes son los operadores relacionales y se muestra un ejemplo de cada uno:

Símbolo	Nombre	Expresión	Resultado
>	Mayor que	4 > 6	Falso
<	Menor que	4 < 6	Verdadero
>=	Mayor o igual que	5 >= 5	Verdadero

Símbolo	Nombre	Expresión	Resultado
<=	Menor o igual que	5 <= 6	Verdadero
<>	Diferente	5 <> 4	Verdadero
==	Igual	5 == 4	Falso

Los **operadores lógicos** se utilizan para establecer relaciones entre valores lógicos: verdadero (V) y falso (F), de esta manera podemos asociar varias expresiones con operadores relacionales pues tienen un resultado lógico.

Por ejemplo ante la expresión 10 < 20 < 30 el resultado es verdadero, pero lo vamos a asociar con un operador lógico, así tenemos dos expresiones relacionales (10 < 20) y (20 < 30) que se deben cumplir.

Los operadores lógicos se presentan a continuación con su símbolo, su tabla de verdad o resultados y un ejemplo.

Símbolo	Nombre	Tabla de verdad	Expresión	Resultado
&&	Y	F && F = F F && V = F V && F = F V && V = V	(10<20) && 20<30) V && V V	Verdadero
||	O	F && F = F F && V = V V && F = V V && V = V	(5<2) && (3<10) F && V V	Verdadero
!	NOT	!F = V	! (4>7) ! F	Verdadero

1. Resuelve las siguientes expresiones matemáticas atendiendo las reglas de precedencia de los operadores y los paréntesis.

a) -6 + (7^2 - 4 * 5 * 6) / 2 * 5 =

b) - 6 + 7^2 - 4 * 5 * 6 / 2 * 5 =

c) (4 + 3) * (-5 + 6 * (12 / 3) – (8*10)) =

d) 4 + 3 * -5 + 6 * 12 / 3 – 8 * 10 =

2. Resuelve las siguientes expresiones relacionales o lógicas colocando V si es verdadero o F si es falso, utilizando el espacio libre para hacer las operaciones necesarias.

a) (5 - 7) > = (7 - 5) …… ( )

b) (5 < 6) && (-1 > 1) …… ( )

c) (40 == 40) || (5 < -5) …. ( )

d) ! (0.100 > 0.200)…… ( )

e) (7 < 4) && (3 < 7) …… ( )

f) ! [(9 / 3) != (45/15)]… ( )

Sección 3.3 Algoritmo, Diagrama de Flujo y Pseudocódigo

Aprendizaje:

7. Conoce el concepto de algoritmo, diagrama de flujo y pseudocódigo.
8. Elabora el algoritmo, diagrama de flujo y pseudocódigo para problemas secuenciales.

Temática

Concepto de algoritmo, diagrama de flujo y pseudocódigo.

Elaboración de algoritmos secuenciales.

Representación de algoritmos secuenciales a través de diagramas de flujo y pseudocódigo.

CONCEPTO DE ALGORITMO, DIAGRAMA DE FLUJO Y PSEUDOCÓDIGO

ALGORITMO

El algoritmo es una secuencia ordenada y cronológica de pasos que llevan a la solución de un problema o a la ejecución de una actividad.

Los pasos del algoritmo deben tener las siguientes características:

- Ser simples, claros, precisos y exactos.
- Tener un orden lógico.
- Tener un principio y un fin, ser finitos.

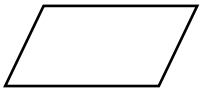
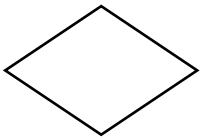
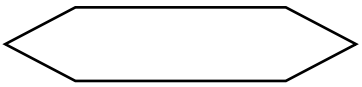
Todas las actividades que llevamos a cabo los seres humanos son algoritmos que hemos aprendido a seguir, como ejemplo: caminar y lavarse los dientes, son secuencias lógicas de pasos. Estamos basados en el orden de las cosas y de acciones las cuales se organizan conforme a secuencias lógicas y a esto se le llama programación.

Cuando se diseña un algoritmo se anotan paso a paso, en secuencia, las acciones que se ejecutarán. En ocasiones hay que repetir uno o varios pasos cierto número de veces, en tal caso se tiene que controlar cada paso para que pueda terminar el proceso, esto se conoce como ciclo repetitivo. En otras ocasiones se tiene que llegar a un resultado partiendo de dos o más condiciones, en este caso se toma en cuenta, cómo llegar desde una parte y cómo desde la otra o alternativa, en estas circunstancias se utiliza la selección.

DIAGRAMA DE FLUJO

Es la representación esquemática de la secuencia de instrucciones de un algoritmo para facilitar la información en un formato gráfico sencillo. Un algoritmo está compuesto por operaciones, decisiones lógicas y ciclos repetitivos que se representan gráficamente por medio de símbolos estandarizados: óvalos para iniciar o finalizar el algoritmo; rombos para comparar datos y tomar decisiones; rectángulos para indicar una acción o instrucción general, entre otros. Son diagramas de flujo porque los símbolos utilizados se conectan en una secuencia de instrucciones o pasos indicada por medio de flechas.

Adicionalmente facilitan la comprensión de la secuencia lógica de la solución planteada y sirven como elemento de documentación en la solución de problemas o en la representación de los pasos de un proceso.

NOMBRE	SÍMBOLO	DESCRIPCIÓN
Inicio		Representa el inicio del diagrama o proceso.
Declarar variables		Declarar los elementos que se necesitan como variables, constantes y los tipos de datos.
Ingresar datos		Entrada de datos por teclado, introducción de datos por el usuario.
Proceso		Representación del proceso u operación a realizar.
Imprimir datos (salida de datos)		Impresión del resultado o datos del proceso.
Fin		Representa el fin del diagrama o proceso.
Línea de flujo		Indica la dirección hacia dónde va el proceso.
Decisión		Evalúa una condición dentro del proceso, así como darle solución con “Si” o “No”
Repetición		Proceso que se repite según la condición a realizar.

PSEUDOCÓDIGO

El pseudocódigo o falso lenguaje es utilizado por los programadores para omitir secciones de código y dar una explicación del paradigma, no es programable sino facilita la programación. El principal objetivo del pseudocódigo es el de representar la solución a un algoritmo de la forma más detallada posible y a su vez lo más parecida al lenguaje de programación que posteriormente se utilizará.

Las principales características de un pseudo lenguaje son:

1. Se puede ejecutar en una computadora.
2. Es una forma de representación sencilla de utilizar y de manipular.
3. Facilita el paso de programación al lenguaje de programación.
4. Es independiente del lenguaje de programación que se vaya a utilizar.
5. Es un método que facilita la programación y solución al algoritmo.

Los ejemplos de pseudocódigo se realizarán con PSeInt, el cual es una herramienta para asistir a los estudiantes en sus primeros pasos en programación, mediante un simple e intuitivo pseudo lenguaje en español (complementado con un editor de diagramas de flujo), que le permite centrar su atención en los conceptos fundamentales de la algoritmia computacional, minimizando las dificultades propias de un lenguaje y proporcionando un entorno de trabajo con numerosas ayudas y recursos didácticos. Este programa se puede descargar de la siguiente página: https://pseint.sourceforge.net/

Todo algoritmo en pseudocódigo tiene la siguiente estructura general: comienza con la palabra clave Algoritmo seguida del nombre, luego le sigue una secuencia de instrucciones y finaliza con la palabra FinAlgoritmo. Una secuencia de acciones es una lista de una o más instrucciones y/o estructuras de control.

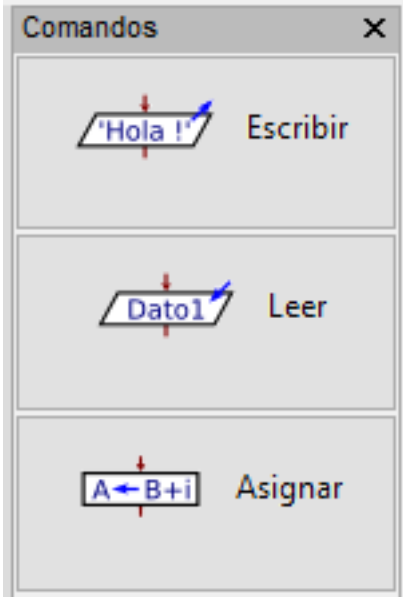
```
Algoritmo SinTitulo
    acción 1;
    acción 2;
    …
    acción n;
FinAlgoritmo
```

Las instrucciones primitivas secuenciales en PSeInt son:

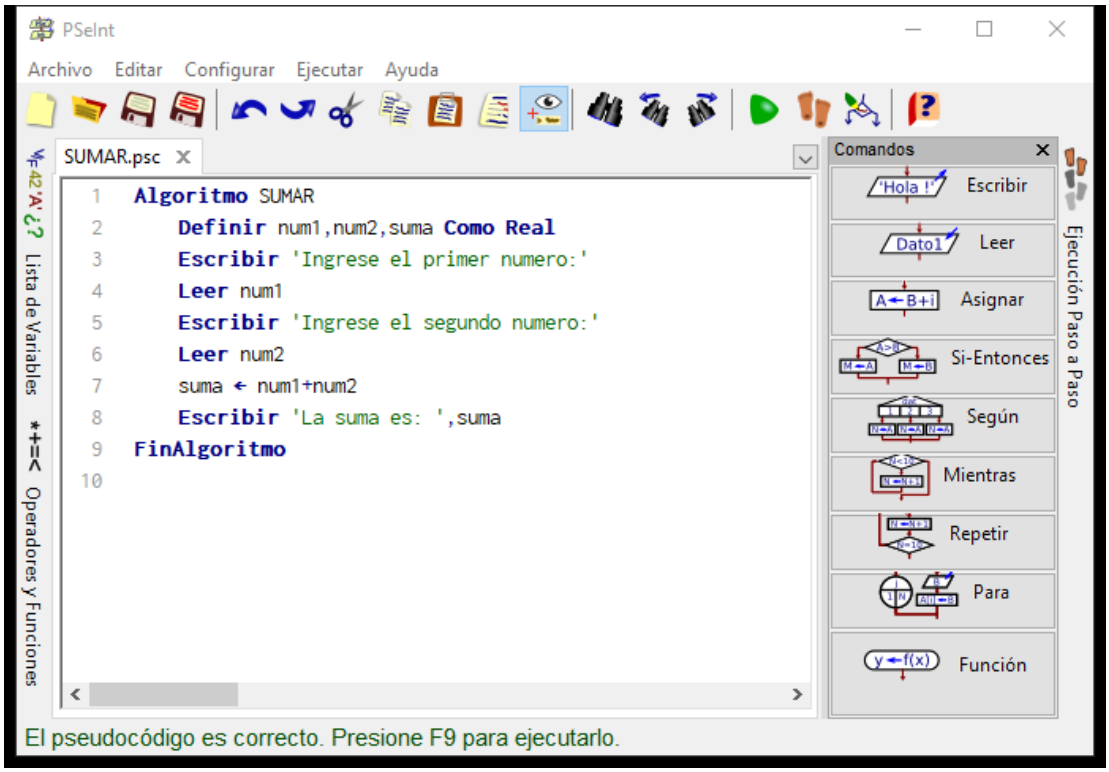
- **Escritura**: la instrucción Escribir permite mostrar valores al ambiente.
  Escribir *<expr1> , <expr2> , ... , <exprN>* ;

- **Lectura**: la instrucción Leer permite ingresar información desde el ambiente.
  Leer *<variable1>, <variable2>, … <variableN>* ;

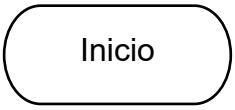
- **Asignación**: permite almacenar un valor en una variable.
  <variable> **<-** *<expresión>* ;

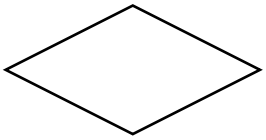
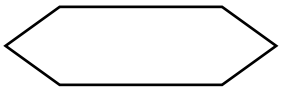


Ejemplo 1. Uso de los comandos de PSeInt para leer dos números, realizar la suma y mostrar el resultado.

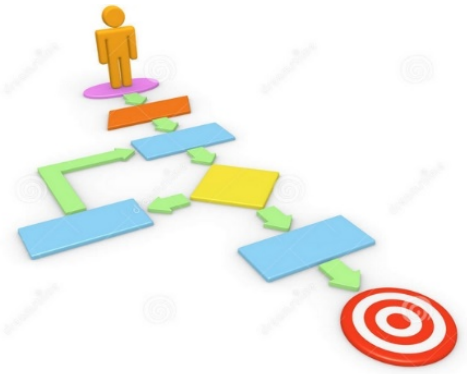


En conclusión, existe una relación al escribir un algoritmo, diagrama de flujo y pseudocódigo que se muestra en la siguiente tabla, esto es para cada acción en un algoritmo se tiene la representación gráfica y la sentencia de pseudocódigo.

ALGORITMO	DIAGRAMA DE FLUJO	PSEUDOCÓDIGO
Algoritmo nombre		Algoritmo SinTitulo
Declarar variables		Definir <var1> Como [REAL/ENTERO/LOGICO/CARACTER];
Pedir datos		Leer <variable >

ALGORITMO	DIAGRAMA DE FLUJO	PSEUDOCÓDIGO
Calcular expresión		<variable > <- <expresión>;
Mostrar datos		Escribir <expr1>
FinAlgoritmo		FinAlgoritmo
Línea de flujo		1 2 3
Si condición verdadera Realizar acciones a De lo contrario Realizar acciones b Fin		Si expresion_logica Entonces acciones_por_verdadero SiNo acciones_por_falso Fin Si
Para i hasta f avanza Realizar acciones Fin		Para Hasta Con Paso paso Hacer secuencia_de_acciones Fin Para

ELABORACIÓN DE ALGORITMO, DIAGRAMA DE FLUJO Y PSEUDOCÓDIGO PARA PROBLEMAS SECUENCIALES



La secuenciación es una estructura que permite controlar la ejecución de un conjunto de acciones en orden secuencial, esto es, ejecuta la primera acción, luego la que sigue y así sucesivamente hasta la última. Dichas acciones pueden consistir en declarar variables, leer datos, calcular alguna expresión o imprimir datos. Es conveniente etiquetar cada acción con números desde el 1 en forma ascendente de uno en uno, para denotar el orden secuencial.

Los elementos que integran la estructura de un algoritmo son los siguientes:

- **Encabezado**: es un identificador, el cual debe empezar con la palabra Algoritmo, seguida de una breve descripción de lo que hace.

- **Declaración**: consiste en declarar los elementos que se necesitan como variables, constantes y los tipos de datos.

- **Lectura de datos**: se empieza a introducir los datos disponibles como materia prima mediante una operación de lectura precedida por una solicitud de los datos.

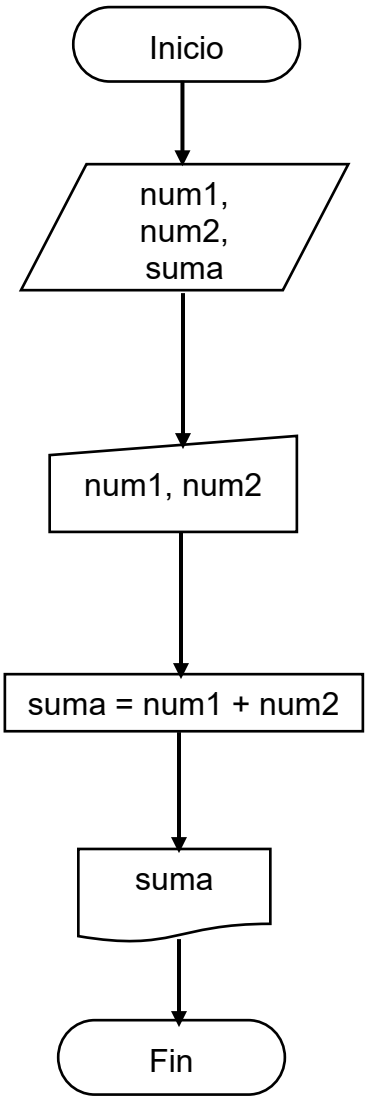
- **Hacer cálculos**: consiste en procesar la entrada para producir la salida mediante la ejecución de cálculos basados en expresiones aritméticas.

- **Impresión de datos**: estriba en dar salida a la información requerida, imprimiendo las variables que las contienen.

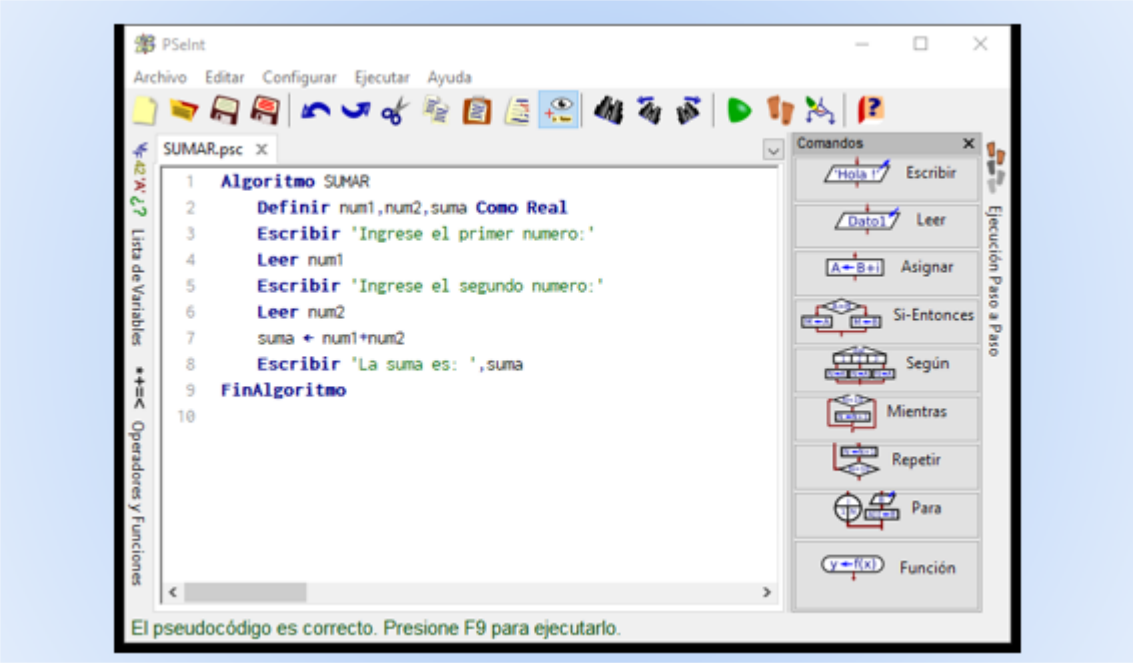
- **Fin del algoritmo**: por último, se tiene el fin del algoritmo.

A continuación, se presenta un ejemplo para explicar la forma como se escribe un algoritmo con sus elementos y el diagrama de flujo con sus símbolos.

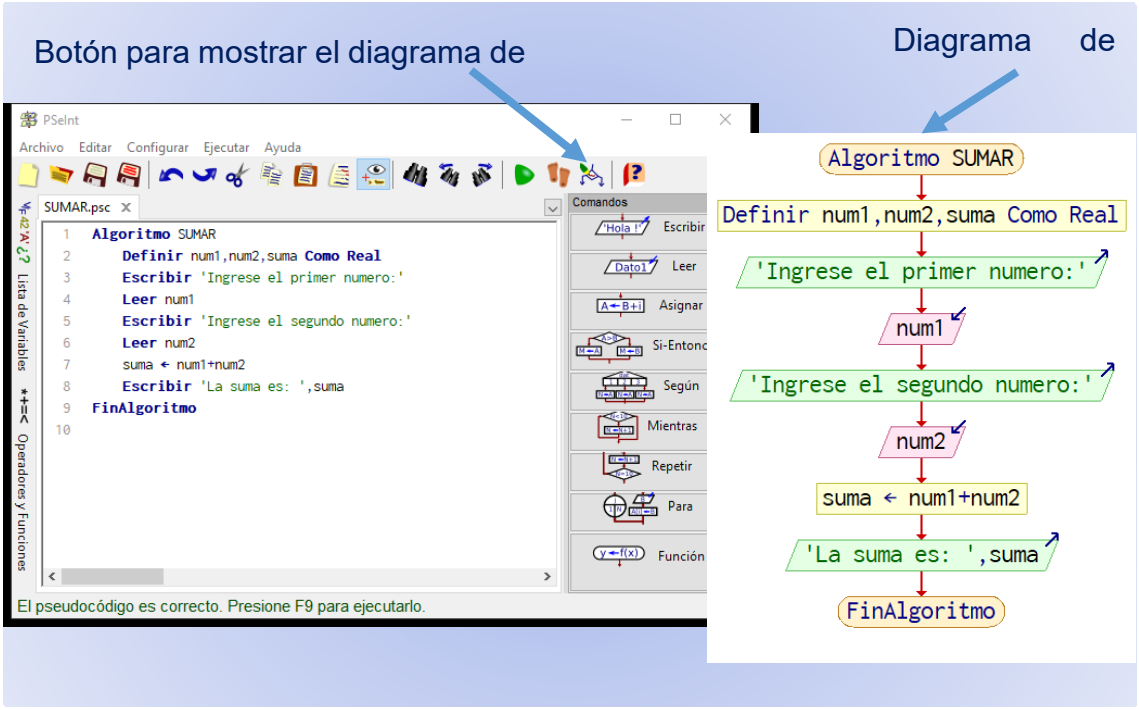
Ejemplo 2: Realizar la suma de dos números.

Elementos	Algoritmo	Diagrama de flujo
Encabezado	Algoritmo SUMAR	
Declaración	1. Declarar variables a. num1: Real b. num2: Real c. suma: Real	
Lectura de datos	2. Pedir el primer número 3. Leer num1 4. Pedir el segundo número 5. Leer num2	
Hacer cálculos	6. Calcular suma = num1 + num2	
Impresión de datos	7. Mostrar la suma	
Fin del algoritmo	FinAlgoritmo	

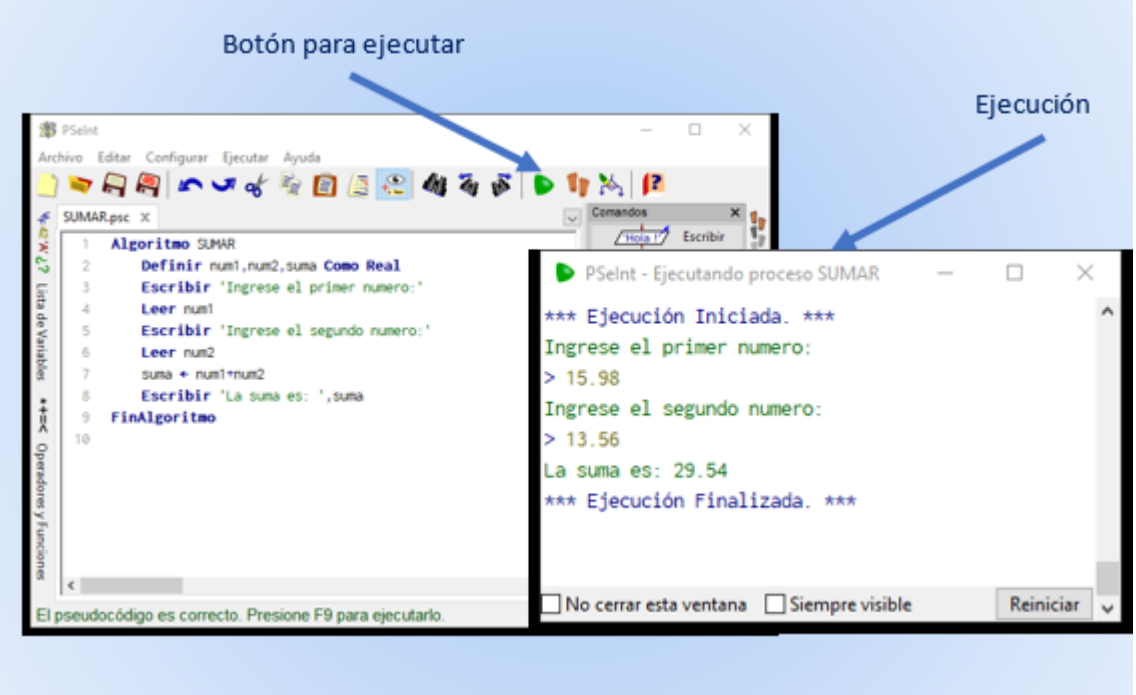
En PSeInt se escriben los comandos como se muestra a continuación:



Para visualizar el diagrama de flujo que PSeInt genera pulsa el botón que a continuación se señala y aparecerá la imagen.



Para ejecutar el pseudocódigo se pulsa el botón que a continuación se señala y la salida es la que se muestra.

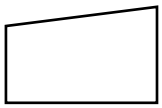


1. Subraya la respuesta correcta:

I. Es una secuencia ordenada y cronológica de pasos que llevan a la solución de un problema.

a) Diagrama de flujo **b)** Algoritmo **c)** Pseudocódigo **d)** Prueba de escritorio

II. Símbolo de un diagrama de flujo para la entrada de datos.

a)  **b)**  **c)**  **d)** 

III. Es el falso lenguaje que sirve para representar la solución de la forma más detallada posible y a su vez lo más parecida al lenguaje de programación.

a) Diagrama de flujo **b)** Algoritmo **c)** Pseudocódigo **d)** Prueba de escritorio

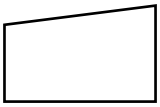
IV. Instrucción primitiva de PseInt que permite mostrar valores al ambiente.

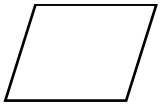
a) Leer **b)** Asignación <- **c)** Escribir **d)** Definir

V. Es la representación gráfica por medio de símbolos estandarizados para facilitar la información en un formato sencillo.

a) Diagrama de flujo **b)** Algoritmo **c)** Pseudocódigo **d)** Prueba de escritorio

VI. Símbolo de un diagrama de flujo para declarar variables.

a) 

b) 

c) 

d) 

VII. Algoritmo para comparar datos y tomar decisiones.

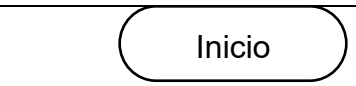
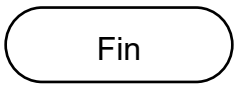
a)	**b)**	**c)**	**d)**
Leer <variable>;	Para i hasta f avanza Realizar acciones Fin	Si condición verdadera Realizar acciones De lo contrario Realizar acciones Fin	<var> <- <expr>;

VIII. No es una característica de un pseudo lenguaje:

a) Se puede ejecutar en una computadora.	**b)** Facilita el paso al lenguaje de programación.	**c)** Depende del lenguaje de programación.	**d)** Es una forma de representación sencilla de utilizar y manipular.

2. Completa el algoritmo, diagrama de flujo y pseudocódigo de los siguientes problemas secuenciales y como actividad extra escribir en PSeInt el pseudocódigo y ejecutar el archivo.

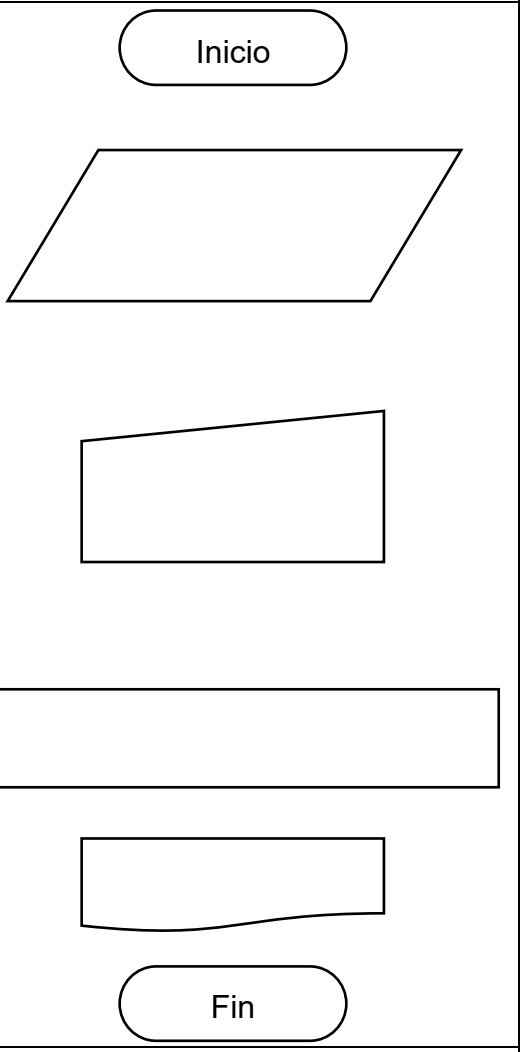
a) Obtener el área de un círculo.

Algoritmo	**Diagrama de flujo**
Algoritmo CIRCULO	
1. Declarar variables a. radio: _____ b. pe = _____ c. área: _____	
2. Pedir el _____ 3. Leer _____	
4. Calcular área = _____	
5. Mostrar el _____	
FinAlgoritmo	

Pseudocódigo

```
Algoritmo CIRCULO
        Definir _____ Como _____;
        pe<-_____;
        Definir _____ Como _____I;
        Escribir "_____";
        _____ radio;
        _____ <- _____;
        _____ "_____", _____
FinAlgoritmo
```

b) Obtener la pendiente de una recta conociendo dos puntos (x1,y1) (x2,y2)

Algoritmo	Diagrama de flujo
Algoritmo RECTA	
1. Declarar variables a. x1: _____ b. y1: _____ c. x2: _____ d. y2: _____ e. m: _____	
2. Pedir _____ 3. Leer _____ 4. Pedir _____ 5. Leer _____ 6. Pedir _____ 7. Leer _____ 8. Pedir _____ 9. Leer _____	
10. Calcular m = _____	
11. Mostrar _____	
FinAlgoritmo	

Pseudocódigo

```
Algoritmo RECTA
        Definir _____ Como Real;
        Escribir " _____ ";
        _____ x1;
        Escribir " _____";
        Leer _____;
        _____"Dame el valor de x2; ";
        _____;
        Escribir " _____ ";
        _____;
        m<-  _____;
        _____ "La pendiente de la recta es m = ", _____;
FinAlgoritmo
```

Sección 3.4 Lenguaje de Programación Java

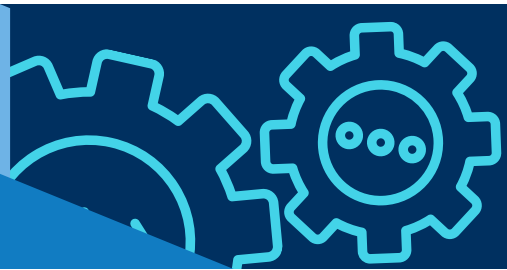
Aprendizaje:

9. Conoce la historia del lenguaje de programación Java.
10. Conoce las características básicas del lenguaje de programación Java.
11. Conoce el entorno de desarrollo para el lenguaje de programación Java.

Temática

Lenguaje de programación Java:

- Historia del lenguaje.
- Características:
  - Tipos de aplicaciones.
  - Arquitectura neutral.
  - Lenguaje orientado a objetos.
  - Disponibilidad de un amplio conjunto de bibliotecas.
  - Interpretado.
  - Robusto.
  - Distribuido.
- Entorno de desarrollo del lenguaje de programación:
  - JDK (kit de desarrollo de Java).
  - IDE (interfaz de entorno de desarrollo).



HISTORIA DEL LENGUAJE DE PROGRAMACIÓN JAVA

Sun Microsystems es la empresa estadounidense creadora del lenguaje Java y comenzó a desarrollarlo con el objetivo de crear un lenguaje independiente de la plataforma y del sistema operativo que permitiera su diseño y construcción en la floreciente electrónica de consumo.

El proyecto original, denominado Green, comenzó apoyándose en C++ pero a medida que progresaba su desarrollo, el equipo se encontró con problemas de portabilidad, por lo que decidieron desarrollar su propio lenguaje, y en agosto de 1991 nació un nuevo lenguaje orientado a objetos y al cual llamaron Oak. En 1993, Green se renombró Fist Person Juc e invirtió presupuesto y esfuerzo para intentar vender esta tecnología, sin tener mucho éxito.

A mitad de 1993 se lanzó el primer navegador web y comenzó a crecer el interés por internet por lo que se rediseñó el lenguaje para desarrollar internet, así el 29 de septiembre de 1994 se termina el prototipo de HotJava el cual tenía soporte para los “applets”, que son las partes de Java que pueden ser cargadas mediante una red de trabajo para después ejecutarse localmente y así lograr soluciones dinámicas acordes al rápido crecimiento del ambiente web. En enero de 1995, Sun forma la empresa Java Soft para dedicarse exclusivamente a crear aplicaciones, herramientas, sistemas de plataforma y servicios para aumentar las capacidades del lenguaje y Oak se convirtió en Java. Existen diferentes versiones sobre el nombre del lenguaje, una de ellas, suponer que le pusieron ese nombre mientras tomaban café (Java es nombre de un tipo de café, originario de Asia), otra versión es que el nombre deriva de las siglas de James Gosling, Arthur Van Hoff y Andy Bechtolsheim.

El 23 de mayo de 1995, en la conferencia SunWorld 95, John Gage, de Sun Microsystems, y Marc Andreessen, cofundador y vicepresidente de Netscape, anunciaron la versión Alpha de Java, en ese momento solo corría en Solaris, y anuncian que Java iba a ser incorporado en Netscape Navigator, el navegador más utilizado de internet en ese momento. Ese fue el factor clave que lanzó a Java a ser uno de los lenguajes de programación más utilizados.

Sun lanzó el entorno JDK 1.0 (java development kit) en 1996 como primera versión del dominio público y se convirtió en la primera especificación formal de la plataforma Java y desde entonces se han lanzado diferentes versiones y la primera versión comercial JDK 1.1 se lanzó a principios de 1997.

En la tabla se muestra la evolución de Java desde la primera versión hasta la 6.0

Versión	Año	Nuevas características	Número de clases e interfaces
1.0	1996	Definición del lenguaje	211
1.1	1997	Clases internas	477
1.2	1998	Ninguna	1524
1.3	2000	Ninguna	1840
1.4	2004	Aserciones	2723
5.0	2004	Clases genéricas, bucles for each, vargas autoboxing, metadatos, enumeraciones, importación estática.	3279
6	2006	Ninguna	3777

En la actualidad encontramos aplicaciones de Java en redes y dispositivos que comprenden desde internet, supercomputadoras científicas hasta portátiles y teléfonos móviles; desde simuladores de mercado hasta juegos de uso doméstico y tarjetas de crédito, Java está en todas partes.

CARACTERÍSTICAS BÁSICAS DEL LENGUAJE DE PROGRAMACIÓN JAVA.

Java es un lenguaje de programación de propósito general utilizado en el desarrollo de programas de software, en especial para internet y actualmente se encuentra en numerosas aplicaciones, dispositivos, redes de comunicaciones, etcétera. Pero no solo es un lenguaje de programación, sino que también constituye una plataforma completa para el desarrollo de software, posee una biblioteca gigantesca de clases y aplicaciones con numerosos códigos reutilizables y un entorno de programación que proporciona

servicios tales como seguridad, portabilidad entre sistemas operativos y recolección automática de basura.

Java tiene un buen lenguaje, un entorno de ejecución de alta calidad y una biblioteca extensa, que lo caracteriza como el lenguaje de programación por excelencia, lo que permite utilizarlo tanto en la enseñanza como en la investigación y el desarrollo profesional de todo tipo de software. Los diferentes tipos de aplicaciones de Java son:

- **Aplicación independiente**: se denominan aplicaciones de área de trabajo o basadas en ventanas y son programas que deben introducirse en cada máquina. Las instancias AWT y Swing se utilizan para hacer aplicaciones independientes.

- **Aplicación web**: es una aplicación que se ejecuta del lado del servidor y crea una página dinámica. En la actualidad JSP, Spring, JSF y otras tecnologías se utilizan para crear este tipo de aplicaciones.

- **Aplicación empresarial**: por ejemplo, aplicaciones bancarias que tienen ventanas de seguridad, ajuste de carga y agrupación. En Java, EJB se utiliza para estas aplicaciones.

- **Aplicación móvil**: una aplicación hecha para teléfonos celulares. En la actualidad Android y Java ME se utilizan para crear aplicaciones móviles.

Algunas características más sobresalientes de Java se describen a continuación:

- **Arquitectura neutral**. El compilador del lenguaje Java crea un objeto de arquitectura neutral, que proporciona el código que se ensambla en numerosos procesadores, con la cercanía del marco de tiempo de ejecución Java.

- **Lenguaje orientado a objetos**. A este tipo de programación la caracterizan los conceptos Objetos, Clases, Encapsulación, Herencia y Polimorfismo. La estructura general de un programa consiste en un conjunto de objetos y cada

objeto se compone por datos y métodos, donde cada método está formado por instrucciones. Este tipo de programación permite manejar mejor la complejidad de los programas y la reutilización de código porque permite una mayor pulverización de los programas a través de los objetos de una forma más eficiente.

- **Disponibilidad de un amplio conjunto de bibliotecas**. Otra fortaleza de Java es que incluye bibliotecas de clases incorporadas; dichos paquetes vienen con los entornos de desarrollo JDK y contienen centenares de clases integradas con millares de métodos.

- **Interpretado**. Para que los programas sean independientes de la máquina y se puedan portar y ejecutar fácilmente, se introdujo la máquina virtual. Un compilador Java traduce un programa fuente en Java a bytecode, el lenguaje máquina de la máquina virtual de Java (JVM), independiente del tipo específico de CPU. El intérprete de Java traduce cada instrucción en bytecode en el tipo específico del lenguaje máquina del CPU y a continuación ejecuta la instrucción.

- **Robusto**. Java fue diseñado para crear software altamente fiable, intenta eliminar las circunstancias inclinadas por errores enfatizando fundamentalmente el tiempo de compilación y la verificación del tiempo de ejecución.

- **Distribuido**. Permite establecer y aceptar conexiones con los servidores o clientes remotos para facilitar la creación de aplicaciones distribuidas ya que proporciona una colección de clases para aplicaciones en red.

ENTORNO DE DESARROLLO PARA EL LENGUAJE DE PROGRAMACIÓN JAVA

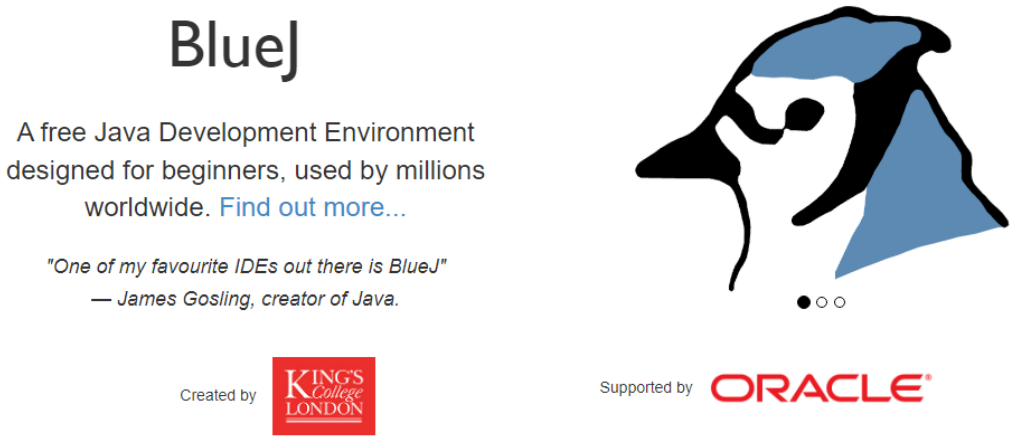
Aprender a programar en Java requiere conocer las técnicas y metodologías descritas anteriormente y también conocer la parte práctica o de laboratorio: edición, compilación

y depuración de los programas fuente; este aprendizaje práctico se realiza de diferentes formas y con herramientas diversas tales como:

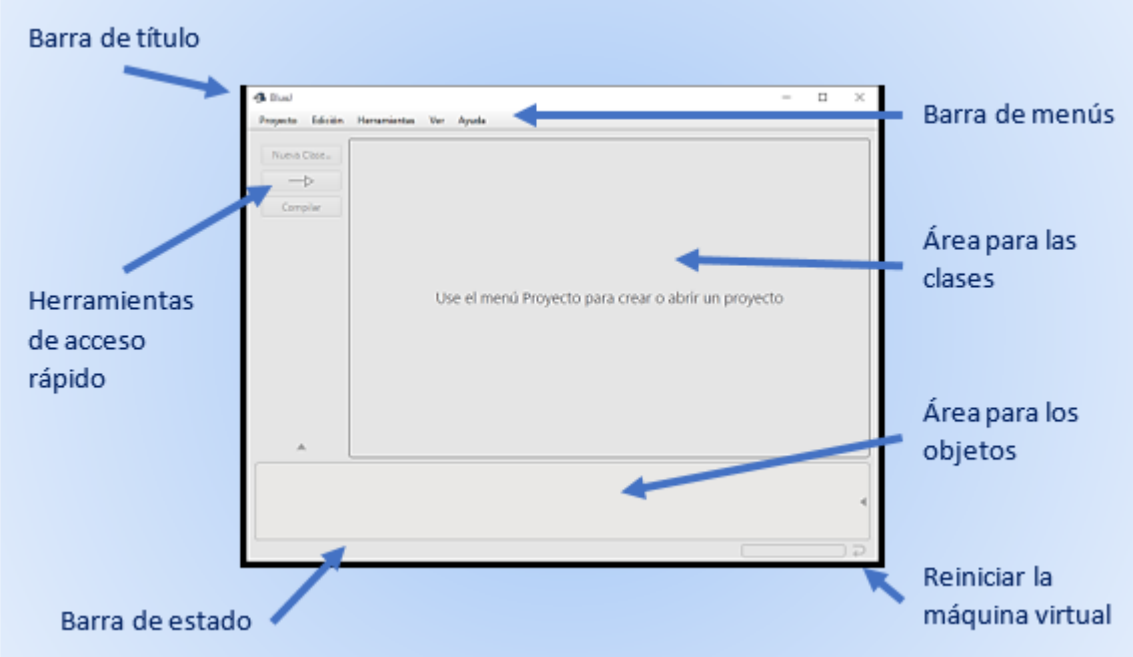
1. El kit de desarrollo Java (JDK, Java development kit)
2. Un entorno de desarrollo integrado o EDI (IED, integrated development environment)
3. Herramientas de línea de órdenes como un editor de textos y un compilador/depurador.

Aunque el JDK es un buen entorno para programación práctica, en la última década han proliferado entornos de desarrollo integrados y profesionales que se convirtieron en herramientas potentes y fáciles de utilizar. Existen diferentes versiones populares de entornos integrados de desarrollo de soporte para Java: Borland, JBuilder, IntelliJ; pero los más adecuados para principiantes son NetBeans y BlueJ.

BlueJ es un IDE, que tiene editor integrado, compilador, máquina virtual y depurador para escritura de programas; también tiene una presentación gráfica para estructuras de clase y soporta edición de texto y gráficos; además de permitir la creación de objetos interactivos, pruebas interactivas y construcción de aplicaciones incrementales. Se puede descargar de la página https://www.bluej.org/ para Windows, Mac OS X, Ubuntu/Debian y otros. La versión Standalone.zip es portable para memorias USB. Se descarga la versión deseada e instala como cualquier programa.

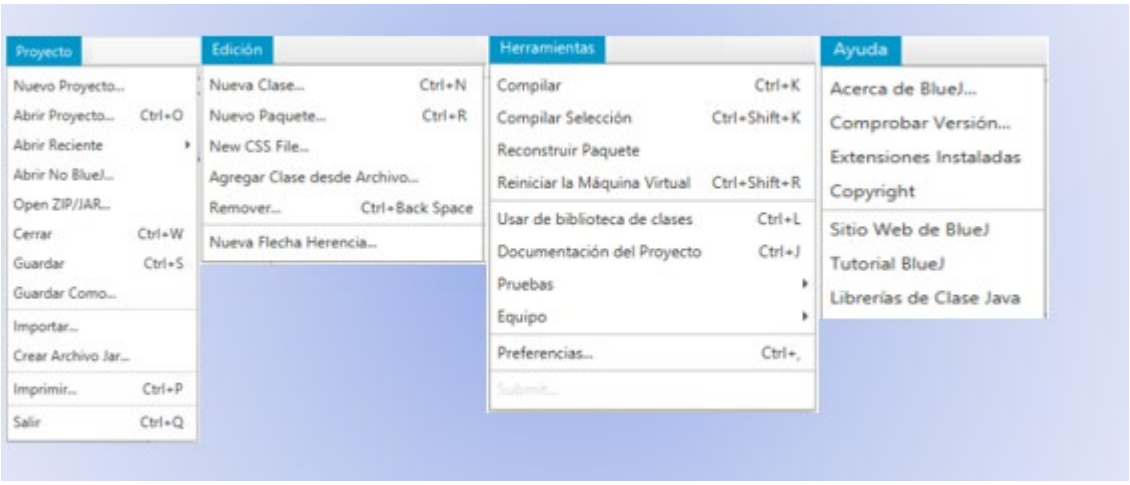


Al abrir por primera vez este programa se visualiza la siguiente ventana en donde se describen sus elementos.



En la barra de título se presenta el nombre del programa BlueJ y el nombre del proyecto en el que se está trabajando. La barra de menús contiene los diferentes comando que se pueden utilizar. Las herramientas de acceso rápido son botones que permiten crear una nueva clase, insertar una relación de herencia y compilar el programa. El área de las clases esta vacía hasta que se crea un nuevo proyecto, después nos muestra el icono de documentación y las clases representadas como cuadros amarillos con su nombre. El área de los objetos los muestra como cuadros rojos cuando se crean. La barra de estado indica cuando se inicializa la máquina virtual de Java y el botón para reiniciar la máquina virtual permite detener la ejecución de un programa.

A continuación se explican las opciones más empleadas de los menús:



MENÚ PROYECTO

- ***Nuevo Proyecto***. Al seleccionar esta opción se le asigna un nombre, lo que generara una carpeta y ahí se guardarán todos los archivos generados por el proyecto.

- ***Abrir Proyecto***. En este comando se debe seleccionar la carpeta que corresponde al proyecto que se quiere abrir.

- ***Abrir Reciente***. Permite abrir un proyecto de la lista de los proyectos usados recientemente.

- ***Cerrar***. Cierra el proyecto en uso.

- ***Guardar***. Almacena la última versión del proyecto en la misma ubicación donde se encuentra el original.

- ***Guardar Como***. Almacena una copia del proyecto con otro nombre.

- ***Salir***. Cierra el proyecto en uso y termina el programa.

MENÚ EDICIÓN

- ***Nueva Clase***. Permite crear una nueva clase solicitando el nombre que se le asignará.

- ***Remover***. Permite eliminar el elemento seleccionado.

MENÚ HERRAMIENTAS

- ***Compilar***. Sirve para compilar el programa fuente.

- ***Compilar Selección***. Sirve solo para compilar la selección.

MENÚ AYUDA

- ***Acerca de Blue***. Mostrará la información del programa y su versión.

- ***Tutorial BlueJ***. Abre la documentación en Internet de BlueJ.

1. Responde las siguientes preguntas.

a) ¿Cuál es la empresa creadora del lenguaje Java?

_____

b) ¿En qué lenguaje se basó originalmente el proyecto Green?

_____

d) ¿Qué es HotJava?

_____

e) ¿En qué año el lenguaje Java tomó ese nombre?

_____

f) ¿Qué es el bytecode?

_____

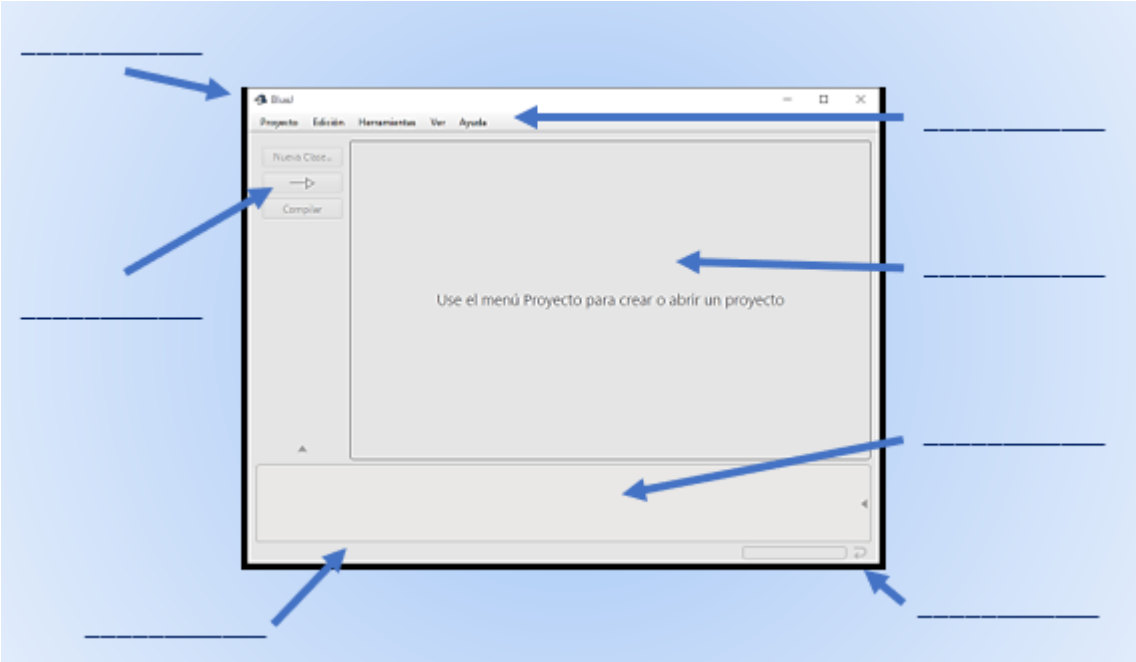
g) ¿Cuál es la característica de Java que permite conexiones con los servidores o clientes y proporciona una colección de clases para aplicaciones de red?

_____

h) ¿Qué realiza el compilador de Java?

_____

2. Escribe los elementos del entorno del IDE BlueJ



Sección 3.5 Implementación de un Programa con el Lenguaje de Programación en Java

Aprendizaje:

12. Realiza programas empleando el método de salida de datos.
13. Realiza programas empleando la Clase Scanner para la entrada de datos.

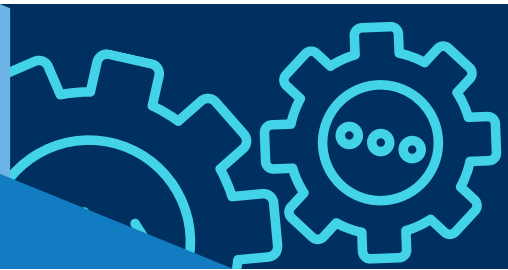
Temática

Pasos para implementar un programa con el lenguaje de programación Java y el entorno de desarrollo.

- Creación de un proyecto.
- Declaración de la Clase.
- Método main.
- Empleo de los métodos System.out. print y System.out.println.
- Errores sintácticos y lógicos.
- Ejecución del programa.

Introducción de datos desde el teclado.

- La Clase Scanner.
- Definición del objeto de la Clase Scanner.
- Método System.in.
- Tipos de datos primitivos.



PASOS PARA IMPLEMENTAR UN PROGRAMA CON EL LENGUAJE DE PROGRAMACIÓN JAVA Y EL ENTORNO DE DESARROLLO.

Los pasos para escribir un programa en BlueJ son los siguientes:

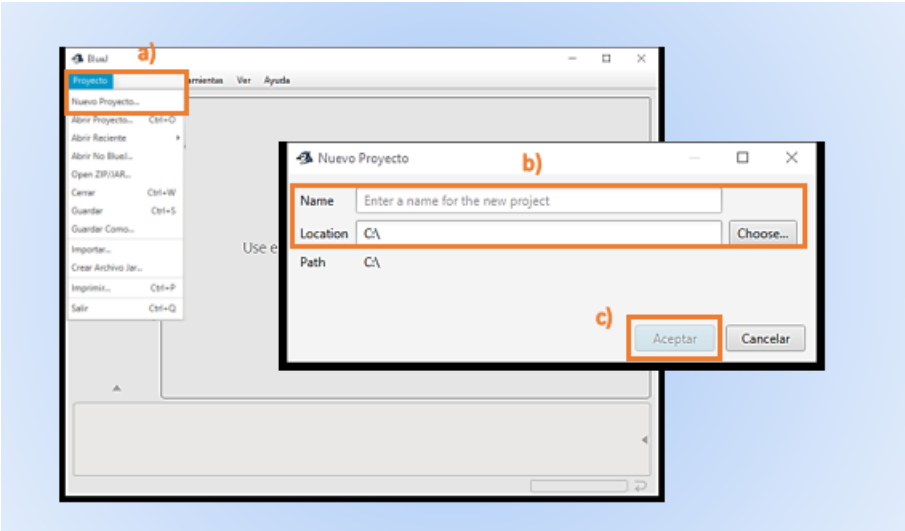
1. Creación de un proyecto.
2. Declaración de la Clase.
3. Escribir el método main.
4. Escribir un mensaje.
5. Compilar el programa.
6. Ejecución del programa.

A continuación, se explica que es un proyecto, una clase, el método principal, los métodos para mostrar mensajes en pantalla y sus consideraciones, los errores sintácticos y lógicos, así como la explicación de cada paso para conseguir el objetivo.

1. CREACIÓN DE UN PROYECTO

Proyecto. Es una carpeta donde se guardarán todos los archivos, para crearlo realiza los siguientes incisos:

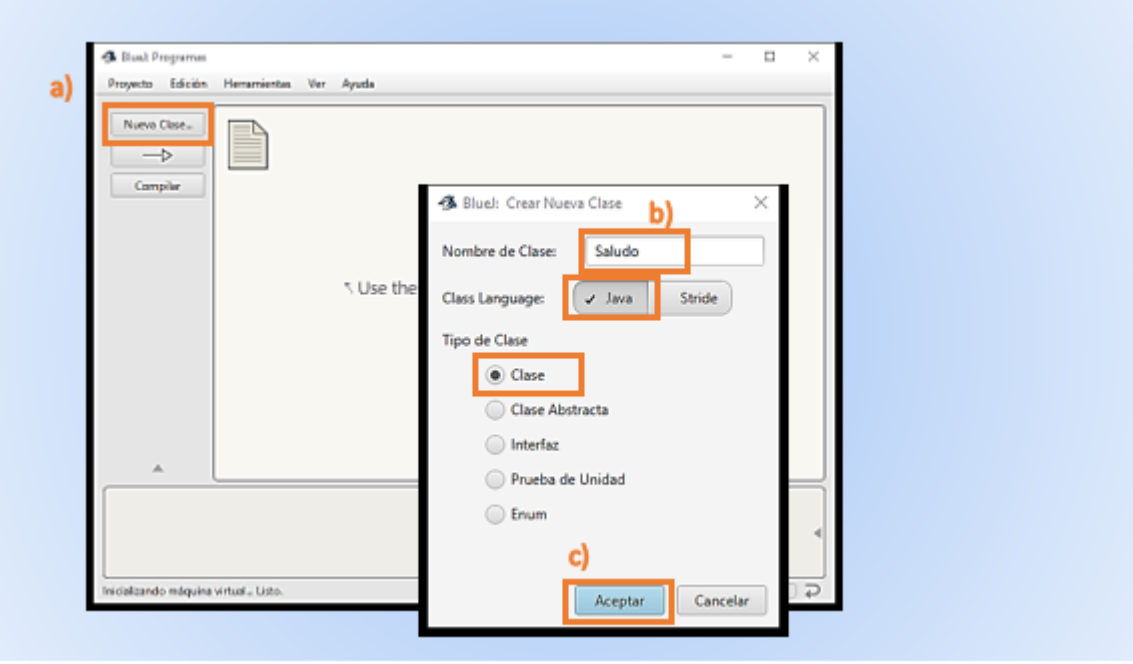
a) Abrir el programa BlueJ y seleccionar el menú Proyecto, opción Nuevo Proyecto.
b) En la ventana que aparece escribir el nombre del proyecto e indicar la ubicación en la computadora dando clic en el botón Chosse.
c) Dar clic en el botón Aceptar.



2. DECLARACIÓN DE LA CLASE.

Clase: contiene datos y métodos para su manipulación, para declararla realiza los siguientes incisos:

- **a)** Seleccionar la opción Nueva Clase de las herramientas de acceso rápido.
- **b)** En la ventana escribir el nombre de la clase, seleccionar el lenguaje Java y tipo de Clase como Clase.
- **c)** Dar clic en el botón Aceptar.



3. ESCRIBIR EL MÉTODO MAIN.

Método main. Es el método principal que permite ejecutar el programa de Java y tendrá la misma declaración en todos los programas:

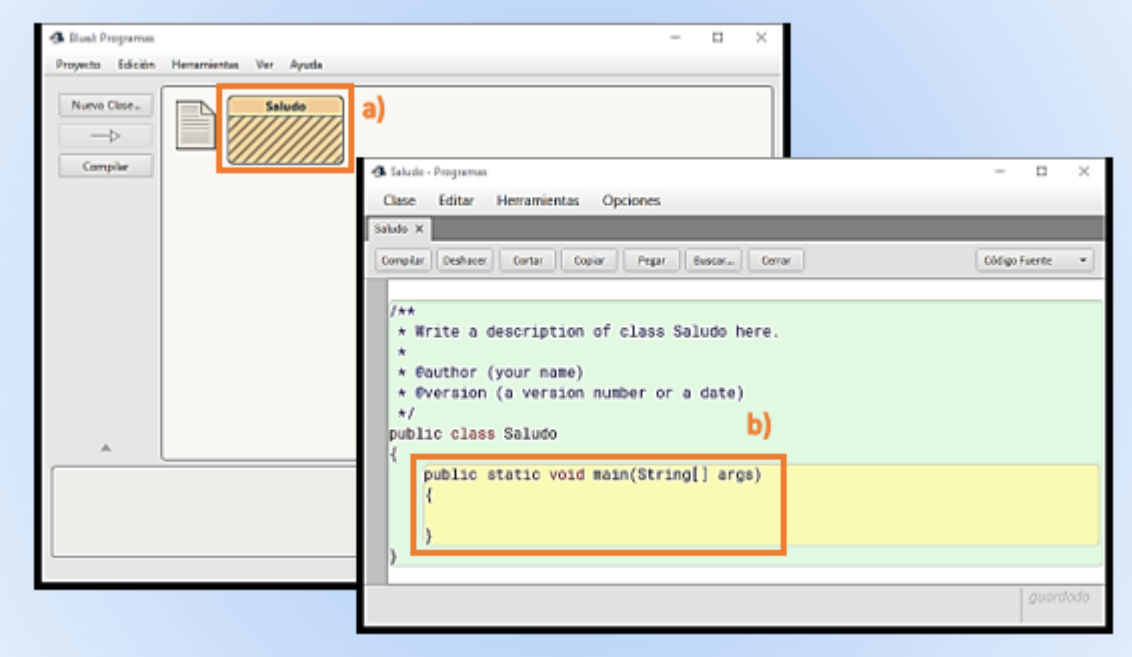
public static void main(String[ ] args)

En donde **public** es el modificador de acceso público para que el método sea accesible a otras clases; **static** es el modificador de acceso para definir el método como de la clase; **void** indica que el método no devuelve ningún valor; **main** es el nombre del método; **String[ ]** tipo de dato arreglo de cadena de caracteres y se utiliza para pasar argumentos en línea de comandos al ejecutar la aplicación; **args** es el nombre del parámetro.

Para escribir el método main en la clase declarada realiza los siguientes incisos:

a) Dar doble clic al icono de la Clase para abrir el editor de texto.

b) Borrar el código que BlueJ muestra y escribir el método main como aparece a continuación.



4. ESCRIBIR EL MENSAJE DE SALIDA.

Empleo de los métodos System.out.print y System.out.println.

En Java la entrada y salida se lee y escribe en flujos; la fuente básica de entrada de datos es el teclado, mientras que la salida es la pantalla. La clase System define dos referencias a objetos static para la gestión de entrada y salida por consola:

- **System.in** para entrada por teclado.
- **System.out** para salida por pantalla.

El objeto out definido en la clase System se asocia con el flujo de salida, que dirige los datos a consola y permite visualizarlos en la pantalla de su equipo.

System.out es una referencia a un objeto de la clase PrintStream y sus siguientes métodos se utilizan con mucha frecuencia:

- **print( )** transfiere una cadena de caracteres al buffer de la pantalla
- **println( )** transfiere una cadena de caracteres y el carácter de fin de línea al buffer de la pantalla.

Con estos métodos se pueden escribir cualquier cadena o dato, empleando además las siguientes consideraciones:

- Con el operador **+** se concatenan ambas cadenas.

```
System.out.println(“Hola mundo ” +” de Java.”);
```

- Como argumento se pueden emplear constantes o variables de los tipos básicos.

```
int x = 500;
System.out.print(x);
```

- Se pueden concatenar cadenas con variables mediante el operador +

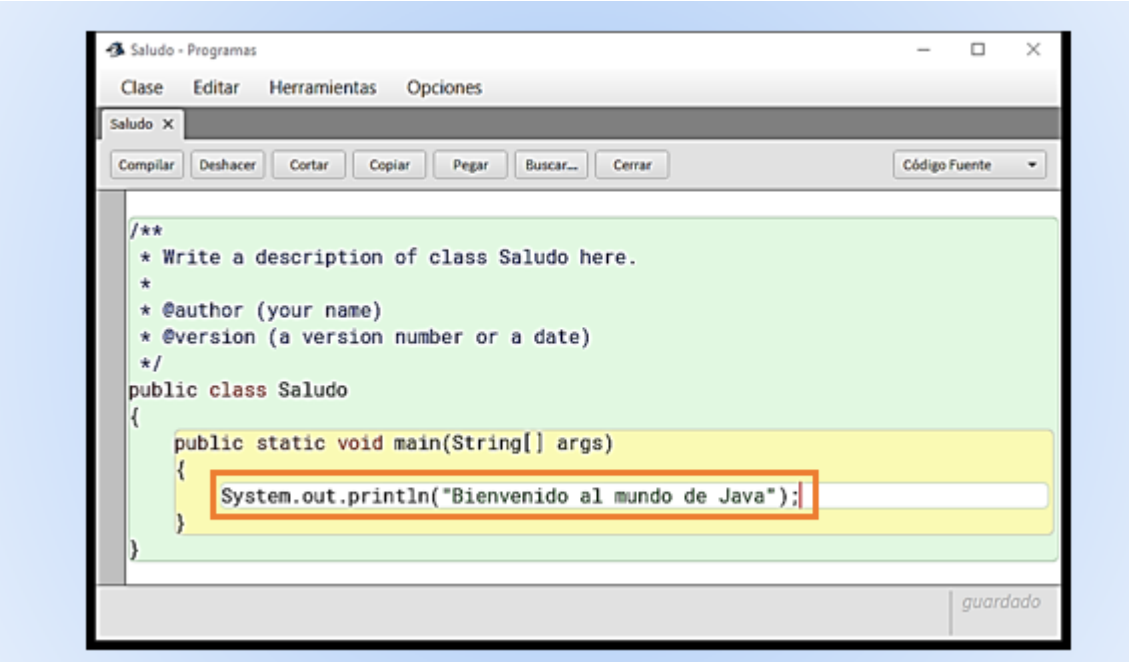
```
int día=10;
int mes=12;
int año=2022;
```

System.out.print("Fecha actual: "+día+"/"+mes+"/"+año);

- Java utiliza secuencias de escape para visualizar caracteres no representados por símbolos tradicionales, tales como \n para nueva línea y \t para tabulación.

System.out.print("\t \t Hola Mundo \n \t ¿Cómo estás?");

Para escribir el mensaje de salida Bienvenido al mundo de Java como se muestra a continuación copia la sentencia en la clase creada.



5. COMPILAR EL PROGRAMA.

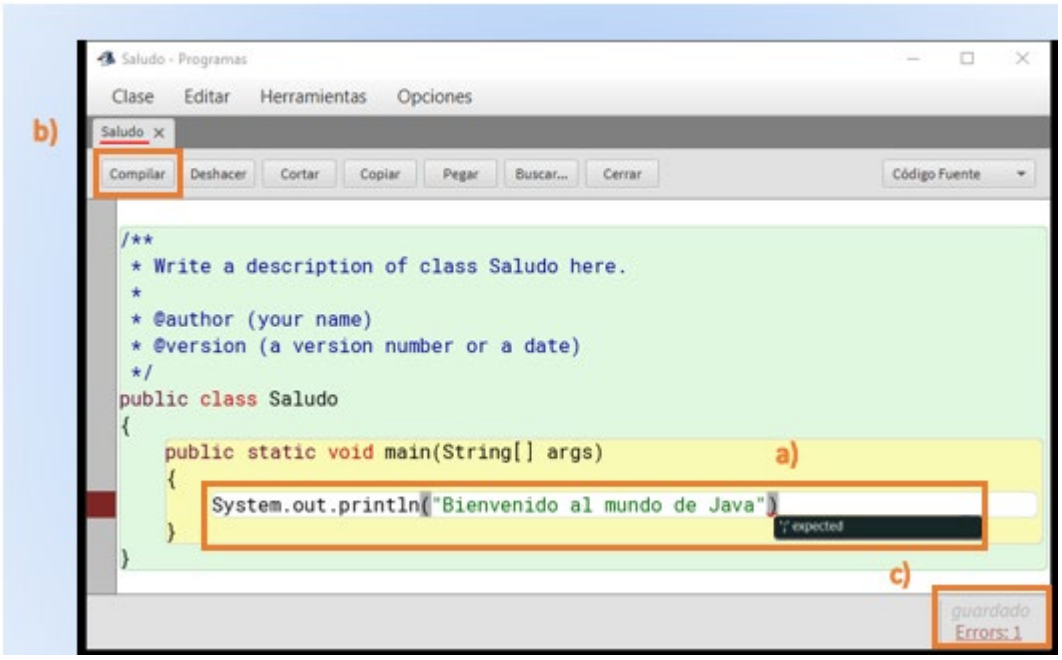
Los errores sintácticos se presentan en la etapa de compilación y ocurren cuando se escribe una sentencia que no va de acuerdo con las reglas de escritura del lenguaje de programación.

Para ejemplificar un error en la compilación del programa y después corregirlo, realiza los siguientes incisos:

a) Borrar el punto y coma (;) al final de la sentencia para mostrar un ejemplo de error.

b) Dar clic en el botón Compilar.

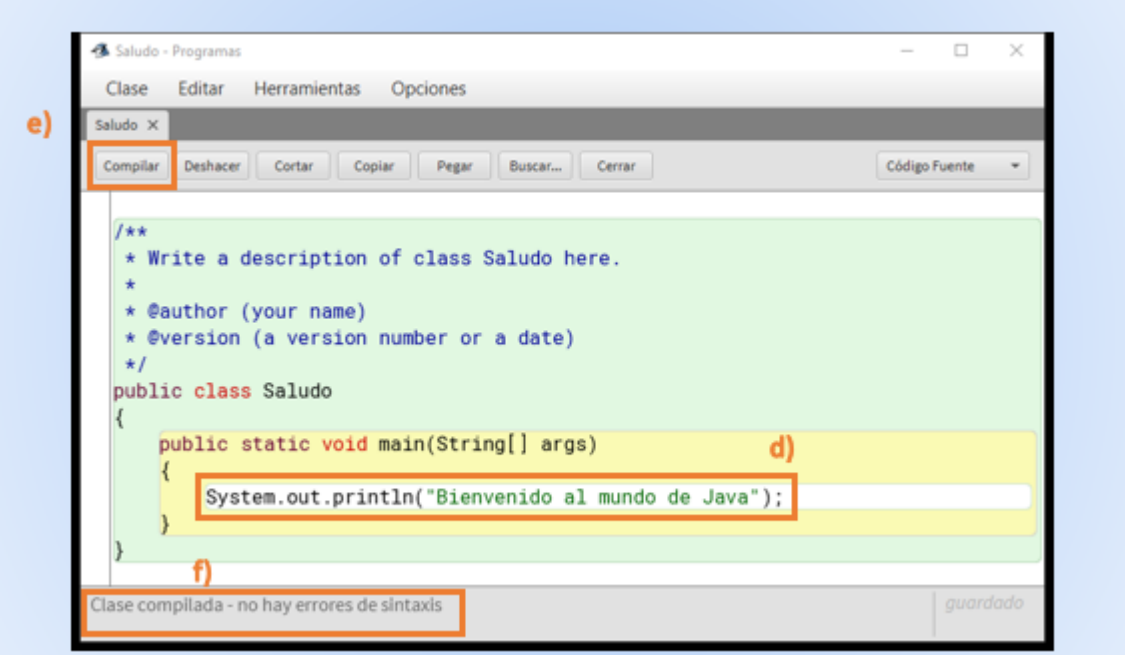
c) Leer la cantidad de errores y la sugerencia para corregirlo.



d) Escribir el punto y coma (;) al final de la sentencia para corregir el error.

e) Dar clic en el botón Compilar.

f) Leer en la barra de estado el mensaje Clase compilada – no hay errores de sintaxis.



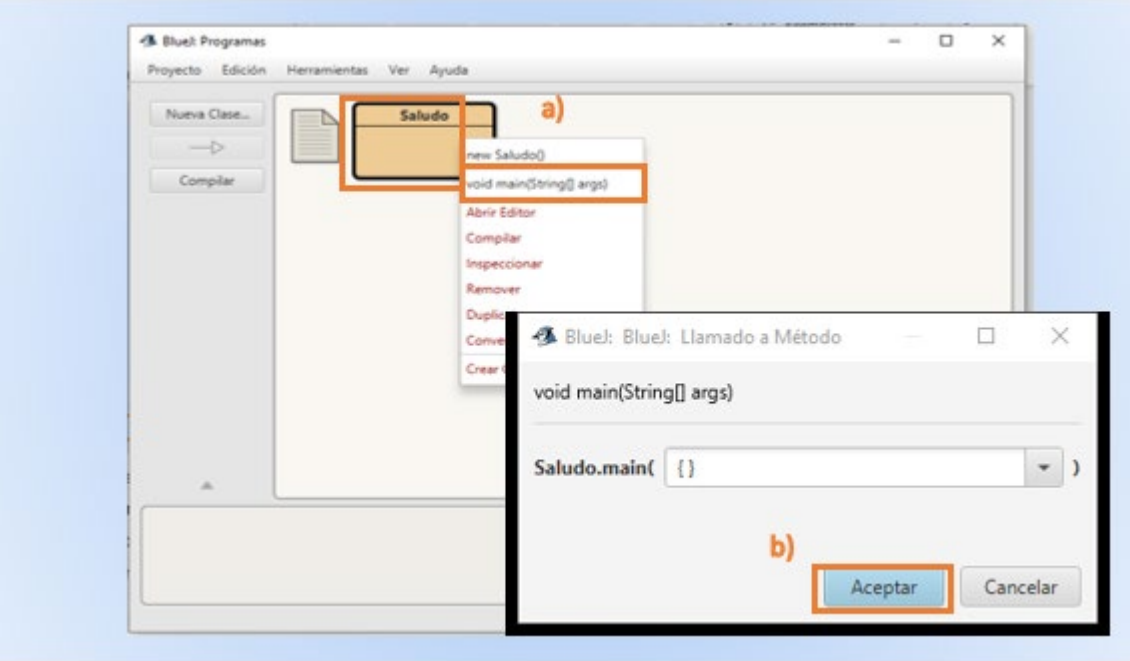
6. EJECUCIÓN DEL PROGRAMA.

Los errores lógicos se observan en la ejecución del programa y ocurren a causa de un mal diseño del programa, esto es que tenga una lógica equivocada y no devuelva el resultado adecuado.

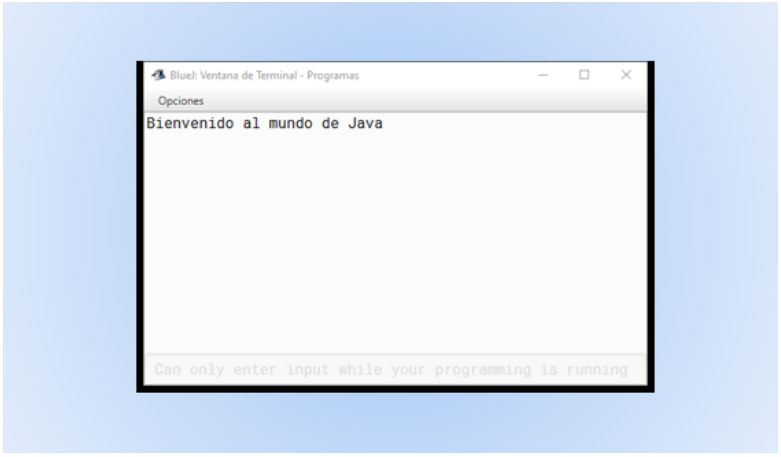
Para ejecutar el programa realiza los siguientes incisos:

a) Dar clic derecho sobre la clase y elegir la opción void main(String[ ] args)

b) En la ventana que aparece, dar clic en el botón Aceptar.



A continuación, se muestra el resultado del programa.



INTRODUCCIÓN DE DATOS DESDE EL TECLADO.

La **clase System** define un objeto de la clase BufferedInputStream cuya referencia resulta en in. El objeto se asocia al flujo estándar de entrada, que por defecto es el teclado; los elementos básicos de este flujo son caracteres individuales y no cadena como ocurre con el objeto out. No resulta práctico el captar la entrada carácter a carácter, es preferible hacerlo línea a línea.

LA CLASE SCANNER.

En la versión 5.0 de Java se incluyó una clase para simplificar la entrada de datos por teclado llamada Scanner, que se conecta a System.in; para leer la entrada a la consola se debe construir primero un objeto de Scanner pasando el objeto System.in al constructor Scanner.

Scanner teclado = new Scanner(System.in);

Una vez creado el objeto Scanner, se pueden utilizar diferentes métodos de su clase para leer la entrada y se explican el los tipos de datos primitivos.

La clase Scanner se define en el paquete java.util y siempre que se utiliza una clase no definida, se utiliza una directiva import cerca del principio del archivo y se escribe la definición de la clase Scanner:

impor java.util.Scanner;

TIPOS DE DATOS PRIMITIVOS.

Un tipo de dato es el conjunto de valores que puede tomar una variable. Los tipos de datos simples o básicos de Java son números, por ejemplo: enteros, flotante, caracteres o boolean.

Tipo	Ejemplo	Tamaño en bytes	Rango mínimo/máximo
char	‘C’	2	‘\0000’ … ‘\FFFF’
byte	-15	1	-128 … 127
short	1024	2	-32768 … 32767
int	42325	4	-2147483648 … 2147483647
long	262144	8	-9223372036854775808 … +9223372036854775807
float	10.5f	4	$3.4*(10^{-38})$ … $3.4*(10^{38})$

Tipo	Ejemplo	Tamaño en bytes	Rango mínimo/máximo
double	0.00045	8	$1.7*(10^{-308})$ … $1.7*(10^{308})$
boolean	true	1 bit	false, true

La forma más simple para declarar variables es poner primero el tipo de dato y a continuación el nombre de la variable; si desea asignar un valor inicial a la variable, el formato de la declaración es:

< tipo de dato > < nombre de la variable > = < valor inicial >

También se pueden declarar múltiples variables en la misma línea:

< tipo de dato > < nombre var1 >, < nombre var2 >, … < nombre varn >

Los métodos que se pueden utilizar con el objeto de la clase Scanner dependen del tipo de dato como se muestra en la tabla.

Tipo de dato	Método para datos	Ejemplo
byte	nextByte( )	byte b=teclado.nextByte( );
double	nextDouble( )	double d = teclado.nextDouble( );
float	nextFloat( )	float f teclado.nextFloat( );
int	nextInt( )	int i = teclado.nextInt( );
long	nextLong( )	long l = teclado.nextLong( );
short	nextShort( )	short s = teclado.nextShort( );
String	next( )	String p = teclado.next( );
String	nextLine( )	String o = teclado.nextLine( );
char	next( ).charAt(0)	char r = teclado.next( ).charAt(0);

Cabe mencionar que el método **nextLine( )** lee una línea completa de entrada, mientras next( ) se emplea cuando se desea leer una palabra sin espacios.

Los pasos para la entrada de datos por teclado utilizando la clase Scanner son los que se enlistan a continuación:

1. Hacer disponible la clase Scanner para utilizarla en el código incluyendo la siguiente línea al comienzo del archivo:

import java.util.Scanner;

2. Crear un objeto de la clase Scanner con la siguiente sentencia:

Scanner teclado = new Scanner(System.in);

3. Enviar un mensaje por pantalla solicitando la información:

System.out.print("Escribe tu nombre: ");

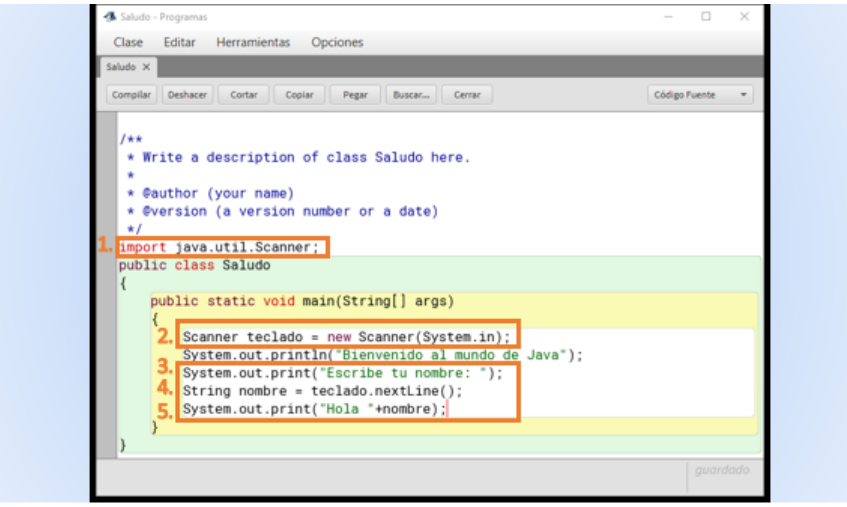
4. Declarar el tipo de dato y utilizar el método correspondiente para leerlo por teclado.

String nombre = teclado.nextLine( );

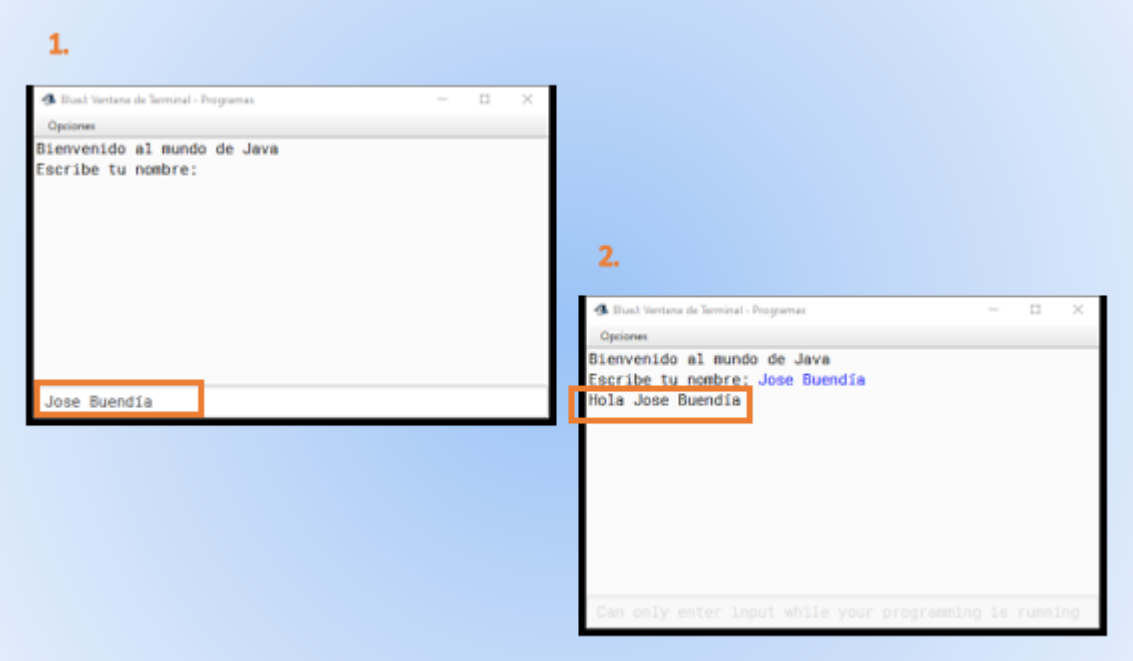
5. Mostrar en pantalla el valor leído por el teclado.

System.out.print("Hola "+nombre);

En seguida, se muestra el orden en que se deben escribir, en el programa Saludo, los pasos antes descritos.



Después de compilar y ejecutar el programa se obtienen las siguientes pantallas. La primera solicita en la consola insertar un valor y al pulsar la tecla enter la segunda pantalla muestra el mensaje concatenado con el valor que se escribió. Nótese que lo escrito por teclado aparece en color azul, diferente a lo escrito en pantalla por el programa.

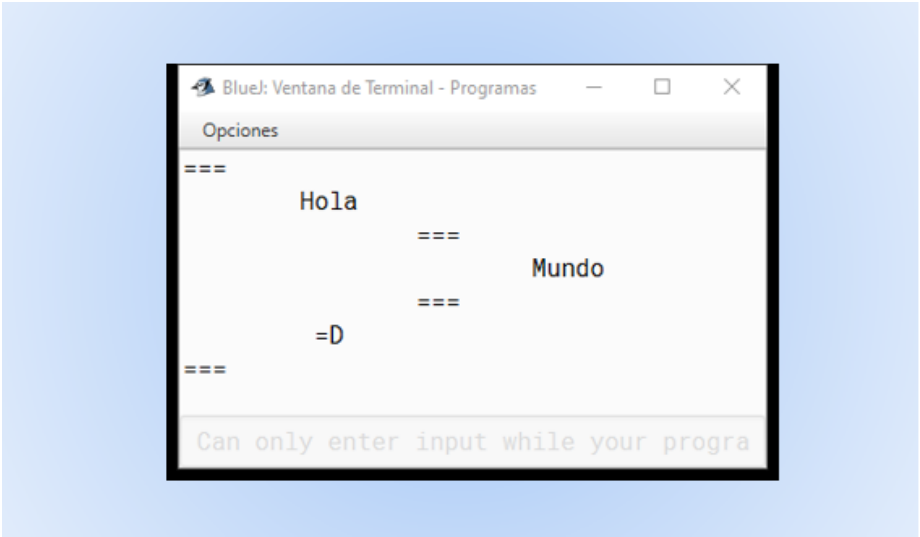

1.
Opciones
Bienvenido al mundo de Java
Escribe tu nombre:
Jose Buendía
2.
Opciones
Bienvenido al mundo de Java
Escribe tu nombre: Jose Buendía
Hola Jose Buendía
Can only enter input while your programming is running

ACTIVIDADES DE APRENDIZAJE
SECCIÓN 3.5

1. **Tacha con color rojo los errores sintácticos y lógicos del siguiente código para obtener la siguiente ejecución del programa.**

```
public clase Errores
{
   Public static void main(string[ ] args)
   {
      system.out.print("===\n \t Hola \t ");
                        System.out.prin("\n \n ===\n \t \n \t Mundo \n");
      Sistem.out.print("\t \t === \n \n =D \t");
      System.out.print("===")
    }
}
```

Ejecución del programa.



Escribe el código correcto en el siguiente cuadro en blanco.

2. **Escribe las palabras que faltan para leer diferentes tipos de datos con la clase Scanner y sus métodos. Obtener la siguiente salida utilizando los métodos print o println, así como las secuencias de escape \t o \n.**

_____

```
public class Datos
{
   public static void main(String[ ] args)
   {
      _________________ teclado = new Scanner(_________________);
      System.out.println("TIPOS DE DATOS POR TECLADO");
      System.out. _________________ ("Escribe tu nombre completo: ");
      _________________nombre = teclado.nextLine();
      System.out. _________________ ("Escribe tu edad: ");
      byte edad = teclado. _________________;
      System.out. _________________ ("Escribe tu peso en kilos: ");
      _________________ peso = teclado.nextFloat();
      System.out. _________________ ("Elige tu sexo: (F o M): ");
      char sexo = teclado. _____________________________;
      System.out. _________________ ("Escribe tu salario: ");
      double salario = teclado. _________________;
      System.out. _________________ ("¿Tienes hijos? (false o true): ");
      _________________ hijo = teclado.nextBoolean();
      System.out. _________________ ("============================");
      System.out. _________________ ("Nombre: "_________________);
      System.out. _________________ ("Edad: "_________________" años");
      System.out. ___________ ("___________Peso: "_________________" kilos");
      System.out. _________________ ("___________Sexo: "_________________);
      System.out. _________________ ("___________Salario: $"_________________);
      System.out. _________________ ("___________Hijos: "_________________);
      System.out. _________________ ("============================");
   }
}
```

Sección 3.6 Sentencia Condicional if y switch

Aprendizaje:

14. Elabora el algoritmo, diagrama de flujo y pseudocódigo para problemas condicionales.

15. Construye programas de computadora que resuelvan problemas condicionales.

16. Elabora el algoritmo, diagrama de flujo y pseudocódigo para problemas condicionales múltiples.

17. Construye programas de computadora que resuelvan problemas que involucren la toma de decisiones múltiples.

Temática

Estructuras condicionales.

- Elaboración de algoritmos, diagramas de flujo y pseudocódigo de problemas condicionales.

Tipo de dato primitivo: Lógico.

Sentencias condicionales:

- Simple if.
- Doble if – else.

Elaboración de algoritmos, diagramas de flujo y pseudocódigo de problemas:

- Sentencias condicionales anidadas.
- Condicionales múltiples.

Sentencias condicionales anidadas.

Sentencias condicionales múltiples.

- Switch.

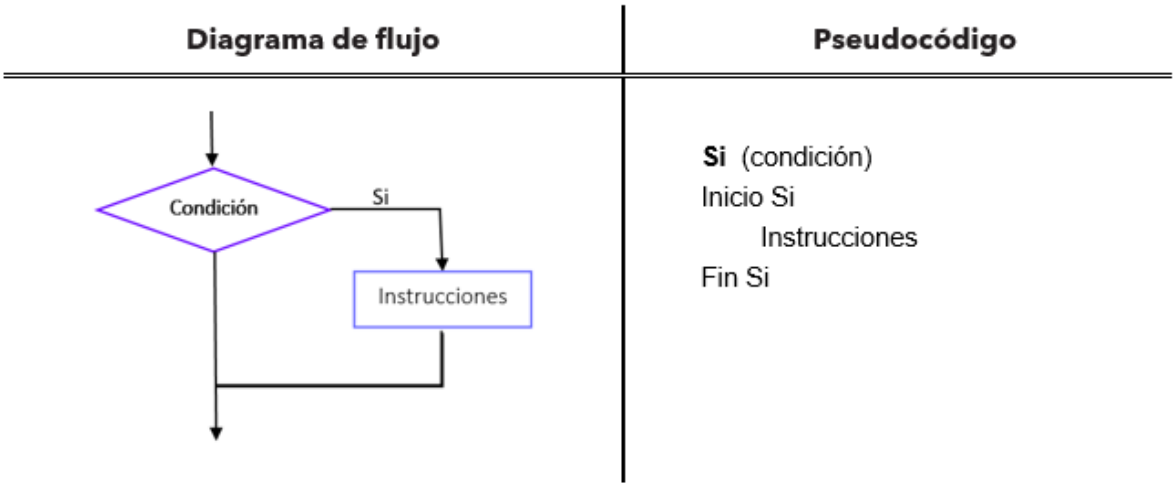
ESTRUCTURAS CONDICIONALES

La solución de un problema condicional depende de la evaluación de una condición. Una condición es una expresión lógica que se puede evaluar dando como resultado un verdadero o falso. Esto es, según el resultado de la evaluación de una condición se van a realizar ciertas acciones.

Hay varias representaciones de una condicional en los algoritmos, se pueden presentar de forma simple, compuesta o anidada.

CONDICIONAL SIMPLE

La representación de la condicional simple en los algoritmos es:



Ejemplo 1. Realizar el diagrama de flujo y el pseudocódigo que calcule el promedio de un alumno, para indicarle el valor de su promedio y **si** aprobó el curso. El nombre de alumno y el valor de las tres calificaciones serán datos introducidos por el usuario, además se conoce que para aprobar el curso se requiere un **promedio mínimo de seis**.

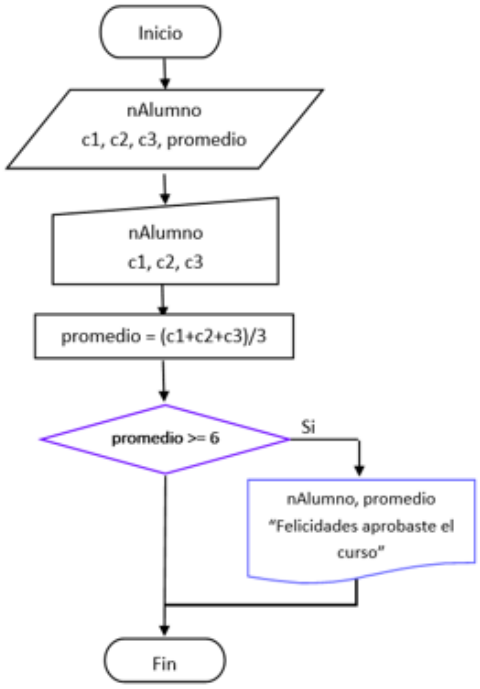
Para determinar la condición primero se deben conocer los valores de las variables o expresiones que se van a relacionar en la comparación. En este caso, es conocer el valor del promedio (se calcula antes de hacer la comparación) y el valor con el que se va a comparar (seis, que se dio a conocer en el planteamiento), conociendo estos datos ahora es verificar el operador relacional que nos permitirá hacer la comparación correcta mayor

o igual (>= para que incluya el valor mínimo que es seis), por ello la condición con la que se trabajará es:

promedio >= 6

Conociendo la condición se procederá al diseño de los algoritmos.

Diagrama de flujo

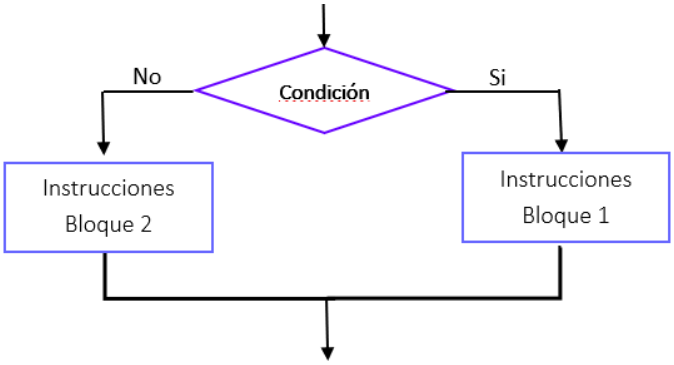


Pseudocódigo

Inicio
Escribir "Introduce el nombre del alumno"
Registrar el valor nombre del alumno (nAlumno)
Escribir "Introduce el valor de la calificación 1"
Registrar el valor de la calificación 1 (c1)
Escribir "Introduce el valor de la calificación 2"
Registrar el valor de la calificación 2 (c2)
Escribir "Introduce el valor de la calificación 3"
Registrar el valor de la calificación 3 (c3)
Calcular *promedio= (c1+c2+c3)/3*
Si *(promedio>=6)*
Inicio si
Escribir "Alumno: " , nAlumno
Escribir "Promedio: " , promedio
Escribir "Felicidades aprobaste el curso"
Fin si
Fin

CONDICIONAL COMPUESTA

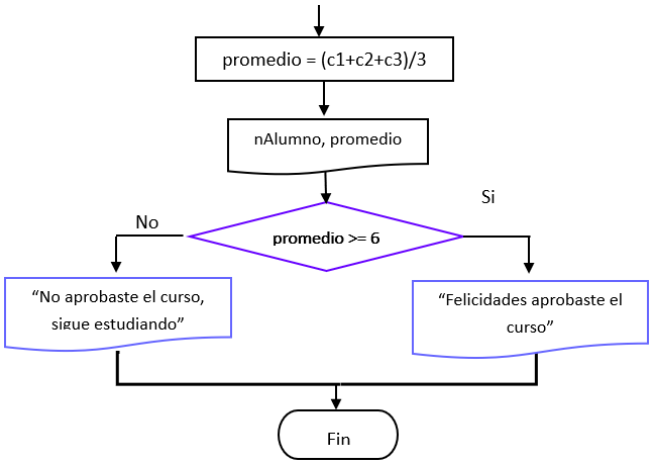
La condicional compuesta en un algoritmo se presenta cuando se ofrecen dos alternativas dada la evaluación de una condición; esto es, se evalúa una condición, si es verdadera se ejecuta un bloque determinado de instrucciones (Bloque 1), si el resultado es falso se ejecuta otro bloque de instrucciones (Bloque 2); la representación en diagrama se flujo y pseudocódigo es:

Diagrama de flujo	Pseudocódigo
	**Si** (condición) Inicio Si Instrucciones (Bloque 1) Fin Si **sino** Inicio Sino Instrucciones (Bloque 2) Fin Sino

Ejemplo 2. Realizar el diagrama de flujo y el pseudocódigo que calcule el promedio de un alumno e indicarle el valor de su promedio y **si** aprobó o **no** aprobó el curso. El nombre de alumno y el valor de las tres calificaciones serán datos introducidos por el usuario, además se conoce que para aprobar el curso se requiere un **promedio mínimo de seis**.

Como puedes observar es muy similar a la situación que se resolvió con la condicional simple, sólo que ahora tenemos dos alternativas para darle al usuario como resultado su promedio y la nota si aprobó o no el curso.

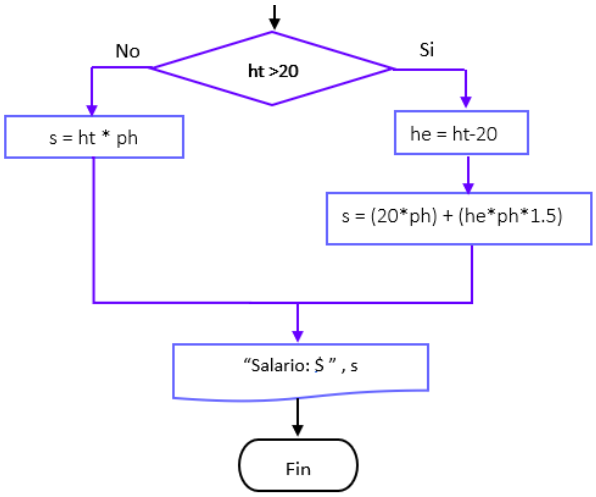
A continuación, se muestra el segmento del diagrama de flujo y pseudocódigo que se modifican para dar solución a la situación actual con estas dos alternativas.

Diagrama de flujo	Pseudocódigo
	**Inicio** **…** Calcular *promedio= (c1+c2+c3)/3* **Escribir** "Alumno: " , nAlumno **Escribir** "Promedio " , promedio ***Si** (promedio>=6)* ***Inicio si*** **Escribir** "Felicidades aprobaste el curso" ***Fin si*** ***sino*** **Escribir** "No aprobaste el curso, sigue estudiando" ***Fin sino*** **Fin**

Como se puede observar en los algoritmos se evalúa la condición - promedio es mayor o igual (>=) a seis, si resulta verdadera se tomará el bloque de instrucciones **Si (verdadero)**; en caso de que resulte falso se ejecutarán las instrucciones del lado **No (falso)**.

De una situación se puede realizar el diseño de un algoritmo para su solución, o se puede contar con un algoritmo (diagrama de flujo o pseudocódigo) para determinar para qué sirve o qué resultados puede dar; para determinar esto último se debe realizar la prueba de escritorio del algoritmo.

Ejemplo 3. Del siguiente fragmento de diagrama de flujo, determinar su salida.
Si ht =24 y ph =220.

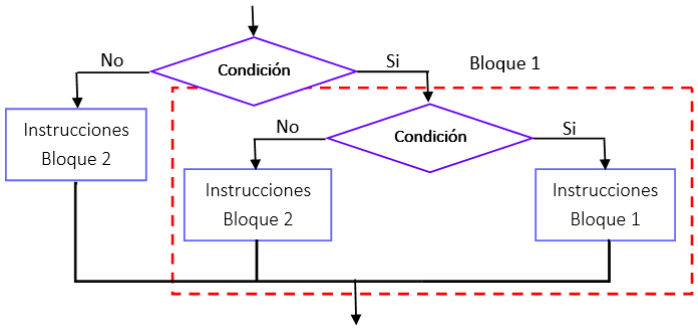


Lo primero que se debe evaluar es la condición con el valor de *ht* de nos proporcionan para determinar que alternativa se va a tomar (Si - verdadero o No – falso).
Sustituimos el valor de ht en la condición para evaluarla, queda

1. 24>20 (24 es mayor a 20)
   El resultado es Si – verdadero por lo que tomamos esa alternativa siguiendo los pasos de ese bloque, sustituyendo los valores de las variables correspondientes.
2. he = 24 – 20
   El resultado es he = 4
3. s = (20*220) + (4*220*1.5)
   s = 4400 + 1320
   s = 5720
   El paso que sigue es la impresión del resultado
4. Salario: $ 5720

CONDICIONAL MÚLTIPLE

Las condicionales múltiples se presentan cuando se tienen más de dos alternativas como resultados, estas se pueden resolver mediante condiciones anidadas, esto es, se utiliza una condición y ya sea en el bloque de instrucciones de verdadero (Si – Bloque 1) o falso (No – Bloque 2), se incluye otra sentencia condicional. A continuación, se muestra el diagrama de flujo y pseudocódigo que muestran un if anidado cuando la condición se cumple.

Diagrama de flujo	Pseudocódigo
	**Si** (condición) Inicio Si (Bloque 1) **Si** (condición) Inicio Si Instrucciones (Bloque 1) Fin Si **Sino** Inicio Sino Instrucciones (Bloque 2) Fin Sino **Sino** Inicio Sino (Bloque 2) Instrucciones (Bloque 2) Fin Sino

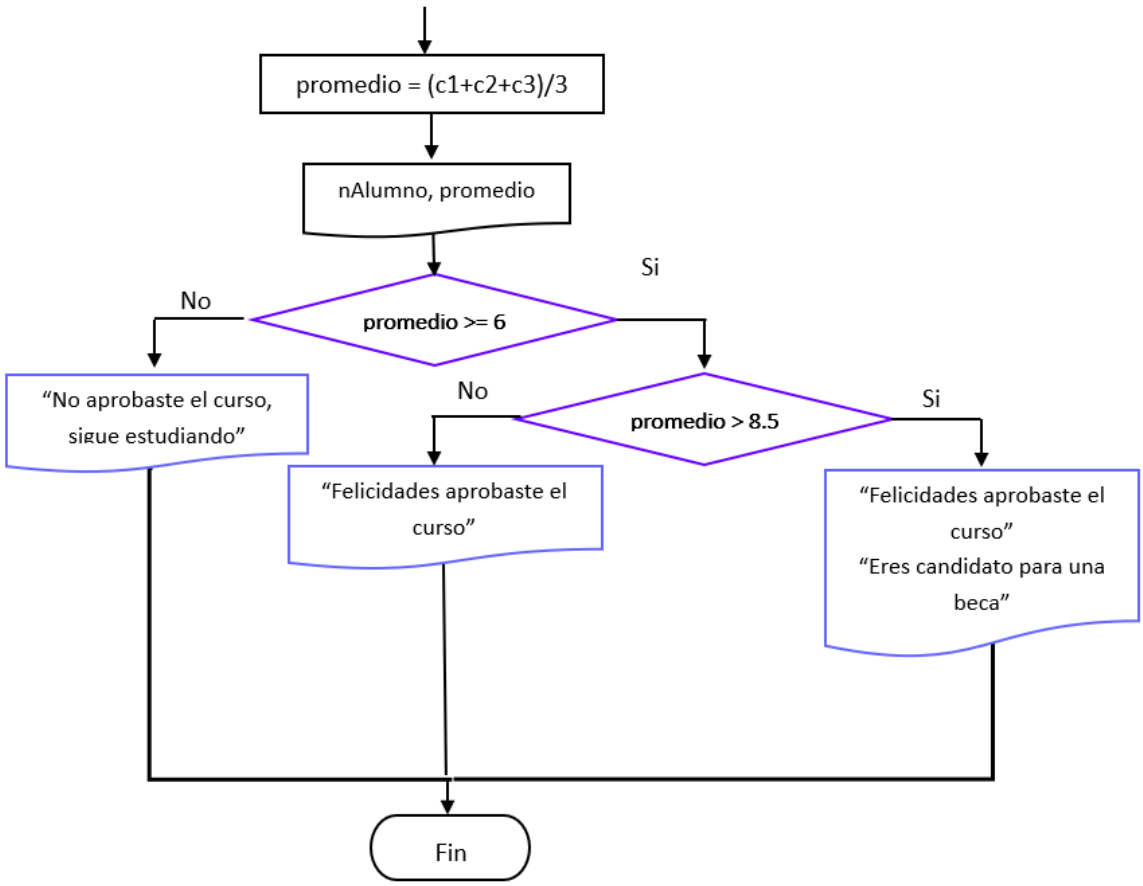
Ejemplo 4. Realizar el diagrama de flujo y el pseudocódigo que calcule el promedio de un alumno e indicarle el valor de su promedio y **si** aprobó o **no** aprobó el curso; también se le tiene que indicar si es candidato para solicitar beca. El nombre de alumno y el valor de las tres calificaciones serán datos introducidos por el usuario, además se conoce que para aprobar el curso se requiere un **promedio mínimo de seis** y para ser candidato para una beca su promedio debe ser mayor a 8.5.

Como se obtuvo la condición en el ejemplo anterior la condición para determinar si aprobó el curso es: promedio>=6.

Y la condición para saber si es candidato para una beca es promedio>8.5

A continuación, se muestra el fragmento de diagrama de flujo y pseudocódigo correspondiente.

Diagrama de flujo



Pseudocódigo

```
Inicio
    …
    Calcular promedio= (c1+c2+c3)/3
    Escribir "Alumno:  "  ,  nAlumno
    Escribir "Promedio " ,  prom

    Si (promedio>=6)
    Inicio si
        Si (promedio>8.5)
        Inicio Si
            Escribir "Felicidades aprobaste el curso"
            Escribir "Eres candidato para una beca"
        Fin si
        Sino
        Inicio sino
            Escribir "Felicidades aprobaste el curso"
        Fin sino
    Fin si
    Sino
    Inicio sino
        Escribir "No aprobaste el curso, sigue estudiando"
    Fin sino

Fin
```

SENTENCIA CONDICIONAL IF

Las estructuras condicionales se utilizan para la toma de decisiones, las cuales consideran la evaluación de una comparación o expresión lógica, en base a su resultado se realizarán ciertas acciones.

La sentencia condicional en java es ***if***, la cual se puede considerar como sentencia simple ***if,*** sentencia compuesta (***if else***) y sentencia ***if*** anidada.

La **sentencia condicional simple *if*** realiza una acción únicamente cuando la expresión a evaluar da como resultado verdadero

La sintaxis de la sentencia condicional ***if*** simple es:

```
if (condición){
    sentencias;
}
```

Ejemplo 5. A continuación, se muestra codificación de la sentencia condicional simple, tomando como referencia el pseudocódigo realizado del ejemplo de la sección de algoritmos de condicional simple.

Pseudocódigo	Codificación
Si *(promedio>=6)* ***Inicio si*** **Escribir** "Alumno: " , nAlumno **Escribir** "Promedio " , promedio **Escribir** "Felicidades aprobaste el curso" ***Fin si***	***if*** (promedio>=6) { System.out.println("Alumno: " + nAlumno); System.out.println("Promedio: " + promedio); System.out.println("Felicidades aprobaste el curso"); }

La codificación de programa completo en BlueJ es:

```
import java.util.Scanner;
public class Aprobado
{
   public static void main(String [] parametro){
       Scanner teclado= new Scanner(System.in);
       String nAlumno;
       double c1, c2, c3, promedio;

       System.out.println("Introduce el nombre del alumno");
       nAlumno=teclado.next();
       System.out.println("Introduce el valor de la calificación 1:");
       c1=teclado.nextDouble();
       System.out.println("Introduce el valor de la calificación 2:");
       c2=teclado.nextDouble();
       System.out.println("Introduce el valor de la calificación 3:");
       c3=teclado.nextDouble();
       promedio=(c1+c2+c3)/3;
       if (promedio>=6)
       {
          System.out.println("Alumno:  "  +  nAlumno);
          System.out.println("Promedio:  "  +  promedio);
          System.out.println("Felicidades aprobaste el curso");

       }
    }
}
```

La ejecución del programa es:

```
BlueJ: Ventana de Terminal - Aprobado
Opciones
Introduce el nombre del alumno
Raúl
Introduce el valor de la calificación 1:
9
Introduce el valor de la calificación 2:
10
Introduce el valor de la calificación 3:
8
Alumno:  Raúl
Promedio:  9.0
Felicidades aprobaste el curso
```

La **sentencia condicional compuesta** ***if - else***, permite elegir entre dos opciones o alternativas posibles, realiza ciertas acciones cuando la evaluación de la condición es verdadera y se realizan otras acciones cuando la evaluación de la condición es falso.

La sintaxis de la sentencia condicional if compuesta es:

```
if (condición){
    sentencias;
}
else{
    sentencias;
}
```

Ejemplo 6. Codificar el pseudocódigo del ejemplo realizado de la sección de algoritmos de condicional compuesta.

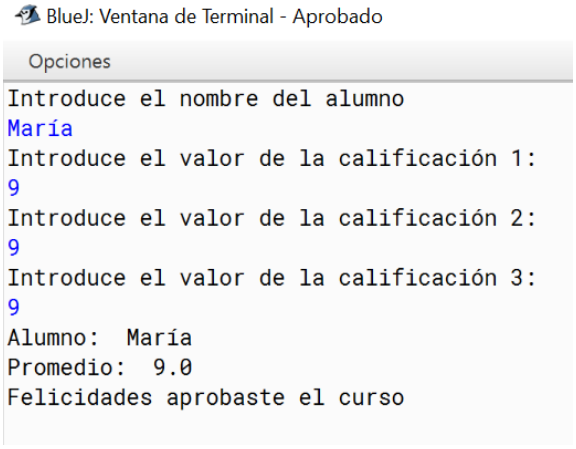
Pseudocódigo	Codificación
Calcular *promedio= (c1+c2+c3)/3* **Escribir** "Alumno: " , nAlumno **Escribir** "Promedio " , prom ***Si*** *(promedio>=6)* **Inicio si** **Escribir** "Felicidades aprobaste el curso" **Fin si** ***Sino*** **Inicio sino** **Escribir** "No aprobaste el curso, sigue estudiando" ***Fin sino***	promedio=(c1+c2+c3)/3; System.out.println("Alumno: " + nAlumno); System.out.println("Promedio: " + promedio); ***if*** (promedio>=6) { System.out.println("Felicidades aprobaste el curso"); } ***else*** { System.out.println("No aprobaste el curso, sigue estudiando "); }

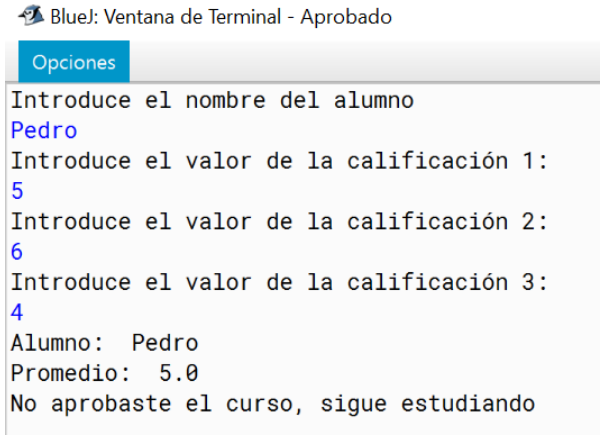
La codificación de la sentencia if compuesta en BlueJ es:

```
promedio=(c1+c2+c3)/3;
System.out.println("Alumno:  "  +  nAlumno);
System.out.println("Promedio:  "  +  promedio);
if (promedio>=6)
{
   System.out.println("Felicidades aprobaste el curso");
}
else
{
   System.out.println("No aprobaste el curso, sigue estudiando");
}
```

Ejemplos de la ejecución.

Caso verdadero	Caso falso





La **sentencia condicional *if múltiple* o anidada** permite la toma de decisión que permiten comparar una variable contra distintos posibles resultados, ejecutando para cada caso una serie de instrucciones específicas.

La sintaxis de la sentencia condicional if múltiple es:

```
if (condición){
        if (condición){
            sentencias;
        }
        else{
            sentencias;
        }
}
else{
      sentencias;
}
```

Ejemplo 7. Codificar el pseudocódigo del ejemplo realizado de la sección de algoritmos de condicional anidadas.

Pseudocódigo	Codificación
Calcular *promedio= (c1+c2+c3)/3* **Escribir** "Alumno: " , nAlumno **Escribir** "Promedio " , prom ***Si** (promedio>=6)* ***Inicio si*** **Si (promedio>8.5)** **Inicio Si** **Escribir** "Felicidades aprobaste el curso" **Escribir** "Eres candidato para una beca" ***Fin si*** ***Sino*** **Inicio sino** **Escribir** "Felicidades aprobaste el curso" **Fin sino** **Fin si** **Sino** ***Inicio sino*** **Escribir** "No aprobaste el curso, sigue estudiando" ***Fin sino***	promedio=(c1+c2+c3)/3; System.out.println("Alumno: " + nAlumno); System.out.println("Promedio: " + promedio); ***if*** (promedio>=6) { ***if*** (promedio>8.5) { System.out.println("Felicidades aprobaste el curso"); System.out.println("Eres candidato a una beca"); } ***else*** { System.out.println("Felicidades aprobaste el curso"); } } ***else*** { System.out.println("No aprobaste el curso, sigue estudiando }

La codificación de la sentencia *if* anidada en BlueJ es:

```
if (promedio>=6)
{
   if (promedio>8.5)
   {
       System.out.println("Felicidades aprobaste el curso");
       System.out.println("Eres candidato a una beca");
   }
   else
   {
       System.out.println("Felicidades aprobaste el curso");
    }
}
else
{
   System.out.println("No aprobaste el curso, sigue estudiando");
}
```

Ejemplos de la ejecución.

Caso verdadero	Caso falso

BlueJ: Ventana de Terminal - Aprobado

Opciones

```
Introduce el nombre del alumno
María
Introduce el valor de la calificación 1:
9
Introduce el valor de la calificación 2:
9
Introduce el valor de la calificación 3:
9
Alumno:  María
Promedio:  9.0
Felicidades aprobaste el curso
```

BlueJ: Ventana de Terminal - Aprobado

Opciones

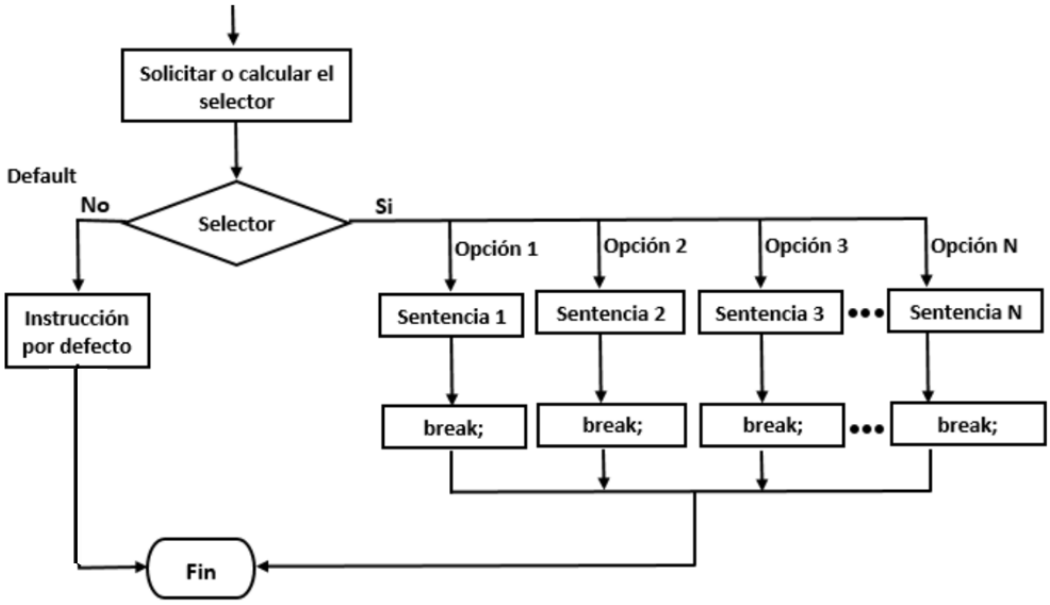
```
Introduce el nombre del alumno
Pedro
Introduce el valor de la calificación 1:
5
Introduce el valor de la calificación 2:
6
Introduce el valor de la calificación 3:
4
Alumno:  Pedro
Promedio:  5.0
No aprobaste el curso, sigue estudiando
```

SENTENCIA CONDICIONAL SWITCH

Una instrucción para los algoritmos de opción múltiple se presenta con la instrucción según el caso, esta instrucción permite comparar el valor de una variable o una expresión con un posible valor que puede tomar para ejecutar un bloque de instrucciones específicas.

A continuación, se presenta el diagrama de flujo y pseudocódigo que la representa.

Diagrama de flujo



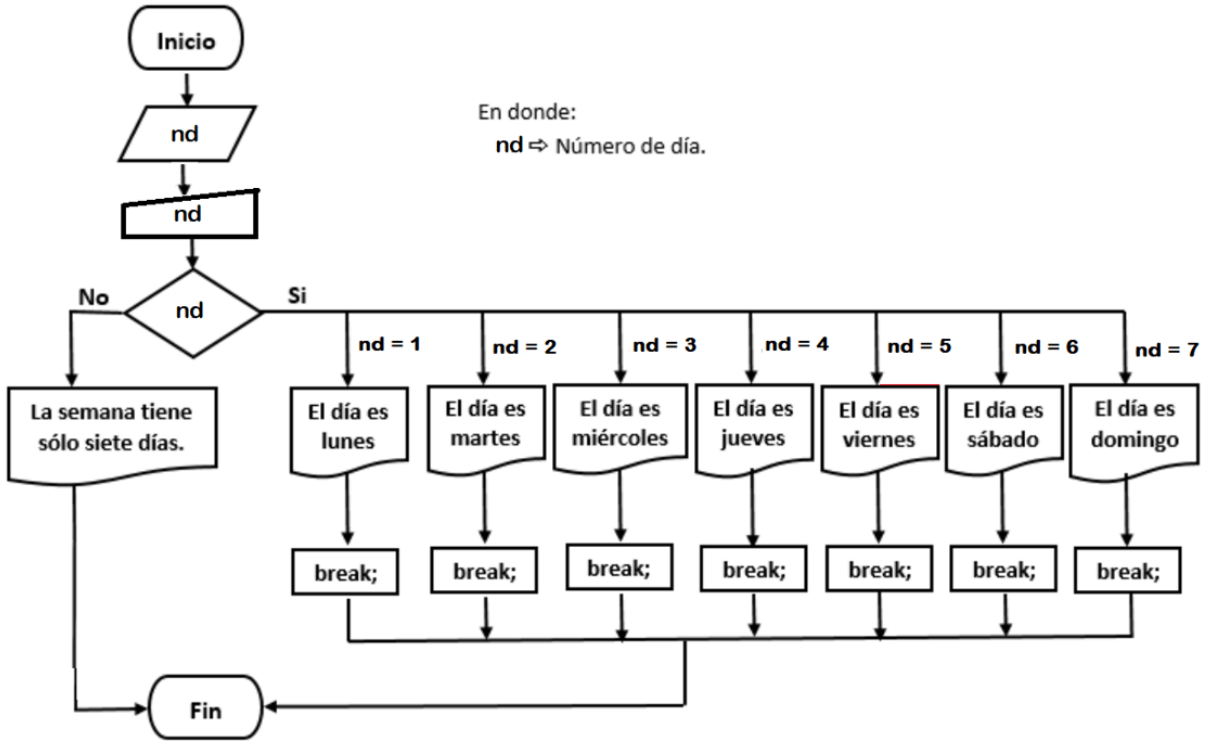
Pseudocódigo

Según [Expresión] Hacer

Caso, valor expresión 1
Instrucción 1
Instrucción 2
Caso, valor expresión 2
Instrucción 1
Instrucción 2…
Caso, valor expresión n
Instrucción 1
Caso, SiNo
Instrucción 1
FinSegún

Ejemplo 8. Elaborar el diagrama de flujo y pseudocódigo, para solicitar un número del 1 al 7 que represente un día de la semana, siendo lunes - 1 y domingo – 7, y nos devuelva el al nombre del día que corresponda.

Diagrama de flujo



Para saber que alternativa se va a ejecutar, se realiza la prueba de escritorio para nd = 4. Se compara el valor del número de día (nd) con los posibles valores que puede tomar, hasta llegar a nd = 4 y se sigue ese bloque de instrucciones, el cual marca que la salida será "El día es jueves" y termina. En otro caso, si nd = 9, este valor se vuelve a comparar con los posibles valores que puede tomar esta variable del lado del Si, y como sólo se tienen los valores del 1 al 7, nos dirigimos del lado del lado del No para ejecutar ese bloque de instruicciones, el cual marca que la salida será "La semana sólo tiene siete días".

En pseudocódigo

```
Inicio
    Escribir "Escribe el número de día de la semana (1-7)"
    Registrar el valor del número del día (nd)
    Según nd Hacer
        Caso, = 1
            Escribir "El día es lunes"
        break;
        Caso, = 2
            Escribir "El día es martes"
        break;
        Caso, = 3
            Escribir "El día es miércoles"
        break;
        Caso, = 4
            Escribir "El día es jueves"
        break;
        Caso, = 5
            Escribir "El día es viernes"
        break;
        Caso, = 6
            Escribir "El día es sábado"
        break;
        Caso, = 7
            Escribir "El día es domingo"
        break;

        Caso, Sino
            Escribir "La semana tiene sólo siete días"
    FinSegún
Fin
```

La **sentencia condicional múltiple switch** permite ejecutar un bloque de instrucciones de varios bloques de instrucciones en función del resultado de la evaluación de una expresión o valor.

La sintaxis de la sentencia switch es:

```
switch (<expresión>) {
case <valor>:
        Sentencias;
break;
case <valor>:
        Sentencias;
break;
…
default:
        Sentencias;
}
```

En donde:

- El tipo de variable de la <expresión> y el <valor> deben coincidir.
- Puede haber tantas cláusulas *case* como se requiera.
- La cláusula *default* es opcional, se utiliza en caso de que ninguno de los valores coincida con el proporcionado.
- El <valor> no puede ser una expresión.

La declaración *break* se usa dentro del switch para finalizar un bloque de instrucciones, esto es, para el *case* de un <valor>. Si se omite la ejecución continuará con el siguiente case.

Funcionamiento:

a. Al llegar a la sentencia *switch*, se debe conocer el valor de la variable o se evalúa la expresión y el resultado se compara con cada <valor> consecutivamente, hasta encontrar uno que coincida.

b. Al encontrar la igualdad con el <valor>, se ejecutan las sentencias de esa cláusula.

c. Si ninguno de los posibles valores (<valor>) coincide, entonces se ejecuta la cláusula default si existe.

Ejemplo 9. Codificación del pseudocódigo para solicitar un número del (1 al 7) que represente un día de la semana, siendo lunes - 1 y domingo – 7, y nos devuelva el nombre del día que corresponda, revisado anteriormente.

Pseudocódigo	Codificación
Inicio **Escribir** "Escribe el número de día de la semana (1-7)" **Registrar** el valor del número del día (nd) **Según nd Hacer** **Inicio Según** Caso, = 1 **Escribir** "El día es lunes" break; Caso, = 2 **Escribir** "El día es martes" break; Caso, = 3 **Escribir** "El día es miércoles" break; Caso, = 4 **Escribir** "El día es jueves" break; Caso, = 5 **Escribir** "El día es viernes" break; Caso, = 6 **Escribir** "El día es sábado" break; Caso, = 7 **Escribir** "El día es domingo" break; Caso, Sino **Escribir** "La semana tiene sólo siete días" **FinSegún**	System.out.println("Introduce el número de día de la semana(1-7)"); nd=teclado.nextInt(); ***switch*** (nd) { *case* 1: System.out.println("El día es lunes"); *break;* *case* 2: System.out.println("El día es martes"); *break;* *case* 3: System.out.println("El día es niércoles"); *break;* *case* 4: System.out.println("El día es jueves"); *break;* *case* 5: System.out.println("El día es viernes"); *break;* *case* 6: System.out.println("El día es sábado"); *break;* *case* 7: System.out.println("El día es domingo"); *break;* *default:* System.out.println("La semana tiene sólo siete días"); }

La codificación del programa completo en BlueJ es:

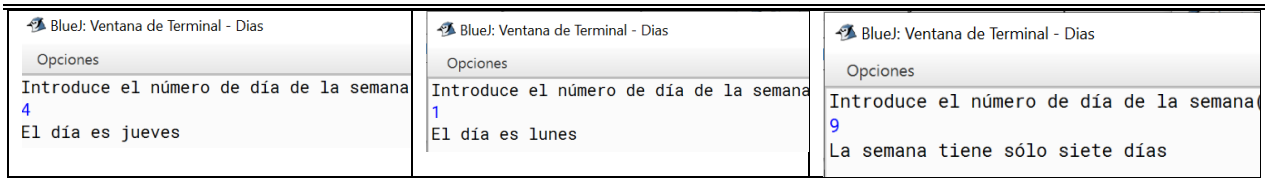
```
import java.util.Scanner;
public class Semana
{
   public static void main(String [] parametro){
       Scanner teclado= new Scanner(System.in);
       int nd;

       System.out.println("Introduce el número de día de la semana(1-7)");
       nd=teclado.nextInt();
       switch (nd)
       {
            case 1: System.out.println("El día es lunes"); break;
            case 2: System.out.println("El día es martes"); break;
            case 3: System.out.println("El día es miércoles"); break;
            case 4: System.out.println("El día es jueves"); break;
            case 5: System.out.println("El día es viernes"); break;
            case 6: System.out.println("El día es sábado"); break;
            case 7: System.out.println("El día es domingo"); break;

            default:
            System.out.println("La semana tiene sólo siete días");
       }
    }
}
```

Nota. Recuerda que el símbolo **;** (punto y coma) es terminador de sentencia, para optimizar espacio los *case* se escribieron en una línea respetando la sintaxis de la sentencia.

Ejemplos de la ejecución del programa

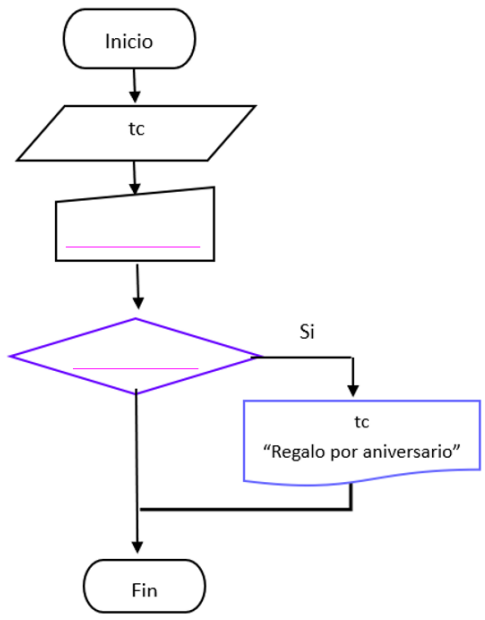


1. Completa el diagrama de flujo y pseudocódigo para la siguiente situación:

   En una camisería hay una promoción por aniversario, si el total de compra (tc) del cliente que indica el ticket es superior a S1000 pesos, se debe imprimir en su ticket "Regalo por aniversario".

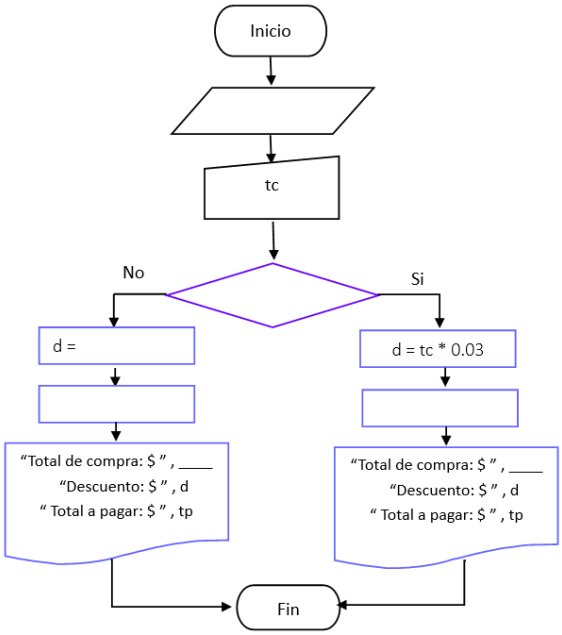
   ¿Cuál es la condición correcta para saber si en el ticket se debe imprimir "Regalo por aniversario"?

   _____

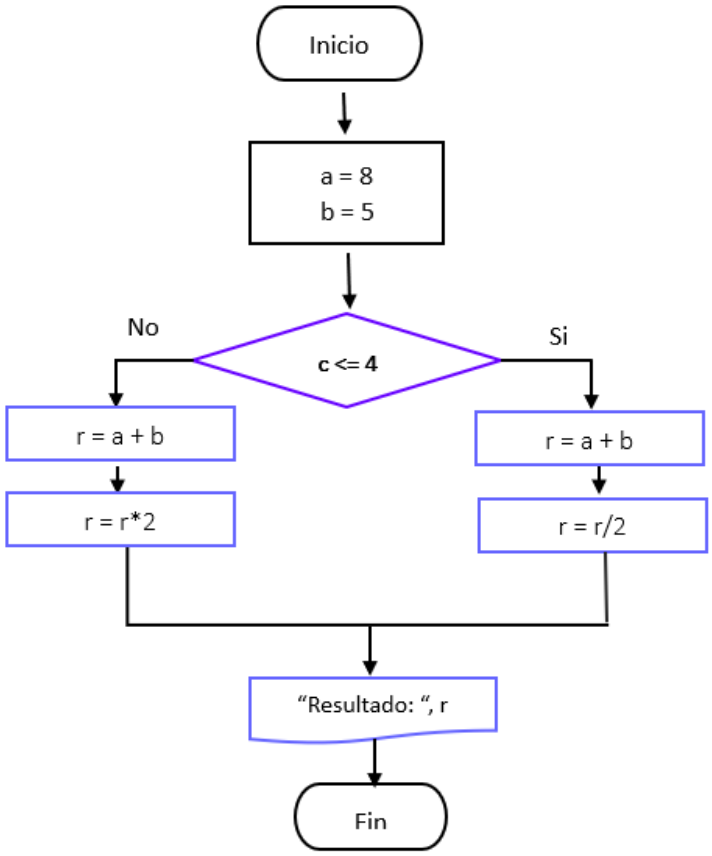
Diagrama de flujo	Pseudocódigo
	**Inicio** **Escribir** "Introduce el total de la compra" **Registrar** el valor del total de la compra (tc) ***Si*** *(_____)* ***Inicio si*** **Escribir** "Total de compra: $ " , tc **Escribir** "_____" ***Fin si*** **Fin**

2. Completa el diagrama de flujo y pseudocódigo para la siguiente situación:

En una camisería hay una promoción de descuentos, en la compra menores de $800 se aplica un descuento del 3%, en las iguales o superiores un descuento del 8%. Imprimir en el ticket, su total de compra (tc), el descuento (d) y el total a pagar (tp).

Diagrama de flujo	Pseudocódigo
	**Inicio** **Escribir** "Introduce el total de compra" (tc) **Registra** el total de compra (tc) ***Si*** *(_____)* ***Inicio si*** **Calcular** d = _____ **Calcular** tp = _____ **Escribir** "Total de compra: $ ", ___ **Escribir** "Descuento: $ ", ___ **Escribir** "Total a pagar: $ ", ____ ***Fin si*** ***Sino*** ***Inicio sino*** **Calcular** d = _____ **Calcular** tp = _____ **Escribir** "Total de compra: $ ", ___ **Escribir** "Descuento: $ ", ___ **Escribir** "Total a pagar: $ ", ____ ***Fin sino*** **Fin**

3. Realiza la prueba de escritorio del siguiente diagrama de flujo para indicar el resultado de su salida, siendo el valor de c = 2.5.

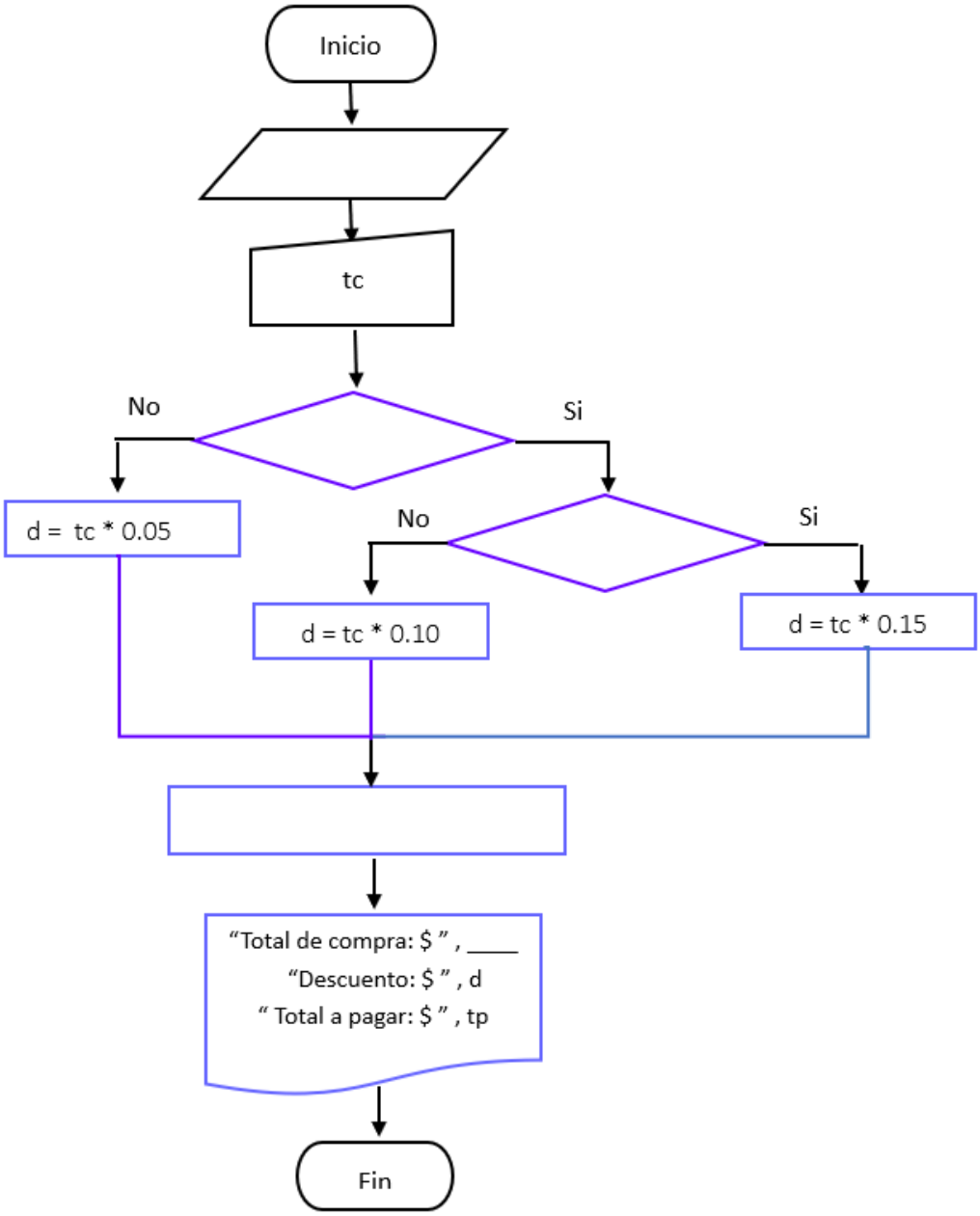


Salida.

4. Completa el diagrama de flujo y pseudocódigo para la siguiente situación:

En una camisería hay una promoción de descuentos, en la compra menores de $800 se aplica un descuento del 5%, en la compra de $800 a $2500 pesos se les aplica un descuento del 10% y las compras de más de $2500 se les descuenta el 15%. Imprimir en el ticket, su total de compra (tc), el descuento (d) y el total a pagar (tp).

Diagrama de flujo



Pseudocódigo

Inicio

Escribir "Introduce el total de compra" (tc)
Registra el total de compra (tc)
Si *(_____)*
Inicio si
Si *(_____)*
Inicio si
Calcular d = _____
Fin si
Sino
Inicio sino
Calcular d = _____
Fin sino
Fin si
Sino
Inicio Sino
Calcular d = _____
Fin sino
Calcular tp = _____
Escribir "Total de compra: $ ", ___
Escribir "Descuento: $ ", ___
Escribir "Total a pagar: $ ", ____

Fin

5. Escribe la condición correcta para que se ejecute la sentencia del if simple

```
if (_____)
{
        System.out.println("El" + numero + "es positivo");
 }
```

6. ¿Cuál es la salida del siguiente fragmento de código?

numero =17; ***if*** *(numero < 0)* *{* *System.out.println("El numero es positivo");* *}* *System.out.println("Fin");*	Salida.

7. En el recuadro escribe el código correspondiente a la sentencia ***if*** para indicarle al usuario si es mayor o menor de edad, la variable que representara la edad es *e*, considerando que una persona es mayor de edad cuando tiene 18 o más años.

8. ¿Cuál es la salida del siguiente fragmento de código, si la *edad = 25*?

```
edad =edad – 10;
if (edad>=18)
 {
        if (edad>=60)
        {
            System.out.println("Perteneces a la tercera edad");
        }
         else
         {
            System.out.println("Eres mayor de edad");
         }
}
 else
 {
          System.out.println("Eres menor de edad");
 }
```

Salida

9. Completa el siguiente pseudocódigo para la situación:

Se requiere de un procedimiento que ofrezca al usuario un menú de figuras: 1. Cuadrado, 2. Triángulo, 3. Círculo, para obtener su área correspondiente. El usuario podrá elegir un número de figura (nf).

Inicio
Escribir "Figuras para calcular el área"
Escribir "1. Cuadrado"
Escribir "2. Triángulo"
Escribir "3. Círculo"
Escribir "Introduce el número de la figura que eliges"
Registrar el valor del número de la figura (nf)
Según _____ Hacer
Caso, = 1
Escribir "Escribe la medida del lado"
Registrar el valor de la medida del lado (l)
Calcular area=l*l;
break;
Caso, = ___
Escribir "Escribe la medida de la base"
Registrar el valor de la medida de la base (b)
Escribir _____
Registrar _____ (h)
Calcular area=
break;
Caso, = 3
Escribir _____
Registrar el valor de la medida del radio (r)
Calcular area = _____
break;

Caso, Sino
Escribir "Esa opción no es válida"
FinSegún

Fin

10. Completa la siguiente codificación

Codificación

```
estacion=teclado.next_______():
 __________(estacion)
 {
     case 1:   System.out.println("Primavera");    break;
     ____2:   System.out.println("Verano");       _______;
     case 3:   System.out.println("Otoño");        _______;
     case _:   System.out.println("Invierno");     break;


  _________
     System.out.println("El año sólo tiene cuatro estaciones");
}
```

11. Realiza la prueba de escritorio del siguiente fragmento de programa

```
int a = 5;
int b = 2;
char op = '*';

System.out.print("El resultado es : ");

switch ( op ) {
      case '+':   System.out.println( a + b );
 break;
      case '-':    System.out.println( a - b );
 break;
      case '*':   System.out.println( a * b );
 break;
      case '/':    System.out.println( a / b );
 break;
default:
    System.out.println("error" );
}
```

Salida:

_____

_____

_____

Sección 3.7 Estructuras Cíclicas

Aprendizaje:

18. Elabora el algoritmo, diagrama de flujo y pseudocódigo para resolver problemas de estructura de ciclo.
19. Construye programas de computadora que empleen la sentencia for.
20. Elabora el algoritmo, diagrama de flujo y pseudocódigo de problemas de ciclo (cíclicos) que satisfagan una condición.
21. Construye programas de computadora que involucren la sentencia while**.**

Temática

Estructuras de ciclo.

- Elaboración de algoritmos de ciclo.
- Representación de algoritmos de ciclo a través de diagramas de flujo.
- Representación de algoritmos de ciclo a través de pseudocódigo.
- Concepto de contador.
- Concepto de acumulador.

Estructuras de control de ciclo.

- Sentencia for.

Estructuras de ciclo.

- Elaboración de algoritmos de ciclo que satisfagan una condición.
- Representación de algoritmos de ciclo a través de diagramas de flujo que satisfagan una condición.
- Representación de algoritmos de ciclo a través de pseudocódigo que satisfagan una condición.

Estructura de control de ciclo.

- Sentencia while.



SENTENCIAS CÍCLICAS

Las estructuras cíclicas nos permiten ejecutar un bloque de instrucciones de forma repetitiva, una cantidad de veces cualquiera o una serie de instrucciones de acuerdo con la evaluación de una condición.

Las estructuras repetitivas para el diseño de algoritmos son: **para desde-hasta, mientras y hacer-mientras.**

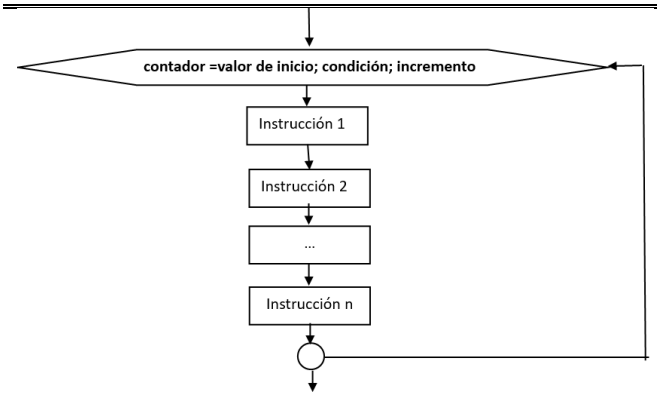
Para el uso de estas estructuras se deben tomar en cuenta los conceptos de contador y acumulador que se involucran en su funcionamiento.

Un contador es una variable de tipo entero que incrementa o decrementa su valor de forma constante, por ello, debe tener un valor de inicio. Se utilizan para contar el número de veces que se repite una acción o bloque de instrucciones.

Un acumulador es una variable numérica que incrementa o decrementa su valor de forma no constante, requiere de un valor de inicio, éste se utiliza para acumular valores en una sola variable.

SENTENCIA *FOR*

La representación en diagrama de flujo y pseudocódigo de la instrucción **para - desde hasta** es:

Diagrama de flujo	Pseudocódigo
	**Para** (c=valor de inicio; condición, paso) Inicio para Instrucciones; Fin_para

En donde:

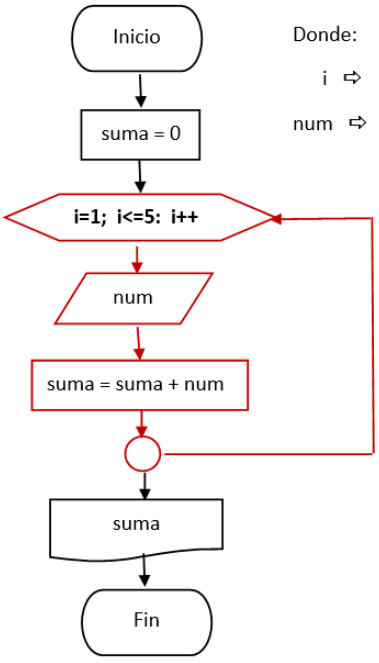
c ⇨ Representa el contador y se le asigna el valor de inicio, puede ser cualquier número que corresponda al tipo de dato asignado.
En lo general el contador es representado como nombre de variable **c** (contador) o **i** (iteración), de tipo entero.

condición En la condición se relaciona al contador con el valor final (número de repeticiones).

paso Este elemento representa el incremento o decremento del tamaño del paso. Se hace uso de los operadores unarios **++** (incremento en 1) o **--** (decremento en 1).

La recomendación para esta instrucción de ciclo es utilizarla cuando se conoce en número de repeticiones de ciertas instrucciones.

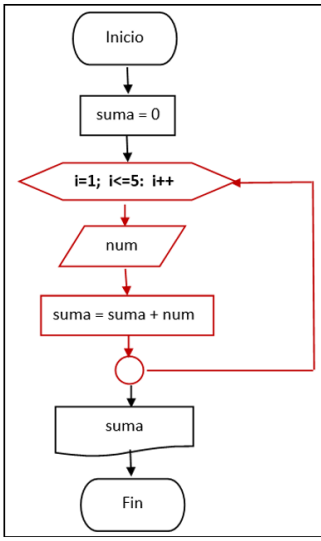
Ejemplo 1. Realiza el diagrama de flujo y pseudocódigo para realizar la suma de cinco números diferentes introducidos por el usuario, haciendo uso de la instrucción ***para***

Diagrama de flujo	Pseudocódigo
	Inicio suma=0; **Escribir** “Algoritmo para sumar 5 números” **Para** (i=1; i<=5; i++) Inicio para **Escribir** “Introduce el valor del número”, i **Registrar** el valor del número (num) **Calcular** suma = suma + num Fin para **Escribir** “La suma total es: ” suma Fin

Para determinar el funcionamiento de la instrucción para se puede realizar la prueba de escritorio.

La prueba de escritorio se realiza haciendo el recorrido de las instrucciones que marca el diagrama de flujo, en primer lugar, nos proporciona el valor inicial de la suma = 0, ese valor se tendrá presente para sustituirlo en donde se solicita. Enseguida, se inicia el ciclo ***para*** en donde el valor inicial del contador (i) es i = 1, la condición a evaluar en cada iteración es i <= 5, si es verdadera se repiten las instrucciones del ciclo, se va a dejar de repetir el ciclo hasta que la condición sea falsa; después viene el incremento en *uno*, el operador unario ++, indica que se debe incrementar en 1 la variable que lo antecede; al cumplirse la condición, se efectúan las instrucciones del ciclo. Por lo tanto, se solicita al usuario el valor del número a sumar (num), después se realiza suma = suma + num, en esta instrucción se irá reemplazando el valor de suma, el valor inicial de esta variable que se tiene desde el principio siendo igual a cero, después su valor va a ser modificado por el resultado de cada suma, y el valor de número va a ir cambiando según el usuario, estas instrucciones se repetirán hasta que se deje de cumplir la condición y se continua con las demás instrucciones del diagrama.

En la siguiente tabla se muestra la prueba de escritorio cada columna significa una repetición.



Prueba de escritorio					
Inicio					
suma = 0					
i = 1; i<=5; i++	i<=5; i++	i<=5; i++	i<=5; i++	i<=5; i++	i<=5; i++
i = 1; 1<=5; 2	2<=5; 3	3<=5; 4	4<=5; 5	5<=5; 6	6<=5; 7
num = 9	num = 7	num = 8	num = 6	num = 10	
suma = suma + num	suma + num	suma + num	suma + num	suma + num	
suma = 0 + 9 = 9	suma = 9 + 7 = 16	suma = 16 + 8 = 24	suma = 24 + 6 = 30	suma = 30 + 10 = 40	
Retorno al ciclo					
					40

Descripción de la tabla.

Columna 1

1. suma = 0	valor inicial de la variable suma inicia el ciclo ***para***
2. i = 1; i <= 5; i++ 1 <= 5; 2	Se sustituye el valor de ***i*** en la condición e incremento en 1 Se evalúa la condición 1 <= 5 , como es verdadero entra al ciclo Se solicita el valor de la variable num, en este caso se asignó el valor de 9
3. num = 9	Se realiza la suma, para ir acumulando la suma de los números introducidos por el usuario
4. suma = suma + num suma = 0 + 9 suma = 9	Se sustituye el valor de suma (paso 1) y num (paso 3) Se actualiza el valor de la variable suma
Se retorna al ciclo y se evalúa la condición, si es verdadera se repiten las instrucciones del ciclo	

Para las columnas 2 a la 5 se repiten las instrucciones de la columna 1, actualizando los valores de las variables *i*, *num* y *suma*.

Columna 6

1. i <= 5; i++ 6 <= 5; 7 suma = 40 **40**	Se retorna al ciclo ***para*** Se sustituye el valor de ***i*** en la condición e incremento en 1 Se evalúa la condición 6 <= 5 , resulta falso, por lo tanto se saltan las instrucciones del ciclo. Se imprime el último valor de suma Resultado

SINTAXIS DE LA SENTENCIA FOR

La sentencia ***for*** se usa en aquellas situaciones en las cuales conocemos la cantidad de veces que queremos que se ejecute el bloque de instrucciones.

La sintaxis de la sentencia es:

```
for (contador =valor de inicio; condición; incremento)
{
        Sentencias;
}
```

Ejemplo 2. Codificar el pseudocódigo correspondiente al ciclo para sumar 5 números.

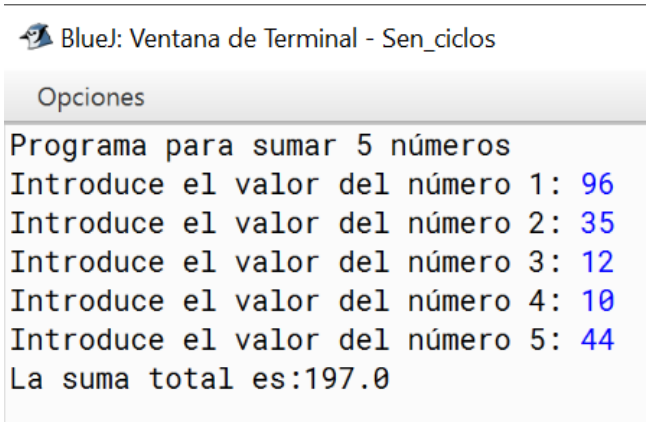
Pseudocódigo
Para (i=1; i<=5; i++) Inicio para **Escribir** “Introduce el valor del número”, i **Registrar** el valor del número (num) **Calcular** suma = suma + num Fin para **Escribir** “La suma total es: ” suma
Codificación
for (i=1; i<=5; i++) *{* *System.out.print("Introduce el valor del número " + i + ": ");* *num=teclado.nextDouble();* *suma=suma+num;* *}* *System.out.print("La suma total es: " + suma);*

El código completo en BlueJ es:

```
import java.util.Scanner;
public class For
{
    public static void main(String [] parametro){
        Scanner teclado= new Scanner(System.in);
        double i;
        double num;
        double suma=0;

        System.out.println("Programa para sumar 5 números");
        for(i=0.1; i<=5; i++){
            System.out.print("Introduce el valor del número " + i + ": ");
            num=teclado.nextDouble();
            suma=suma+num;
        }
       System.out.println("La suma total es:" + suma);
     }
}
```

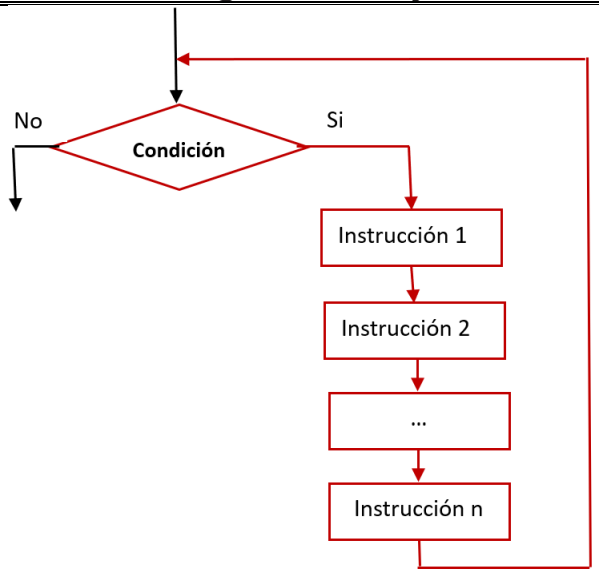
Ejemplo de la ejecución:



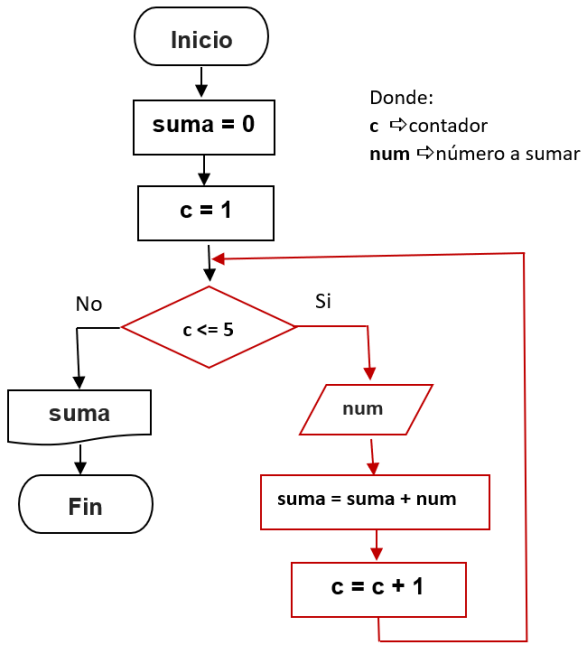
SENTENCIA WHILE

La instrucción mientras nos permite hacer repeticiones cuando se cumple su condición, para que se ejecute este ciclo primero se evalúa la condición si es verdadera entra al bloque de instrucciones que contempla el ciclo, se detiene cuando la condición se deja de cumplir. Este ciclo puede no ejecutarse, si la primera vez que se evalúa la condición es falsa, en este caso, se salta el ciclo.

La representación del diagrama de flujo y pseudocódigo es:

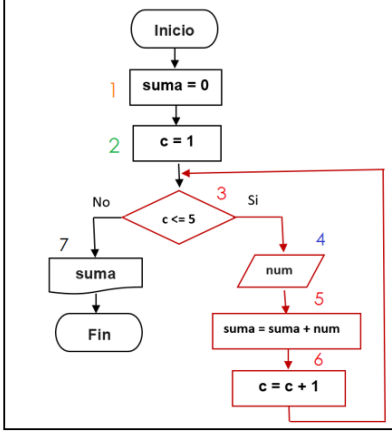
Diagrama de flujo	Pseudocódigo
	**Mientras** [Condición] Hacer Inicio mientras Instrucción 1 Instrucción 2 ... Instrucción n Fin mientras

Ejemplo 3. Realizar el diagrama de flujo y pseudocódigo para sumar 5 números introducidos por el usuario haciendo uso de la instrucción ***mientras***.

Diagrama de flujo	Pseudocódigo
	Inicio suma=0 c = 1 **mientras (c<=5)** Inicio mientras **Escribir** "Introduce el valor del número", c **Registrar** el valor del número (num) **Calcular** suma = suma + num **Calcular** c = c +1 Fin para **Escribir** "La suma total es: " suma Fin

El funcionamiento de la instrucción mientras se puede revisar con la prueba de escritorio de diagrama de flujo.

La siguiente tabla muestra la prueba de escritorio en donde en cada columna se tiene una instrucción del diagrama de flujo y en cada fila se van ejecutando la instrucción que marca la columna correspondiente. Si la condición (3) es verdadera se sigue con el paso 4, sino (falso) se pasa hasta la instrucción 7.



Prueba de escritorio

		Si				No
1	2	3	4	5	6	7
suma = 0	c = 1	c <= 5	num	suma =suma + num	c = c + 1	Suma
suma = 0	c = 1	1 <= 5	4	suma = 0 + 4 suma = 4	c = 1 + 1 c = 2	
		2 <= 5	17	suma = 4 + 17 suma = 21	c = 2 + 1 c = 3	
		3 <= 5	6	suma = 21 + 6 suma = 27	c = 3 + 1 c = 4	
		4 <= 5	9	suma = 27 + 9 suma = 36	c = 4 + 1 c = 5	
		5 <= 5	14	suma = 36 + 14 suma = 50	c = 5 + 1 c = 6	
		6 <=5				**50**

La **estructura repetitiva *while*** se ejecuta mientras una condición se cumpla, en cuanto la condición no se cumple el ciclo deja de ejecutarse. Si la condición se evalúa por primera vez como falsa, el ciclo no se ejecutará. Esta sentencia se utiliza cuando no se conoce el número de veces que las acciones o el ciclo deben repetirse, sin embargo, se puede utilizar en otros casos.

Sintaxis de la sentencia while

```
while (condición) {
  Sentencias;
}
```

Ejemplo 4. Codificación del pseudocódigo para sumar 5 números introducidos por el usuario utilizando la sentencia ***while.***

Pseudocódigo

```
suma=0
c = 1

mientras (c<=5)
Inicio mientras
    Escribir "Introduce el valor del número", i
    Registrar el valor del número (num)
    Calcular suma = suma + num
    Calcular c = c +1
Fin mientras
Escribir "La suma total es: " suma
```

Código

```
int suma=0
int c = 1

while (c<=5)
{
        System.out.print("Introduce el valor del número " + c + ": ");
        num=teclado.nextDouble();
        suma=suma+num;
        c = c+1;
}
System.out.print("La suma total es: " + suma);
```

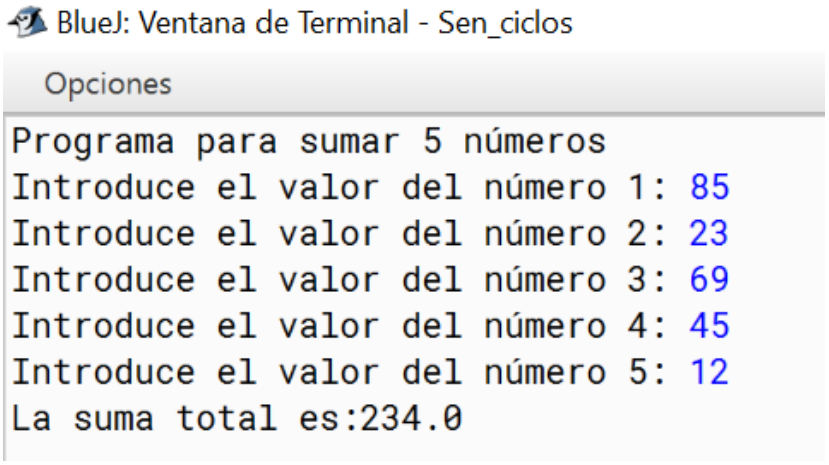
El código completo en BlueJ es:

```java
import java.util.Scanner;
public class While

{
    public static void main(String [] parametro){
        Scanner teclado= new Scanner(System.in);
        double num;
        int c=1;  //Se declara el contador con su valor de inicio
        double suma=0;  //Se declara el acumulador con su valor de inicio

        System.out.println("Programa para sumar 5 números");
        while(c<=5){
            System.out.print("Introduce el valor del número " + c + ": ");
            num=teclado.nextDouble();
            suma=suma+num;
            c=c+1;      //Se incrementa el contador en 1
        }
            System.out.println("La suma total es:" + suma);
     }
}
```

Ejemplo de la ejecución del programa.

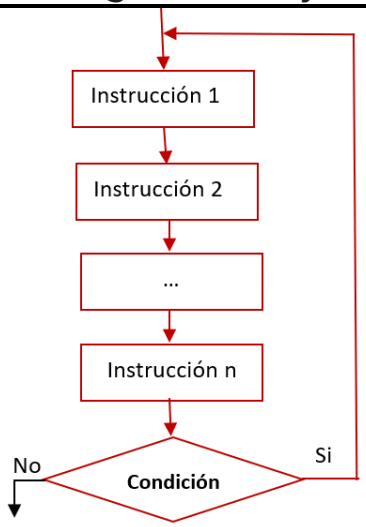


SENTENCIA DO WHILE

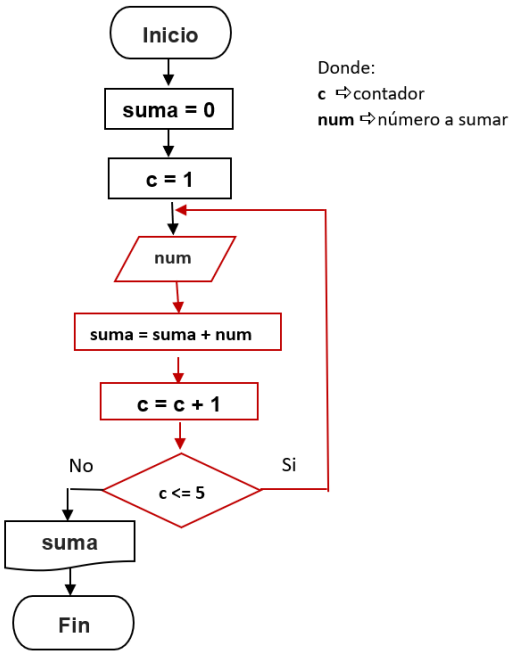
La ***estructura repetitiva*** *do while* se ejecuta mientras una condición se cumpla, en cuanto la condición no se cumple el ciclo deja de ejecutarse. Esta sentencia tiene la característica

que al menos se van a efectuar todas las instrucciones del ciclo, pues se evalúa al final la condición.

La representación de esta sentencia en diagrama de flujo y pseudocódigo es:

Diagrama de flujo	Pseudocódigo
	**Hacer** **Inicio hacer** Instrucción 1 Instrucción 2 ... Instrucción n **Fin hacer mientras** (condición)

Ejemplo 5. Realizar el diagrama de flujo y pseudocódigo para sumar 5 números introducidos por el usuario haciendo uso de la instrucción ***hacer - mientras***.

Diagrama de flujo	Pseudocódigo
	Inicio suma=0 c = 1 **hacer** Inicio hacer **Escribir** "Introduce el valor del número", i **Registrar** el valor del número (num) **Calcular** suma = suma + num **Calcular** c = c +1 Fin hacer **mientras (c<=5)** **Escribir** "La suma total es: " suma Fin

Como se puede ver en el diagrama de flujo anterior las instrucciones que pertenecen al ciclo se ejecutan al menos una vez, porque hasta el final se realiza la comparación, si es verdadera la condición se repite el ciclo, en caso contrario se deja de repetir.

Sintaxis de la sentencia *do while*

```
do{
    Sentencias;
}while (condición);
```

Ejemplo 6. Codificación del pseudocódigo para sumar 5 números introducidos por el usuario utilizando la sentencia ***do while.***

Pseudocódigo

```
suma=0
c = 1

do (c<=5)
Inicio mientras
    Escribir "Introduce el valor del número", c, ":"
    Registrar el valor del número (num)
    Calcular suma = suma + num
    Calcular c = c +1
Fin mientras while (c<=5)
Escribir "La suma total es: " suma
```

Código

```
int suma=0
int c = 1

do
{
        System.out.print("Introduce el valor del número " + c + ": ");
        num=teclado.nextDouble();
        suma=suma+num;
        c = c+1;
} while (c<=5)

System.out.print("La suma total es: " + suma);
```

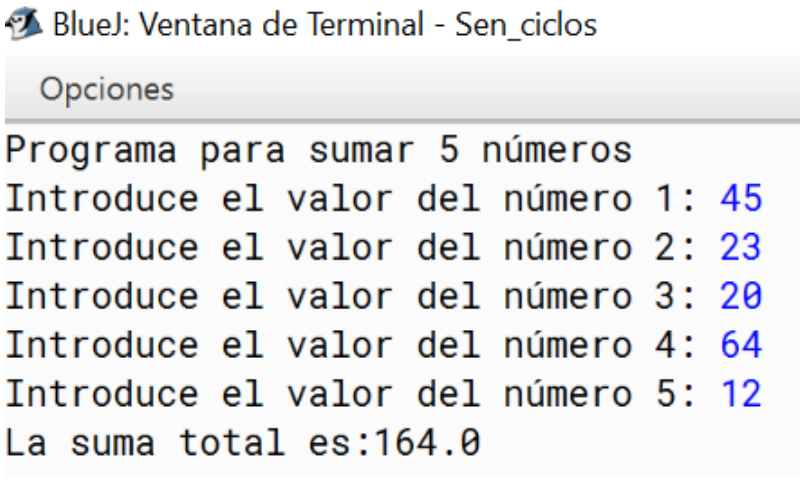
El código completo en BlueJ es:

```
import java.util.Scanner;
public class Do_while

{
   public static void main(String [] parametro){
       Scanner teclado= new Scanner(System.in);
       double num;
       int c=1;  //Se declara el contador con su valor de inicio
       double suma=0;  //Se declara el acumulador con su valor de inicio

       System.out.println("Programa para sumar 5 números");
       do{
           System.out.print("Introduce el valor del número " + c + ": ");
           num=teclado.nextDouble();
           suma=suma+num;
           c=c+1;      //Se incrementa el contador en 1
       }while(c<=5);
           System.out.println("La suma total es:" + suma);
    }
}
```

Ejemplo de la ejecución:



1. Realiza la prueba de escritorio del siguiente algoritmo y anota la salida en el recuadro correspondiente.

Pseudocódigo	Prueba de escritorio
Para i=1;i<=10; i++ inicio para **Calcular** r = 7 * i **Escribir** (“7 x ”, i, “= ”, r) fin para **Escribir** “Fin”	

2. Realiza la prueba de escritorio del siguiente algoritmo y anota la salida en el recuadro correspondiente.

Pseudocódigo	Prueba de escritorio
Para i=1;i>=10; i++ inicio para **Calcular** r = 1+ i **Escribir** (“1 + ”, i, “= ”, r) fin para **Escribir** “Fin”	

3. Completa el siguiente código para imprimir los números consecutivos del 1 al 20, separados por un guion medio (-).

Salida: 1 – 2 – 3 - … - 19 - 20

```
for (i=____; _______; _______)
{
    System.out.print( i + "_____");
}
```

4. Escribe el código de un ciclo *for* para escribir 8 veces “Buen día” una por renglón.

5. Realiza la prueba de escritorio del siguiente algoritmo y anota la salida en el recuadro correspondiente.

Pseudocódigo	Prueba de escritorio
num = 1 **Mientras** (num<=10) inicio mientras **Escribir** num **Calcular** num = num + 2 fin mientras **Escribir** “Fin”	

6. Realiza la prueba de escritorio del siguiente algoritmo y anota la salida en el recuadro correspondiente.

Pseudocódigo	Prueba de escritorio
num = 0 **Mientras** (num<=10) inicio mientras **Calcular** num = num + 2 **Escribir** num fin mientras **Escribir** "Fin"	

7. ¿Cuál es la salida del siguiente fragmento de programa?

```
int c = 1;
while ( c <= 5 ) {
        System.out.println("*****");
        c =c+1;
 }
```

Respuesta:

8. ¿Cuál es la salida del siguiente fragmento de programa?

```
int c = 1;
    while ( c <= 3 ) {
        System.out.println(c*0.5);
        c =c+1;
    }
```

Respuesta:

9. ¿Cuál es la salida del siguiente fragmento de programa?

```
int c = 1;
do{
        System.out.println("Prueba");
        c =c+1;
 }while ( c > 5 );
System.out.println("Fin");
```

Respuesta:

10. ¿Cuál es la salida del siguiente fragmento de programa?

```
int c = 1;
while ( c > 5 )
{
        System.out.println("Prueba");
        c =c+1;
 }
 System.out.println("Fin");
```

Respuesta:

11. ¿En qué ejercicio 9 o 10 se realizaron las instrucciones que contiene el ciclo, no importando si la condición es verdadera por primera vez?

_____

1. **Con la ayuda de los cuadros elabora el diagrama entrada, proceso, salida (EPS) de cada problema.**

   a) Calcular el tiempo (t) que tarda un objeto en caer de un edificio con cierta altura (h) con la siguiente fórmula

$$h = \frac{g * t^2}{2}$$

   b) Obtener la molaridad como el número de moles de soluto contenido en un litro de solución.

2. **Resuelve las siguientes expresiones matemáticas atendiendo las reglas de precedencia de los operadores y los paréntesis**

a) 9 * 8 + 2 - 7 * 10 / 2 + 8 =

b) 9 * (8 + 2) - 7 * (10 / 2) + 8 =

3. Resuelve las siguientes expresiones relacionales o lógicas colocando V si es verdadero o F si es falso, utilizando el espacio libre para hacer las operaciones necesarias.

a) (6>3) || (8<4) || (10/2)==5 ( )

b) (55 - 11) == 44 && (0<1) ( )

c) !F && (F || V) ……. ( )

d) (5-6)<(6-5)|| (! (4<3)) ( )

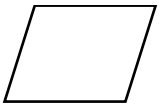
4. Subraya la respuesta correcta:

I. No es una característica de un algoritmo.

a) Tener un orden lógico. b) Ser finitos. c) Ser simples, claros y precisos. d) Un formato gráfico sencillo.

II. Símbolo que se relaciona con el pseudocódigo Escribir <expr1>

a) 

b) 

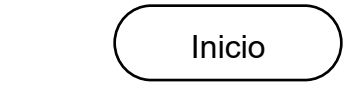
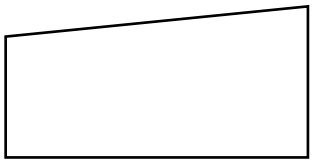
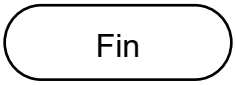
c) 

d) 

5. Completa el algoritmo, diagrama de flujo y pseudocódigo del siguiente problema secuenciales y como actividad extra escribir en PSeInt el pseudocódigo y ejecutar el archivo.

Calcular el descuento del 8% del precio de un producto y su precio final.

Algoritmo	Diagrama de flujo

Algoritmo COMPRA	
1. Declarar variables a. _____= 0.08 b. precio: _____ c. descuento: _____ d. preciofinal: _____	
2. Pedir _____ 3. Leer _____	
4. Calcular _____ = _____ 5. Calcular preciofinal = _____	
6. Mostrar _____ 7. Mostrar _____	
FinAlgoritmo	

Pseudocódigo

```
Algoritmo COMPRA
        porcentaje <-_____;
        _____ precio,descuento,preciofinal _____ Real;
        _____ "_____";
        _____  _____;
        _____<-_____;
        _____=_____;
        _____ "El descuento es de: $",_____;
        _____ "El precio final es de $",_____;
FinAlgoritmo
```

6. Responde a las siguientes preguntas:

a) ¿Cómo se llamó el lenguaje orientado a objetos en 1991?

_____

b) ¿Tipo de aplicación que se ejecuta del lado del servidor y crea una página dinámica? Aplicación web

_____

c) ¿Qué realiza el intérprete de Java?

_____

7. Realiza la prueba de escritorio del siguiente pseudocódigo

```
Inicio
    a = 8
    b = 5
    c = 4
    Si c<=2.5
    Inicio si
        Si b>3
        Inicio si
            r = (a - b + 2)/5
        Fin si
        Sino
        Inicio sino
            r = (a + b * 2)/13
        Fin sino
    Fin si
    Sino
    Inicio sino
        r = (a * b + 2)/7
    Fin sino
    Escribir “Resultado: ”, r
Fin
```

Salida

8. Relaciona las columnas de la definición con su concepto correspondiente.

		Definición	Concepto
I.	(___)	Sentencia que tiene sólo dos alternativas.	**a.** swtich
II.	(___)	Es la sentencia condicional múltiple que evalúa más de una condición.	**b.** if simple
III.	(___)	Sentencia condicional múltiple.	**c.** If anidada
IV.	(___)	Sentencia que tiene instrucciones sólo cuando la condición es verdadera.	**d.** If compuesta

9. Selecciona la respuesta correcta.

I. ¿Cuál es el fragmento de código que permite determinar si un número es par?, representando al número con la variable n.

a.
```
residuo = n%2;
if (residuo == 0){
    System.out.println( "par" );
}
```

b.
```
residuo = n/%2;
if (residuo != 0){
    System.out.println( "par" );
}
```

c.
```
residuo = n/2;
if (residuo != 0){
    System.out.println( "par" );
}
```

d.
```
residuo = n/2;
if (residuo == 0){
    System.out.println( "par" );
}
```

II. ¿Cuál es el fragmento de código que permite determinar si un número es impar?, representando al número con la variable n.

a.
```
residuo = n%2;
if (residuo == 0){
    System.out.println( "impar" );
}
```

b.
```
residuo = n%2;
if (residuo != 0){
    System.out.println( "impar" );
}
```

c.
```
residuo = n/2;
if (residuo != 0){
    System.out.println( "impar" );
}
```

d.
```
residuo = n/2;
if (residuo == 0){
    System.out.println( "impar" );
}
```

III. ¿Cuál es el fragmento de código que permite ordenar dos números diferentes de menor a mayor?

a.

```
if (num1 >= num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1)
}
```

b.

```
if (num1 > num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1);
}
```

c.

```
if (num1 <= num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1);
}
```

d.

```
if (num1 < num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1);
}
```

IV. ¿Cuál es el fragmento de código que permite ordenar dos números diferentes de mayor a menor?

a.

```
if (num1 >= num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1)
}
```

b.

```
if (num1 > num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1);
}
```

c.

```
if (num1 <= num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1);
}
```

d.

```
if (num1 < num2){
    System.out.println( "Orden: " + n1 + " " + n2 );
}
else{
    System.out.println( "Orden: " + n2 + " " + n1);
}
```

V. Es el fragmento de código que muestra el resultado de una operación básica con dos números.

a.

```
switch ( operacion ); {
    case '+':   System.out.println( a + b );
break;
    case '-':    System.out.println( a - b );
break;
    case '*':   System.out.println( a * b );
break;
    case '/':    System.out.println( a / b );
break;
default: System.out.println( “No válida” );
}
```

b.

```
switch ( operacion ) {
    case '+':   System.out.println(“ a + b“);
break
    case '-':    System.out.println(“ a - b “);
break;
    case '*':   System.out.println( “a * b” );
break;
    case '/':    System.out.println(“ a / b” );
break;
default: System.out.println( “No válida” );
}
```

c.

```
switch ( operacion ) {
    case '+':   System.out.println( a + b );
break;
    case '-':    System.out.println( a - b );
break;
    case '*':   System.out.println( a * b );
break;
    case '/':    System.out.println( a / b );
break;
default: System.out.println( “No válida” );
}
```

d.

```
switch ( operacion ):
    case '+':   System.out.println( a + b );
break;
    case '-':    System.out.println( a - b );
break;
    case '*':   System.out.println( a * b );
break;
    case '/':    System.out.println( a / b );
break;
default: System.out.println( “No válida” );
}
```

10. Relaciona las columnas de la definición con su concepto correspondiente.

		Definición	Concepto
I.	(___)	Sentencia cíclica que primero evalúa la condición para saber si se van a ejecutar las instrucciones del ciclo.	**e.** Acumulador
II.	(___)	Variable numérica que incrementa o decrementa su valor de forma no constante.	**f.** do - while
III.	(___)	Sentencia cíclica que su incremento o decremento lo realiza dentro de su sintaxis.	**g.** Contador
IV.	(___)	Variable de tipo entero que incrementa o decrementa su valor de forma constante.	**h.** for
V.	(___)	Sentencia cíclica en la que se realizan al menos una vez las sentencias del ciclo.	**i.** while

11. Coloca dentro del paréntesis si el enunciado es verdadero (V) o falso (F).

a) La variable que representa al contador requiere de un valor de inicio. - - - - (___)

b) Se recomienda utilizar la sentencia ***for*** cuando se conoce el número de repeticiones. - (___)

c) En la variable que representa al acumulador no se pueden realizar decrementos. - (___)

d) La sentencia ***while*** es la única sentencia cíclica que depende de una condición para conocer si se van a repetir las sentencias del ciclo. - - - - - - - - - - - - - - - (___)

e) Las sentencias cíclicas nos permiten ejecutar un bloque de instrucciones de forma repetitiva. - (___)

12. Selecciona la respuesta correcta.

I. ¿Cuáles son los elementos de la sintaxis de la sentencia ***for***?

a)	**b)**	**c)**	**d)**
valor de inicio condición valor de acumulador	valor de inicio condición tamaño de paso	valor de inicio condición valor de término	valor de inicio condición y contador valor de acumulador

II. Es el encabezado de un ciclo que no se efectuará, aunque la condición sea verdadera. Para esta pregunta se va a suponer que todas las condiciones son verdaderas.

a) ***while (letra = = 'A')*** **b)** ***while (num < 5)*** **c)** ***while (c <= 3);*** **d)** ***while (i <= 50)***

III. De la valoración de este elemento depende si se repite un ciclo.

a) contador **b)** ciclo **c)** acumulador **d)** condición

IV. Son ejemplos de sentencias cíclicas, excepto:

a) ***while - do*** ***for***

b) ***while*** ***for***

c) ***for*** ***do - while***

d) ***do - while*** ***while***

V. ¿Qué fragmento de código representa un ciclo infinito, esto es, no termina de forma natural, habrá que interrumpirlo?

a)
```
c = 1;
while (c <= 10) {
  System.out.print (c);
}
```

b)
```
c = 1;
while (c >= 10) {
  System.out.print (c);
  c = c + 1;
}
```

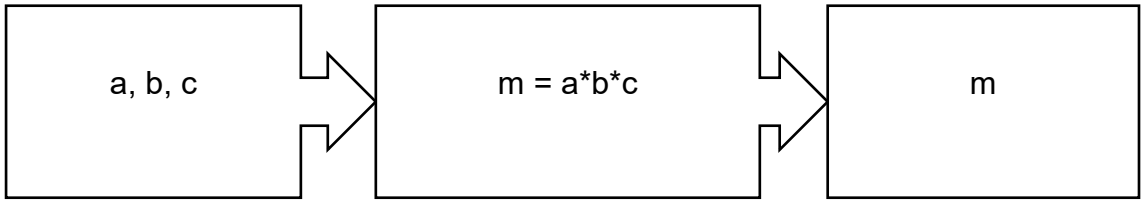
c)
```
c = 1;
while (c >= 10) {
  System.out.print (c);
}
```

d)
```
c = 1;
while (c <= 10) {
  System.out.print (c);
  c= c + 1;
}
```

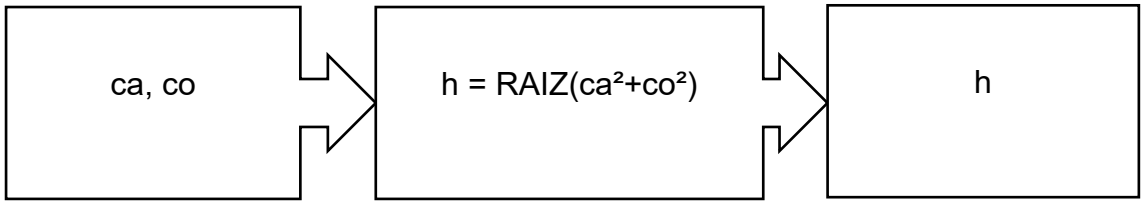
SECCIÓN 3.1

1.

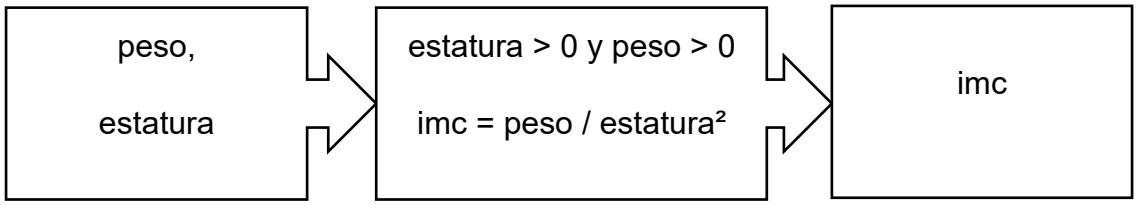
a) Realizar la multiplicación de tres números a, b, c



b) Obtener el valor de la hipotenusa (h) de un triángulo rectángulo conociendo sus catetos (ca,co).

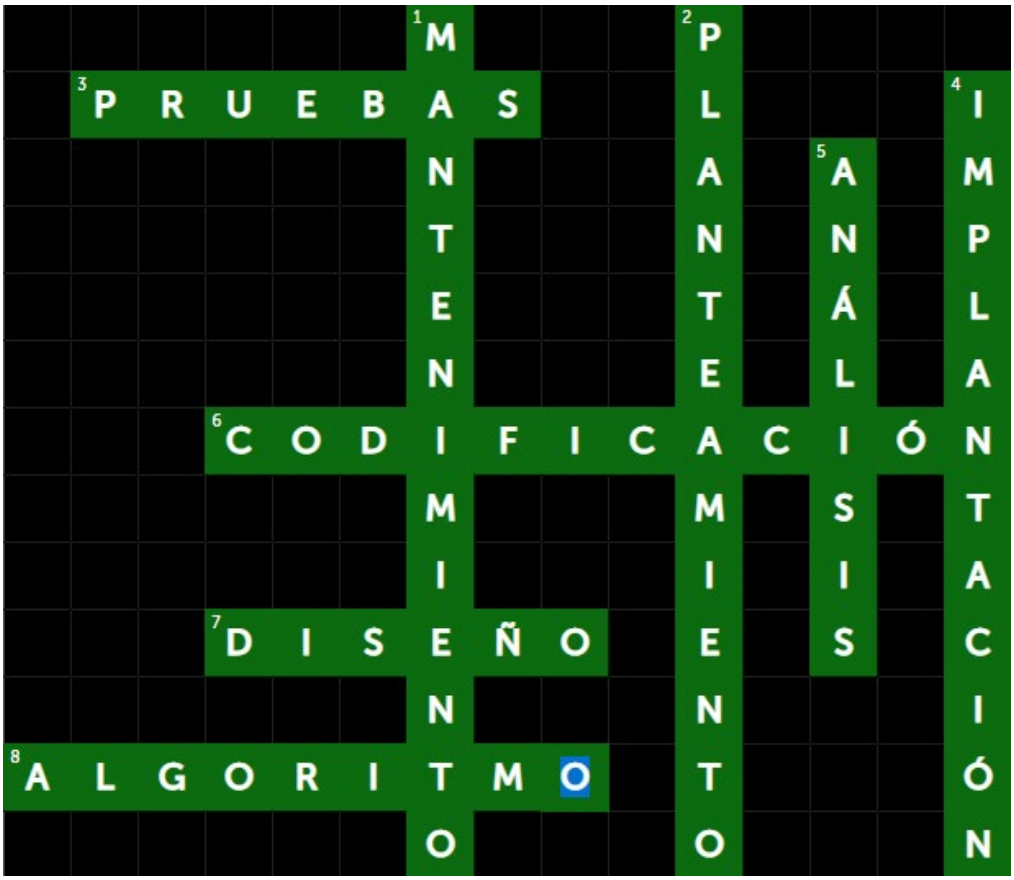


c) Calcular el índice de masa corporal de un adulto dividiendo el peso entre el cuadrado de la estatura.



2. 1. (J) 2. (H) 3. (E) 4. (B) 5. (I) 6. (G) 7. (A) 8. (F) 9. (D) 10. (C)

3. Crucigrama metodología de solución de problemas



SECCIÓN 3.2

1.

a) - 13.1
2. - 257
a) - 427
b) – 67

2.

a) F
b) F
c) V
d) V
e) F
f) V

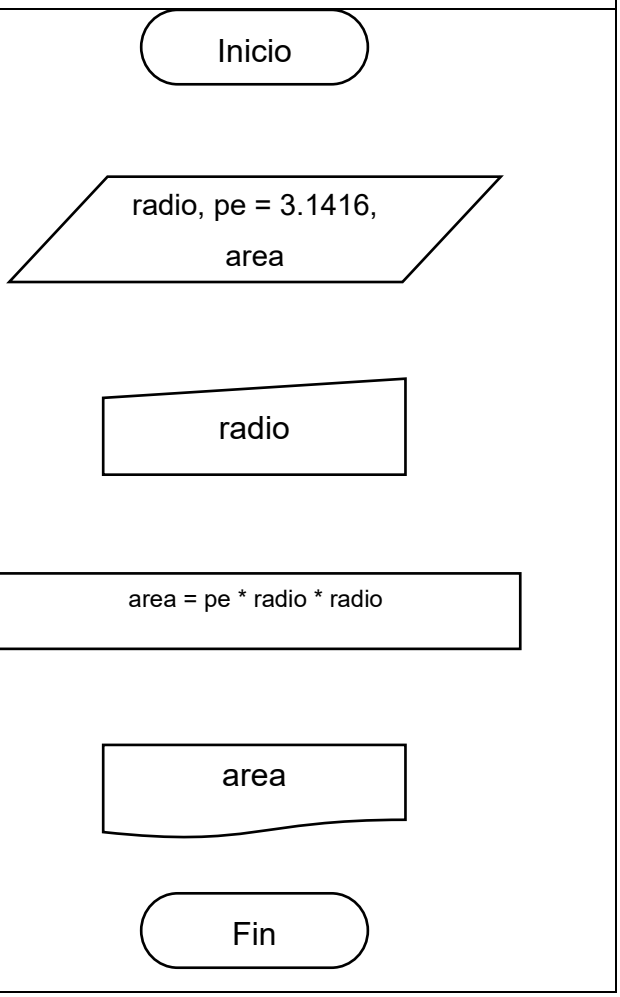
SECCIÓN 3.3

1.

I. b
II. a
III. c
IV. c
V. a
VI. b
VII. c
VIII. c

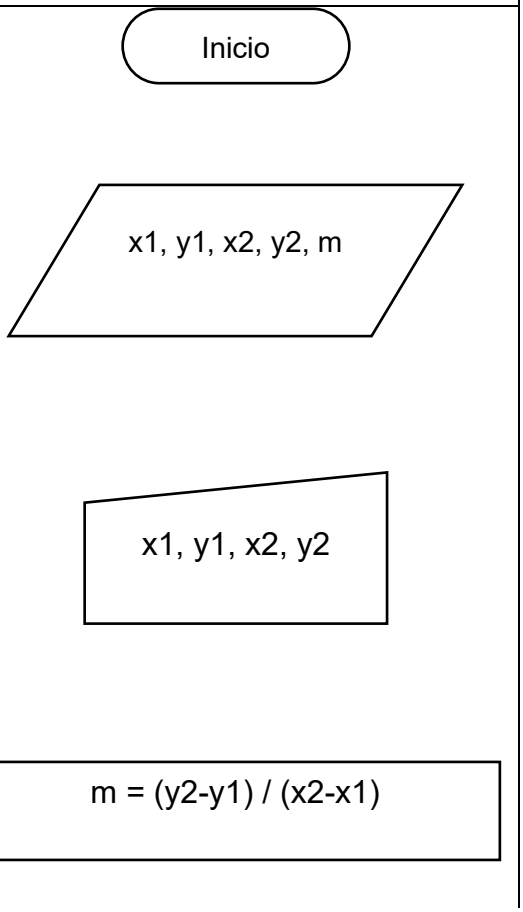
2.

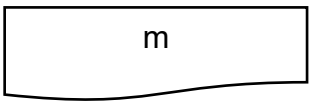
a) Obtener el área de un círculo.

Algoritmo	Diagrama de flujo
Algoritmo CIRCULO	
1. Declarar variables a. radio: real b. pe = 3.1416 c. area: real	
2. Pedir el radio 3. Leer radio	
4. Calcular área = pe*radio*radio	
5. Mostrar el area	
FinAlgoritmo	

Pseudocódigo
Algoritmo CIRCULO Definir __radio__ Como __Real__; pe <- _3.1416_; Definir __area_ Como __Real__; Escribir "__Dame el radio del círculo_"; __Leer_radio; area_<- pe*radio*radio; Escribir "El área del círculo es: ", area FinAlgoritmo

b) Obtener la pendiente de una recta conociendo dos puntos (x1,y1) (x2,y2)

Algoritmo	Diagrama de flujo
Algoritmo RECTA	
1. Declarar variables a. x1: __real__ b. y1: __real__ c. x2: __real__ d. y2: __real__ e. m: __real__	
2. Pedir ___x1__ 3. Leer ___x1__ 4. Pedir ___y1__ 5. Leer ___y1__ 6. Pedir ___x2__ 7. Leer ___x2__ 8. Pedir ___y2__ 9. Leer ___y2__	
10. Calcular m = (y2-y1) / (x2-x1)	

11. Mostrar __m__	
FinAlgoritmo	

Pseudocódigo

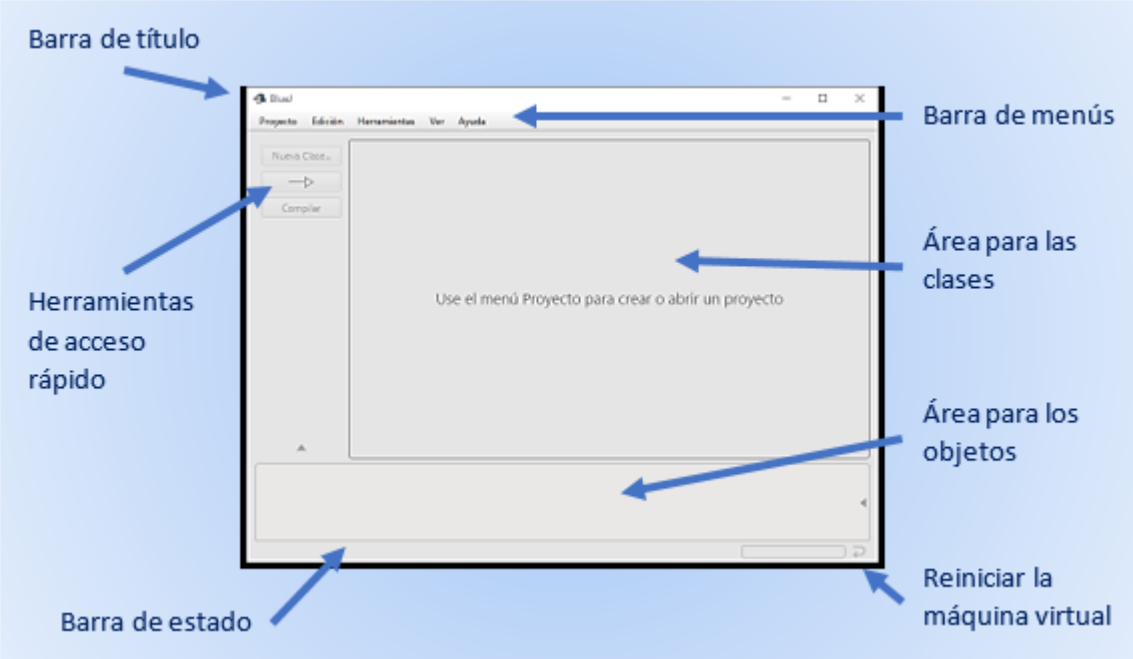
```
Algoritmo RECTA
        Definir __x1, y1, x2, y2, m__ Como Real;
        Escribir " Escribe el valor de la coordenada x1: ";
        _Leer_ x1;
        Escribir " Escribe el valor de la coordenada y1: ";
        Leer _y1_ ;
        Escribir “Dame el valor de x2: ";
        _Leer x2_ ;
        Escribir " Escribe el valor de la coordenada y2: ";
        _Leer y2_ ;
        m<- (y2-y1) / (x2-x1);
        _Escribir_ "La pendiente de la recta es m = ", _m_ ;
FinAlgoritmo
```

SECCIÓN 3.4

1.

a) Sun Microsystems
b) C++
c) Un prototipo que tenía soporte para los applets en un ambiente web
d) 1995
e) El lenguaje máquina de la máquina virtual de Java
f) Un lenguaje distribuido
g) Traduce un programa fuente a bytecode.

2.



SECCIÓN 3.5

1. Tacha con color rojo los errores sintácticos y lógicos del siguiente código para obtener la siguiente ejecución del programa.

```
public clase Errores
{
   Public static void main(string[ ] args)
   {
      system.out.print("===\n \t Hola \t ");
      System.out.print("\n \n ===\n \t \n \t Mundo \n");
      Sistem.out.print("\t \t === \n \n =D \t");
      System.out.print("===");
    }
}
```

Escribe el código correcto.

```
public class Errores
{
   public static void main(String[] args)
```

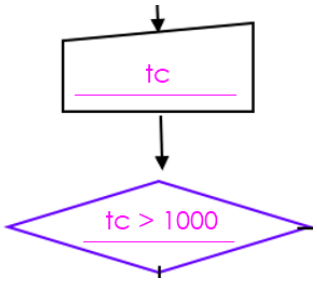
```
    {
        System.out.print("===\n \t Hola \n");
        System.out.print("\t \t===\n \t \t \t Mundo \n");
        System.out.print("\t \t===\n \t =D \n");
        System.out.print("===");
    }
}
```

2.

```
impor java.util.Scanner;
public class Datos
{
    public static void main(String[ ] args)
    {
        Scanner_ teclado = new Scanner(System.in);
        System.out.println("TIPOS DE DATOS POR TECLADO");
        System.out. _print_ ("Escribe tu nombre completo: ");
        _String_ nombre = teclado.nextLine();
        System.out. _print_  ("Escribe tu edad: ");
        byte edad = teclado. _nextByte() ;
        System.out. _print_ ("Escribe tu peso en kilos: ");
        _float_ peso = teclado.nextFloat();
        System.out. _print_  ("Elige tu sexo: (F o M): ");
        char sexo = teclado. _teclado.next().charAt(0) _;
        System.out. _print_  ("Escribe tu salario: ");
        double salario = teclado. _nextDouble() ;
        System.out. _print_  ("¿Tienes hijos? (false o true): ");
        _boolean_ hijo = teclado.nextBoolean();
        System.out. _println_  ("==================================");
        System.out. _println_   ("Nombre: "_+nombre_);
        System.out. _print_  ("Edad: "_+edad_" años");
        System.out. _print_   (" \t Peso: "_+peso_" kilos");
        System.out. _println_   ("\t Sexo: "_+sexo_);
        System.out. _print_   ("\t Salario: $"_+salario_);
        System.out. _println_  ("\t Hijos: "_+hijo_);
        System.out. _print_  ("==================================");
    }
}
```

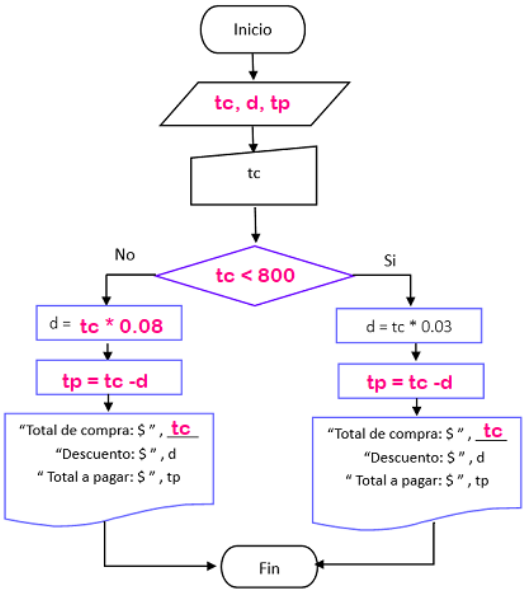
SECCIÓN 3.6

1. ¿Cuál es la condición correcta para saber si en el ticket se debe imprimir “Regalo por aniversario”? tc > 1000



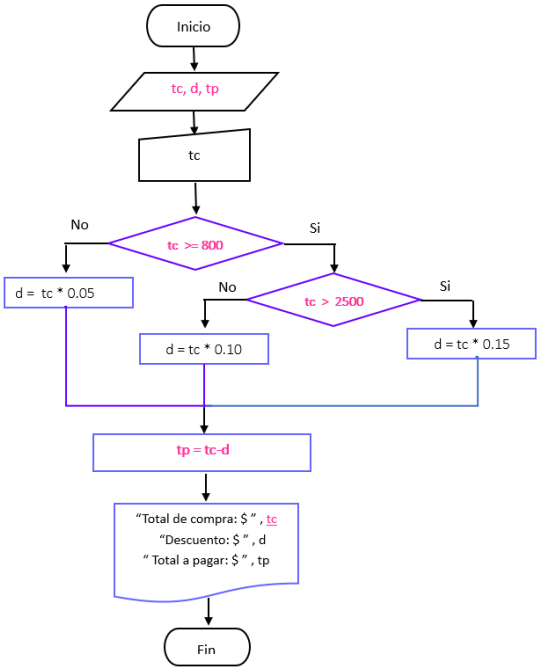
Si *(tc >1000)*
Inicio si
Escribir “Total de compra: $ ” , tc
Escribir “Regalo por aniversario **”**
Fin si

2.

Diagrama de flujo	Pseudocódigo
	**Inicio** **Escribir** “Introduce el total de compra” (tc) **Registra** el total de compra (tc) ***Si*** *(* tc < 800 *)* ***Inicio si*** **Calcular** d = tc * 0.03 **Calcular** tp = tc - d **Escribir** “Total de compra: $ “, tc **Escribir** “Descuento: $ “, d **Escribir** “Total a pagar: $ “, tp ***Fin si*** ***Sino*** ***Inicio sino*** **Calcular** d = tc * 0.08 **Calcular** tp = tc - d **Escribir** “Total de compra: $ “, tc **Escribir** “Descuento: $ “, d **Escribir** “Total a pagar: $ “, tp ***Fin sino*** **Fin**

3. Resultado: 6.5

4.



Inicio

Escribir "Introduce el total de compra" (tc)

Registra el total de compra (tc)

Si (tc >= 800)

Inicio si

Si (tc >2500)

Inicio si

Calcular d = **tc * 0.15**

Fin si

Sino

Inicio sino

Calcular d = **tc * 0.10**

Fin sino

Fin si

Sino

Inicio Sino

Calcular d = **tc * 0.05**

Fin sino

Calcular tp = **tc - d**

Escribir "Total de compra: $ ", **tc**

Escribir "Descuento: $ ", **d**

Escribir "Total a pagar: $ ", **tp**

Fin

5. numero > 0

6. Fin

7.

Alternativa 1	Alternativa 2
if (e >= 18) { System.out.println("Eres mayor de edad"); } **else** { System.out.println("Eres mayor de edad"); }	**if** (e >= 18) System.out.println("Eres mayor de edad"); **else** System.out.println("Eres mayor de edad");

8. Eres menor de edad

9.

Según nf **Hacer**
Caso, = 1
Escribir "Escribe la medida del lado"
Registrar el valor de la medida del lado (l)
Calcular area=l*l;
break;
Caso, = 2
Escribir "Escribe la medida de la base"
Registrar el valor de la medida de la base (b)
Escribir "Escribe la medida de la altura"
Registrar el valor de la medida de la altura (h)
Calcular area = b * h /2;
break;
Caso, = 3
Escribir "Escribe la medida del radio"
Registrar el valor de la medida del radio (r)
Calcular area= 3.1416 * r *r
break;

10.

Codificación

```
estacion=teclado.nextInt():
switch (estacion)
 {
    case 1:  System.out.println("Primavera");    break;
    case 2:  System.out.println("Verano");       break;
    case 3:  System.out.println("Otoño");        break;
    case 4:  System.out.println("Invierno");     break;

  default:
    System.out.println("El año sólo tiene cuatro estaciones");
 }
```

11. El resultado es: 10

SECCIÓN 3.7

1.

```
7 x  1 = 7
7 x  2 = 14
7 x  3 = 21
7 x  4 = 28
7 x  5 = 35
7 x  6 = 42
7 x  7 = 49
7 x  8 = 56
7 x  9 = 63
7 x  10 = 70
Fin
```

2. Fin

3.

```
for (i= 1; i<=20; i++)
{
    System.out.print( i + " - ");
}
```

4.

```
for (i= 1; i<=8; i++)
{
    System.out.println( "Buen día ");
}
```

5.

Salida

1
3
5
7
9

Fin

6.

Salida

2
4
6
8
10
12

Fin

7.

Salida

*****
*****
*****
*****
*****39

8.

Salida

0.5

1.0

1.5

9.

Prueba
Fin

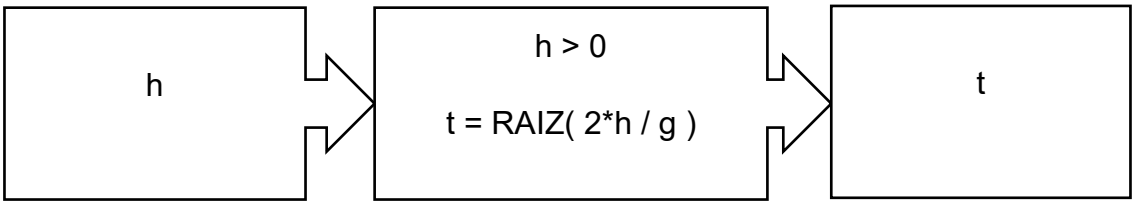
10. Fin

11. 9

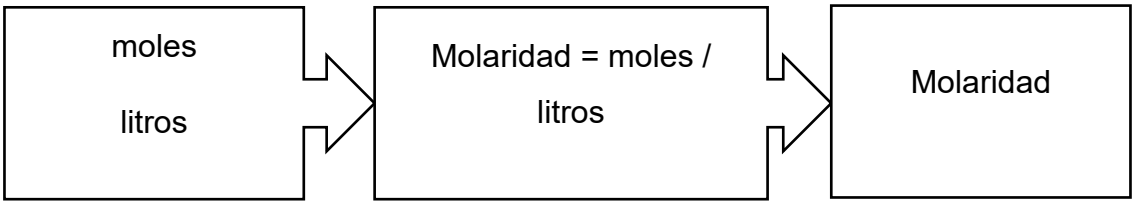
AUTOEVALUACIÓN

1.

a) Calcular el tiempo (t) que tarda un objeto en caer de un edificio con cierta altura (h) con la siguiente fórmula h= (g * t²) / 2



b) Obtener la molaridad como el número de moles de soluto contenido en un litro de solución.



2.

a) 47
b) 63

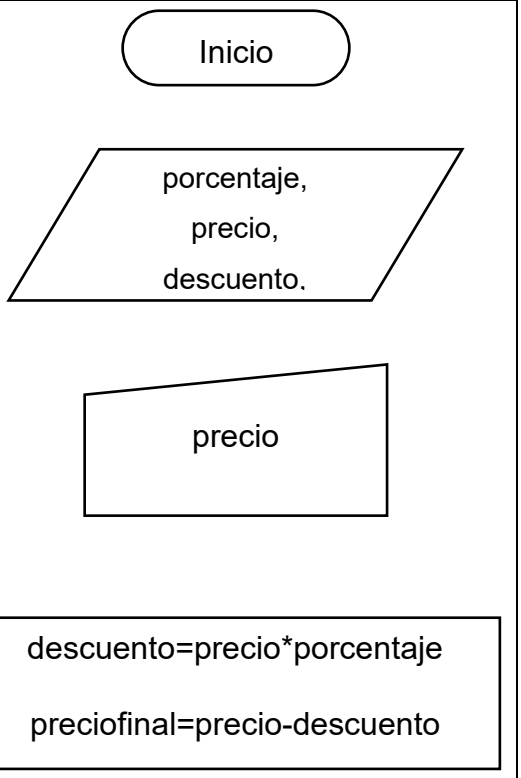
3.

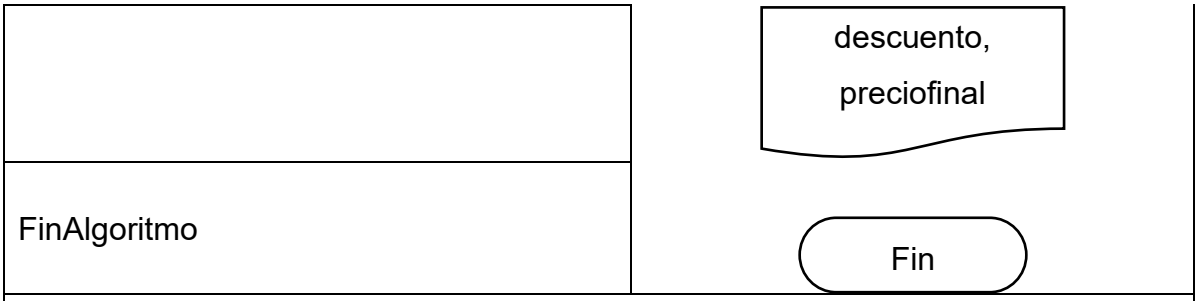
a) (V)
b) (V)
c) (V)
d) (V)

4.

I. d
II. c

5. Calcular el descuento del 8% del precio de un producto y su precio final.

Algoritmo	Diagrama de flujo
Algoritmo COMPRA	
1. Declarar variables a. _porcentaje_ = 0.08 b. precio: _real_ c. descuento: _real_ d. preciofinal: _real_	
2. Pedir _precio_ 3. Leer _precio_	
4. Calcular descuento= precio*porcentaje 5. Calcular preciofinal= precio-descuento	
6. Mostrar _descuento_ 7. Mostrar _preciofinal_	



Pseudocódigo

```
Algoritmo COMPRA
        porcentaje <- 0.08 ;
        Definir precio,descuento,preciofinal Como Real;
        Escribir " Cual es el precio del producto: ";
        Leer  precio ;
        descuento <- precio*porcentaje ;
        preciofinal = precio-descuento ;
        Escribir "El descuento es de: $", descuento ;
        Escribir "El precio final es de $", preciofinal ;
FinAlgoritmo
```

6.

a) Oak
b) Aplicación web
c) Traduce cada instrucción en bytecode al lenguaje máquina del CPU y ejecuta la instrucción.

7. Resultado: 6

8.

I. d
II. c
III. a
IV. b

9.

I. a
II. b
III. d
IV. b
V. c

10.

I. e
II. a
III. d
IV. c
V. b

11.

a) V
b) V
c) F
d) F
e) V

12.

I. b
II. c
III. d
IV. a
V. a

BIBLIOGRAFÍA

- Aguilar, L. J. (2011). *Programación en Java 6. Algoritmos, programación orientada a objetos e interfaz gráfica de usuarios.* México: McGrawHill.
- Aguilar, L. J. (2023). *Portal tecnológico y de conocimiento*. Obtenido de https://highered.mheducation.com/sites/8448118952/information_center_view0/documentacion.html
- Aguilar, L. J. (2023). *Programación en Java 6*. Obtenido de https://highered.mheducation.com/sites/6071506184/student_view0/
- Barrera Arrestegui, L. (2012). Fundamentos históricos y filosóficos de la Inteligencia Artificial. *UCV-HACER. Revista de Investigación y Cultura*, 87-92.
- Bravo, J. M. (1968). *Introducción e historia de la cibernética.* México: Grijalbo.
- Cebral Loureda, M. (2019). *La revolución cibernética desde la filosofía de Gilles Deleuze: una revisión crítica de las herramientas de minería de datos y Big Data.* Santiago de Compostela: Universidad de Santiago de Compostela.
- Desarrollo Web. (7 de noviembre de 2022). *Operadores y operandos*. Obtenido de https://desarrolloweb.com/articulos/operadores-operandos-programacion
- Ecured. (s.f.). *Pseudocódigo*. Obtenido de https://www.ecured.cu/Pseudoc%C3%B3digo#Desarrollo_de_algoritmos
- Education-wiki. (2023). *Versiones de Java*. Obtenido de https://es.education-wiki.com/8387738-versions-of-java#:~:text=A%20continuaci%C3%B3n%20aprenderemos%20diferentes%20versiones%20de%20java%20con,%2828%20de%20julio%20de%202011%29%20...%20M%C3%A1s%20elementos
- Eduteka. (1 de marzo de 2001). *Organizadores gráficos diagramas de flujo*. Obtenido de https://eduteka.icesi.edu.co/articulos/definicion-diagramas-flujo
- González Casanova, P. (2017). *Las Nuevas Ciencias y las Humanidades.* Ciudad Autónoma de Buenos Aires: CLACSO.
- Guzik Glantz, R. (s.f.). Entre la experimentación y los modelos abstractos Breve historia de vida de Arturo Rosenblueth (1900-1970). *Antropología*, 20-36.
- M. Layús, A. (s.f.). Modelo de comunicación de Shannon y Weaver .
- Morris, M. Diseño digital [en línea]. Recuperado el 18 de mayo de 2023 en https://urielectronics.files.wordpress.com/2010/11/diseno-digital-morris-mano-en-espanol.pdf

- Morris, M. (1988). Ingeniería computacional. México: Prentice–Hall Hispanoamericana.
- Murcia López, M. I., & Crespo-Blanc, A. (2008). La formación de oceanos y cadenas de montañas a partir de modelos analógicos: maquetas y nuevos materiales. *Enseñanza de las Ciencias de la Tierra*, 173-177.
- Novara, P. (2023). *PseInt*. Obtenido de https://pseint.sourceforge.net/
- O'Brien, J. A., & Marakas, G. M. (2006). *Sistemas de Información Gerencial.* México: Mc Graw Hill.
- Perez Porto, J., & Gardey, A. (18 de julio de 2013). *Problemas matemáticos - Qué son, definición y concepto.* Obtenido de Definición de.: https://definicion.de/problemas-matematicos/
- Pérez, J. P. (25 de abril de 2008). *Problema - Qué es, en la filosofía, definición y concepto.* Obtenido de Definicio.de: https://definicion.de/problema/
- Pradilla Rueda, M. (2014). Alan Turing, su obra y los efectos sobre la calculabilidad. *Rev. Ingeniería, Matemáticas y Ciencias de la Información*, 93-122.
- Puertas lógicas [en línea]. Recuperado el 17 de junio de 2023 en http://electronikatualcance.blogspot.com/2011/10/puertas-logicas-and-or-not.html
- Ríos Estavillo, J. J. (1997). *Derecho e informática en México. Informática jurídica y derecho de la informática.* México: UNAM.
- Román, L. L. (2013). *Metodología de la programación orientada a objetos.* México: Alfaomega Grupo Editor.
- Sistema de educación Digital. (11 de agosto de 2022). *Clase digital 5. Probabilidad de eventos - Bachillerato virtual.* Obtenido de https://blogs.ugto.mx/bachilleratovirtual/clase-digital-5-probabilidad-de-eventos/
- Wiener, N. (1985). *CIBERNETICA o el control y comunicación en animales y maquinas.* Barcelona: Tusquets Editores.