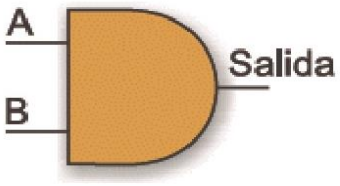
Escuela Nacional
Colegio de Ciencias y Humanidades
Plantel Oriente

Paquete Didáctico para la asignatura de Cibernética y Computación I.



A	B	Salida
0	0	0
0	1	0
1	0	0
1	1	1

María del Socorro Ávila Nicolás
Juan Gutiérrez Quiroz (Coordinador)
José Luís Hermoso Sandoval
Carmen Yadira Martínez Valdés
Alejandro Muñoz Navarrete
Rosangela Zaragoza Pérez

Contenido

Indicaciones de uso

¿Cómo utilizar el material?

El contenido de este paquete didáctico está dirigido a alumnos que cursan la asignatura de Cibernética y Computación I, con el propósito de ofrecer un material de apoyo para lograr los aprendizajes planteados en el programa de la asignatura.

El paquete didáctico es un conjunto estructurado de materiales necesarios para la enseñanza aprendizaje de la asignatura de Cibernética y Computación I, que están adecuados al nivel y profundidad de los contenidos especificados en el programa de estudios de esta asignatura.

Este material se puede utilizar en la modalidad de Curso taller, ya que en cada unidad temática contiene conceptos, ejemplos desarrollados y ejercicios para los diferentes temas. Para su desarrollo se contemplaron los propósitos de cada una de las unidades, los aprendizajes a alcanzar por el alumno y la temática que conforma dicho programa.

Consta de tres unidades y está organizado de la siguiente forma:

En cada una de las unidades:

- Se explican los conceptos correspondientes a la temática.
- Se proponen ejemplos para visualizar y reflexionar los conceptos abordados.
- Se proponen actividades de evaluación para que el alumno estime el aprendizaje. obtenido.

Al final del paquete se anexa la bibliografía y fuentes de información utilizadas para su desarrollo.

Se puede utilizar en clase después de revisar los conceptos y ejemplos en el desarrollo de la solución a los ejercicios, esto le permitirá observar al profesor las dificultades o dudas de los alumnos proporcionando un asesoramiento adecuado con sugerencias y orientaciones para obtener el resultado pertinente. También es recomendable que el profesor explique alguna de las soluciones obtenidas para una mejor comprensión del tema.

Para lograr un mejor proceso de enseñanza - aprendizaje, se recomienda que:

El alumno:

- Revise los conceptos y ejemplos previamente a la clase
- Desarrolle las actividades de evaluación
- Compare con sus compañeros los resultados obtenidos
- Plantee dudas en clase.

El profesor:

- Exponga el tema por revisar.
- Solucione las dudas de los alumnos.
- Oriente en la solución de los ejercicios.

Con lo anterior se pretende lograr el aprendizaje indicado.

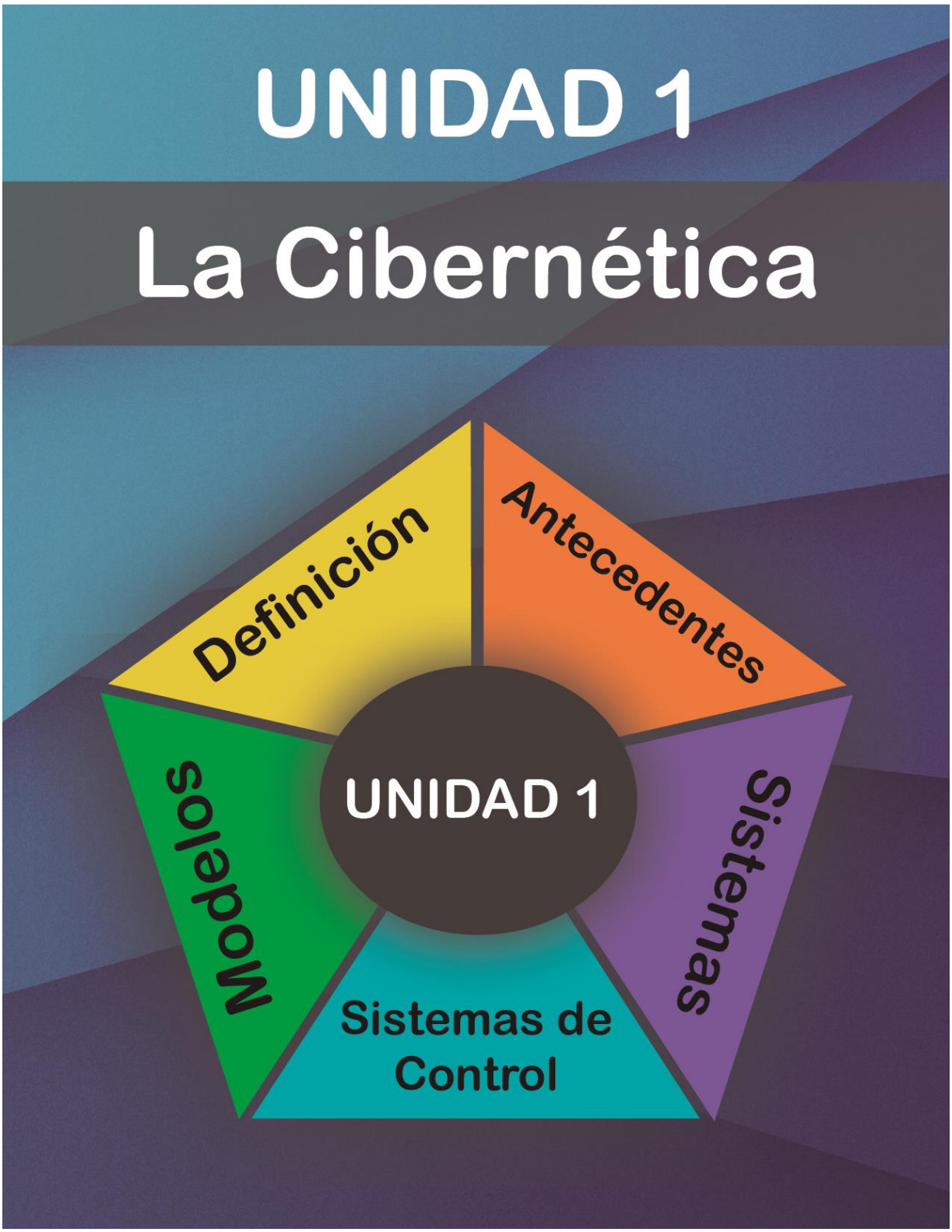
Este paquete didáctico aborda el curso completo de Cibernética y Computación I y tiene como objetivos primordiales:

Que los profesores:

- Cuenten con un material adecuado para preparar su clase.

Que los alumnos:

- Comprendan y apliquen los conocimientos adquiridos en la asignatura.
- Comprendan la influencia de la Cibernética en el desarrollo de la ciencia.
- Adquieran la metodología para elaborar Circuitos Lógicos.
- Adquieran una metodología para resolver problemas utilizando la computadora.
- Sena capaces de desarrollar programas sencillos en el lenguaje de programación Java.


UNIDAD 1
La Cibernética
Definición
Antecedentes
Modelos
UNIDAD 1
Sistemas
Sistemas de Control

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.

Aprendizajes:

El alumno:

- 🕮 Comprende la influencia de la Cibernética en el desarrollo de la ciencia.
- 🕮 Describe el trabajo científico sobre la Cibernética de Norbert Wiener, Arturo Rosenblueth, Claude Shannon, entre otros.
- 🕮 Comprende los componentes de un sistema.
- 🕮 Comprende los componentes esenciales de un sistema de control.
- 🕮 Comprende el concepto y la importancia del modelo.
- 🕮 Desarrolla el modelo de un sistema.

Introducción

La Cibernética es la ciencia que se encarga del estudio de los sistemas de control y de comunicación, la cual nació en 1948 impulsada por Norbert Wiener conocido como el padre de la Cibernética, quien la denomina como la rama de la ciencia "encargada de efectuar la dirección y la comunicación entre los seres vivos y las máquinas". (Jramoi, 1966, pág. 61)

La Cibernética dio gran impulso a la teoría de la información a mediados de los 60, en la que Shannon comenzó los fundamentos de la teoría de información en el cual propone la idea de convertir los datos a dígitos binarios, (bits de información) en código binario (0, 1). El objetivo del desarrollo de esta teoría es encontrar fundamentos en las operaciones de procesamiento de tales como compresión de datos, almacenamiento y comunicación, de tal manera que la información sea transmitida sobre el medio de comunicación.

Por lo tanto, la teoría de la información, se refiere principalmente a definir una medida cuantitativa de datos para resolver los problemas relativos, aplicándolo en diversos campos

entre ellos la física, química, biología, la robótica, la criptografía, la computación, la lingüística y la teoría de la comunicación.

La Cibernética es una disciplina íntimamente vinculada con la teoría general de sistemas, al grado en que muchos la consideran inseparable de esta, y se ocupa del estudio del mando, el control, las regulaciones y el gobierno de los sistemas.

TEMA **1**

Definición del concepto de Cibernética

Objetivo:

- Comprender el concepto de la Cibernética.

1 Introducción

La Cibernética es una ciencia que trata en lo general de los procesos y sistemas de dirección en los dispositivos técnicos, en los organismos vivos y en las organizaciones humanas. Los principios de la Cibernética fueron expuestos principalmente por Wiener.

2 Concepto de Cibernética

La Cibernética se define como:

- La ciencia que estudia los sistemas de control y comunicación de los seres vivos y de las máquinas. (Norbert Wiener)
- La ciencia que se encarga del estudio de los mecanismos de control, regulación de los sistemas humanos y mecánicos, incluyendo los ordenadores, la cual se ocupa de los procesos de dirección en los sistemas dinámicos complejos y que tiene por fundamento teórico, las matemáticas y la lógica, así como el empleo de dispositivos especialmente de calculadoras electrónicas y de máquina de control.

- Se considera sistema dinámico complejo aquel que modifica su estado y que incluye toda una serie de sistemas y elementos más simples, interrelacionados entre sí y que actúan en conjunto.
- Es una ciencia que trata los principios generales de dirección y de su aplicación en la técnica, la sociedad humana y los organismos vivos.
- La Cibernética se subdivide en Cibernética teórica (se encarga de los fundamentos matemáticos y lógicos), Cibernética técnica (se encarga de la construcción y explotación de los medios técnicos aplicando mecanismos de dirección y de cálculo) y Cibernética aplicada (se enfoca a la solución de problemas relacionados con los sistemas concretos de dirección de las diferentes esferas de la actividad humana, en la industria, el suministro de energía, el transporte, el servicio de comunicaciones, etc.).

Con base en estos conceptos se puede entender a la Cibernética como la ciencia encargada de estudiar la comunicación en los seres vivos y las máquinas, así como el control de los sistemas en todos sus aspectos y mecanismos comunes, partiendo de la teoría de la información la cual se enfoca en el desarrollo del procesamiento y transmisión de información mediante el medio de comunicación.

TEMA **2**

Antecedentes de la Cibernética

Objetivo:

- Comprender la evolución de la Cibernética.

1 Introducción

El surgimiento de la Cibernética fue provocado por una serie de resultados técnicos y científicos obtenidos con la resolución de problemas que han surgido a través del desarrollo técnico y tecnológico de la humanidad, desde el tiempo de los griegos a través de sus filósofos, neurofisiólogos e ingenieros, con la finalidad de mejorar procedimientos de construcción, procesos administrativos y en general cualquier aplicación que se pueda facilitar aplicando un método.

2 Antecedentes de la Cibernética

La Cibernética se fue desarrollando en base algunos descubrimientos, como el androide de Ptolomeo Filadelfo, mecanismo que imitaba los movimientos humanos (siglo III a. C.); Demetrio de Faleria (siglo IV-III) construyó un caracol que se arrastraba; Arquitas de Tarento (siglo V-IV) construyó una paloma voladora.

La técnica en el (siglo IV) donde se realizaron intentos para construir sistemas automáticos que realizarán movimientos de los seres vivos por Pascal y Leibniz.

Juanelo Turriano en (siglo XVI) preparó para Carlo V numerosos juguetes automáticos como soldados armados, pájaros voladores, etc.

En el (siglo XVII) se tiene un antecedente relevante en la fisiología por el descubrimiento de W. Harvey que descubrió el sistema de circulación de sangre.

J. Muller (1436-1476) creo una serie de autómatas, en los cuales figuraba una mosca que corría alrededor de la mesa y un águila que fue colocada en las puertas de Nuremberg.

Leonardo da Vinci (1452-1519) construyó un mecanismo automático en forma de león que se movía solo de un lugar a otro, la retroacción fue C. Huygens (1629-1695) quien descubrió el reloj de péndulo.

Blaise Pascal (1641-1662) construyó la primera sumadora automática y Leibnitz (1646-1716) creó el primer mecanismo de multiplicar, ambos fueron los primeros en tratar de reproducir con ayuda de medios mecánicos una de las facultades mentales del hombre. Chébyshev (1896) una máquina calculadora de multiplicar y dividir.

Norbert Wiener (1948) el “padre de la Cibernética”, realizó aportaciones de gran importancia en varias áreas de las matemáticas, principalmente en análisis, probabilidad y teoría de control, junto con otros autores, como Alan Turing, John von Neumann y Claude Shannon, quienes dieron lugar al nacimiento de las ciencias computacionales.

Wiener mencionó que el fenómeno de la retroalimentación y la sincronización están presentes en la naturaleza, la sociedad y las máquinas creadas por el hombre, y que ambos son necesarios para mantener una organización efectiva y en equilibrio de los sistemas dinámicos complejos. De aquí parte la importancia que ha adquirido la teoría de control y de la información, ya que ambas han desarrollado herramientas y algoritmos necesarios para analizar, diseñar y diagnosticar.

Jhon Von Neumann (1949) fundamentó la relación que existe entre la Cibernética y las computadoras, implementando los principios básicos de la programación, elaboró la teoría de los juegos y demostró la posibilidad que tienen las máquinas de reproducirse.

TEMA **3**

Relación de la Cibernética con otras ciencias

Objetivo:

- Identificar las diversas disciplinas que integran la Cibernética.

1 Introducción

La Cibernética es una ciencia que ha apoyado al avance de otras ciencias pero también se ha retroalimentado de estos, por lo que se considera una ciencia multidisciplinaria, esto se refleja en los grandes avances científicos que se han derivado de la colaboración, entre otros, de matemáticos, químicos, físicos, biólogos, fisiólogos e ingenieros, los cuales se integran en equipos multidisciplinarios para el desarrollo de sus proyectos.

2 Relación de la Cibernética con otras ciencias

La Cibernética colabora con diversas ramas de la ciencia y la técnica moderna influyendo favorablemente en el desarrollo de cada una, algunas de estas son:

- Biónica: Se encarga de estudiar los sistemas biológicos aplicándolo con la física, biología y robótica, es decir tratar de similar el comportamiento de los seres vivos.
- Neurofisiología: Descarga las neuronas de forma análoga para determinar un digito en escala binaria.
- Computación: Comprende la creación de algoritmos para la solución de problemas y el procesamiento de la información.
- Medicina: Su finalidad es curar enfermedades y deficiencias físicas.
- Física: Estudia las propiedades de la materia.

- Matemáticas: Se enfoca al estudio de las propiedades de los entes abstractos y de sus relaciones.

En el desarrollo de la Cibernética se han involucrado diversas ramas del saber tales como: la teoría de la regulación automática, y de los sistemas vigilantes; la termodinámica; la teoría estadística de transmisión de la información; la lógica matemática; la economía matemática, entre otras.

En cuanto a la aplicación de la Cibernética a la técnica, se pueden señalar dos esferas principales:

1) El vínculo que existe con la dirección de máquinas y complejos de máquinas en la industria, los transportes, el arte militar, etc.

2) La que se encarga del manejo de los dispositivos que brinda la Cibernética, principalmente las calculadoras, que se utilizan para efectuar cálculos complejos y modelar diferentes procesos dinámicos.

TEMA **4**

Aplicaciones de la Cibernética en la actualidad

Objetivo:

- Identificar las distintas ramas que se relacionan con la Cibernética.

1 Introducción

Las aportaciones de la Cibernética fueron, en un momento dado, de uso casi exclusivo de la Física y como la Cibernética a su vez era una disciplina común a varios sectores de investigación, trajo como consecuencia que ramas como la psicología, la sociología y la biología, pudieran de alguna manera formalizar sus teorías, esto proporcionó métodos de experimentación a través de la creación de máquinas que permitieran, a través de diversos sistemas, estudiar conductas, reacciones, reflejos, aprendizaje, etc.

2 Aplicaciones de la Cibernética en la actualidad

Algunas disciplinas técnicas que están relacionadas con la Cibernética son:

- Robótica: Fundamentalmente se dedica al diseño, control y programación de robots.
- Informática: Se especializa en el Cálculo Automático y las máquinas que lo realizan.
- Telemática: Estudia los procesos de transmisión de información mediante telecomunicaciones aplicando la informática.
- Inteligencia artificial: Se encarga del estudio y desarrollo de técnicas para conseguir que una computadora resuelva problemas en el ámbito convencional con intervención humana.
- Mecatrónica: Estudia la integración da las tecnologías implicadas en los sistemas electromecánicos.

Los avances en la Cibernética son interesantes, actualmente se ha logrado conectar el sistema nervioso a una máquina e incluso, se ha mandado la información de los impulsos nerviosos a través de una red de computadoras conectándose en otro sistema nervioso humano o acabar en un brazo robótico que puede ser controlado a distancia con el simple movimiento de la mano.

Ejercicio

Relaciona las siguientes columnas.

1. Se encarga de efectuar la dirección y la comunicación entre los seres vivos y las máquinas.	( ) Norbert Wiener
2. Es aquel que modifica su estado y que incluye toda una serie de sistemas y elementos simples.	( ) Teoría de la información

3. Se le considera el padre de la Cibernética	( ) Cibernética
4. Su objetivo es encontrar fundamentos en las operaciones de procesamiento de señales tales como compresión de datos, almacenamiento y comunicación.	( ) Neurofisiología
5. Ciencia que descarga las neuronas de forma análoga para determinar un digito en escala binaria.	( ) Sistema dinámico complejo
6. Griego que descubrió el sistema de circulación de sangre.	( ) Blaise Pascal
7. Construyó la primera sumadora automática.	( ) Robótica
8. Ciencia que se enfoca al estudio de las propiedades de los entes abstractos y de sus relaciones.	( ) Biónica
9. Aplicación que se dedica al diseño, control y programación de robots.	( ) W. Harvey
10. Ciencia que se encarga de estudiar los sistemas biológicos aplicándolo con la física, biología.	( ) Matemáticas

TEMA **5**

Obra de distintos autores en trabajos científicos sobre la Cibernética

Objetivo:

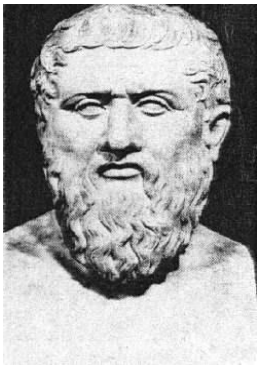
- Mencionar algunos precursores científicos de la Cibernética.

1 Introducción

El surgimiento de la Cibernética se debe a la integración de estudios realizados por matemáticos, físicos, ingenieros, filósofos y técnicos con el propósito desarrollar mecanismos que imiten acciones realizadas por los seres vivos y que se puedan comunicar con los seres humanos.

Norbert Wiener es considerado el padre de La Cibernética, porque propuso el término "Cibernética" para denominar a la rama de la ciencia encargada de estudiar los sistemas de control y la comunicación entre las máquinas y los organismos vivos, así mismo se mencionan pero como, como es natural, existen aportes relevantes de otros científicos en esta área.

2 Platón



Nacido el 427 a. C. en Atenas o Egina y murió próximo a los 80 años en Atenas en el año 348 o 347 a.C.

Fue un filósofo de la Grecia Antigua, es reconocido como una de las figuras más importantes de la filosofía occidental; incluso las prácticas religiosas deben mucho a su pensamiento. Fue el fundador de la Academia, el primer instituto de enseñanza superior de aquella época. Fuel quien utilizó por primera vez el termino *kibernetes* para definir el piloteo o gobierno de naves.

La Academia fundada por Platón no era un centro de enseñanza abstracta. Las ciencias manejadas hasta el momento (geometría, aritmética, astronomía, armonía) eran los campos fundamentales de investigación dentro del recinto. Platón desarrolló y mejoró las técnicas didácticas existentes hasta el momento.

3 Blaise Pascal

Nació el 19 de junio del año 1623 en la ciudad de Clermont-Ferrand, región de Auvernia, Francia y murió el 19 de agosto del año 1662 en París

Físico y matemático inventó una de las primeras calculadoras mecánicas en 1642, el aparato llamada Pascalina que calculadora basada en el uso de ruedas y engranes para realizar operaciones aritméticas de suma y resta. Algunas aportaciones fueron el diseño y construcción de calculadoras mecánicas, investigaciones sobre los fluidos, aportes a la teoría de la probabilidad y la clarificación de los conceptos de vacío y presión.

Pascal demuestra una gran inteligencia, por lo que su padre decidió iniciarlo rápidamente en las matemáticas y geometría.

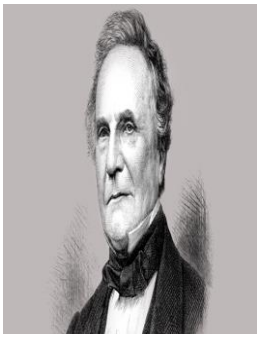
4 Joseph Marie Jacquard

Nació en Lyon, Francia el 7 de julio de 1752 y falleció el 7de agosto de 1834, Oullins, Francia.

Inventor de máquinas sofisticadas, su fama fue creciendo, hasta que en 1799 Napoleón le dio trabajo en el Conservatorio de Artes y Oficios como maestro inventor. En 1805 presentó el telar de los Jacquar. Este era un telar que utilizaba tarjetas perforadas, era una máquina que permitía fabricar telas con hilos de distintos colores y complicados dibujos mediante el uso de tarjetas perforadas, y podía ser manejada por un solo operario. Los hilos de urdimbre se movían de forma independiente para conseguir el dibujo deseado. También automatizó las 'arcadas' para poder tejer piezas de dibujo complicado.

Esta máquina trabajaba mediante un paquete de tarjetas de cartón perforadas que se cambiaban accionando un pedal, y que activaban un complejo mecanismo de cuerdas y plantas para elevar de forma alternativa un número diferente de hilos de urdimbre para la colocación de los hilos de la trama.

5 Charles Babbage



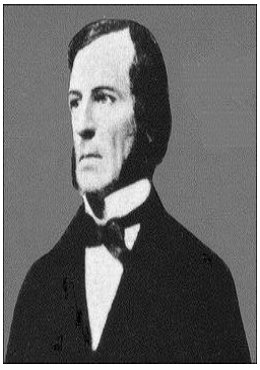
Nació: 1792 en Teignmouth, Inglaterra y falleció 1871 en Londres, Inglaterra.

Se considera el padre de las computadoras digitales, pues en 1822 construyó un modelo de calculadora mecánica (*la máquina diferencial*), así llamada porque para realizar sus cálculos utilizaba la teoría matemática de las diferencias finitas.

En 1834, diseñó *la máquina analítica*, capaz de realizar toda clase de operaciones con una estructura semejante a la de las computadoras modernas, pero no llegó a construirse, quedando olvidada hasta que se publicaron las notas de Babbage en 1937.

La máquina se habría programado por medio de tarjetas perforadas, según un procedimiento inventado por la condesa Ada Byron, a quien se considera la primera programadora de la historia.

6 George Boole



Nació el 2 de noviembre de 1815 en Lincoln, Lincolnshire y falleció el 8 de diciembre de 1864 en Ballintemple, Irlanda.

En 1854 publicó las leyes del pensamiento sobre las cuales son basadas las teorías matemáticas de Lógica y Probabilidad.

Inicia el álgebra de la lógica llamada Algebra Booleana la cual ahora encuentra aplicación en la construcción de computadores, circuitos eléctricos, etc. El legado se basa en una teoría matemática que simplifica los enunciados que tenían por respuesta «Sí» o «No», usando para ello la aritmética binaria.

Se le considera padre de las ciencias computacionales, por su invención del álgebra booleana.

En 1859 se publicó el cálculo de las diferencias finitas, "Tratado sobre el Cálculo de las Diferencias Finitas" (1860), y métodos generales en probabilidad.

7 Herman Hollerith

Herman Hollerith nace en Bufallo, Nueva York, el 29 de febrero de 1860 y fallece el 17 de noviembre de 1929.

Al momento de estudiar las preguntas de los censos se percata de que todas son preguntas cerradas “sí” o “no”. Esto hizo que Herman creara una tarjeta de cartulina de 80 columnas, en la cual se podría responder las preguntas con perforaciones en determinadas posiciones. La máquina que creó fue considerada como la primera computadora y fue usada por el gobierno de los Estados Unidos para el censo de 1890. Hollerith fue el primero en utilizar la cinta de papel para codificar los datos. La cinta estaba compuesta por campos marcados con tinta, en los que se podía codificar datos perforándola o no, según el dato a registrar.

Las tarjetas perforadas fueron utilizadas por primera vez por J. Jacquard en 1902 para controlar el funcionamiento de los telares de forma mecánica. Pero Hollerith fue quien las utilizó para trabajar la información y para la informática, usando herramientas electromecánicas.

8 Norbert Wiener

Matemático estadounidense. Nació: 1894 en Columbia (Misuri), E. U. Falleció: 1964 en Estocolmo, Suecia, conocido como el padre de la Cibernética.

Trabajó sobre los procesos estocásticos, en los que intervienen variables aleatorias (como el movimiento browniano de Brown) y en la descomposición de funciones en componentes periódicas. Pero se le conoce especialmente como padre de la Cibernética, porque la definió como " la ciencia del control y la comunicación entre los seres vivos y las máquinas". La idea le vino de su trabajo durante la segunda guerra

mundial, cuando se encargó de diseñar el sistema de control automático de un arma de defensa antiaérea, capaz de seguir y derribar un blanco móvil.

Wiener concebía la Cibernética como una ciencia multidisciplinaria, basada en el estudio de los procesos comunes a los seres vivos y las máquinas, como el control y las comunicaciones.

9 Arturo Rosenblueth

Nació en Chihuahua, Chih., en 1900; murió en México, D.F., el 20 de septiembre de 1970.

Fue profesor de fisiología en la Universidad Nacional Autónoma de México (1927-1930), profesor e investigador en la Escuela de Medicina de la Universidad de Harvard, donde trabajó con el matemático Norbert Wiener con quién sentó las bases de la Cibernética, jefe del Laboratorio de Fisiología del Instituto Nacional de Cardiología (1944-1960) y jefe del Departamento de Fisiología y director del Centro de Investigación Científica y Estudios Superiores del Instituto Politécnico Nacional. Investigó el mecanismo químico de la trasmisión de los impulsos nerviosos y elaboró, con Walter B. Cannon, la teoría de las dos simpatinas.

Sus investigaciones sobre la forma en que se transmiten señales en el sistema nervioso sirvieron de guía para desarrollar la cibernética.

Las investigaciones de Wiener se apoyaron en su trabajo junto al neurofisiólogo mexicano Arturo Rosenblueth a quien había conocido en 1942, durante un congreso en Nueva York. Con él estudió las semejanzas entre el cerebro humano y los robots y sistemas automáticos.

10 John Von Neumann

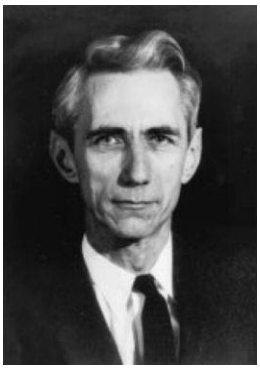
Nació en Budapest el 28 de diciembre de 1903, en el seno de una familia de banqueros acomodada. Murió el 8 de febrero de 1957 bajo seguridad militar por miedo a que revelase secretos militares mientras se estaba medicando.

Realizó varias contribuciones en la física cuántica, análisis funcional, teoría de conjuntos, ciencias de comunicación, economía, análisis numérico, cibernética, hidrodinámica de expresiones, estadística y otros campos de las matemáticas.

Incursionó en diferentes ramas de la ciencia durante toda su vida, desde la Teoría de conjuntos y mecánica cuántica, el diseño de computadoras y teoría de juegos, astrofísica y meteorología.

Escribió un ensayo sobre la construcción y diseño de las computadoras, describiendo la estructura de una computadora dividiendo el diseño en una unidad de procesamiento, unidad de control, memoria interna y dispositivos de entrada y salida. Este modelo es en el que se basa la arquitectura de las computadoras actuales.

11 Claude Shannon

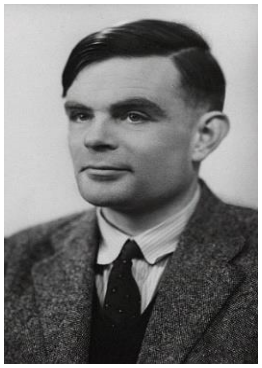


Nació el 30 de abril de 1916 en Petoskey (Michigan, EEUU). Shannon sienta los fundamentos matemáticos de la revolución tecnológica de la segunda mitad del siglo XX.

Desde el estudio del álgebra 'booleana', teoria acerca del código binario, que es la base del lenguaje digital, a partir de unidades básicas del concepto de información, definidas por dos estados: el 'si' y el 'no', el 0 y el 1, abierto/cerrado, verdadero/falso, blanco/negro.

Más allá de la formulación teórica, Shannon construyó circuitos y máquinas basadas en los flujos binarios de información, mediante interruptores y en las que se anticipaban los embriones de muchos de los desarrollos de las décadas posteriores.

12 Alan Turing



Nació en Londres, 1912, muere Wilmslow, Reino Unido, 1954.

Publicó en 1793 un célebre artículo en el que definió una máquina calculadora de capacidad infinita (máquina de Turing) que operaba basándose en una serie de instrucciones lógicas, sentando así las bases del concepto moderno de algoritmo.

Turing describió en términos matemáticos precisos, cómo un sistema automático con reglas extremadamente simples, podía efectuar toda clase de operaciones matemáticas expresadas en un lenguaje formal determinado. La máquina de Turing era tanto un ejemplo de su teoría de

computación como una prueba de que un cierto tipo de máquina computadora podía ser construida.

Definió además un método teórico para decidir si una máquina era capaz de pensar como un hombre (test de Turing) y realizó contribuciones a otras ramas de la matemática aplicada, como la aplicación de métodos analíticos y mecánicos al problema biológico de la morfogénesis.

Ejercicio

I. Subraya la respuesta correcta.

1. Científico que presentó el Telar con tarjetas perforadas.
   a) Platón b) Blaise Pascal c) Joseph Marie Jacquard d) Norbert Wiener

2. Utilizó por primera vez el término Cibernética.
   a) Platón b) Blaise Pascal c) Joseph Marie Jacquard d) Norbert Wiener

3. Publicó las leyes del pensamiento sobre las cuales son basadas las teorías matemáticas de Lógica y Probabilidad.
   a) Platón b) George Boole c) Joseph Marie Jacquard d) Norbert Wiener

4. Científico mexicano que trabajó con el matemático Norbert Wiener con quién sentó las bases de la Cibernética.
   a) Platón b) George Boole c) Joseph Marie Jacquard d) Arturo Rosenblueth

5. Diseñó el modelo sobre el que se basa la arquitectura de las computadoras actuales.
   a) John Von Neumann b) George Boole c) Blaise Pascal d) Arturo Rosenblueth

6. Inventó la primera calculadora basada en el uso de ruedas y engranes para realizar operaciones aritméticas de suma y resta.
   a) Platón b) Blaise Pascal c) Joseph Marie Jacquard d) Norbert Wiener

7. Construyó un modelo de calculadora mecánica (*la máquina).* Por lo que es considerado el padre de la computación.
   a) Charles Babbage b) Blaise Pascal c) George Boole d) Norbert Wiener

8. Fue el primero en utilizar la cinta de papel para codificar los datos.
   a) Charles Babbage b) Blaise Pascal c) George Boole d) Herman Hollerith

9. Definió a la Cibernética como la ciencia del control y la comunicación entre el animal y e la máquina.

a) Platón b) Blaise Pascal c) Joseph Marie Jacquard d) Norbert Wiener

10. Construyó circuitos y máquinas basadas en los flujos binarios de información

a) John Von Neumann b) Claude Shannon c) Blaise Pascal d) Arturo Rosenblueth

TEMA **6**

Sistema

Objetivo:

- Definir lo que es un sistema y cada uno ellos en base a sus elementos.

1 Introducción

Los elementos de un sistema pueden interactuar entre si hasta lograr lo que se pretende recibiendo datos, o energía. Por lo tanto, un sistema está integrado por entradas, salidas y procesos de tal manera que pueden clasificarse de acuerdo al medio ambiente, según su naturaleza y según su origen.

2 Concepto

Un sistema es un conjunto de partes o elementos organizados y relacionados que interactúan entre sí para cumplir un determinado objetivo. Los sistemas reciben datos (entrada), energía o materia del ambiente y proveen información (salida), energía o materia.

3 Elementos

Un sistema está compuesto por los siguientes elementos:

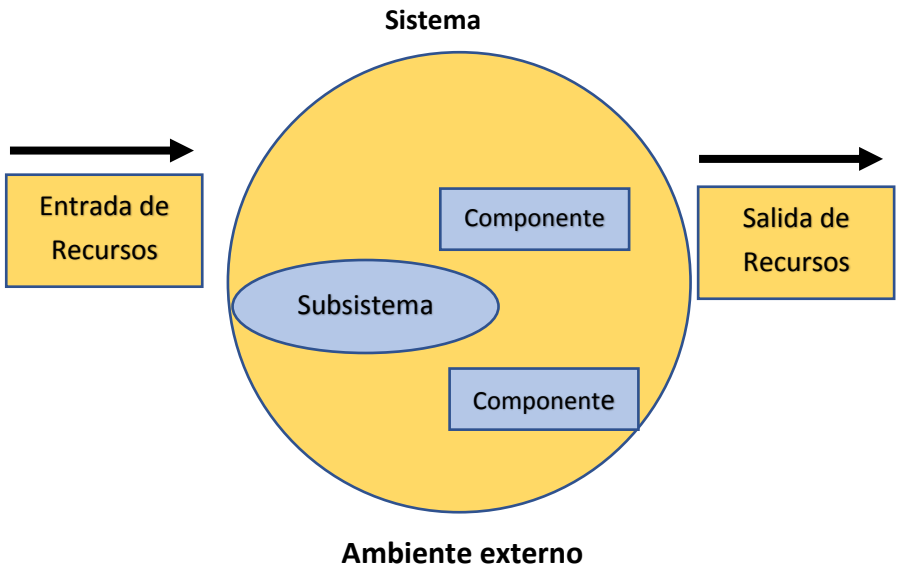
Entrada: es una variable del sistema tal que una modificación de su magnitud o condición puede alterar el estado del sistema.

Salida: es una variable del sistema cuya magnitud o condición se mide.

Proceso: aquellos elementos que al ingresar al sistema presentan una transformación.

4 Ambiente

Es el medio externo que envuelve física o conceptualmente a un sistema. Por lo tanto un sistema tiene interacción con el ambiente, mediante el cual recibe entradas y devuelve salidas. El ambiente también puede ser una amenaza para el sistema.



Ambiente o entorno próximo: es aquel entorno que ingresa el sistema, puede influir en este y en el sistema.

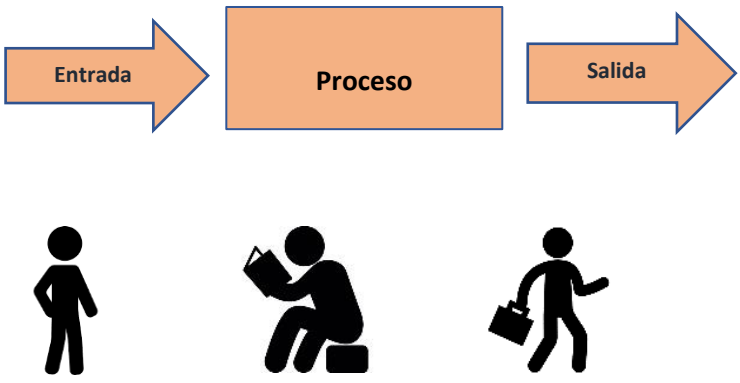
Ambiente o entorno lejano: el sistema no puede influir en este, pero este sí puede influir en el sistema

5 Clasificación de los Sistemas

Sistemas abiertos: presentan intercambio con el ambiente, a través de entradas y salidas. Intercambian energía y materia con el ambiente. Se adaptan al medio ambiente.

Su estructura es óptima cuando el conjunto de elementos del sistema se organiza, aproximándose a una operación adaptativa.

Ejemplo: **Biblioteca**



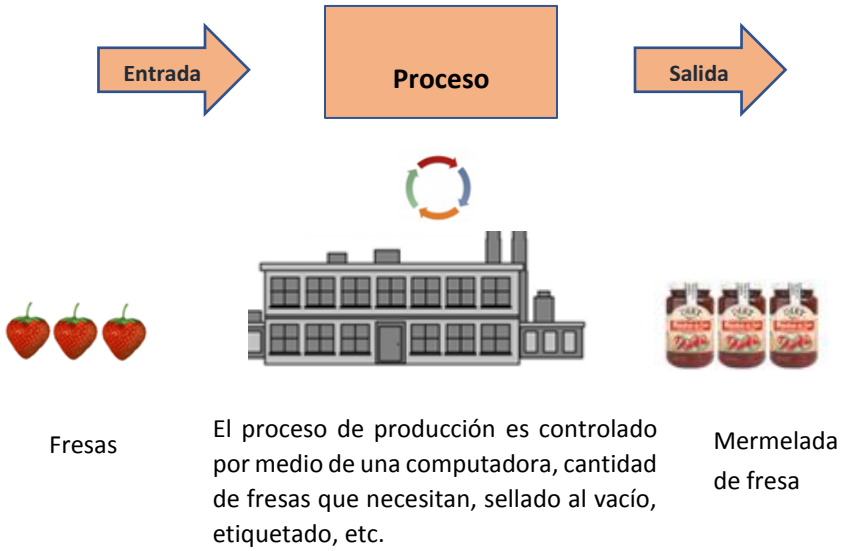
Ingresa la persona - Realiza consulta de libros - Solicita un libro

Sistemas cerrados: son herméticos a cualquier influencia ambiental. No reciben ningún recurso externo y no producen nada que sea enviado hacia fuera. Se le da el nombre de sistema cerrado aquellos sistemas cuyo comportamiento es determinista y programado y que opera

con un pequeño intercambio de energía y materia con el ambiente. Estos no interactúan con su ambiente, son completamente autónomos.

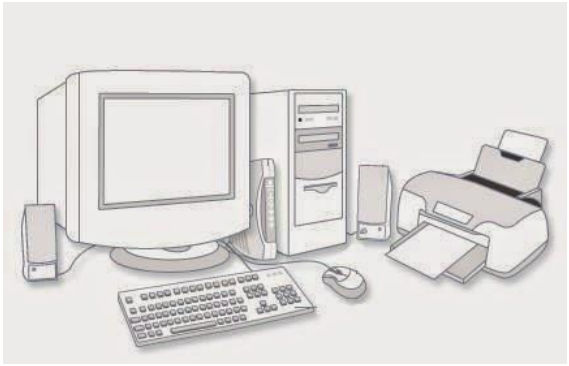
Un sistema cerrado es aquel que a las mismas entradas produce las misma salidas.

Ejemplo: Procesadora de **mermelada**



Sistemas físicos o concretos*:* están compuestos por equipos, maquinaria, objetos y cosas reales.

Ejemplo: Una **computadora**, que tiene pantalla, componentes de entrada, microprocesador, memoria RAM, disco duro, etc.



Sistemas abstractos: se encuentran compuestos por conceptos, planes, hipótesis, pensamientos e ideas.

Ejemplo: El **software** permite controlar e interactuar con un sistema.

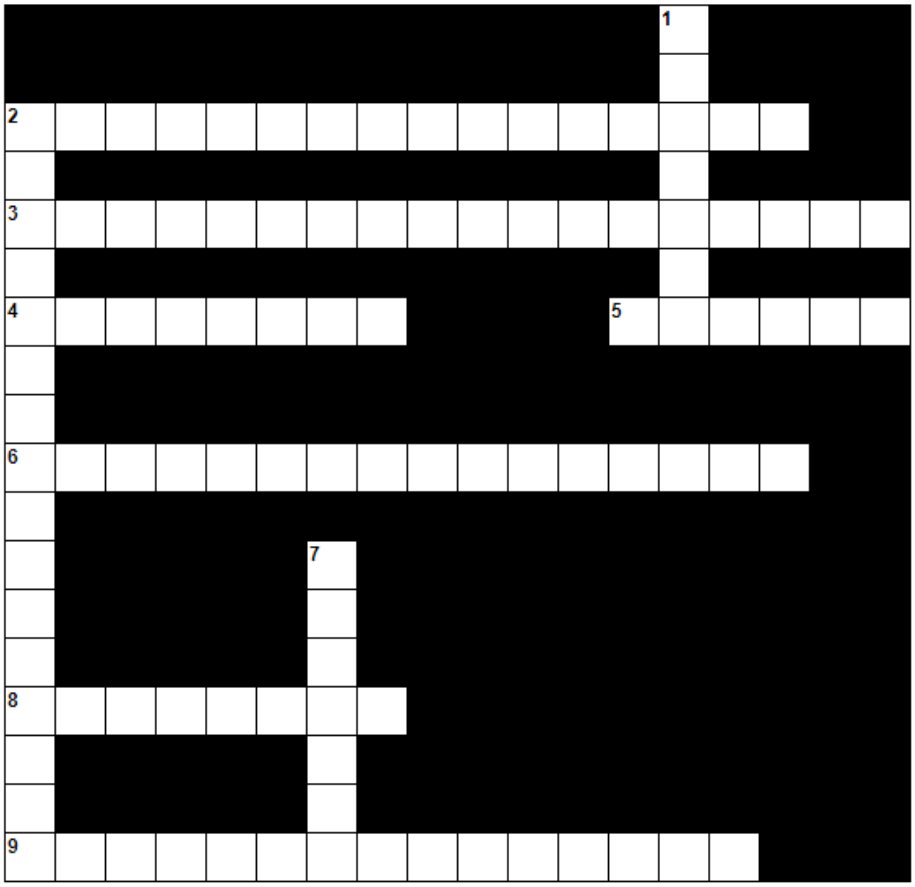
Sistemas aislados: son aquellos sistemas en los que no se produce intercambio de materia ni energía.

Ejemplo: **Termo.** Durante un cierto tiempo, los termos logran aislar el calor contenido en su interior y evitar la fuga de energía hacia el medio ambiente, al mismo tiempo impidiendo el derramamiento del contenido o la introducción del mismo. No obstante, dado el tiempo suficiente, la inevitable fuga del calor ocurrirá y el contenido volverá a estar frío.

Entropía: Es la tendencia natural a la pérdida de orden en un sistema.

Ejercicio

Resuelve el siguiente crucigrama en base a los siguientes conceptos.



1. Conjunto de elementos organizados y relacionados que interactúan entre sí para cumplir un determinado objetivo.
2. Es una variable del sistema cuya magnitud o condición se mide.
3. Es una variable del sistema tal que una modificación de su magnitud o condición puede alterar el estado del sistema.
4. Es el medio externo que envuelve física o conceptualmente a un sistema.
5. Presentan intercambio con el ambiente, a través de entradas y salidas.
6. Son herméticos a cualquier influencia ambiental.
7. Es la tendencia natural a la pérdida de orden en un sistema.
8. Están compuestos por equipos, maquinaria, objetos y cosas reales.
9. Se encuentran compuestos por conceptos, planes, hipótesis, pensamientos e ideas.
10. Son aquellos sistemas en los que no se produce intercambio de materia ni energía.

TEMA 7

Sistema de Control

Objetivo:

- Identificar la clasificación de los sistemas de control.

1 Introducción

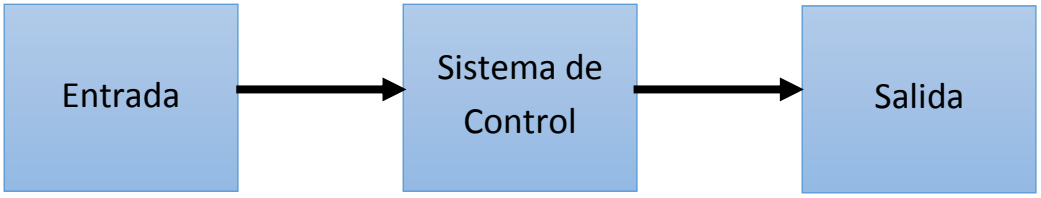
Un sistema de control está definido como un conjunto de elementos que pueden regular su propia conducta o la de otro sistema con el fin de lograr un funcionamiento predeterminado. Los sistemas de control se clasifican en lazo abierto y lazo cerrado.

2 Sistemas de control

Es aquel que está integrado por una serie de elementos que actúan en conjunto para cumplir un objetivo. Un sistema no es independiente, sino que está estrechamente relacionado entre sí, de manera que los cambios que se producen en uno de ellos pueden influir en los demás.

Un sistema de control se lleva a cabo mediante un conjunto de componentes mecánicos, hidráulicos, eléctricos y/o electrónicos que, interconectados, reúnen información acerca del funcionamiento, relacionan este funcionamiento con datos anteriores y, de ser necesario, cambian el proceso para lograr el objetivo.

Para entenderlo mejor, es necesario considerar que sus elementos reciben una orden o entrada y producen una respuesta o salida, comparándolas y usando la diferencia como medio de control.



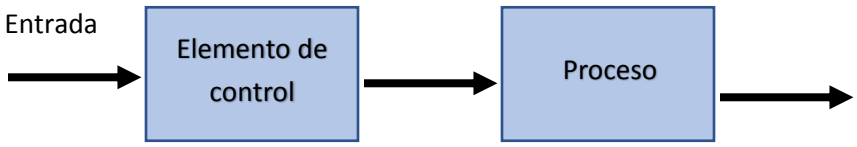
3 Lazo abierto

En este sistema se obtienen los datos de entrada y se ejecuta el proceso de control, en los cuales la salida no afecta la acción de control. En este tipo de sistemas de control en lazo abierto la señal de salida no influye sobre su regulación, ni se realimenta para compararla con la entrada.

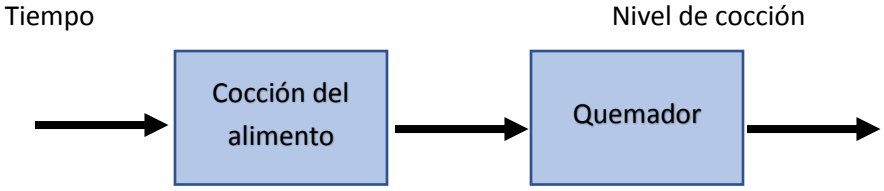
Cada entrada determina una posición de funcionamiento fijo en los elementos de control.

Elementos de un sistema de lazo abierto

- Entrada: Señal que se espera obtener mediante el proceso.
- Elemento de control: Se encarga de determinar qué acción se tomara dada la entrada al sistema de control.
- Proceso: Se encarga de controlar la variable.
- Salida: Variable controlada



Ejemplo: **Horno de Microondas** en este caso la variable controlada debería ser la cocción del alimento que introducimos, la variable de entrada sería el tiempo el cual lo ingresamos a través del tablero de control que vendría a ser parte de los componentes del sistema de control. Como sabemos intuitivamente nosotros digitamos el tiempo de cocción este puede ser o no el correcto sin embargo el microondas como tal no mide ni se retroalimenta de esta variable, en caso de no ser suficiente el tiempo nosotros volvemos a introducir otro tiempo.



4 Lazo cerrado

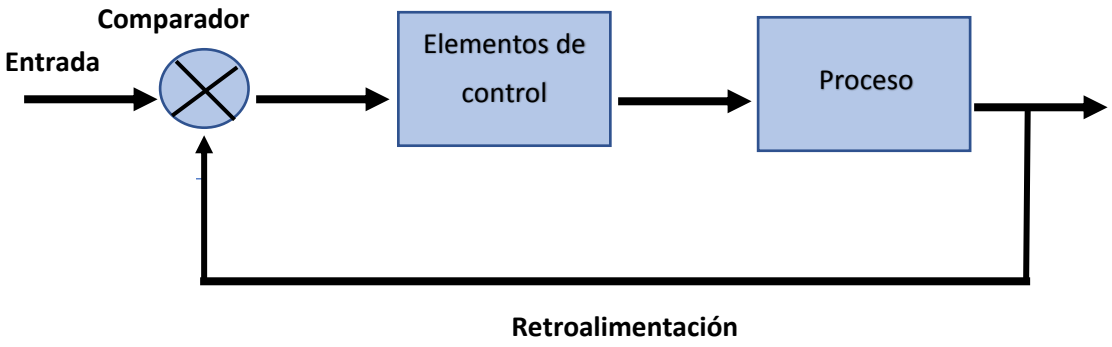
Son aquellos sistemas que poseen retroalimentación de la señal de salida que interviene en la regulación para reducir el error del sistema.

En este tipo de sistemas se alimenta al controlador, que es la diferencia entre la señal de entrada y la salida de realimentación, con el fin de reducir el error y llevar la salida del sistema a un valor conveniente.

Elementos de un sistema de lazo cerrado:

- Entrada: Señal que se espera obtener mediante el proceso.
- Elemento de comparación: Este elemento compara el valor requerido o de referencia de la variable por controlar con el valor medido de lo que se obtiene a la salida, y produce una señal de error la cual indica la diferencia del valor obtenido a la salida y el valor requerido.
- Elemento de control: Se encarga de determinar qué acción se tomará dada la entrada al sistema de control.
- Proceso: Se encarga de controlar la variable.
- Salida: Variable controlada.
- Retroalimentación: Proporciona la señal al elemento de comparación para determinar si hay o no error.

En el siguiente esquema se puede visualizar los descrito arriba:

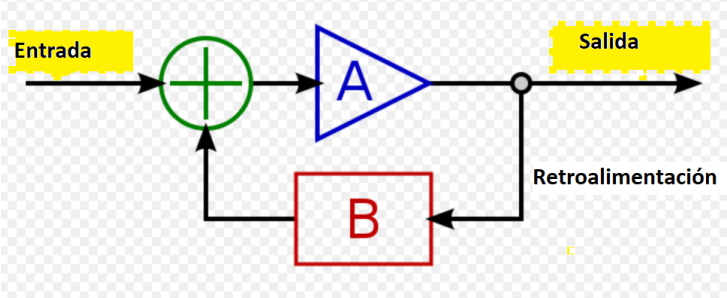


Esquema de un sistema de lazo cerrado

5 Retroalimentación

Es la información que recibe el sistema sobre los resultados de su propia acción, en función con la entrada y la salida.

La retroalimentación es el proceso mediante el cual un sistema obtiene y procesa información sobre las funciones que ejecuta para generar acciones correctivas, preventivas o de optimización.



Ejercicio

Marca con una X si es Falso o Verdadero el ejemplo qué se menciona de acuerdo con el tipo de sistema de control.

(F) (V) Un semáforo es un sistema de lazo abierto, ya que la señal de entrada es el tiempo asignado a cada luz (rojo, amarilla y verde) en cada una de las calles. El sistema cambia las luces según el tiempo indicado, sin importar que la cantidad de tránsito varíe en las calles.

(F) (V) Un equipo de aire acondicionado es un sistema de lazo cerrado, ya que cuenta con un sensor que permanentemente registra la temperatura ambiente, y con un comparador, que determina sí la temperatura es la deseada.

(F) (V) Un sistema de riego es lazo cerrado, porque no se detendrá al cabo de un tiempo fijo, sino cuando detecte que se está consiguiendo el objetivo buscado, es decir, que la humedad de las plantas es la adecuada. Y se pondrá en marcha, no a una hora determinada, sino en cualquier momento en que la humedad se sitúe por debajo de un valor determinado.

(F) (V) Una caja fuerte es un sistema abierto, el contenido en las cajas fuertes está separado por gruesas capas herméticas de metal de su entorno, aislado de la materia y de la energía, al menos en condiciones normales: si la arrojamos a un volcán es seguro que se derrita y se incinere su contenido.

(F) (V) Un software es un sistema cerrado, porque es un conjunto de líneas de código organizadas según reglas, gramáticas y métricas que fueron creadas por el mismo hombre o un grupo de personas y son conceptos generados del intelecto, pero que en la naturaleza puramente no tienen ningún lugar; son parte y producto del pensamiento pero constituyen un sistema.

TEMA **8**

Modelos

Objetivo:

- Definir que es un modelo y los tipos de modelos.

1 Introducción

Un modelo es la representación de algo real en la forma más completa posible, sin pretender aportar una réplica de lo que existe en la realidad. Los modelos son útiles para describir, explicar o comprender mejor la realidad, se clasifican en varios tipos como naturales y artificiales, analógicos y digitales, matemáticos y conceptuales.

2 Concepto

Los modelos Son aquellos que representan la realidad por medio de abstracciones, Se enfocan en ciertas partes de un sistema por lo menos, en aquellos por los que se tiene un interés especifico. Los modelos son creados empleando herramientas de modelado.

3 Tipos

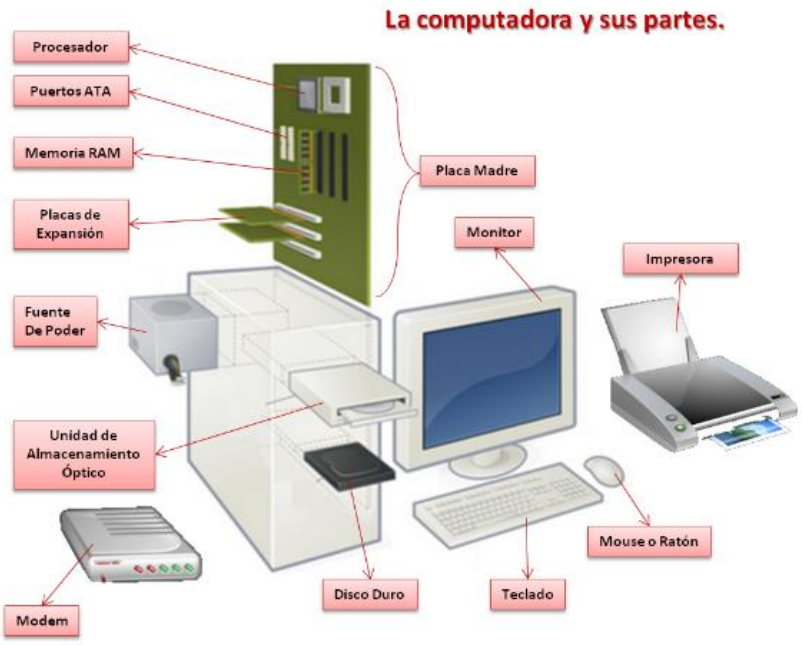
Modelo Natural y Artificial

El **Modelo Natural** es un conjunto basado en una o varias interpretaciones de carácter cognitivo, el cual se encuentra impuesto.

Ejemplo. Un **árbol**, su objetivo es proveer de oxígeno al medio ambiente, albergar especies vivas, ser alimento de especies vivas. Sus elementos: hojas, clorofila, ramas, frutos, etc.

El **Modelo Artificial** es creado por el hombre para representar situaciones de acuerdo con sus necesidades.

Ejemplo: Una **computadora**, su objetivo es automatizar tareas. Sus elementos son el teclado, el mouse, las placas, etc.



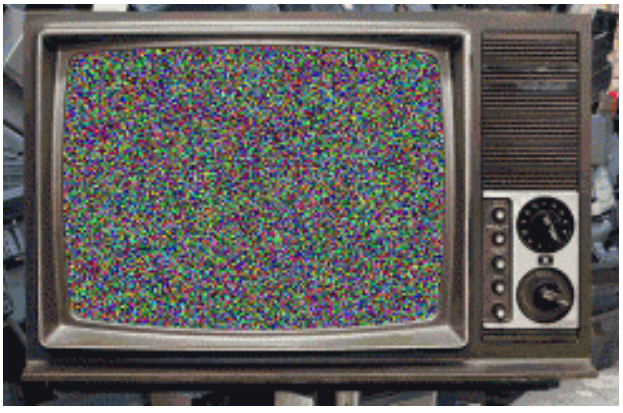
Modelo Analógicos y Digitales

El **Modelo Analógico** es una representación material de un objeto o un proceso para entender mejor su origen, formación o funcionamiento.

Se aplica en diferentes ciencias e ingenierías para validar hipótesis formando un modelo conceptual de cierto proceso u objeto mediante el cálculo numérico.

En este tipo de modelos es difícil almacenar, manipular, comparar, calcular y recuperar información con exactitud cuándo ha sido guardado.

Ejemplo: La **televisión analógica** ocupa excesivos recursos del espectro electromagnético; lo que conlleva un aumento de las estaciones reemisoras y problema de interferencias.



El **Modelo Digital** es una estructura numérica de datos el cual puede ser analizado uno a uno. Los modelos digitales son importantes en la tecnología moderna, especialmente en la computación y sistemas de control automático.

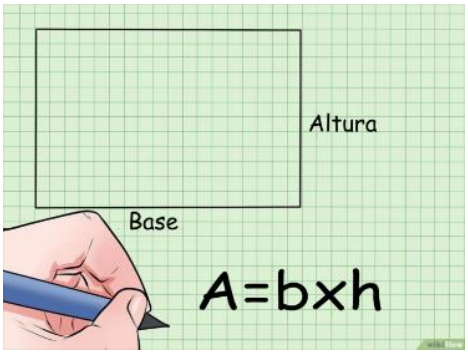
Ejemplo: La **televisión digital** consiste en un sistema de codificación de la señal de vídeo en forma de valores numéricos en formato binario.

Modelo Matemático

Es uno de los modelos científicos, que se basa en expresar los instrumentos de la teoría matemática, declaraciones, relaciones, proposiciones sustantivas de hechos o de contenidos simbólicos: en el cual se encuentran variables, parámetros, entidades y relación entre variables y entidades.

Es decir, es la traducción de la realidad física para poder aplicar los instrumentos y técnicas de las teorías matemáticas para estudiar el comportamiento de sistemas complejos, y posteriormente invertirlo para traducir los resultados numéricos a la realidad física.

Ejemplo: Calcular el área de un rectángulo.



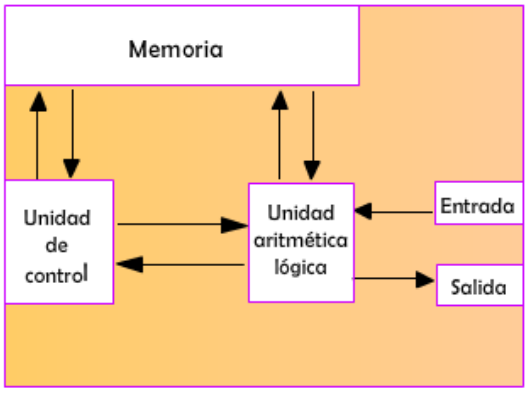
Modelos Conceptuales

El modelo conceptual de un dispositivo se forma mediante la interpretación de sus acciones percibidas y su estructura visible.

Se puede entender como un mapa de conceptos, planes, hipótesis e ideas y sus relaciones acerca de la naturaleza.

Este tipo de modelos implican un alto nivel de abstracción basándose en aspectos conceptuales y semánticos los cuales se consideran para la comprensión de su representación mediante símbolos los cuales representan atributos y objetos, que algunas veces sólo existen en el pensamiento de las personas.

Ejemplo: Estructura del Modelo de Von Neumann



Ejercicio

I. Relaciona las siguientes columnas en base a los siguientes ejemplos de acuerdo con el tipo de modelo.

( ) **a) Modelo conceptual**

b) Modelo analógico

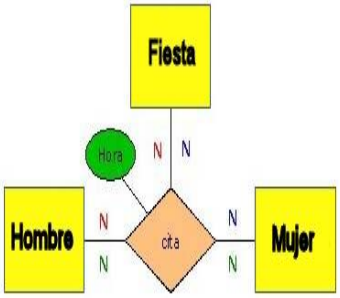
( )

c) Modelo natural

( )

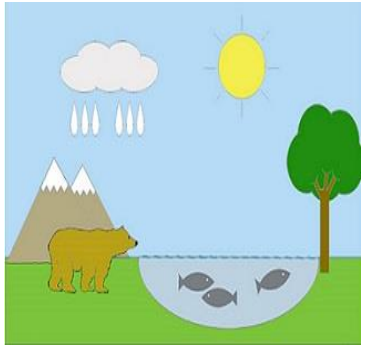
d) Modelo artificial

( )



e) Modelo digital

( )



f) Modelo matemático

( )

TEMA 9

Elementos para modelar un sistema

1 Entrada y salida

Entrada: Son los ingresos al sistema tales como recursos materiales o de información.

Pueden ser de diferente índole, por ejemplo: recursos materiales, recursos humanos o información. Las entradas pueden clasificarse de la siguiente manera:

- ✓ En serie: es el resultado o salida de un sistema anterior con el cual el sistema en estudio está relacionado en forma directa.
- ✓ Aleatorio o al azar: representan entradas potenciales para un sistema, esto significa que pueden o no llegar.
- ✓ Retro acción o retroalimentación: son entradas que modifican el funcionamiento futuro del sistema a partir del estudio o control de las salidas anteriormente producidas por el propio sistema.

Salida: Son el resultado del funcionamiento del sistema.

La salida es el resultado del procesamiento de la información de entrada por parte del sistema y que, por otra parte, cumple con el propósito para el cual existe el sistema. Las salidas de un sistema se convierten en la entrada de otro sistema, que la procesara para convertirla en otra salida, repitiéndose este ciclo indefinidamente.

2 Proceso

Proceso: Es lo que transforma una entrada en salida.

Esta acción la podemos ver realizada por una máquina, un individuo, una computadora, un producto químico, una tarea realizada por un miembro de la organización, etc.

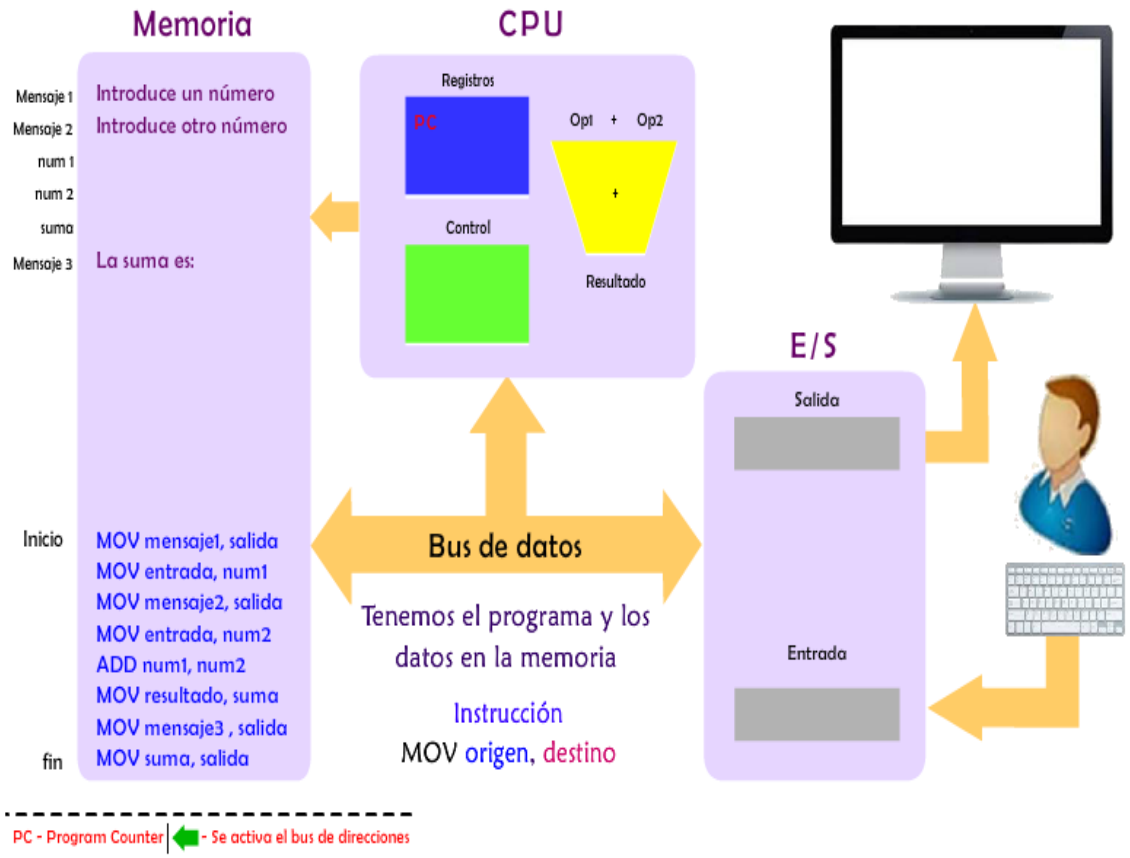
El proceso de transformación de entradas en salidas puede ser conocido o no. Cuando tenemos conocimiento de las entradas, las salidas y como se lleva a cabo el proceso que transforma las entradas en salidas, denominamos a este proceso "**caja blanca**". Cuando por el contrario se conocen las entradas y las salidas, pero no se conoce en detalle el proceso mediante el cual las entradas se transforman en salidas, decimos que se trata de una "**caja negra**". Generalmente el proceso de transformación puede ser desconocido porque es complejo, o simplemente porque cuando se está desarrollando la solución no conocemos como se llevará ca cabo el procesamiento de los recursos de entrada.

La "**caja negra**" se utiliza para representar a los sistemas cuando no sabemos qué elementos o cosas componen al sistema o proceso, pero sabemos que a determinadas entradas le corresponden determinadas salidas.

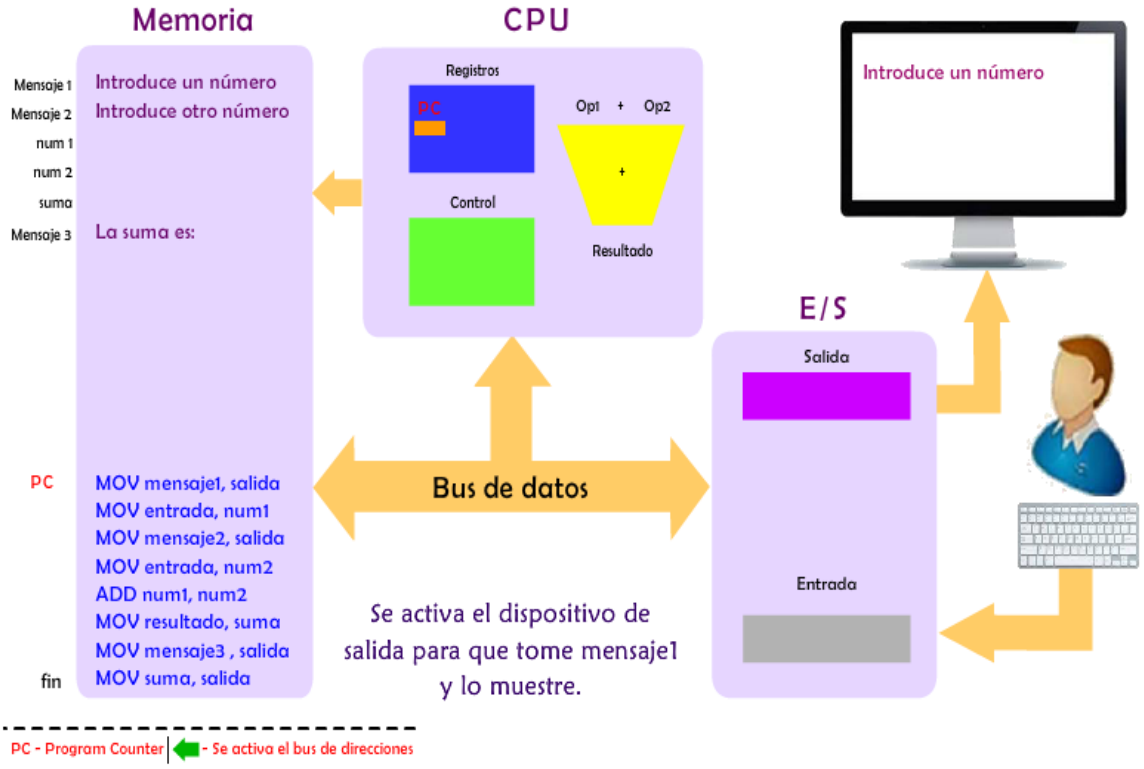
Veamos a continuación un ejemplo de los conceptos anteriormente expuestos.

Ejemplo: **La suma de dos números mediante el Modelo de Jhon Von Neumann**

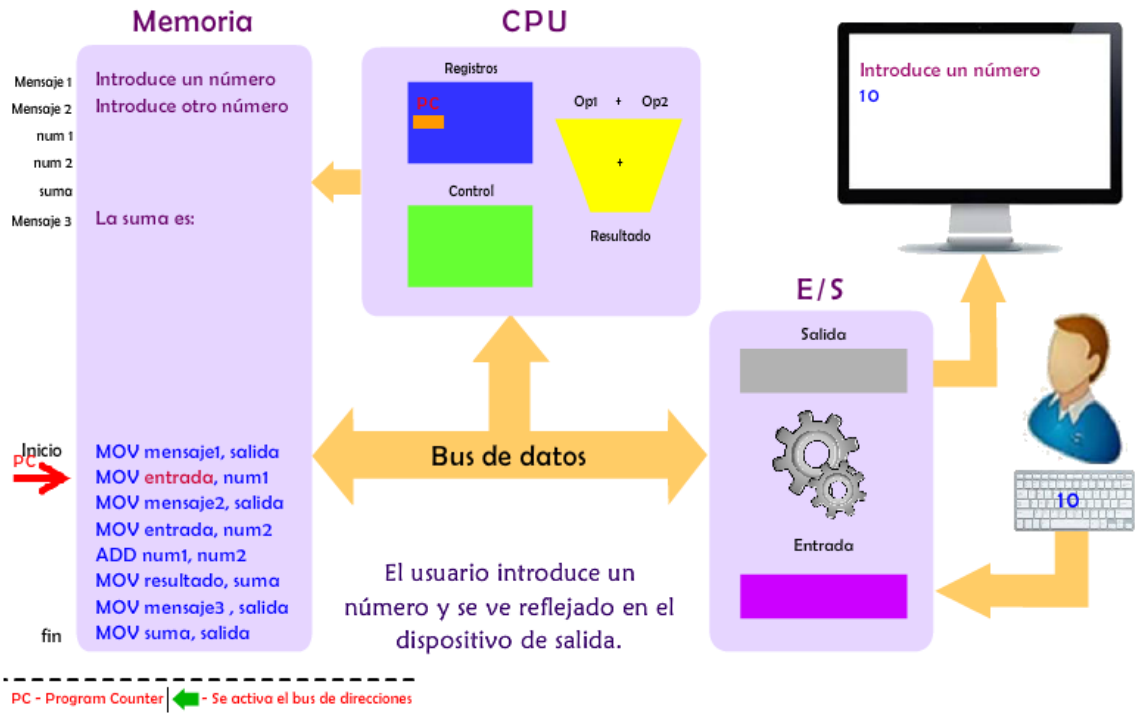
- Los datos y el programa se encuentran en la memoria.



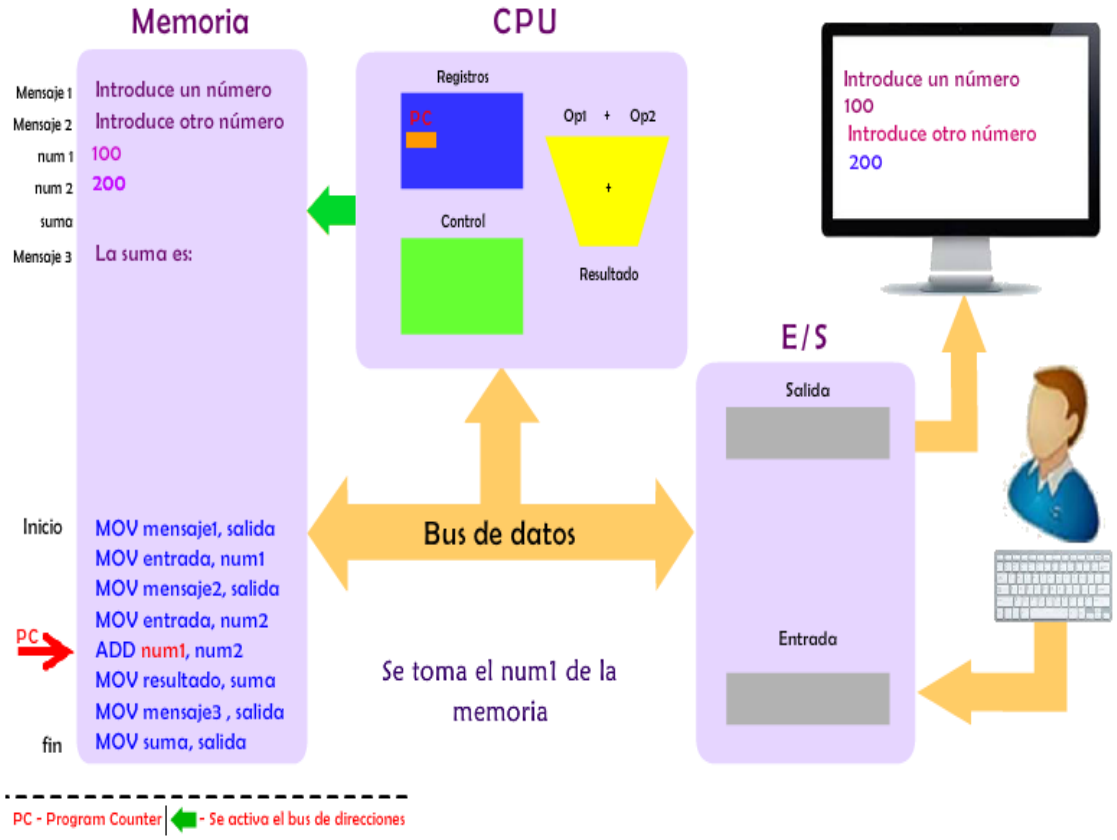
- Se inicia la ejecución, se toma el mensaje 1.



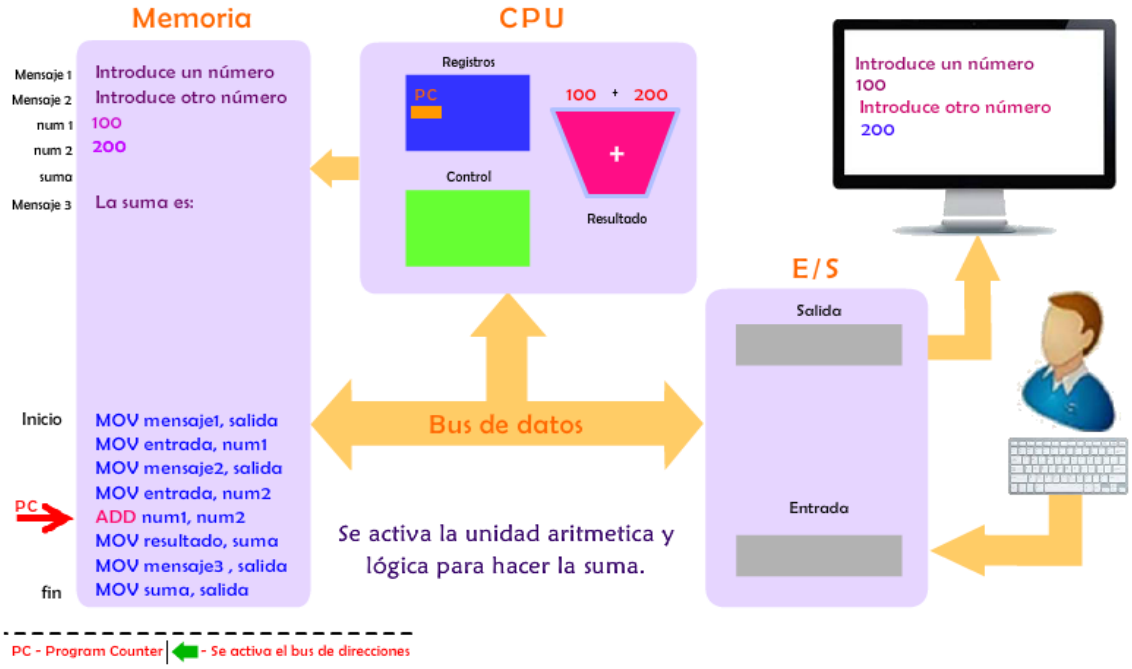
- Se introduce un número y se ve reflejado en la entrada.



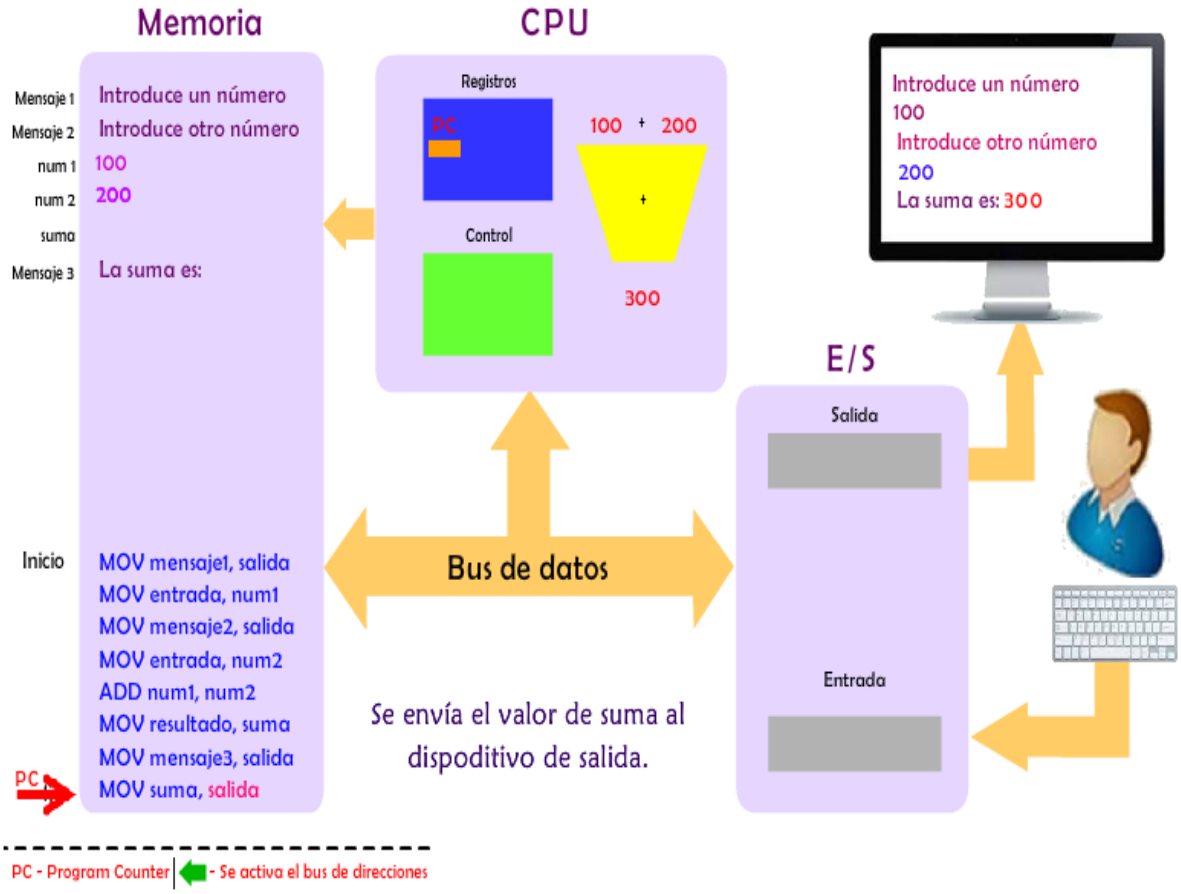
- Se introduce otro un número de manera que ambos valores pasan por el bus de datos y queden cargados en la memoria.



- Se activa la unidad aritmética lógica para la suma.



- El resultado se envía a la memoria y al dispositivo de salida.




UNIDAD 2
Circuitos Lógicos
Sistemas de Numeración
Circuitos Lógicos
Funciones Booleanas
Simplificación
Diseño de circuitos lógicos
UNIDAD 2

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.

Aprendizajes:

El alumno:

- Convierte números entre los sistemas de numeración binario, octal, decimal y hexadecimal.
- Realiza operaciones aritméticas con el sistema de numeración binario.
- Describe los conceptos de interruptor, circuito eléctrico, compuerta lógica y circuito lógico.
- Construye tablas de verdad de funciones booleanas.
- Construye la función booleana a partir de la tabla de verdad, empleando suma de productos.
- Simplifica funciones booleanas utilizando postulados y teoremas básicos.
- Aprende a utilizar el protoboard o un simulador.
- Construye un semisumador.
- Construye un sumador completo.
- Diseña circuitos lógicos a partir de un problema cotidiano usando la metodología aprendida.

TEMA 1

Sistemas de numeración

Objetivo:

- Convertir números entre los sistemas de numeración binario, octal, decimal y hexadecimal.

1 Introducción

Para poder representar cantidades utilizamos un sistema numérico que se conoce como **Decimal**, esto significa que utilizamos diez dígitos diferentes para representar una cantidad.

Imagen 1. Dígitos del sistema decimal.

El sistema decimal es además es un sistema posicional, esto quiere decir que cada uno de estos dígitos tienen un valor diferente con base a la posición que tienen entre sí. Al estar basado el sistema decimal en 10 dígitos, cualquier cantidad se puede expresar **como una suma de potencias de 10** y es precisamente estas potencias las que dan un valor relativo conforme a la posición en la que se expresa una cantidad.

Hagamos un ejemplo, pensemos en la cantidad 21. El número 2 y el número 1 tienen un valor diferente de acuerdo a la posición que ocupan. Tal como mencionamos en el párrafo anterior, formamos las cantidades como la suma de potencias de 10, veamos la siguiente tabla:

10^1	10^0
2	1

El dígito 2 tiene un valor diferente por la posición que ocupa, en este caso se multiplica por 10 elevado a la primera potencia. Además, el número uno tiene otro valor que también depende de su posición la cual corresponde a multiplicarlo por 10 elevado a la potencia cero.

Por tanto, la cantidad 21 se forma de esta forma:

$(2 \times 10^1) + (1 \times 10^0)$

Es igual a $(2 \times 10) + (1 \times 1) = 20 + 1 =$ **21**

Veamos un segundo ejemplo, para la cantidad 325 si la representamos por las potencias de 10 que le corresponden:

10^2	10^1	10^0
3	2	5

Podemos observar que **cada digito tiene un valor diferente conforme a su posición**, centenas para el número 3, decenas para el número 2 y unidades para el número 5. Es decir:

$(3 \times 10^2) + (2 \times 10^1) + (5 \times 10^0)$

$(3 \times 100) + (2 \times 10) + (5 \times 1) = 300 + 20 + 5 =$ **325**

Un último ejemplo, para la cantidad 1987, representado por las potencias de 10 correspondientes tenemos:

10^3	10^2	10^1	10^0
1	9	8	7

Las unidades de miles le corresponden al 1, las centenas al 9, las decenas al 8 y las unidades al 7, por lo tanto:

$(1 \times 10^3) + (9 \times 10^2) + (8 \times 10^1) + (7 \times 10^0)$

$(1 \times 1000) + (9 \times 100) + (8 \times 10) + (7 \times 1) = 1000 + 900 + 80 + 7 = 1987$

Es importante mencionar que podemos utilizar tantas potencias de 10 como se requieran para formar una cantidad.

2 Sistema numérico binario

El sistema binario utiliza únicamente 2 dígitos para representar cualquier cantidad, veamos a continuación cómo podemos representar estas cantidades y vamos a compararlas con el sistema decimal.

DECIMAL	BINARIO
0	0
1	1
2	**10**
3	11
4	**100**
5	101
6	110
7	111
8	**1000**
9	1001
10	1010
11	1011
12	1100

13	1101
14	1110
15	1111
16	**10000**

Para la cantidad cero y uno no tenemos problema, puesto que ambas son iguales tanto para el sistema decimal como para el sistema binario. Sin embargo, representar la cantidad 2 en binario no es tan simple de comprender en un inicio.

Debemos empezar por entender que **el sistema binario también es posicional** y, por tanto, **el lugar en donde se coloque un digito binario modifica su valor relativo**. De esta forma, para representar esta cantidad lo que hemos de hacer es agregar una posición más a la izquierda y utilizar una nueva combinación entre los dígitos con los que cuenta el sistema binario (0 y 1) de tal forma que para representar la cantidad (2) del sistema decimal utilizamos (10) en el sistema binario.

Para evitar caer en confusiones entre ambos sistemas numéricos, vamos a utilizar una notación que consiste en encerrar entre paréntesis la cantidad que queremos representar indicando por medio de un subíndice la base numérica a utilizar, es decir:

Para representar el numero binario 10 lo hacemos como $(10)_2$

Mientras que para representar la cantidad 12 en decimal lo hacemos como $(12)_{10}$

De esta forma podemos afirmar que $(2)_{10}$ es igual a $(10)_2$ esto significa: La cantidad (2) en decimal, es igual a la cantidad (10) en binario.

Para la cantidad $(3)_{10}$ utilizamos la posición adicional que agregamos y la siguiente combinación posible $(11)_2$ pero, para representar la cantidad $(4)_{10}$ nuevamente tenemos el mismo problema, ya que se nos han acabado las combinaciones posibles por lo cual debemos de agregar una posición más a la izquierda y representar entonces la cantidad $(4)_{10}$ como $(100)_2$

De esta manera continuaremos tal como puedes ver en la tabla y es importante que observes los cambios que se producen en el sistema binario en $(2)_{10}$, $(4)_{10}$, $(8)_{10}$, y $(16)_{10}$ ya que en cada uno de estos casos es necesario agregar un digito más.

Ahora bien, **¿Cómo podemos hacer la conversión para cualquier cantidad del sistema binario al sistema decimal y viceversa?** Para ello vamos a desarrollar los siguientes ejemplos que te permitirán entender este proceso de conversión entre sistemas numéricos.

Conversión del sistema binario al sistema decimal.

Para realizar este tipo de conversión vamos a tomar en cuenta la siguiente tabla, recordando que el sistema binario también es posicional pero que, a diferencia del decimal, la base es 2:

2^5	2^4	2^3	2^2	2^1	2^0

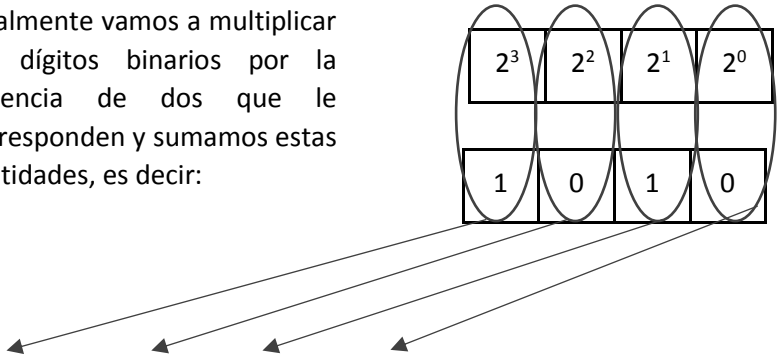
Cada potencia de 2 equivale a una cantidad por la cual vamos a multiplicar el número binario que deseamos convertir, por lo tanto, las potencias anteriores equivalen a las siguientes cantidades:

2^5	2^4	2^3	2^2	2^1	2^0
↕	↕	↕	↕	↕	↕
32	16	8	4	2	1

En caso de que el número binario que deseamos convertir sea más grande que las potencias anteriores, podemos agregar tantas potencias de 2 como se necesiten, es decir podemos continuar con 2^6, 2^7, 2^8, etc.

Empecemos por el número binario $(1010)_2$ el cual deseamos convertir a su equivalente decimal, vamos a representar este número con las potencias de dos que le corresponden:

Finalmente vamos a multiplicar los dígitos binarios por la potencia de dos que le corresponden y sumamos estas cantidades, es decir:



$(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$ lo cual equivale a:

$(1 \times 8) + (0 \times 4) + (1 \times 2) + (0 \times 1) = 8 + 0 + 2 + 0 =$ **10**

Por tanto, el equivalente decimal del número binario $(1010)_2$ es $(10)_{10}$

Llevemos a cabo un segundo ejemplo, en este caso deseamos convertir a su equivalente decimal el número binario 101110. La representación de este número y las potencias de 2 que le corresponden son las siguientes:

2^5	2^4	2^3	2^2	2^1	2^0
1	0	1	1	1	0

Realizando la multiplicación de los dígitos binarios por las potencias de 2 correspondientes y la suma de estas cantidades tenemos:

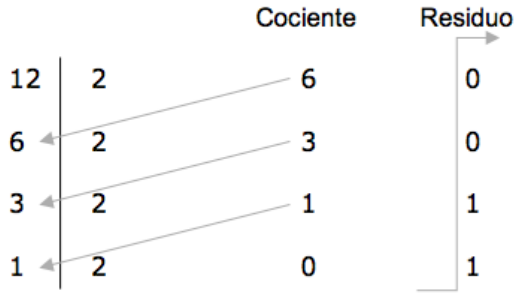
$(1 \times 2^5) + (0 \times 2^4) + (1 \times 2^3) + (1 \times 2^2) + (1 \times 2^1) + (0 \times 2^0)$ lo cual equivale a:

$(1 \times 32) + (0 \times 16) + (1 \times 8) + (1 \times 4) + (1 \times 2) + (0 \times 1) = 32 + 8 + 4 + 2 = 46$

Por lo tanto, el número binario $(101110)_2$ equivale al número decimal $(46)_{10}$

Conversión del sistema decimal al sistema binario.

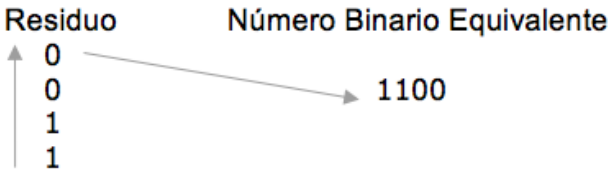
Para encontrar el equivalente binario de una cantidad decimal debemos de dividir a esta sucesivamente por 2. Por ejemplo, supongamos que deseamos convertir el número $(12)_{10}$ en su correspondiente número en binario, para ello realizamos lo siguiente:



Dividimos el número 12 entre 2, para lo cual vamos a escribir el cociente y el residuo correspondiente. En este caso 12 entre 2 nos toca a 6 (Cociente) y nos sobran 0 (Residuo).

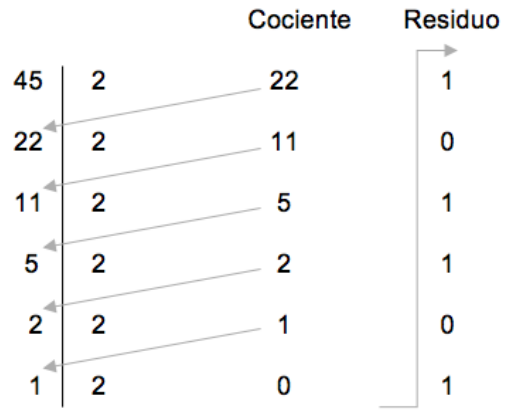
Bajamos el cociente que se encontró en la división anterior y se vuelve a dividir entre 2, en este caso 6 entre 2 nos toca a 3 y también nos sobran 0.

Las divisiones entre 2 van a continuar sucesivamente hasta que se tenga el caso de dividir de 1 entre 2. **Aquí el resultado siempre será cociente igual a 0 y residuo igual a 1 y con ello se termina el procedimiento**. El número binario correspondiente lo obtendremos del **Residuo**, formándose de abajo hacia arriba, lo que quiere decir que el número binario más significativo es el que se encuentra en la parte inferior y el menos significativo en la parte superior del Residuo, es decir:

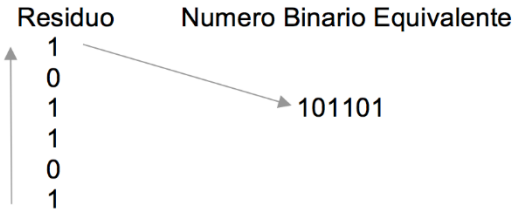


Por tanto, la cantidad binaria que es equivalente al decimal $(12)_{10}$ es $(1100)_2$

Hagamos un segundo ejemplo, supongamos ahora que deseamos convertir a binario el número decimal 45. Siguiendo los pasos que se explicaron anteriormente tenemos:



El resultado de dividir consecutivamente entre 2 nos da el siguiente resultado:



Por lo tanto: $(45)_{10}$ = $(101101)_2$

Ejercicios

Convierte los siguientes números decimales a su equivalente en binario:

a. 15
b. 23
c. 45
d. 103
e. 235
f. 457
g. 934
h. 1024
i. 2356
j. 4567

Convierte los siguientes números binarios a su equivalente en decimal:

1. 101
2. 1001
3. 1101
4. 10111
5. 11001
6. 11101
7. 110110
8. 101011
9. 1001101
10. 1011101

3 Sistema numérico octal

En este sistema numérico, tenemos solo 8 dígitos posibles para poder representar una cantidad, es decir del 0 al 7. Además, como en el caso del sistema decimal y del binario, también es un sistema posicional con lo cual los valores de sus dígitos dependen de la posición en donde se encuentren, veamos a continuación una tabla en donde comparamos los valores decimal, binario y octal.

DECIMAL	BINARIO	OCTAL
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6

7	111	7
8	**1000**	**10**
9	1001	11
10	1010	12
11	1011	13
12	1100	14
13	1101	15
14	1110	16
15	1111	17
16	10000	20

Como puedes observar, cuando se requiere representar la cantidad $(8)_{10}$ debemos agregar una posición más al sistema octal y al haber utilizado todos los dígitos posibles entonces se deben retomar dígitos que ya conocemos, es decir el 1 y el 0, por este motivo la cantidad $(8)_{10}$ es equivalente a $(10)_8$

Al ser el sistema octal posicional, este se representa también como una suma potencias, en este caso de 8, es decir:

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

Lo cual es equivalente a:

4096	512	64	8	1

Conversión del sistema octal al sistema decimal.

De forma similar a lo que realizamos en el sistema binario, basta con multiplicar el digito por la potencia de 8 correspondiente y sumarlos, hagamos un primer ejemplo, supongamos que deseamos conocer el equivalente decimal del numero octal 357.

Representado los dígitos y las bases de 8 correspondientes tenemos:

8^2	8^1	8^0
3	5	7

Por tanto, tenemos:

$(3 \times 8^2) + (5 \times 8^1) + (7 \times 8^0) = (3 \times 64) + (5 \times 8) + (7 \times 1) = 192 + 40 + 7 =$ **239**

El numero octal $(357)_8$ equivale al decimal $(239)_{10}$

En un segundo ejemplo encontremos el numero decimal que equivale del octal 4025, colocando las potencias de 8 y dígitos correspondientes:

8^3	8^2	8^1	8^0
4	0	2	5

Realizando las operaciones correspondientes:

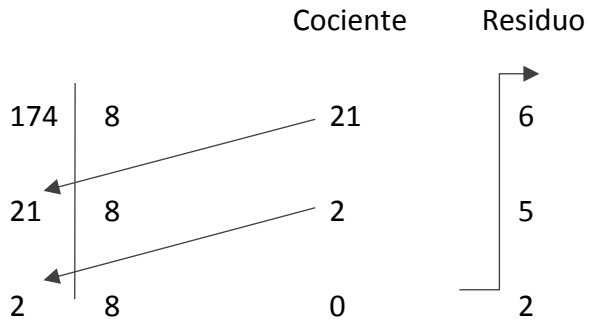
$(4 \times 8^3) + (0 \times 8^2) + (2 \times 8^1) + (5 \times 8^0) = (4 \times 512) + (0 \times 64) + (2 \times 8) + (5 \times 1) = 2048 + 0 + 16 + 5 =$ **2069**

Por tanto, el numero octal $(4025)_8$ equivale al decimal $(2069)_{10}$

Conversión del sistema decimal al sistema octal.

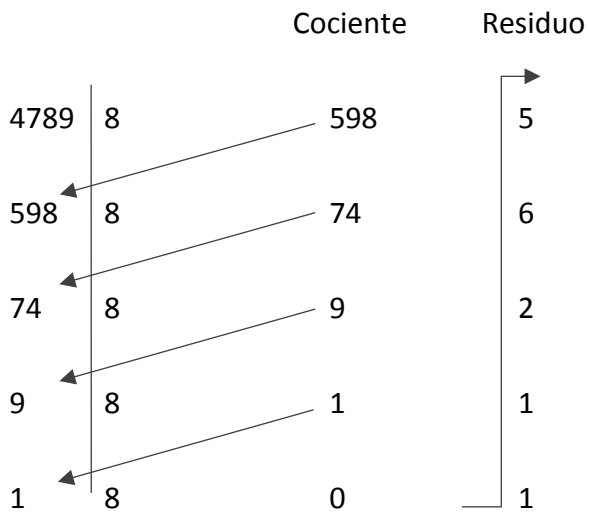
Para convertir un numero decimal a octal vamos a utilizar un método muy similar al que utilizamos en el sistema binario, la única diferencia es que, en lugar de realizar divisiones sucesivas entre dos, en este caso serán entre ocho. Hagamos un primer ejemplo, vamos a convertir a su equivalente octal el numero decimal $(174)_{10}$

Por tanto, dividimos consecutivamente este número entre 8, anotando el cociente y el residuo, realizamos nuevamente este procedimiento utilizando el cociente que se encontró en la división anterior y esto se repetirá indefinidamente hasta llegar al caso en que tengamos la división de cualquier número menor a 8, aquí el cociente será 0 y el residuo será el número que es menor a 8.



Observa que el resultado se genera en el residuo, para ello el numero octal se forma de abajo hacia arriba con lo cual concluimos que **el numero decimal $(174)_{10}$ tiene su equivalente octal $(256)_8$**

Realicemos un segundo ejemplo, vamos a encontrar el equivalente octal del numero decimal $(4789)_{10}$



Por lo tanto, **el numero decimal $(4789)_{10}$ equivale al octal $(11265)_8$**

Ejercicios

Convierte los siguientes números decimales a su equivalente en octal:

1. 12
2. 34
3. 99
4. 154
5. 234
6. 579
7. 1989
8. 4567
9. 7899
10. 10000

Convierte los siguientes números octales a su equivalente en binario:

1. 25
2. 46
3. 94
4. 146
5. 362
6. 765
7. 1345
8. 4256
9. 6543
10. 12546

4 Sistema numérico hexadecimal

Finalmente vamos a mencionar un sistema numérico que es importante que conozcas el cual se denomina hexadecimal, en este caso tenemos hasta **16 dígitos posibles** para poder representar una cantidad, por tal motivo los dígitos faltantes (recordemos que solo conocemos 10 del sistema decimal que hemos utilizado) se van a representar con letras. Veamos su tabla correspondiente:

DECIMAL	BINARIO	HEXADECIMAL
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6

7	111	7
8	**1000**	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10

De igual forma que en los sistemas numéricos que vimos anteriormente, el hexadecimal es posicional por lo que los valores de sus dígitos cambian con respecto a su posición, y cada una de estas posiciones posibles están dadas por las potencias de su base, que en este caso es 16. Es decir:

16^4	16^3	16^2	16^1	16^0

Lo cual equivale a:

65536	4096	256	16	1

Conversión del sistema hexadecimal al sistema decimal.

La conversión se realiza de forma similar, es decir debemos multiplicar los dígitos hexadecimales por la potencia de 16 que le corresponde y finalmente llevar a cabo la suma de todos estos términos, hagamos un primer ejemplo, encontremos el equivalente decimal del numero hexadecimal $(8EA)_{16}$

Representado el numero hexadecimal con las potencias que le corresponden:

16^2	16^1	16^0

8	E	A

Al efectuar la multiplicación es importante que cuando tengas dígitos representando por una letra, (en este caso E y A) coloques la cantidad que le corresponde, esto lo puedes observar en la tabla que elaboramos al principio, es decir: E equivale a 14 y A equivale a 10.

$(8 \times 16^2) + (E \times 16^1) + (A \times 16^0) = (8 \times 16^2) + (14 \times 16^1) + (10 \times 16^0)$

$(8 \times 256) + (14 \times 16) + (10 \times 1) = 2048 + 224 + 10 =$ **2282**

Por tanto, el número hexadecimal $(8EA)_{16}$ equivale al decimal $(2282)_{10}$

En un segundo ejemplo vamos a convertir el número hexadecimal 4BD5 a su equivalente decimal. Para ello lo representamos con las potencias que corresponden:

16^3	16^2	16^1	16^0

4	B	D	5

Realizando las operaciones correspondientes:

$(4 \times 16^3) + (B \times 16^2) + (D \times 16^1) + (5 \times 16^0) = (4 \times 4096) + (11 \times 256) + (13 \times 16) + (5 \times 1)$

$16384 + 2816 + 208 + 5 =$ **19413**

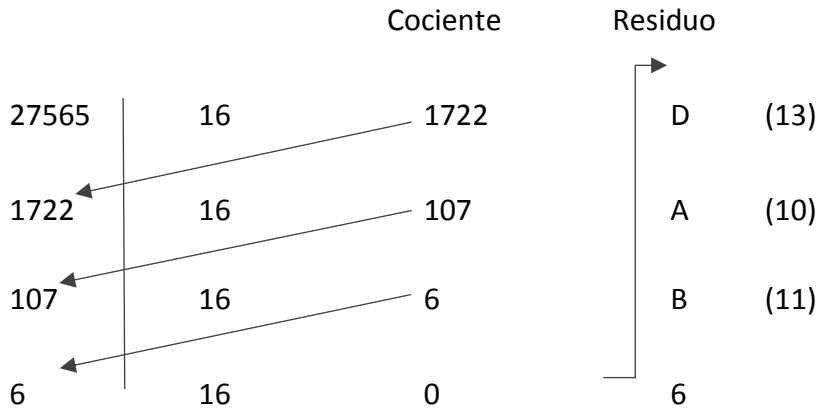
Por lo tanto, el número hexadecimal $(4BD5)_{16}$ equivale al decimal $(19413)_{10}$

Conversión del sistema decimal al sistema hexadecimal.

Para llevar a cabo la conversión entre estos sistemas el procedimiento es similar a lo que hemos visto, como podrás imaginar, la diferencia consiste en que ahora las divisiones se realizaran entre 16, es decir entre la base del sistema hexadecimal. Veamos un primer ejemplo, hagamos la conversión del número decimal 27565 a su equivalente hexadecimal.

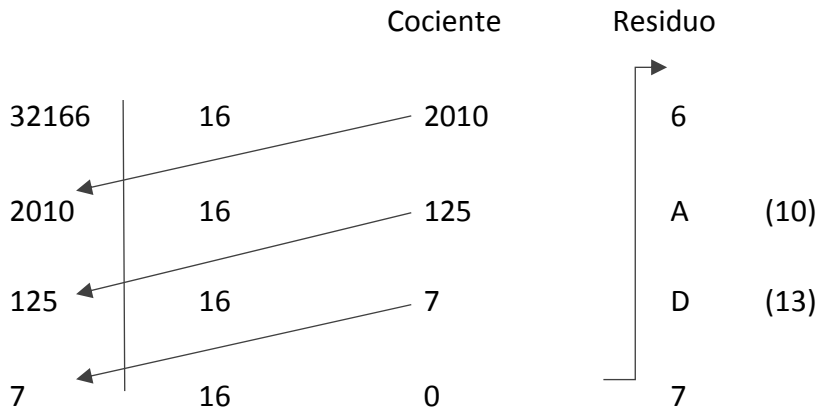
Realicemos las divisiones consecutivas entre 16, el procedimiento terminará cuando el número a dividir sea menor a 16 y en tal caso el cociente será 0 y el residuo será precisamente este número menor a 16.

Es importante que observes que cuando en el residuo se tienen cantidades mayores a 9, se debe colocar la letra que corresponda de acuerdo a la tabla que se mostró al inicio de este apartado.



Por lo tanto, el número decimal $(27565)_{10}$ equivale a número hexadecimal $(6BAD)_{16}$

En un segundo ejemplo vamos a encontrar el equivalente hexadecimal del número $(32166)_{10}$



Por tal motivo, el número hexadecimal equivalente es **$(7DA6)_{16}$**

Ejercicios

a. Convierte los siguientes números decimales a su equivalente en hexadecimal:

1. 34
2. 97
3. 156
4. 256
5. 987
6. 1056
7. 3456
8. 8567
9. 9869
10. 12395

b. Convierte los siguientes números hexadecimales a su equivalente en decimal:

1. 2E
2. 5A
3. 48B
4. BCD
5. CFE
6. 2EA1
7. 5FE2
8. 7FDA
9. FFE1
10. 1B2ED

5 Relación entre el sistema binario, octal y hexadecimal.

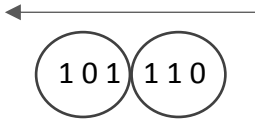
Dado que nuestro objetivo es trabajar en binario, es necesario que conozcas que la conversión entre el binario y los sistemas octal y hexadecimal es muy sencilla, basta que realices una agrupación de dígitos en el sistema binario. Veamos la siguiente tabla:

BINARIO	OCTAL
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

Como puedes observar, a cada digito octal le corresponden 3 dígitos binarios. Por lo tanto, para convertir cualquier número binario a su equivalente octal, basta con que agrupes el número binario en conjuntos de 3 dígitos de derecha a izquierda, es decir empezando con los dígitos menos significativos. En caso de que hagan falta dígitos a la izquierda puedes colocar 0's.

Empecemos con un primer ejemplo, vamos a convertir a su equivalente octal al binario 101110

Agrupamos este número binario en conjuntos de tres dígitos de derecha a izquierda:



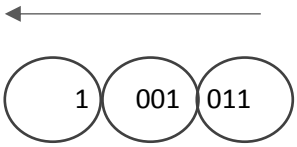
La primera agrupación es 110, a lo cual le corresponde el 6 octal.

Nuestra segunda agrupación 101 que le corresponde el 5 octal, por tanto, el equivalente octal corresponde a 56.

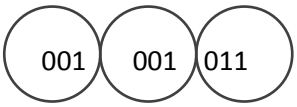
Esto quiere decir que: $(101110)_2$ es igual a $(56)_8$

Realicemos un segundo ejemplo, vamos a encontrar el equivalente octal del numero binario $(1001011)_2$

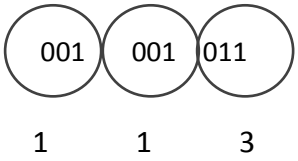
Agrupemos este número binario en conjuntos de 3 dígitos de derecha a izquierda:



La ultima agrupación no contiene tres dígitos, pero podemos agregar 0's a la izquierda tantos como se necesites sin que cambie el valor de nuestra agrupación:



De la tabla que ya conocemos vemos los valores correspondientes:



Por tanto, el número binario $(1001011)_2$ equivale al octal $(113)_8$

Para convertir de Octal a Binario es igual de sencillo, cada digito octal equivale a 3 dígitos binarios, por ejemplo, **vamos a convertir el número octal $(431)_8$ a su equivalente binario.**

Tenemos por tanto de la tabla de equivalencias:

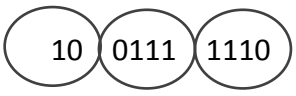
4	3	1
100	011	001

Con lo cual $(431)_8$ equivale a $(100011001)_2$

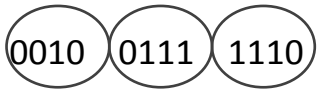
En el caso del sistema Hexadecimal hay también una relación que nos permite convertir valores de manera sencilla entre ambos sistemas, partimos primero de la tabla de equivalencias:

BINARIO	HEXADECIMAL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	A
1011	B
1100	C
1101	D
1110	E
1111	F

Para convertir entre ambos sistemas, tal como puedes observar de la tabla anterior, cada digito hexadecimal corresponde a 4 dígitos binarios. Empecemos por convertir el numero binario 1001111110 a su equivalente hexadecimal, agrupando de 4 en 4 de derecha a izquierda tenemos:



Agreguemos 0's a la izquierda en la tercera agrupación para dejarlo como un grupo de 4 dígitos:



El equivalente de estas agrupaciones es:

2 7 E

Por tanto, el numero binario $(1001111110)_2$ equivale al número hexadecimal $(27E)_{16}$

En el caso inverso, en el cual queremos saber el equivalente binario de un número hexadecimal, a cada digito hexadecimal lo sustituimos por el numero binario equivalente, por ejemplo, para el hexadecimal $(8E3A)_{16}$, hagamos las agrupaciones correspondientes:

8	E	3	A
1000	1110	0011	1010

Por tanto, el número hexadecimal (8E3A)16 equivale al binario $(1000111000111010)_2$

Ejercicios

Convierte los siguientes números binarios a su equivalente en octal:

1. 1001
2. 11001
3. 110010
4. 1110111
5. 10011001

Convierte los siguientes números binarios a su equivalente en hexadecimal:

1. 1011011
2. 11100110
3. 101011100
4. 1010101011
5. 11110011100

TEMA **2**

Operaciones aritméticas con números binarios.

Objetivos:

- Realizar operaciones aritméticas con el sistema de numeración binario.

1 Introducción

Desde el desarrollo de las primeras computadoras, se buscó una solución en la cual se pudieran realizar cálculos de una forma simple y rápida, los elementos con los cuales están construidas las computadoras tienen 2 posibles estados, es decir encendido y apagado. Es por este motivo que el sistema binario es ideal para que una computadora pueda realizar operaciones aritméticas, ya que si deseáramos utilizar el sistema decimal entonces deberíamos de utilizar dispositivos que tuvieran 10 estados posibles lo cual hasta el momento no es factible.

Para comprender de una mejor manera como es que la computadora realiza este tipo de operaciones aritméticas, es que a continuación te mostramos la forma en que se realizan las operaciones aritméticas básicas en el sistema binario.

2 Suma binaria

Para realizar una suma binaria se sigue el proceso normal de una suma, para ello vamos a tomar en cuenta lo siguiente:

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

Como podrás observar, la suma de 1 + 1 es igual a la cantidad dos, lo cual en el sistema binario se representa como 10.

Empecemos por realizar un ejemplo sencillo, en este caso vamos a sumar las siguientes cantidades binarias: 101 + 110

Tal como lo hemos hecho en el sistema decimal, la suma la representamos de la siguiente forma:

$$\begin{array}{r} 101 \\ +\ 110 \\ \hline \end{array}$$

Comenzamos la suma con los dígitos menos significativos, es decir aquellos que se encuentran a la derecha, en este caso tenemos la suma de 1 + 0, cuyo resultado es 1. Por tanto, nuestra primera suma nos queda de la siguiente forma:

$$\begin{array}{r} 101 \\ +\ 110 \\ \hline 1 \end{array}$$

Continuamos con los siguientes dígitos a la izquierda, en este caso tenemos 0 + 1 lo cual también nos da como resultado 1, por tanto, en este segundo paso la suma nos queda así:

$$101$$

$$\begin{array}{r} +\ 110 \\ \hline 11 \end{array}$$

La última suma a realizar es 1 + 1, lo cual nos da como resultado el número 10, con ello finalizamos nuestra suma:

$$\begin{array}{r} 101 \\ +\ 110 \\ \hline \mathbf{1011} \end{array}$$

En un segundo ejemplo hagamos la suma de 1101 + 1001, representándola en el formato que vimos anteriormente:

$$\begin{array}{r} 1101 \\ +\ 1001 \\ \hline \end{array}$$

Comencemos al igual que el ejemplo anterior, de derecha a izquierda, por tanto, los primeros dígitos que vamos a sumar son 1 + 1 cuyo resultado es 10, al ser un resultado de dos cifras no lo podemos colocar de esta forma, por lo que al igual que en sistema decimal colocamos la cifra menos significativa, es decir el 0 y llevamos 1.

Con lo cual:

$$\begin{array}{r} 1101 \\ +\ 1001 \\ \hline 0 \end{array}$$

Los siguientes dígitos que sumar es 0 + 0 cuyo resultado es 0, sin embargo, como llevamos 1 de la suma anterior, entonces el resultado es 0 + 1 lo cual es igual a 1, por tanto, para este segundo paso:

$$\begin{array}{r} 1101 \\ +\ 1001 \\ \hline 10 \end{array}$$

Los siguientes dígitos que sumar son 1 + 0 cuyo resultado es 1.

$$\begin{array}{r} 1101 \\ +\ 1001 \\ \hline 110 \end{array}$$

Para concluir, realizamos la suma de los últimos dígitos, es decir 1 + 1 cuyo resultado es 10, al ser la última cantidad a sumar entonces no tenemos problemas por representarla con los dos dígitos que le corresponden al resultado, por lo que finalmente:

$$\begin{array}{r} 1101 \\ +\ 1001 \\ \hline \mathbf{10110} \end{array}$$

En un tercer ejemplo hagamos la suma de 1111 + 1111, es decir:

$$\begin{array}{r} 1111 \\ +\ 1111 \\ \hline \end{array}$$

Los primeros dígitos a sumar son 1 + 1, cuyo resultado es 10, colocamos el 0 y llevamos 1.

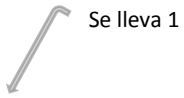
$$\begin{array}{r} 1111 \\ +\ 1111 \\ \hline 0 \end{array}$$

En la siguiente suma tenemos 1 + 1 cuyo resultado es 10, a este resultado debemos sumar el 1 que llevamos de la suma anterior por este motivo nos queda como 10 + 1 cuyo resultado es 11, se coloca el 1 y se vuelve a llevar 1.

$$\begin{array}{r} 1111 \\ +\ 1111 \\ \hline \end{array}$$

10

Los siguientes dígitos a sumar son 1 + 1 cuyo resultado es 10, más 1 que llevamos de la suma anterior es 11, de igual forma se coloca 1 y se vuelve a llevar 1.



1111
+ 1111
110

Los últimos dígitos a sumar son 1 + 1, el resultado es 10, más el 1 que se lleva de la suma anterior es 11, al ser la cantidad final a sumar entonces podemos colocar este resultado con sus dos cifras, por tanto, el resultado de la suma es:

1111
+ 1111
11110

Ejercicios

Realiza las siguientes sumas binarias:

1. 101 + 111
2. 111 + 100
3. 1010 + 1100
4. 1110 + 1101
5. 1111 + 1100
6. 11101 + 10110
7. 11101 + 11011
8. 10111 + 10110
9. 110111 + 101100
10. 111110 + 101111

3 Resta binaria

Resta Binaria con complemento a 1.

Para realizar la resta binaria, debemos primero introducir un concepto denominado **complemento a 1**, el cual consiste en cambiar a un número binario los 0's por 1's y viceversa, por ejemplo: si tenemos el numero binario x es igual a 10011, entonces su complemento se representa como x', es decir:

Valor Original: x = 10011

Complemento a 1: **x'** = 01100

El procedimiento para llevar a cabo la resta consiste en lo siguiente:

Para la resta de a – b, donde a y b son dos números binarios, a es el minuendo (el número a restar) y b el sustraendo (el número que se resta).

Paso 1. Sacar el complemento del sustraendo (b) el cual se expresa como b'.

Paso 2. Sumar al minuendo el complemento obtenido en el paso anterior, es decir a + b'

Paso 3. Si en el resultado de la suma anterior obtenemos un número binario con una cifra más de las que teníamos originalmente, debemos quitar este 1 del resultado de la suma anterior y sumarlo al resultado, ese es el resultado final de la resta.

Paso 4, En el caso de que la suma no genere un dígito más de los que se tenían originalmente, entonces realizamos el complemento a 1 del resultado de esta suma, agregamos un signo negativo y este es el resultado final de la resta.

Ejemplo No. 1

Realizar la resta de 1011 - 1001

Para la resta de a – b, donde a=1011 y b=1001 tenemos:

Paso 1. Hacer el complemento a 1's de b, el cual es: b' = 0110

Paso 2. Sumar a + b':

```
  1 0 1 1
+ 0 1 1 0
---------
1 0 0 0 1
```

El resultado de la suma tiene un dígito más de los 4 dígitos originales, por lo cual hacemos:

Paso 3.

El 1 que estaba de más lo hemos pasado sumando al resultado anterior:

```
  1 0 1 1
+ 0 1 1 0
---------
  0 0 0 1

+       1
---------
  0 0 1 0
```

Por lo tanto, el resultado de a – b = 0 0 1 0

Ejemplo No. 2

Realizar la resta de 11011 - 10011

Para la resta de a – b, donde a = 11011 y b = 10011 tenemos:

Paso 1. Hacer el complemento a 1's de b, el cual es: b' = 01100

Paso 2. Sumar a + b':

```
  1 1 0 1 1
+ 0 1 1 0 0
-----------
1 0 0 0 1 1
```

El resultado de la suma tiene un dígito más de los 5 dígitos originales, por lo cual hacemos:

Paso 3.

El 1 que estaba de más, lo hemos pasado sumando al resultado anterior:

```
  1 1 0 1 1
+ 0 1 1 0 0
-----------
  0 0 1 1 1

+         1
-----------
  1 0 0 0
```

Por lo tanto, el resultado de a – b = 1 0 0 0

Ejemplo No. 3

Realizar la resta de 1001 - 1110

Para la resta de a – b donde a = 1001 y b = 1110

Paso 1. Hacer el complemento a 1's de b, el cual es: b' = 0001

Paso 2. Sumar a + b',

$$\begin{array}{rr} & 1\ 0\ 0\ 1 \\ + & 0\ 0\ 0\ 1 \\ \hline & 1\ 0\ 1\ 0 \end{array}$$

El resultado de la suma **No** tiene un dígito adicional de los 4 dígitos originales, por lo cual hacemos:

Paso 3. Sacar el complemento a 1's del resultado de la suma anterior, es cual es: 0 1 0 1

Paso 4. Agregar un signo negativo al resultado: - 0 1 0 1

Por lo tanto, el resultado de a – b = - 0 1 0 1

Ejercicios

Realiza las siguientes restas binarias:

1. 111 - 100
2. 110 - 111
3. 1110 - 1100
4. 1100 - 1101
5. 1111 - 1100
6. 11101 - 11110
7. 11101 - 10111
8. 10111 - 11110
9. 110111 - 101100
10. 110110 - 111001

4 Multiplicación binaria

En la multiplicación binaria partimos de la siguiente tabla:

0 x 0 = 0
0 x 1 = 0
1 x 0 = 0
1 x 1 = 1

Teniendo en cuenta estos resultados, podemos realizar multiplicaciones de forma similar a como las llevamos a cabo en el sistema decimal, por ejemplo, hagamos la multiplicación de 101 x 10

```
  101
x  10
-----
```

Multiplicando 101 x 0 tenemos:

```
  101
x  10
-----
  000
```

Realizando la siguiente multiplicación, es decir 101 x 1:

```
  101
x  10
-----
  000
 101
```

Finalmente, como en la multiplicación decimal hacemos la suma y con ello se obtiene el resultado final:

```
  101
x  10
-----
  000
+ 101
-----
 1010
```

Por lo tanto 101 x 10 = 1010

Hagamos un segundo ejemplo: 1010 x 110

```
   1010
x   110
-------
   0000
  1010
 1010
-------
 111100
```

Por tanto 1010 x 110 = 111100

Ejercicios

Realiza las siguientes multiplicaciones binarias:

1. 111 x 100
2. 110 x 111
3. 100 x 110
4. 1110 x 1111
5. 1011 x 1000
6. 1100 x 1011
7. 1110 x 1001
8. 1000 x 1010
9. 1010 x 1100
10. 1101 x 1110

5 División binaria

Para el caso de la división binaria utilizaremos el mismo método que aprendimos para dividir números decimales, supongamos como primer ejemplo que deseamos dividir el numero $(1101)_2$ entre $(11)_2$

$$11 \overline{)\ 1101}$$

Empecemos por la primera división, 1 entre 11 nos toca a cero y se resta este valor, posteriormente se baja el siguiente dígito:

$$\begin{array}{r} 0 \\ 11 \overline{)\ 1101} \\ -0 \\ \hline 11 \end{array}$$

Ahora dividimos 11 entre 11, lo cual toca a 1 y nos sobra 0.

$$\begin{array}{r} 01 \\ 11 \overline{)\ 1101} \\ -0 \\ \hline 11 \\ -11 \\ \hline 0 \end{array}$$

Bajamos el siguiente dígito

$$\begin{array}{r} 01 \\ 11 \overline{)\ 1101} \\ -0 \\ \hline 11 \\ -11 \\ \hline 00 \end{array}$$

La división de 00 entre 11 nos toca a cero y restamos cero.

```
   010
11 | 1101
  -0
   11
  -11
    00
   -0
    0
```

Se baja el último dígito:

```
   010
11 | 1101
  -0
   11
  -11
    00
   -0
    01
```

01 entre 11 nos toco a cero y entonces sobran 1

```
   0100
11 | 1101
  -0
   11
  -11
    00
   -0
    01
   -0
    01
```

Por tanto, 1101 entre 11 nos toca a 100 y nos sobra 1

Ejemplo 2, hacer la división de 1110 entre 10

$$10 \overline{)\ 1110}$$

La primera división nos toca a 0 y nos sobra 1, bajamos el siguiente dígito.

$$\begin{array}{r} 0 \\ 10 \overline{)\ 1110} \\ -0 \\ \hline 11 \end{array}$$

Ahora hacemos la división de 11 entre 10, la cual nos toca a 1 y restamos 10 del valor anterior cuyo resultado es 1.

$$\begin{array}{r} 01 \\ 10 \overline{)\ 1110} \\ -0 \\ \hline 11 \\ -10 \\ \hline 1 \end{array}$$

Bajamos el siguiente dígito:

$$\begin{array}{r} 01 \\ 10 \overline{)\ 1110} \\ -0 \\ \hline 11 \\ -10 \\ \hline 11 \end{array}$$

Dividimos 11 entre 10, lo cual nos toca 1, restamos de igual forma 10 del resultado anterior y nos da como resultado 1.

$$\begin{array}{r} 011 \\ 10 \overline{)\ 1110} \\ -0 \\ \hline 11 \\ -10 \\ \hline 11 \\ -10 \\ \hline 1 \end{array}$$

Se baja el último dígito:

```
    011
10 ┌1110
  -0
   11
  -10
    11
   -10
     10
```

10 entre 10 toca a 1 y nos sobra 0:

```
    0111
10 ┌1110
  -0
   11
  -10
    11
   -10
     10
    -10
      0
```

Por tanto, el resultado de 1110 entre 10 es: 111

Ejercicios

Realiza las siguientes divisiones binarias:

1. 111 ÷ 10
2. 1010 ÷ 11
3. 1110 ÷ 10
4. 11001 ÷ 111
5. 10111 ÷ 101

TEMA 3

Interruptores, circuitos eléctricos y circuitos lógicos.

Objetivos:

- Describir los conceptos de interruptor, circuito eléctrico, compuerta lógica y circuito lógico.

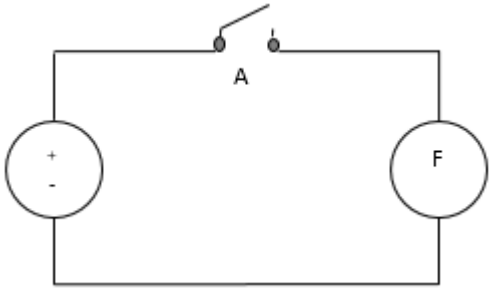
1 Introducción

Para entender el funcionamiento interno de una computadora, debemos de comenzar comprendiendo como podemos implementar circuitos básicos mediante interruptores, ya que en las computadoras tenemos millones de ellos que se encienden o apagan para poder realizar instrucciones. Comenzaremos con la explicación de circuitos eléctricos, en los cuales vamos a identificar la función lógica Y así como la función lógica O.

Posteriormente estos circuitos eléctricos los vamos a representar con compuertas lógicas y finalmente, desarrollaremos circuitos con estas compuertas.

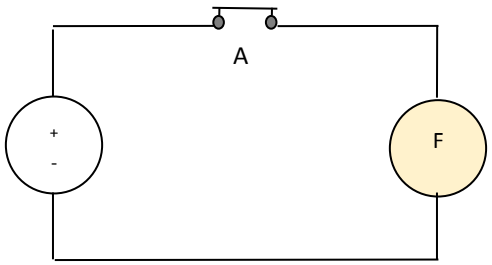
2 Interruptores y circuitos eléctricos.

Vamos a desarrollar un circuito eléctrico básico que va a estar compuesto de una batería, un interruptor y un foco. Vamos a realizar el diagrama correspondiente:



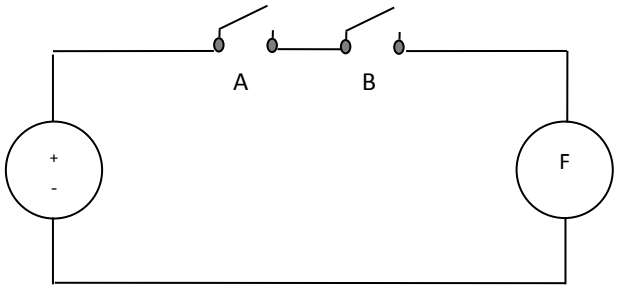
A la Izquierda representamos a la batería la cual está representada por un circulo en donde se muestra el voltaje positivo y negativo, esta batería se conecta al interruptor A, el cual se encuentra abierto o apagado, por lo cual la corriente no puede circular para llegar al foco F que se representa por el circulo de la derecha, por tanto, en este circuito eléctrico el foco permanece apagado.

Si el interruptor A esta cerrado o encendido, entonces tendremos el siguiente circuito en el cual la corriente puede circular por el interruptor y con ello el foco se encenderá.



Función Lógica Y

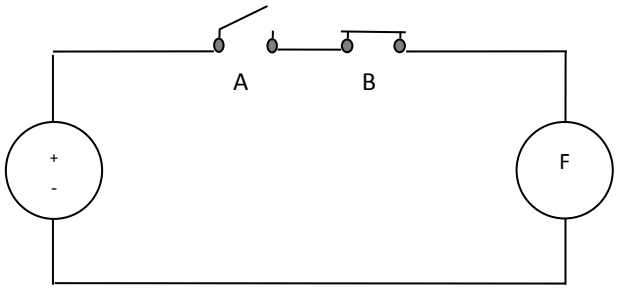
Vamos a modificar el circuito anterior agregando un interruptor más, si ambos interruptores están apagados tendremos el siguiente circuito:



Realicemos una tabla en la cual vamos a mostrar el estado de los interruptores y del foco:

A	B	F
Apagado	Apagado	Apagado

Modifiquemos el estado del interruptor B, de apagado a encendido, con lo cual nuestro circuito es:

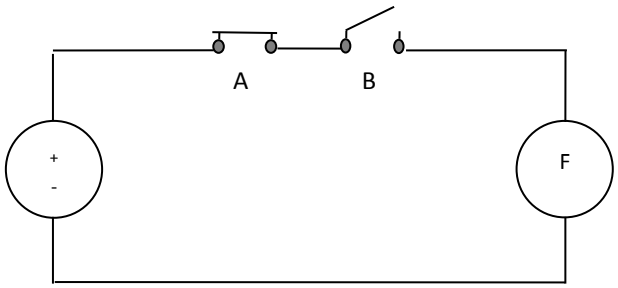


La tabla para esta combinación de interruptores es:

A	B	F
Apagado	Encendido	Apagado

Ya que la corriente puede no puede circular por el circuito ya que, aunque B se encienda, A permanece apagado.

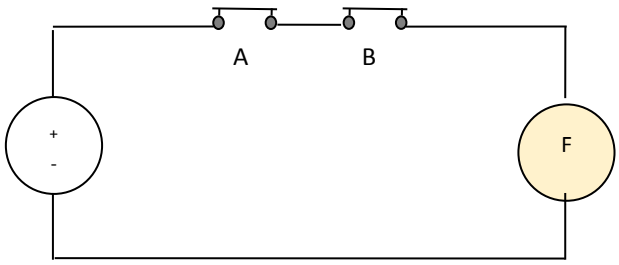
Ahora modifiquemos el estado del interruptor A de apagado a encendido y regresemos al interruptor B a su estado inicial.



Nos queda entonces la siguiente tabla para esta combinación de los interruptores:

A	B	F
Encendido	Apagado	Apagado

Hay una última combinación posible entre estos dos interruptores cuando ambos están encendidos, en este caso nos queda el siguiente circuito:



En esta combinación nos queda la siguiente tabla:

A	B	F
Encendido	Encendido	Encendido

Si agrupamos todas las combinaciones que hemos desarrollado para estos dos interruptores tendremos finalmente la siguiente tabla:

A	B	F
Apagado	Apagado	Apagado
Apagado	Encendido	Apagado
Encendido	Apagado	Apagado
Encendido	Encendido	Encendido

Una función lógica binaria contiene variables que solo pueden tener 2 valores posibles y se relacionan entre sí mediante operadores, la tabla anterior corresponde a la función lógica Y, en la cual la salida será verdadera (encendido) únicamente cuando todas las variables de entrada (A Y B) sean verdaderas (encendidas).

Esto lo podemos comprobar de la tabla anterior ya que el foco, que corresponde a la función, se enciende únicamente cuando ambos interruptores (que corresponden a las variables) están encendidos.

Utilicemos la notación binaria para mencionar los estados de encendido o verdadero los cuales vamos a representar con 1's, mientras que los estados de apagado o falso se representaran con 0's. Por tanto, la tabla anterior se representa en notación binaria como:

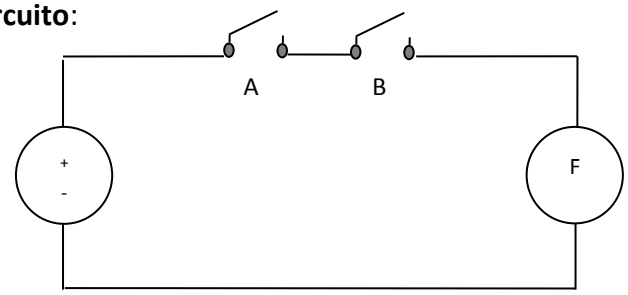
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

La función lógica que representa a la tabla anterior es: **F = A · B**, lo cual significa el producto de A y B, ya que el operador que los relaciona es ·

Esta función será verdadera o 1 solo cuando ambas variables sean verdaderas, en cualquier otro caso el valor de la función será falso o 0.

Recapitulando, podemos visualizar la función lógica Y de tres formas las cuales son equivalentes entre sí:

1) **Como un circuito**:



2) **Como una tabla**:

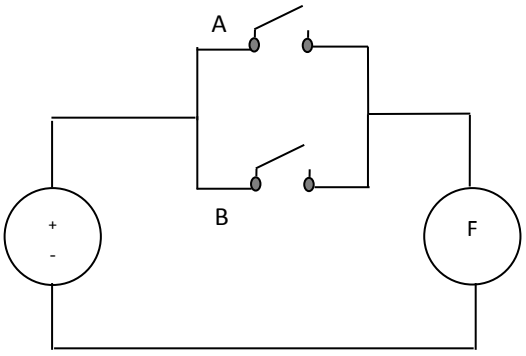
A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

3) **Como una función lógica por si misma**:

$$F = A \cdot B$$

Función Lógica O

En este caso vamos a representar primeramente el circuito haciendo una modificación en la distribución de los interruptores:



En este caso, la corriente puede circular tanto por el interruptor A o por el interruptor B cuando uno de ellos o los dos se encuentren encendidos, únicamente en el caso de que ambos se encuentren apagados, que es el caso que se muestra en el diagrama, el foco permanecerá apagado.

Los cuatro casos posibles utilizando una tabla y notación binaria nos queda como:

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Observa que las combinaciones de las entradas, es decir los estados de los interruptores A y B no cambian y permanecen igual, pero la salida, es decir estado del foco cambia y como mencionamos anteriormente, estos serán 1 cuando uno de los interruptores o ambos sean 1.
La función lógica que le corresponde a este circuito se denomina O y se expresa de la siguiente forma:

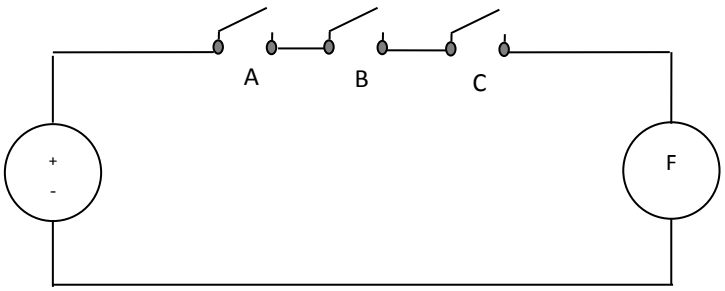
$$F = a + b$$

Donde la suma de a + b representa a la **función lógica O** ya que la relación entre ambos se expresa con el operador +
Es importante que, tal como en la sección anterior observes que el circuito eléctrico se puede visualizar como una función y como una tabla.

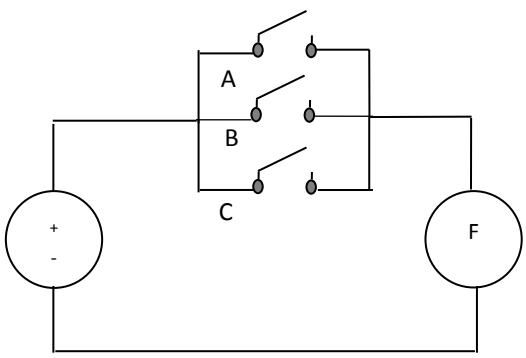
Ejercicios

Realiza la tabla de verdad y la función lógica de los siguientes circuitos eléctricos:

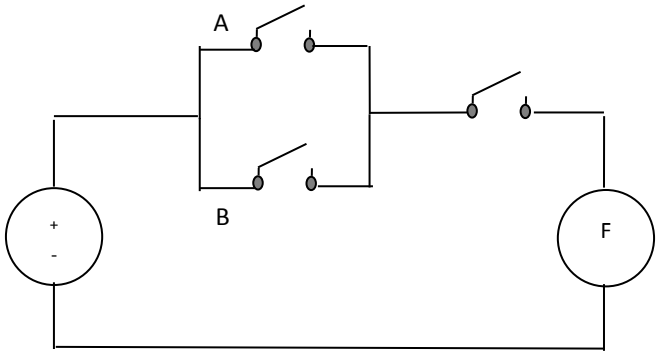
1)



2)



3)



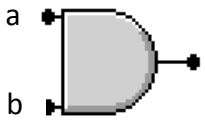
3 Compuertas lógicas.

Los circuitos eléctricos con base a interruptores que vimos anteriormente evolucionaron para que la misma función pueda realizarse con componentes electrónicos, a estos se les conoce como compuertas lógicas.

Una compuerta lógica está construida por varios elementos electrónicos tales como transistores, resistencias, diodos etc., cuya salida está en función de las entradas y en la cuales se lleva a cabo una función lógica en especial. Las principales compuertas lógicas son las siguientes:

Compuerta Lógica AND

La representación gráfica de esta compuerta, su función lógica y la tabla correspondiente son las siguientes:



F

$F = a \cdot b$

a	b	F
0	0	0
0	1	0
1	0	0
1	1	1

Tal como vimos en la sección anterior, la salida de esta compuerta es 1 únicamente cuando ambas entradas son 1, en cualquier otra combinación la salida será 0.

Compuerta Lógica OR

Para esta función tenemos:

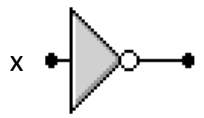
F = a + b

a	b	F
0	0	0
0	1	1
1	0	1
1	1	1

En el caso de esta compuerta la salida será 1 cuando alguna o ambas entradas sean 1, por tanto, será cero únicamente cuando ambas entradas sean 0.

Compuerta Lógica NOT o Inversor

Para esta función tenemos:



x = x'

x	x'
0	1
1	0

a

Esta compuerta solo invierte el valor de la entrada, cuando esta es 0 la salida es 1 y viceversa.

Compuerta Lógica XOR

Para esta función tenemos:

F = a ⊕ b

a	b	F
0	0	0
0	1	1
1	0	1
1	1	0

Esta compuerta es una modificación de la compuerta OR, se conoce como XOR o OR Exclusiva, el valor de la salida es 1 cuando alguna de las entradas es 1 **pero no ambas**, de igual forma es 0 la salida cuando ambas entradas son 0.

Compuerta Lógica NAND

Para esta función tenemos:

$F = (a \cdot b)'$

a	b	F
0	0	1
0	1	1
1	0	1
1	1	0

Esta compuerta es una AND en cuya salida se encuentra una compuerta NOT o Inversor, por tanto, la salida será la negación de la función Y tal como se puede ver en la tabla.

Compuerta Lógica NOR

Para esta función tenemos:

$F = (a + b)'$

a	b	F
0	0	1
0	1	0
1	0	0
1	1	0

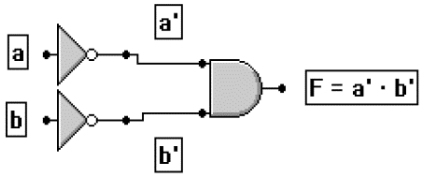
Esta compuerta es similar a la anterior ya que es una compuerta OR en cuya salida se encuentra un inversor, como se puede ver en la tabla tenemos la negación de la compuerta OR.

4 Circuitos lógicos.

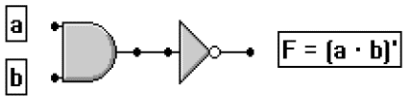
Con la utilización de las compuertas lógicas que vimos anteriormente, es posible crear circuitos lógicos que cumplan una función en específico, por ejemplo, realicemos el circuito lógico que corresponda a la siguiente función:

$$\mathbf{F = a' \cdot b' + (a \cdot b)'}$$

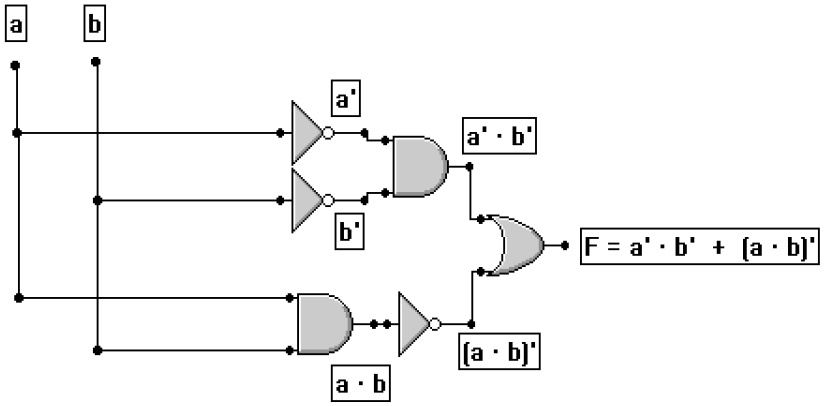
En primer lugar, vamos a desarrollar el término a'·b' para ello observa que (a') es una negación que implica una compuerta inversor, también tenemos (b') que implica otra negación y por tanto otra compuerta inversor, finalmente estos dos elementos están relacionados con el operador · lo que implica una compuerta AND, por tanto, para este primer termino tenemos el siguiente circuito:



El siguiente término: (a·b)' consta de las entradas a y b la cuales están relacionadas mediante el operador · el que corresponde a una compuerta AND, después de esta compuerta existe una negación, la cual se implementa con una compuerta inversor, por tanto, esta parte de nuestro circuito se representa de la siguiente forma:

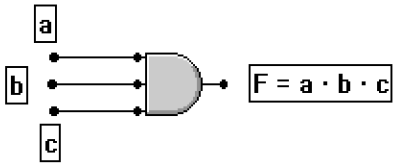


Los términos anteriores se relacionan entre ellos mediante el operador + el cual corresponde a una compuerta OR, por tanto, el circuito completo que corresponde a la F = a'·b' + (a·b)' es:

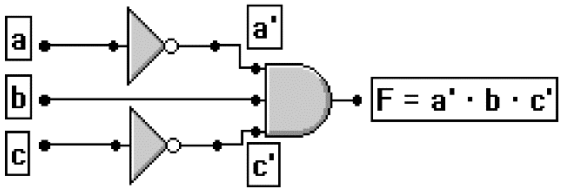


Realicemos ahora un circuito lógico utilizando una variable más: F = a·b·c + a'·b·c'

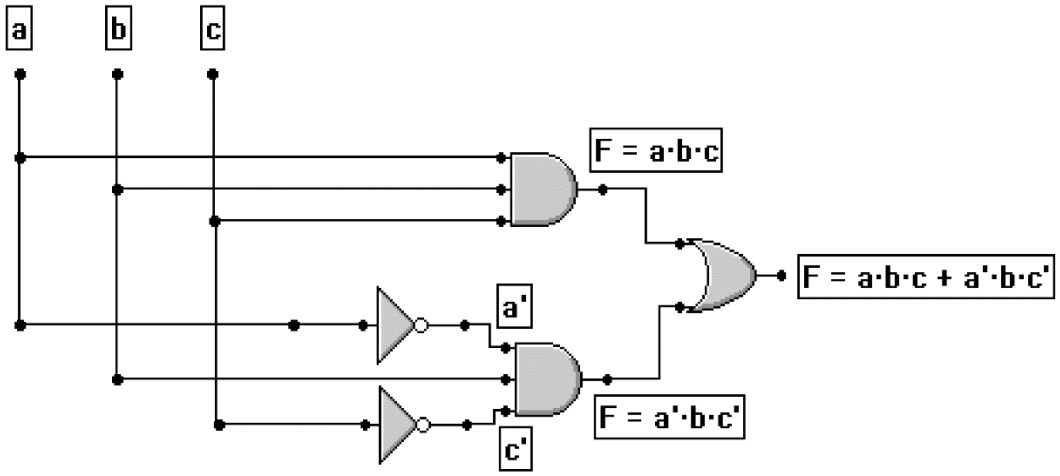
De forma similar a como realizamos el circuito anterior, vamos a desarrollarlo por cada uno de los términos que lo conforman, de esta forma comenzamos con: a · b · c, esto implica que las variables a, b y c están relacionadas mediante una compuerta AND, la cual puede ser de tres entradas tal como se muestra a continuación:



El siguiente término corresponde de igual forma a una compuerta AND, pero en este caso tanto las variables a y c se encuentran negadas, por tal motivo es necesario colocar primero un inversor antes de que ingresen a la compuerta AND.

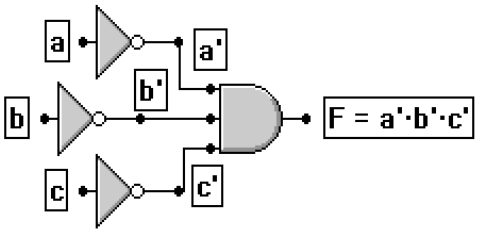


El circuito final es la unión de los términos anteriores mediante una compuerta OR, de esta forma el circuito final es el siguiente:

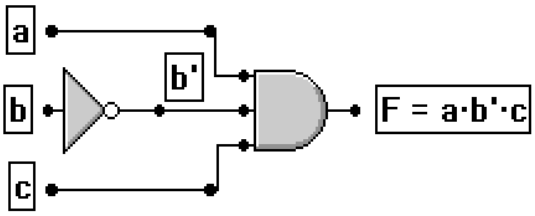


Realizar el circuito lógico para la función: a'·b'c' + a·b'·c + a·b'·c'

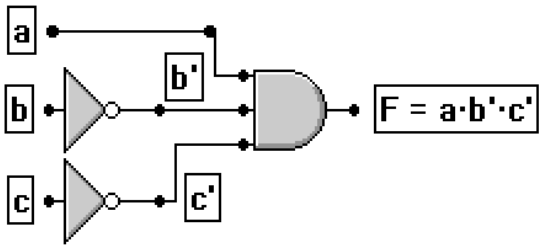
Hagamos el circuito correspondiente al primer término: a'·b'·c' el cual como puedes observar consiste de una compuerta AND de tres entradas y de tres compuertas inversor. Es decir:



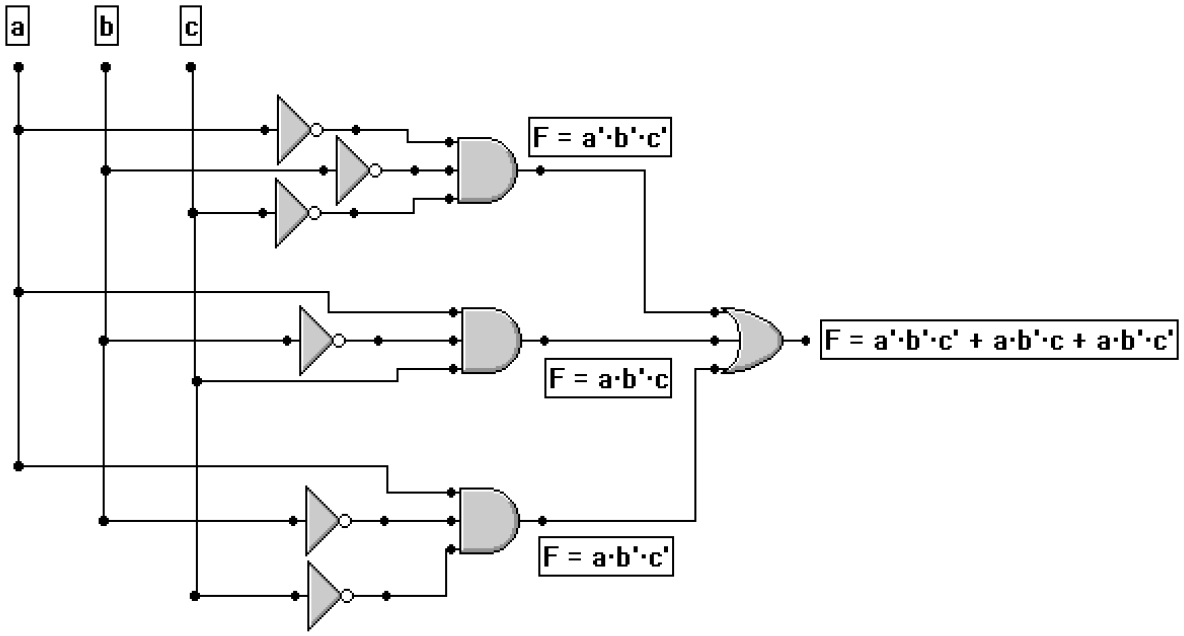
El siguiente término nos queda de la siguiente forma:



El último término corresponde a:



El circuito final es la agrupación de los términos anteriores mediante una compuerta OR de tres entradas, con lo cual finalmente:



Ejercicios

Realiza el circuito lógico de las siguientes funciones:

1) F = a·b’ + a’·b
2) F = a’·b’ + a·b
3) F = a·b·c + a’·b·c
4) F = a’·b’·c + a’·b·c + a·b’·c’
5) F= a’·b’·c’ + a’·b·c’ + a·b·c

TEMA 4

Funciones booleanas y tablas de verdad

Objetivos:

- Construir tablas de verdad de funciones booleanas.
- Construir la función booleana a partir de la tabla de verdad, empleando suma de productos.

1 Introducción

Las funciones booleanas son una forma de representar el comportamiento de los circuitos lógicos que vimos en el capitulo anterior, en ellas es importante comprender que la función es igual a una serie de relaciones entre las entradas mediante los operadores · + ‘, es decir las compuertas básicas AND, OR e Inversor, por otra parte, las tablas de verdad son una herramienta que nos permite visualizar cada una de las combinaciones posibles entre las entradas y la salida que le corresponde a cada una de ellas. La tabla de verdad y la función booleana correspondiente son equivalentes entre si y podemos pasar de una a otra tal como veremos en este capítulo.

2 Funciones booleanas

Una función booleana es aquella que esta representada por variables y constantes cuyo valor solo puede tener dos valores posibles (0, 1) y se relacionan mediante los siguientes operadores:

- · Operador multiplicación, utiliza una función lógica Y entre variables.
- + Operador suma, utiliza una función lógica O entre variables.
- ‘ Operador Negación, lo cual implica el complemento de la variable.

Por ejemplo, para la función booleana:

$$F = a \cdot b$$

La salida (F) es el resultado de la relación entre las variables (a, b) mediante el operador · lo que nos indica que se vinculan mediante una función lógica Y entre ellas, recordemos también que estas variables y el valor de F solo pueden tener dos valores posibles (0,1).

Las funciones booleanas pueden estar compuestas de un número n de variables, en nuestro caso veremos ejemplos con 2 y 3 variables. De igual forma es importante que comprendas que el valor de la función booleana se obtiene cuando sustituimos 0 y 1 en las variables que las representan.

Algunos ejemplos de funciones booleanas son las siguientes:

1. F = x’·y + x·y
2. F = a’·b·c + a’·b’·c’ + a·b·c’
3. F = x·y·z + x’·y·z + (x·y·z)’

3 Construcción de tablas de verdad a partir de una función booleana

Recordemos que una expresión lógica puede tener tres formas de expresarse: como una función, como una tabla y como un circuito. Por tanto, vamos a mostrar la forma en la cual se puede generar la tabla de verdad a partir de una función lógica.

Hagamos un primer ejemplo, utilicemos:

F = x·y + x’·y’

Para construir la tabla, primero **debemos identificar cuantas variables** utilza la función, tal como puedes ver en el ejemplo existen solo **dos** variables (x,y).

La tabla de verdad se conforma de dos partes, la primera de ellas son todas aquellas combinaciones que pueden existir de acuerdo al número de variables, de forma general la cantidad de combinaciones posibles se expresan de la siguiente forma:

Número de combinaciones = 2^n donde n es la cantidad de variables que contiene la función.

Regresando al ejemplo, al tener 2 variables la cantidad de combinaciones posibles son: $2^2 = 4$
Por tanto, la primera parte de nuestra tabla de verdad estará conformada por las cuatro combinaciones posibles, cuando a cada una de las variables se les asigna el valor de 0 y 1, es decir:

x	y
0	0
0	1
1	0
1	1

Ahora bien, esta tabla solo contiene las combinaciones posibles, la segunda parte de nuestra tabla de verdad consiste en agregar los términos de la función a la derecha de la tabla, es decir:

x	y	x · y	x'· y'
0	0		
0	1		
1	0		
1	1		

Debemos empezar por el primer término de la función (x · y) por tanto, en cada una de las combinaciones se va a escribir el resultado correspondiente, empecemos con:

x	y	x · y
0	0	0

Ya que si x = 0 y = 0, el valor de x · y = 0 tal como vimos cuando analizamos la función lógica Y.

Para la segunda combinación:

x	y	x · y
0	1	0

Para la tercera combinación:

x	y	x · y
1	0	0

Para la cuarta combinación:

x	y	x · y
1	1	1

Por tanto, la tabla de verdad para el primer término (x · y) es:

x	y	x · y
0	0	0
0	1	0
1	0	0
1	1	1

En el caso del segundo término tenemos:

x	y	x’ · y’
0	0	1

El resultado es 1, ya que:

si x = 0, entonces x’ = 1

si y = 0, entonces y’ = 1

Por lo cual, si ambos valores valen 1, al relacionarlos con la función lógica Y, el valor será 1.

Para la segunda combinación:

x	y	x' · y’
0	1	0

Para la tercera combinación:

x	y	x' · y'
1	0	0

Para la última combinación:

x	y	x' · y'
1	1	0

Finalmente:

x	y	x' · y'
0	0	1
0	1	0
1	0	0
1	1	0

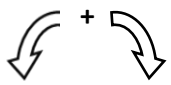
Si juntamos los resultados de los dos términos que conforman a la función tenemos la siguiente tabla:

x	y	x · y	x' · y'
0	0	0	1
0	1	0	0
1	0	0	0
1	1	1	0

Agregaremos una tercera columna con la expresión final de la función:

x	y	x · y	x' · y'	F = x · y + x'· y'
0	0	0	1	
0	1	0	0	
1	0	0	0	
1	1	1	0	

Para finalizar, debemos realizar la suma (+) (función lógica O) entre la columna (x · y) + (x' · y')



x · y	x' · y'	x · y + x'· y'
0	1	1
0	0	0
0	0	0
1	0	1

Por tanto, la **tabla de verdad** que le corresponde a la función **F = x·y + x'·y'** es:

x	y	F
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Es esta tabla de verdad se conforma por todas las variables y sus correspondientes combinaciones, además del valor de la función que resulta de las relaciones que existen entre sus términos.

Hagamos un segundo ejemplo: **F = a· b' + (a · b)' + a'· b**

Al analizar la función vemos que de igual forma solo contamos con dos variables, por tanto, las combinaciones posibles de la tabla de verdad serán 4.

a	b
0	0
0	1
1	0
1	1

Agregamos los términos correspondientes a la función:

a	b	a · b'	(a · b)'	a'· b
0	0			
0	1			
1	0			
1	1			

Para el primero término tenemos:

a	b	a · b'
0	0	0
0	1	0
1	0	1
1	1	0

Para el siguiente término:

a	b	(a · b)’
0	0	1
0	1	1
1	0	1
1	1	0

Y para el último término:

a	b	a’· b
0	0	0
0	1	1
1	0	0
1	1	0

El valor final del tabla de verdad sera realizando el suma (+) entre los terminos anteriores lo cual equivale a realizar la función lógica O

a	b	a · b’	(a · b)’	a’· b
0	0	**0**	**1**	**0**
0	1	**0**	**1**	**1**
1	0	**1**	**1**	**0**
1	1	**0**	**0**	**0**

Recuerda que la función lógica O nos dice que el resultado sera 1 cuando al menos una de las entradas sea 1, Por tanto, la **tabla de verdad** que le corresponde a la función **F = a· b’ + (a · b)’ + a’· b**

a	b	**F**
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

En un tercer ejemplo realicemos la tabla de verdad que le corresponde a la función:

F = a·b·c + a’·b’·c + a’·b’·c’

Para este ejemplo, la cantidad de variables se ha incrementado a tres, por tanto, las combinaciones posibles entre las tres variables es 8.

La tabla de verdad que corresponde a las combinaciones posibles entre las tres variables es la siguiente:

a	b	c
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

De igual forma a los ejemplos pasados, vamos a desarrollar la tabla para cada uno de los términos que componen a la función, por tanto, para el primero de ellos:

a	b	c	a·b·c
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	0
1	1	1	1

La tabla de verdad para el segundo término:

a	b	c	a’·b’·c
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

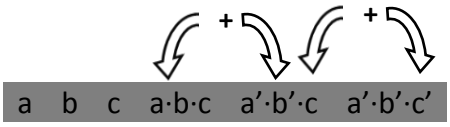
Y para el tercer término:

a	b	c	a'·b'·c'
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

De esta forma, la tabla de verdad con los tres términos que encontramos anteriormente es:

a	b	c	a·b·c	a'·b'·c	a'·b'·c'
0	0	0	0	0	**1**
0	0	1	0	**1**	0
0	1	0	0	0	0
0	1	1	0	0	0
1	0	0	0	0	0
1	0	1	0	0	0
1	1	0	0	0	0
1	1	1	**1**	0	0

Para finalizar necesitamos realizar la suma (+) (Función lógica O) entre los tres términos que componen a la función:



De igual forma que en los ejemplos anteriores, el resultado de la función lógica O será 1, cuando al menos uno de los términos sea 1, por lo tanto, finalmente la tabla de verdad que le corresponde a la función **F = a·b·c + a'·b'·c + a'·b'·c'** es:

a	b	c	**F**
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Hagamos un ultimo ejemplo, es este caso vamos a encontrar la tabla de verdad correspondiente a la función:

$$F = (x \cdot y \cdot z)' + x' \cdot y \cdot z' + x \cdot y' \cdot z$$

De forma similar a los ejemplos anteriores, primero verificamos de cuantas variables consta la función, en este caso son 3 y por tanto necesitamos una tabla de verdad con 8 combinaciones posibles:

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Una vez que se tienen las entradas de la tabla de verdad, procedemos a encontrar las salidas, por tanto, evaluamos cada uno de los términos de los que consta la función:

x	y	z	$(x \cdot y \cdot z)'$
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

El siguiente término:

x	y	z	$x' \cdot y \cdot z'$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

El último término de la función:

x	y	z	x·y'·z
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0

La tabla de verdad con las entradas y las salidas de los tres términos nos queda como:

x	y	z	(x·y·z)'	x'·y·z'	x·y'·z
0	0	0	1	0	0
0	0	1	1	0	0
0	1	0	1	1	0
0	1	1	1	0	0
1	0	0	1	0	0
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	0	0	0

Al relacionar los términos mediante la Función lógica O tenemos:

x y z (x·y·z)' x'·y·z' x·y'·z

x	y	z	F
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Ejercicios

Encuentra la tabla de verdad de las siguientes funciones booleanas:

1. $F_1 = x' \cdot y + x \cdot y'$
2. $F_2 = a \cdot b + a' \cdot b' + (a \cdot b)'$
3. $F_3 = x \cdot y \cdot z + x' \cdot y' \cdot z'$
4. $F_4 = x' \cdot y \cdot z' + x \cdot y' \cdot z + x \cdot y \cdot z$
5. $F_{5=} = x \cdot y' \cdot z + x' \cdot y \cdot z' + x \cdot y \cdot z + x' \cdot y' \cdot z'$

4 Generación de una función booleana a partir de la tabla de verdad.

Una función booleana puede ser representada de dos formas:

1) Como una suma de minitérminos
2) Como el producto de maxitérminos

Para entender el concepto de minitérmino y de maxitérmino partamos de una tabla de verdad de 3 variables:

x	y	z
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Para el caso de los minitérminos, cada una de estas combinaciones puede representarse como el producto de las variables que lo conforman expresándose en su forma original o como su negación. Para entender mejor esto representemos la tabla de verdad de la siguiente manera:

minitérmino	x	y	z
m_0	0	0	0
m_1	0	0	1
m_2	0	1	0
m_3	0	1	1
m_4	1	0	0
m_5	1	0	1
m_6	1	1	0
m_7	1	1	1

Cada una de las 8 combinaciones posibles esta asignada a un minitérmino el cual se representa como m_n

El primer minitérmino, m_0 corresponde a la combinación x=0, y=0, z=0, para representar este minitérmino en función de las variables que lo conforman tomaremos las siguientes reglas:

1) Si la variable es igual a 0, la variable se representa como negada, es decir su complemento.
2) Si la variable es igual a 1, la variable se representa con su valor original.

Por tanto, el minitérmino m_0 se representa como el producto de: x'·y'·z'

Hagamos una lista con todos los miniterminos y sus valores correspondientes:

minitérmino	valor
m_0	x'· y'· z'
m_1	x'· y'· z
m_2	x'· y · z'
m_3	x'· y · z
m_4	x · y'· z'
m_5	x · y'· z
m_6	x · y · z'
m_7	x · y · z

Como puedes observar, **los minitérminos son el producto de las variables tanto en su forma original como con sus valores negados (Complemento)**

La otra forma de representar a una función booleana es **mediante los maxitérminos**, en este caso hablamos de **la suma de variables**. Hagamos una tabla para identificar a los maxitérminos:

maxitérmino	x	y	z
M_0	0	0	0
M_1	0	0	1
M_2	0	1	0
M_3	0	1	1
M_4	1	0	0
M_5	1	0	1
M_6	1	1	0
M_7	1	1	1

Comencemos por el primer maxitérmino M_0 este será la suma de los términos que representa con las siguientes reglas:

1) Si la variable es igual a 0, la variable se representa con su valor original.
2) Si la variable es igual a 1, la variable se representa como negada, es decir su complemento.

Por tanto, el **maxitérmino M_0 es la suma de: x + y + z**

Si hacemos una tabla de los maxitérminos y la suma de variables correspondientes tenemos:

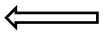
maxitérmino	valor
M_0	x + y + z
M_1	x + y + z’
M_2	x + y’ + z
M_3	x + y’ + z’
M_4	x' + y + z
M_5	x' + y + z’
M_6	x' + y’ + z
M_7	x' + y’ + z’

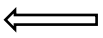
Como puedes observar de la tabla anterior, **las combinaciones se expresan como la suma de sus variables correspondientes, las cuales son negadas cuando valen 1 y se representan en su forma original cuando valen 0.**

Por tanto, una función booleana puede representarse como la suma de minitérminos o como el producto de maxitérminos.

Para ejemplificar lo anterior hagamos la siguiente tabla de verdad:

a	b	c	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**





Para expresarlo con minitérminos o con maxitérminos **debemos identificar en primer lugar en que combinaciones la función F vale 1**, en este ejemplo es en las siguientes combinaciones:

a	b	c
0	0	1

a	b	c
1	1	1

Si lo expresamos como miniterminos, la primera combinación corresponde al minitérmino m_1 y la siguiente al minitérmino m_7 por lo tanto, **la función F será la suma de estos miniterminos**:

$$\mathbf{F = m_1 + m_7}$$

Lo cual expresado con las variables correspondientes es:

$$\mathbf{F = a' \cdot b' \cdot c + a \cdot b \cdot c}$$

En el caso de los maxitérminos, la primera combinación corresponde a M_1 y la segunda a M_7 por tanto, **la función F será el producto de M_1 y M_7**

$$\mathbf{F = M_1 \cdot M_7}$$

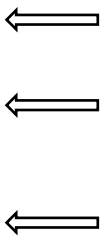
Expresado con las variables correspondientes:

$$\mathbf{F = (a + b + c') \cdot (a' + b' + c')}$$

Aunque se puede trabajar indistintamente con cualquier de las dos formas, se utiliza más comúnmente el uso de la suma de minitérminos.

Hagamos un segundo ejemplo, encontremos la función booleana correspondiente a la siguiente tabla de verdad:

a	b	c	F
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**



El primer paso consiste en identificar las combinaciones en donde la función vale 1, para este ejemplo son las siguientes:

a	b	c
0	0	0
0	1	0
1	0	1

Por tanto, los minitérminos correspondientes son: m_0, m_2 y m_5

La función correspondiente será la suma de los minitérminos anteriores:

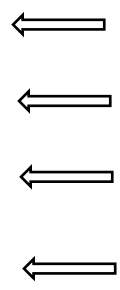
$$F = m_0 + m_2 + m_5$$

Sustituyendo los términos correspondientes a cada minitérmino, tenemos finalmente:

$$\mathbf{F = a'\cdot b'\cdot c' + a'\cdot b\cdot c' + a\cdot b'\cdot c}$$

Veamos un ultimo ejemplo, encontremos la función booleana correspondiente de la siguiente tabla de verdad:

a	b	c	F
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**



Identificamos en primer lugar las combinaciones de la tabla en donde la función vale 1, en este ejemplo son las siguientes cuatro:

a	b	c
0	0	1
0	1	1
1	0	1
1	1	1

Estas combinaciones corresponden a los minitérminos: m_1, m_3, m_5 y m_7

La función correspondiente es la suma de los minitérminos anteriores:

$$F = m_1 + m_3 + m_5 + m_7$$

Sustituyendo los valores de cada minitérmino, finalmente tenemos la función booleana correspondiente:

F = a'·b'·c + a'·b·c + a·b'·c + a·b·c

Ejercicios

Encuentra las funciones booleanas (F1, F2, F3, F4 y F5) de la siguiente tabla de verdad.

a	b	c	F1	F2	F3	F4	F5
0	0	0	**0**	**1**	**1**	**1**	**0**
0	0	1	**0**	**0**	**0**	**1**	**1**
0	1	0	**0**	**0**	**0**	**0**	**1**
0	1	1	**0**	**0**	**1**	**0**	**0**
1	0	0	**0**	**0**	**1**	**1**	**1**
1	0	1	**0**	**0**	**1**	**0**	**1**
1	1	0	**1**	**1**	**0**	**0**	**0**
1	1	1	**1**	**1**	**0**	**1**	**0**

TEMA **5**

Simplificación de funciones.

Objetivos:

- Simplificar funciones booleanas utilizando postulados y teoremas básicos.

1 Introducción

El álgebra de Boole esta definida por una serie de elementos, operadores y postulados la cual fue desarrollada por George Boole en 1854. Posteriormente en 1938, Claude Elwood Shannon publicó el álgebra de Boole de solo dos valores y demostró que los circuitos de conmutación de dos estados se representan por este tipo de álgebra. En este capitulo veremos que a partir de los postulados y teoremas que definen al álgebra de Boole es posible simplificar una función lógica. Posteriormente veremos un método de simplificación de funciones lógicas llamado mapas de Karnaugh.

2 Teoremas básicos del álgebra de Boole

A continuación, vamos a enunciar los 6 postulados y los 6 teoremas del álgebra de Boole de dos estados, los postulados son enunciados tan evidentes que no necesitan demostración. Los teoremas pueden comprobarse a partir de los postulados o mediante el uso de tablas de verdad.

Postulado 1	Conjunto cerrado sobre el operador + cuyo resultado siempre es 1 o 0.	Conjunto cerrado sobre el operador · cuyo resultado siempre es 1 o 0.
Postulado 2	$x + 0 = x$	$x \cdot 1 = x$
Postulado 3 (Conmutativo)	$x + y = y + x$	$x \cdot y = y \cdot x$
Postulado 4	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	$x + (y \cdot z) = (x + y) \cdot (x + z)$ *

(Asociativo)		
Postulado 5	$x + x' = 1$	$x \cdot x' = 0$
Postulado 6	Contiene solo dos elementos diferentes (0 y 1), donde $0 \neq 1$	

Los postulados 3 y 4 son similares al álgebra que conoces, sin embargo, en el álgebra de Boole también existe la propiedad asociativa con respecto a la suma *.

Teorema 1	$x + x = x$	$x \cdot x = x$
Teorema 2	$x + 1 = 1$	$x \cdot 0 = 0$
Teorema 3 (Involución)	$(x')' = x$	
Teorema 4 (Asociativo)	$x + (y + z) = (x + y) + z$	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
Teorema 5 (De Morgan)	$(x + y)' = x' \cdot y'$	$(x \cdot y)' = x' + y'$
Teorema 6 (Absorción)	$x + (x \cdot y) = x$	$x \cdot (x + y) = x$

Comprobación del Teorema 1.

$x + x = x$

x	x + x
0	0
1	1

x	x · x
0	0
1	1

Comprobación del Teorema 2.

$x + 1 = 1$

x	x + 1
0	1
1	1

$x \cdot 0 = 0$

x	x · 0
0	0
1	0

Comprobación del Teorema 3

$(x')' = x$

x	x'	(x')'
0	1	0
1	0	1

Comprobación del Teorema 4

$x + (y + z) = (x + y) + z$

x	y	z	(x + y)'	(x + y) + z
0	0	0	**0**	**0**
0	0	1	**1**	**1**
0	1	0	**1**	**1**
0	1	1	**1**	**1**
1	0	0	**1**	**1**
1	0	1	**1**	**1**
1	1	0	**1**	**1**
1	1	1	**1**	**1**

$x \cdot (y \cdot z) = (x \cdot y) \cdot z$

x	y	z	x · (y · z)	(x · y) · z
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	**1**	**1**

Una función booleana es equivalente a otra si su representación en una tabla de verdad es igual para cada una de las combinaciones de ambas funciones, tal como puedes observar en la primer tabla, la función ***x + (y + z)*** es equivalente a la función ***(x + y) + z*** ya que los valores de estas dos funciones son los mismos para todas las combinaciones posibles entre las 3 variables. Es el mismo caso para la siguiente tabla de verdad en donde se representa la función ***x · (y · z)*** y la función (***x · y) · z***

Comprobación del Teorema 5

$(x + y)' = x' \cdot y'$

x	y	x + y	(x + y)'	x'	y'	x' · y'
0	0	0	**1**	1	1	**1**
0	1	1	**0**	1	0	**0**
1	0	1	**0**	0	1	**0**
1	1	1	**0**	0	0	**0**

$(x \cdot y)' = x' + y'$

x	y	x · y	(x · y)'	x'	y'	x' + y'
0	0	0	**1**	1	1	**1**
0	1	0	**1**	1	0	**1**
1	0	0	**1**	0	1	**1**
1	1	1	**0**	0	0	**0**

Comprobación del Teorema 6

$x + (x \cdot y) = x$

x	y	x · y	x + (x · y)
0	0	0	**0**
0	1	0	**0**
1	0	0	**1**
1	1	1	**1**

$x \cdot (x + y) = x$

x	y	x + y	x · (x + y)
0	0	0	**0**
0	1	1	**0**
1	0	1	**1**
1	1	1	**1**

3 Simplificación de funciones mediante el álgebra de Boole

Veamos las siguientes dos funciones:

$F_1 = a' \cdot b' \cdot c' + a' \cdot b' \cdot c + a' \cdot b \cdot c + a \cdot b \cdot c$

$F_2 = a' \cdot b' + b \cdot c$

Hagamos la tabla de verdad para estas dos funciones:

a	b	c	a'·b'·c'	a'·b'·c	a'·b·c	a·b·c	F_1
0	0	0	1	0	0	0	**1**
0	0	1	0	1	0	0	**1**
0	1	0	0	0	0	0	**0**
0	1	1	0	0	1	0	**1**
1	0	0	0	0	0	0	**0**
1	0	1	0	0	0	0	**0**
1	1	0	0	0	0	0	**0**
1	1	1	0	0	0	1	**1**

a	b	c	a'·b'	b·c	F_2
0	0	0	1	0	**1**
0	0	1	1	0	**1**
0	1	0	0	0	**0**
0	1	1	0	1	**1**
1	0	0	0	0	**0**
1	0	1	0	0	**0**
1	1	0	0	0	**0**
1	1	1	0	1	**1**

Si comparamos estas dos tablas:

a	b	c	F_1	F_2
0	0	0	**1**	**1**
0	0	1	**1**	**1**
0	1	0	**0**	**0**
0	1	1	**1**	**1**
1	0	0	**0**	**0**
1	0	1	**0**	**0**
1	1	0	**0**	**0**
1	1	1	**1**	**1**

Ambas funciones tienen la misma tabla de verdad, ya que en cada combinación para las variables de entrada (a,b,c) se tienen los mismos valores en la salida, tanto para F_1 como para F_2. Por tanto, tal como lo mencionamos anteriormente, ambas funciones son equivalentes ya que la tabla de verdad es la misma.

Esto es muy importante, imagina que deseamos construir el circuito de la tabla de verdad anterior, para ello podemos usar cualquiera de las dos funciones, sin embargo, ten en cuenta la cantidad de compuertas que se requieren:

Para F_1: 6 Inversores, 4 compuertas AND de 3 entradas, 1 compuerta OR de 4 entradas.
Para F_2: 2 Inversores, 2 compuertas AND de 2 entradas, 1 compuerta AND de 2 entradas.

Si tuvieras que armar el circuito comprando los elementos necesarios, **gastarías más dinero** si eliges hacer la función F1 en lugar de la F2 además de que **seria más complejo** poder realizarlo.

En las computadoras, los microprocesadores que hacen posible realizar las operaciones aritméticas y lógicas están desarrollados con base a compuertas lógicas. Cuando se hace el diseño de estos microprocesadores, siempre se eligen **las funciones más simples** ya que estas ahorran espacio y dinero.

La pregunta es **¿Cómo se puede obtener una función equivalente que sea más simple?** Para ello es necesario utilizar siempre que nos sea posible, los postulados y teoremas del álgebra de Boole.

Veamos un primer ejemplo, para la función F_1 que vimos anteriormente:

$$F_1 = a'\cdot b'\cdot c' + a'\cdot b'\cdot c + a'\cdot b\cdot c + a\cdot b\cdot c$$

Aplicamos el postulado 4 (asociativo) a los términos:	$a'\cdot b'\cdot c' + a'\cdot b'\cdot c$
Lo cual nos queda como:	$a'\cdot b'(c'+ c)$
Aplicando el postulado 5 ($x + x' = 1$):	$a'\cdot b'\cdot(1)) = a'\cdot b'$
De igual forma, aplicando el postulado 4 a los términos:	$a'\cdot b\cdot c + a\cdot b\cdot c$
Nos queda:	$b\cdot c\cdot(a'+ a)$
Aplicando nuevamente el postulado 5:	$b\cdot c\cdot(1)) = b\cdot c$

Por tanto, la función simplificada F1s será la suma de los dos términos simplificados anteriormente, es decir:

$$F_{1s} = a'\cdot b' + b\cdot c$$

Hagamos un segundo ejemplo, encontrar la función simplificada de:

$$F_2 = x'\cdot y\cdot z' + x'\cdot y\cdot z + x\cdot y'\cdot z' + x\cdot y'\cdot z$$

Aplicamos el postulado 4 a los términos:	$x'\cdot y\cdot z' + x'\cdot y\cdot z$
Lo cual nos queda como:	$x'\cdot y\ (z' + z)$
Aplicando el postulado 5:	$x'\cdot y$

De igual forma, aplicando el postulado 4 a los términos:	x·y'·z' + x·y'·z
Nos queda:	x·y'·(z'+ z)
Aplicando nuevamente el postulado 5:	x·y'

Sumando los términos que encontramos anteriormente tenemos finalmente que:
F_{2s} = x'·y + x·y'

Hagamos un último ejemplo, para la siguiente función booleana:

$$\mathbf{F_3 = x'\cdot y'\cdot z' + x'\cdot y'\cdot z + x'\cdot y\cdot z' + x'\cdot y\cdot z + x\cdot y'\cdot z' + x\cdot y\cdot z}$$

Factorizar el término z' y el término z:	**z'·(x'·y' + x'·y + x·y') + z·(x'·y' + x'·y + x·y)**
Factorizar el término x':	**z'·(x'·(y' + y) + x·y') + z·(x'·(y'+ y) + x·y)**
Aplicando *y + y' = 1* tenemos:	**z'·(x'·1 + x·y') + z·(x'·1 + x·y)**
Aplicando *x' ·1 = x'* tenemos:	**z'·(x' + x·y') + z·(x' + x·y)**
Aplicando *x' + x·y' = x' + y' x' + x·y = x' + y*	**z'·(x' + y') + z·(x' + y)**
Eliminado la factorización:	**x'·z' + y'·z' + x'·z + y·z**
Factorizando x' :	**x'·(z' + z) + y'·z' + y·z**
Aplicando *z + z' = 1*	**x'·1 + y'·z' + y·z**
Aplicando *x'·1 = x'*	**x' + y'·z' + y·z**
Finalmente:	
F_{3s} = x' + y'·z' + y·z	

4 Simplificación de funciones mediante mapas de Karnaugh.

Existe un método alternativo para llevar a cabo la simplificación de una función booleana mediante el uso de un mapa y la agrupación de términos, este se conoce como mapas de Karnaugh. Se puede aplicar para una función booleana con un numero de variables mayor o igual a dos, sin embargo, dado que las funciones de 2 variables son fáciles de simplificar mediante el algebra de Boole, nos enfocaremos en utilizarlo para funciones booleanas con tres variables.

Para explicar este método primero recordemos la tabla de verdad y los minitérminos asociados.

minitérmino	x	y	z
m_0	0	0	0
m_1	0	0	1
m_2	0	1	0
m_3	0	1	1
m_4	1	0	0
m_5	1	0	1
m_6	1	1	0
m_7	1	1	1

El siguiente paso es elaborar un "mapa" el cual es una distribución especial de estos minitérminos, este se representa de la siguiente forma:

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

Si representamos a este mapa con los valores correspondiente a cada minitérmino el mapa nos queda de la siguiente manera:

x'· y'· z'	x'· y'· z	x'· y · z	x'· y · z'
x · y'· z'	x · y'· z	x · y · z	x · y · z'

Hagamos un ejemplo para mostrar como nos ayuda el mapa anterior para simplificar una función, comencemos con la función F_1 que vimos en la sección anterior.

$$\mathbf{F_1 = a'\cdot b'\cdot c' + a'\cdot b'\cdot c + a'\cdot b\cdot c + a\cdot b\cdot c}$$

Hagamos la tabla de verdad correspondiente a la función F_1:

a	b	c	$\mathbf{F_1}$
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

Agreguemos a la tabla anterior los minitérminos correspondientes:

minitérmino	a	b	c	F_1
m_0	0	0	0	**1**
m_1	0	0	1	**1**
m_2	0	1	0	**0**
m_3	0	1	1	**1**
m_4	1	0	0	**0**
m_5	1	0	1	**0**
m_6	1	1	0	**0**
m_7	1	1	1	**1**

Vamos a identificar cada uno de los minitérminos donde la tabla de verdad de la función F_1 vale 1, es decir, los minitérminos: $m_0\, m_1\, m_3\, m_7$

Si expresamos estos valores de 1 en el mapa que vimos anteriormente, nos quedaría de la siguiente manera:

1	**1**	**1**	m_2
m_4	m_5	**1**	m_6

Los minitérminos que no valen 1 tendrán el valor de 0, por lo tanto, nos quedará de esta forma:

1	**1**	**1**	**0**
0	**0**	**1**	**0**

Este mapa es la pieza principal que utilizaremos para lograr simplificar nuestra función, para ello debemos de aplicar las siguientes reglas:

1) En un mapa de Karnaugh siempre debemos de **agrupar únicamente los términos 1**
2) Las agrupaciones de 1's se harán **únicamente en numero pares**, es decir grupos de 2, 4 u 8 elementos.
3) Las agrupaciones se harán **únicamente horizontales o verticales**, no son validas las agrupaciones en diagonal.
4) Se deben agrupar **la mayor cantidad posible 1's**. Una función es más simple si hay una agrupación de cuatro 1's que otra con dos agrupaciones de un par de 1's, es decir, la siguiente agrupación es correcta:

1	1	1	1
0	0	0	0

Con respecto a esta otra:

1	1	1	1
0	0	0	0

5) Es posible agrupar **los extremos** del mapa de la siguiente forma:

1	0	0	1
0	0	0	0

1	0	0	1
1	0	0	1

6) Una vez realizada las agrupaciones de 1's, estas se representarán con los minitérminos correspondientes y las variables que los conforman, **se eliminaran todas aquellas variables que cambien de valor en las agrupaciones correspondientes y nos quedaremos con aquellas que no cambian de valor**. Esto lo veremos mas a fondo cuando sigamos explicando el ejemplo.

Regresando a ejemplo que estábamos desarrollando, podemos agrupar los 1's correspondientes de la siguiente forma:

1	1	1	0
0	0	1	0

Al hacer esta agrupación observa que estamos agrupando:

$$m_0 \text{ y } m_1$$

Y también:

$$m_3 \text{ y } m_7$$

Si expresamos los minitérminos anteriores **mediante las variables correspondientes**:

Para la primera agrupación:

m_0	**a'· b'· c'**
m_1	**a'· b'· c**

Y para la segunda:

m_3	**a'· b · c**
m_7	**a· b · c**

En la primera agrupación observa que los términos a' b' no cambian entre m_0 y m_1, mientras que el **término c' en m_0 cambia a c en m_1** por tanto, lo eliminamos de la agrupación y conservamos solamente **a'·b'**

En la segunda agrupación ahora observa los términos (b, c); estos no cambian entre m_3 y m_7, mientras que el **término a' en m_3 cambia al valor de a en m_7** por tanto, lo debemos eliminar de la agrupación y nos quedamos solamente con **b·c**

La función simplificada será la suma de todas las agrupaciones obtenidas, por tanto, finalmente tendremos:

$$F_{1s} = a'\cdot b' + b\cdot c$$

Ejemplo 2. Encontrar la simplificación de la función:

$$F_2 = x'\cdot y'\cdot z' + x'\cdot y'\cdot z + x'\cdot y\cdot z$$

La tabla de verdad que le corresponde a esta función es:

x	y	z	F_2
0	0	0	**1**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**0**

Utilizando el mapa de Karnaugh para los minitérminos correspondiente nos queda:

1	**1**	**1**	**0**
0	**0**	**0**	**0**

La primera agrupación nos queda como:

1	**1**	**1**	**0**
0	**0**	**0**	**0**

Observa que existe un 1 que le corresponde al término m_3 para poder simplificarlos es posible agruparlo con el 1 que corresponde a m_1 sin importar que esta ya se haya utilizado para agruparse con m_0 por lo tanto la segunda agrupación nos queda de la siguiente forma:

1	**1**	**1**	**0**
0	**0**	**0**	**0**

Nos quedan entonces las siguientes agrupaciones:

m_0	**x'· y'· z'**
m_1	**x'· y'· z**

m_1	**x'· y'· z**
m_3	**x'· y · z**

Para la primera agrupación nos quedamos con el término: **x'·y'**

Y para la siguiente agrupación nos queda el término: **x'·z**

Por lo tanto, la función simplificada será: F_{2s} = **x'·y' + x'·z**

Ejemplo 3. Hagamos la simplificación de la función:

$$F_3 = a'\cdot b'\cdot c' + a'\cdot b\cdot c' + a\cdot b'\cdot c' + a\cdot b'\cdot c$$

La table de verdad que le corresponde a esta función es:

a	b	c	F_3
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**1**
1	0	1	**1**
1	1	0	**0**
1	1	1	**0**

El mapa de Karnaugh correspondiente es:

1	**0**	**0**	**1**
1	**1**	**0**	**0**

Las agrupaciones serian las siguientes:

1	**0**	**0**	**1**
1	**1**	**0**	**0**

Es decir:

m_0	**a'· b'· c'**
m_2	**a'· b· c'**

m_4	**a· b'· c'**
m_5	**a· b' · c**

Eliminando los términos que cambian tenemos para la primera agrupación: **a'·c'**

Y para la segunda: **a·b'**

Por lo tanto, la función simplificada será: F_{3s} **= a'·c' + a·b'**

Ejemplo 4. Encontrar la función simplificada de la función: $F_4 = x' \cdot y' \cdot z + x' \cdot y \cdot z + x \cdot y' \cdot z + x \cdot y \cdot z$

Realizando la tabla de verdad:

x	y	z	F_4
0	0	0	**0**
0	0	1	**1**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

Utilizando el mapa de Karnaugh:

0	**1**	**1**	**0**
0	**1**	**1**	**0**

Agrupando procurando abarcar la mayor cantidad de 1's, es decir una agrupación de cuatro 1's. nos queda de la siguiente forma:

0	**1**	**1**	**0**
0	**1**	**1**	**0**

Los términos de esta agrupación son:

m_1	$x' \cdot y' \cdot z$
m_3	$x' \cdot y \cdot z$
m_5	$x \cdot y' \cdot z$
m_7	$x \cdot y \cdot z$

Si observas bien, te darás cuenta de que el único término que permanece sin cambios es **z**. Aquí es importante mencionar que **siempre que se tienen agrupaciones de cuatro 1's la simplificación se reduce a una sola variable**, por tanto, la función nos queda como:

$$F_{4s} = z$$

Ejemplo 5. Encontrar la función simplificada de:

$$F_5 = a'\cdot b'\cdot c' + a'\cdot b\cdot c + a'\cdot b\cdot c' + a\cdot b'\cdot c + a\cdot b\cdot c'$$

Si encontramos su tabla de verdad:

a	b	c	F_5
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

El mapa de karnaugh correspondiente:

1	**0**	**1**	**1**
0	**1**	**0**	**1**

Este ejemplo es importante por las siguientes razones:

1. Observa que el término m_2 se puede usar en tres ocasiones, para agrupar con m_3, con m_0 y con m_6
2. El término m_5 dado que no se puede agrupar con otro término, es necesario incluirlo en la funcion simplificada con su valor original es decir: **a · b'· c**

Por lo tanto, tendremos las siguientes agrupaciones:

m_2	**a'·b·c'**
m_3	**a'·b·c**

m_2	**a'·b·c'**
m_0	**a'·b'·c'**

m_2	**a'·b·c'**
m_6	**a·b·c'**
m_5	**a·b'·c**

Las agrupaciones nos dan los siguientes términos:

a'·b

a'·c'

b·c'

Entonces sumando estos términos mas m_5 nos que la función simplificada como:

$$F_{5s} = a'\cdot b + a'\cdot c' + b\cdot c' + a\cdot b'\cdot c$$

Un último ejemplo, encontremos la función simplificada de la función:

$$F_6 = x'\cdot y'\cdot z' + x'\cdot y\cdot z + x\cdot y'\cdot z + x\cdot y\cdot z'$$

Encontramos la tabla correspondiente:

a	b	c	F_5
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**1**
1	0	0	**0**
1	0	1	**1**
1	1	0	**1**
1	1	1	**0**

El mapa de karnaugh correspondiente:

1	**0**	**1**	**0**
0	**1**	**0**	**1**

En este mapa no permite hacer ninguna agrupación ya que estas solo pueden ser horizontales o verticales, por lo tanto, esta función no puede ser simplificada y la función se queda igual.

Ejercicios

Encuentra la simplificación de las siguientes funciones booleanas:

1. x’·y·z’ + x’·y·z + x·y’·z’ + x·y’·z
2. x’·y’·z’ + x·y’·z+ x·y·z + x’·y·z
3. x·y’·z’ + x·y’·z+ x’·y’·z + x’·y·z’ + x’·y·z
4. x’·y’·z’ + x’·y’·z+ x’·y·z’ + x·y’·z’ + x·y·z
5. x’·y’·z’ + x·y’·z+ x·y·z + x’·y·z’ + x’·y·z

TEMA **6**

Diseño de circuitos lógicos

Objetivos:

- Construir un semisumador.
- Construir un sumador completo.
- Diseñar circuitos lógicos a partir de un problema cotidiano usando la metodología aprendida.

1 Introducción

El diseño de los circuitos lógicos es el objetivo final de esta unidad, para ello vamos a utilizar los conceptos que vimos anteriormente. El procedimiento que vamos a realizar para diseñar nuestros circuitos lógicos es el siguiente:

- ✓ Entender el problema.
- ✓ Visualizar el problema como caja negra.
- ✓ Encontrar la tabla de verdad.
- ✓ Obtener la función booleana.
- ✓ Simplificar la función booleana.
- ✓ Construir el circuito correspondiente.

2 Problema de la escalera

En una casa de dos pisos la escalera se ilumina mediante un foco F. En la planta baja hay un interruptor A y en el segundo piso, se encuentra el interruptor B. Se necesita construir un dispositivo donde inicialmente el foco se encuentra apagado y ambos interruptores también se encuentran apagados con las siguientes condiciones:

Al entrar se oprime el interruptor A y el foco enciende.
Al subir la escalera se oprime el interruptor B y el foco se apaga.
Al llegar otra persona oprime nuevamente el interruptor A y el foco enciende.
Al subir nuevamente la escalera se oprime el interruptor B y el foco se apaga.

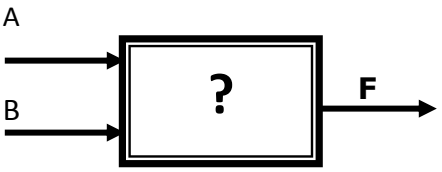
Diseñar un circuito lógico que cumpla con las condiciones anteriores.

Solución:

1. Planteamiento del problema.

Las variables del problema están dadas por las mismas condiciones, podemos deducir que nuestras variables de entrada son A y B, y dependiendo del estado que en que se encuentren van a producir una salida en F que representa al foco.

2. Visualizando el problema como caja negra.



F = 0 ⇒ El foco se apaga.
F = 1 ⇒ El foco se enciende.

3. Obteniendo la tabla de verdad.

La tabla de verdad se forma con base a los valores que se van obteniendo de las variables, de acuerdo a las condiciones enunciadas. Para poder expresar los valores debemos tomar en cuenta lo siguiente:

Encendido = 1 **Apagado = 0**

Al inicio la persona esta debajo de la escalera, presiona el interruptor A y el foco F enciende.
La persona sube, presiona el interruptor B y el foco F se apaga.
Llega otra persona, presiona el interruptor A y el foco F se vuelve a encender.
Sube la escalera, presiona el interruptor B y el foco F se apaga.

Condición	A	B	F
1	1	0	1
2	1	1	0
3	0	1	1
4	0	0	0

Reordenando la tabla de verdad

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

4. Obtener la función boolena.

Recordando que la función booleana se expresa como **la suma de los minitérminos donde la función vale 1,** tenemos:

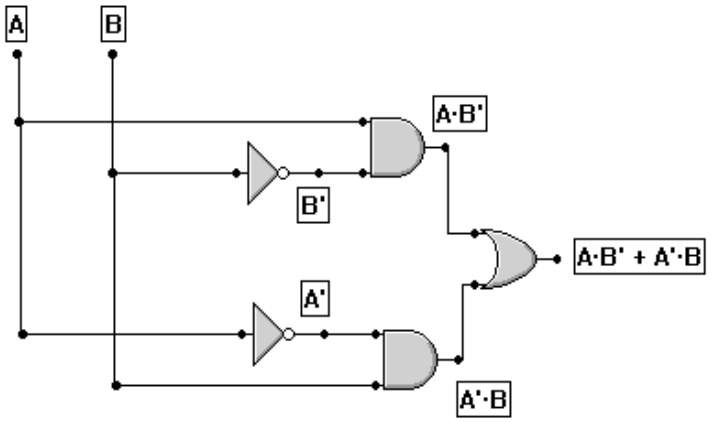
F = A·B' + A'·B

5. Simplificación de la función booleana.

En este caso la ecuación es muy sencilla y por lo tanto ya no se puede optimizar o simplificar más ya que no es posible factorizarse.

6. Implementación del circuito lógico.

El diagrama del circuito lógico resultante es el siguiente:



3 Semisumador

Diseñar un circuito para sumar dos números binarios de un digito.

Solución:

1. Planteamiento del problema.

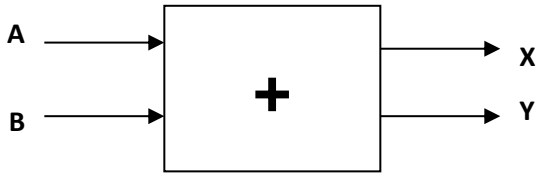
La suma de dos números binarios es la siguiente:

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

Al resultado anterior vamos a colocarle un cero a la izquierda, como sabes, una cantidad no se altera si colocamos un cero a la izquierda. Entonces tenemos lo siguiente:

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

2. Visualizándolo como caja negra.



A y B son las variables de entrada que representan a los sumandos.

X y Y son las variables de salida que representan el resultado.

Si definimos que los números binarios la suma de los números binarios sea la entrada de nuestra tabla de verdad y el resultado de la suma sea la salida, entonces tendríamos lo siguiente:

3. Obtener la tabla de verdad.

Representamos los sumandos y los resultados en la tabla de verdad.

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

4. Obtener la función booleana.

La variable X es igual a 1 cuando las variables de entrada son A = 1 y B=1, lo cual se representa como X·Y, por tanto la función booleana es:

$$X = A \cdot B$$

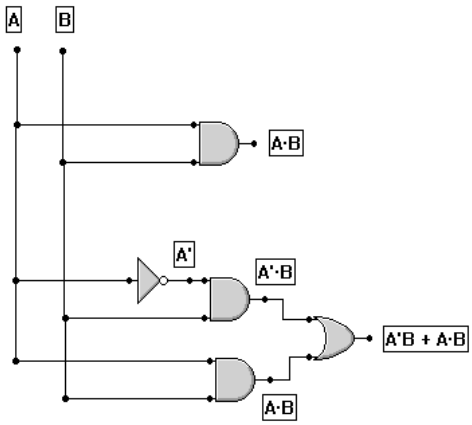
La variable Y es igual a 1 en las combinaciones A=0 y B=1, A=1 y B=0, por lo tanto la función booleana es:

$$Y = A' \cdot B + A \cdot B$$

5. Simplificar de la función booleana.

Debido a que esta función es sencilla, no es necesario la simplifición ya que no es posible factorizar.

6. Implementando el circuito.



4 Sumador Completo

Diseñar un circuito que realice la suma de tres números binarios.

Solución:

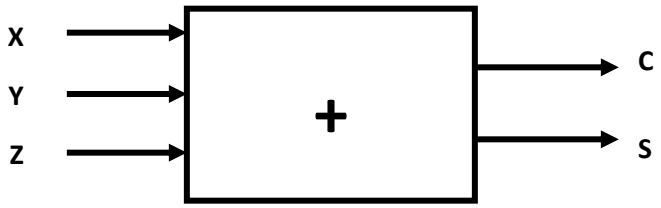
1. Planteamiento del problema.

La suma de tres números binarios (agregando un cero a la izquierda tal como en el ejemplo anterior) es la siguiente:

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

Sean los sumandos las entradas de la tabla de verdad y el resultado las salidas de esta. Por tanto:

2. Visualizando el problema como caja negra.



3. Obtener la tabla de verdad.

Representamos los sumandos y los resultados en la tabla de verdad.

X	Y	Z	C	S
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

4. Obtener la función booleana.

La función C es igual a 1 en los siguiente minitérminos:

$$C = X' \cdot Y \cdot Z + X \cdot Y' \cdot Z + X \cdot Y \cdot Z' + X \cdot Y \cdot Z$$

Mientras que para la función S, esta es igual a 1 en los siguientes minitérminos:

$$S = X' \cdot Y' \cdot Z + X' \cdot Y \cdot Z' + X \cdot Y' \cdot Z' + X \cdot Y \cdot Z$$

5. Simplificar la función booleana.

Para la función **C= X'·Y·Z + X·Y'·Z + X·Y·Z' + X·Y·Z** vamos a simplificarla utilizando los postulados y teoremas del álgebra de Boole que se mencionaron anteriormente.

Paso 1. Factorizar el término X:	**X'·Y·Z + X·(Y'·Z + Y·Z' + Y·Z)**
Paso 2. Factorizar el término Y:	**X'·Y·Z + X·(Y'·Z + Y·(Z' + Z))**
Paso 3. Aplicando $Z + Z' = 1$:	**X'·Y·Z + X·(Y'·Z + Y·(1))**
Paso 4. Aplicando $Y \cdot 1 = Y$:	**X'·Y·Z + X·(Y'·Z + Y)**
Paso 5. Aplicando $Y + Y' \cdot Z = Y + Z$:	**X'·Y·Z + X·(Y + Z)**
Paso 6. Eliminado la factorización:	**X'·Y·Z + X·Y + X·Z**
Paso 7. Factorizar el término *Y*:	**Y·(X'·Z + X) + X·Z**
Paso 8. Aplicando $X + X' \cdot Z = X + Z$	**Y·(X + Z) + X·Z**
Paso 9. Eliminando la factorización:	**Y·X + Y·Z + X·Z**

Finalmente:

C_s = X·Y + Y·Z + X·Z

Para la función **S = X'·Y'·Z + X'·Y·Z' + X·Y'·Z' + X·Y·Z** no es posible simplificarla mediante el álgebra de Boole por lo cual se queda tal como esta.

Si queremos utilizar el metodo que utiliza el mapa de Karnaugh, debemos de colocar en él los valores en los cuales los miniterminos son 1's y agrupar en conjuntos de dos o cuatro, vertical u horizontal. Es decir:

0	**0**	**1**	**0**
0	**1**	**1**	**1**

Primera agrupación: m_3 y m_7
Segunda agrupación: m_5 y m_7
Tercera agrupación: m_7 y m_6

De la primera agrupación se reduce a: Y·Z
De la segunda agrupación se reduce a: X·Z
Y de la última agrupación: X·Y

Por tanto, la simplificación de la función C es: **C_s = Y·Z + X·Z + X·Y**

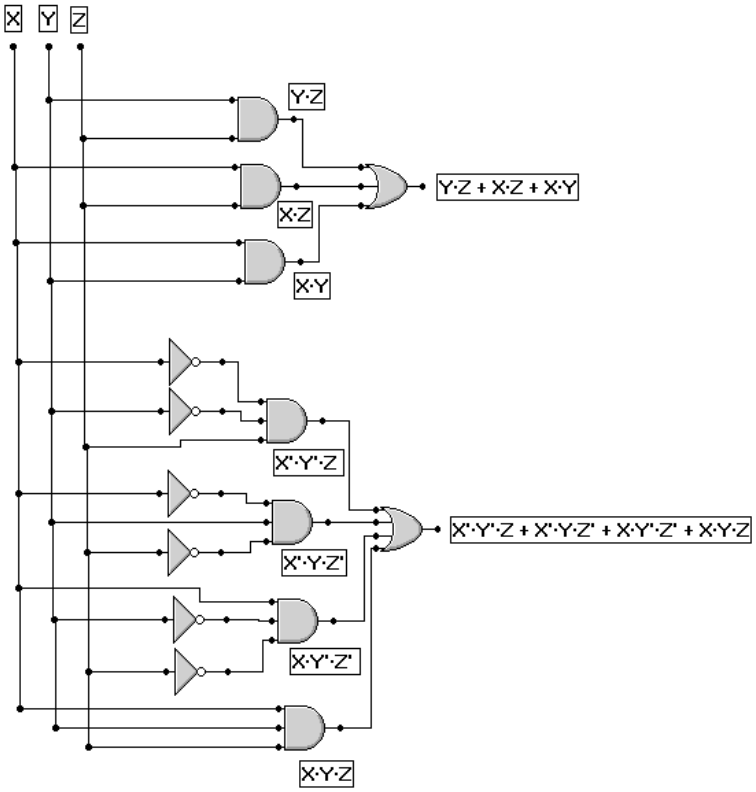
Para la función S, utilizando el mapa de Karnaugh:

0	**1**	**0**	**1**
1	**0**	**1**	**0**

Como puedes observar, no es posible realizar una agrupación, por tanto, la función no se puede simplificar y se queda igual, es decir: **S = X'·Y'·Z + X'·Y·Z' + X·Y'·Z' + X·Y·Z**

6. Implementando el circuito.

El circuito correspondiente nos queda como:



5 Semirestador

Diseñar un circuito para sumar restar dos números binarios de un digito.

Solución:

1. Planteamiento del problema.

La resta de dos números binarios es la siguiente:

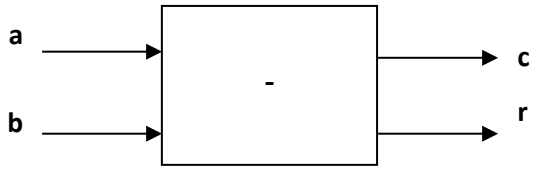
0 - 0 = 0
0 - 1 = **1** 1
1 - 0 = 1
1 - 1 = 0

Observa que el resultado de 0 – 1, el uno a la izquierda significa que se lleva uno en el resultado. Agregando un cero a la izquierda en los demás resultados tenemos:

0 - 0 = 0 0
0 - 1 = 1 1
1 - 0 = 0 1
1 - 1 = 0 0

Hagamos que los números a restar sean la entrada del circuito y el resultado de la resta sea la salida.

2. Visualizándolo como caja negra.



3. Obtener la tabla de verdad.

De acuerdo a las condiciones del problema.

a	b	c	r
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

4. Obtener la función booleana.

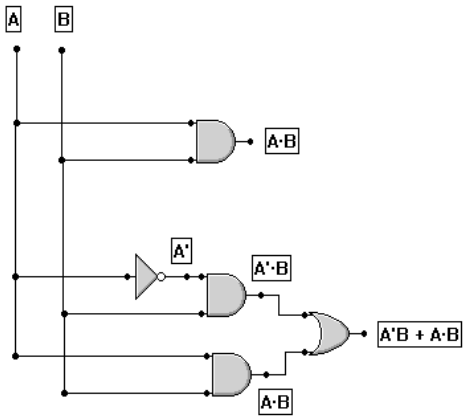
Para la función c, la representación en minitérminos es: **c= a'·b**

Para la función r, tenemos: **r = a'·b + a·b'**

5. Simplificar de la función booleana.

Debido a que esta función es sencilla, no es necesario la simplifición ya que no es posible factorizar.

6. Implementando el circuito.



6 Detector de números pares a la entrada

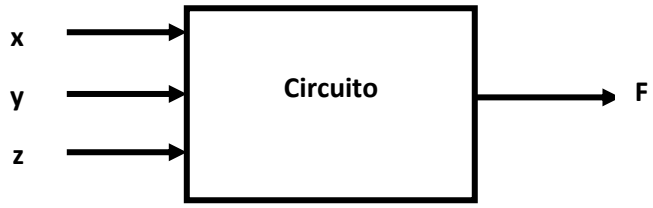
Diseñar un circuito de tres entradas que nos indique cuando el número total de 1's a la entrada es un número par.

Solución:

1. Planteamiento del problema.

Dado que nos piden un circuito de tres entradas, entonces tendremos 8 combinaciones posibles. Aquí deberemos identificar cuando se forma un número par.

3. **Visualizándolo como caja negra**.



3. Obtener la tabla de verdad.

De acuerdo a las condiciones del problema.

x	y	z	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Observa que los valores de x,y,z en donde hay una cantidad par de 1's son en las combinaciones:

0	1	1
1	0	1
1	1	0

Por esta razón, la función F vale 1 en estas combinaciones.

4. Obtener la función booleana.

Los minitérminos en los cuales la función F es igual a 1 son:

$$F = x'\cdot y\cdot z + x\cdot y'\cdot z + x\cdot y\cdot z'$$

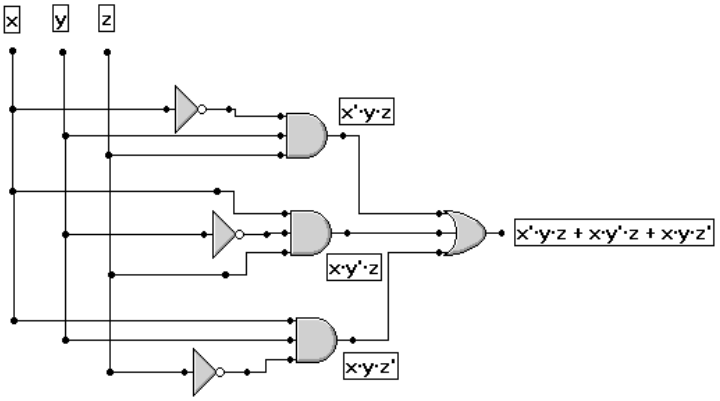
5. Simplificar de la función booleana.

Mediante la utilización del álgebra de Boole no es posible simplificar la función, adicionalmente si utilizamos el mapa de Karnaugh para esta función:

0	**0**	**1**	**0**
0	**1**	**0**	**1**

No es posible realizar una simplificación por lo cual la función lógica permanece igual.

6. Implementando el circuito.



7 Detector de números impares a la entrada

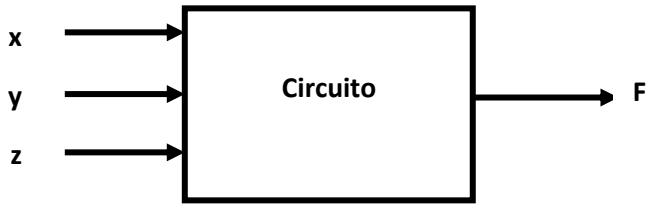
Diseñar un circuito de tres entradas que nos indique cuando el número total de 1's a la entrada es un número impar.

Solución:

1. Planteamiento del problema.

Como en el ejemplo anterior, para un circuito de tres entradas, tendremos 8 combinaciones posibles. La diferencia es que ahora deberemos identificar cuando se forma un número impar de 1's.

2. Visualizándolo como caja negra.



3. Obtener la tabla de verdad.

De acuerdo a las condiciones del problema.

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

Observa que los valores de x,y,z en donde hay una cantidad impar de 1's, son en las combinaciones:

0	0	1
0	1	0
1	0	0
1	1	1

Por esta razón, la función F vale 1 en estas combinaciones.

4. Obtener la función booleana.

Los minitérminos en los cuales la función F es igual a 1 son:

F = x'·y'·z + x'·y·z' + x·y'·z' + x·y·z

5. Simplificar de la función booleana.

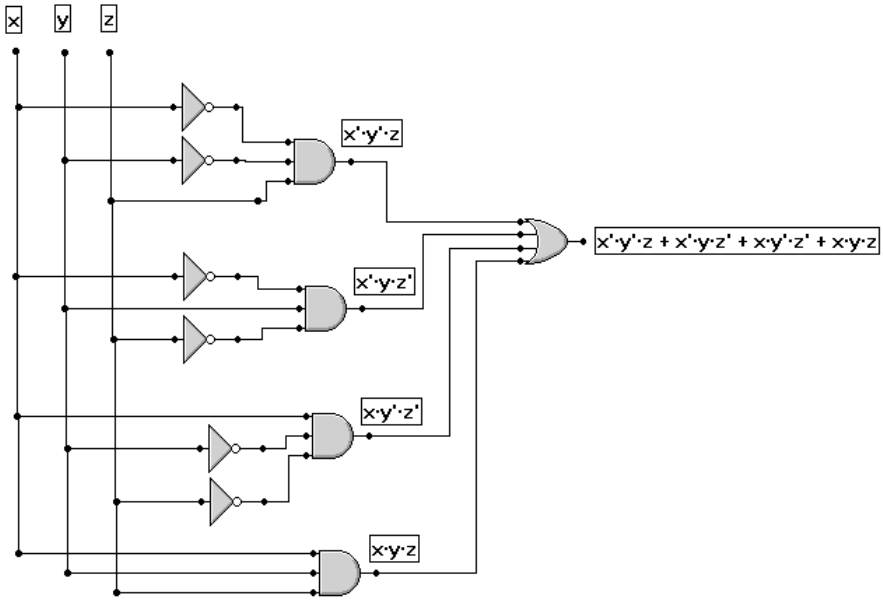
No es posible simplificar la función mediante el álgebra de Boole.

Adicionalmente, utilizando el mapa de Karnaugh para esta función:

0	**1**	**0**	**1**
1	**0**	**1**	**0**

Vemos también que no es posible realizar una simplificación, por lo cual la función lógica permanece igual.

6. Implementando el circuito.



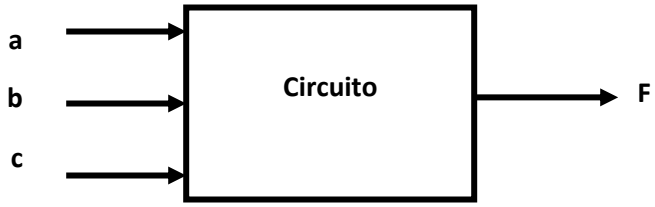
8 Detector de estados

Diseñar un circuito que tenga 3 entradas y una salida. La salida es igual a 1 cuando todas las entradas sean iguales a 1, o cuando ninguna de las entradas sea igual a 1 o cuando alguna de las 3 entradas sea 1.

1. Planteamiento del problema.

En este caso el problema esta planteado claramente, por lo cual podemos obtener la tabla de verdad directamente de las condiciones establecidas.

2. Visualización como caja negra.



3. Obtener la tabla de verdad.

De acuerdo al planteamiento del problema:

	A	B	C	F
Cuando ninguna entrada es 1	0	0	0	1
Cuando al menos una entrada es 1	0	0	1	1
Cuando al menos una entrada es 1	0	1	0	1
	0	1	1	0
Cuando al menos una entrada es 1	1	0	0	1
	1	0	1	0
	1	1	0	0
Cuando las todas las entradas son 1	1	1	1	1

4. Obtener la función booleana.

Observado en cuales minitérminos la función F es 1, tenemos:

$$\mathbf{F = a'\cdot b'\cdot c' + a'\cdot b'\cdot c + a'\cdot b\cdot c' + a\cdot b'\cdot c' + a\cdot b\cdot c}$$

5. Simplificar la función lógica.

Para la función **a'·b'·c' + a'·b'·c + a'·b·c' + a·b'·c' + a·b·c** vamos a simplificar la función utilizando los postulados y teoremas del álgebra de Boole que se mencionaron anteriormente.

Paso 1. Factorizar el término a':	**a'·(b'·c' + b'·c + b·c') + a·b'·c' + a·b·c**
Paso 2. Factorizar el término b':	**a'·(b'·(c' + c) + b·c') + a·b'·c' + a·b·c**
Paso 3. Aplicando $c' + c = 1$:	**a'·(b'·1 + b·c') + a·b'·c' + a·b·c**
Paso 4. Aplicando $b' \cdot 1 = b$:	**a'·(b'+ b·c') + a·b'·c' + a·b·c**
Paso 5. Aplicando $b' + b\cdot c = b' + c$:	**a'·(b'+ c') + a·b'·c' + a·b·c**
Paso 6. Eliminado la factorización:	**a'·b'+ a'·c' + a·b'·c' + a·b·c**
Paso 7. Factorizar el término b':	**b'·(a' + a·c') + a'·c'+ a·b·c**
Paso 8. Aplicando $a' + a\cdot c' = a' + c'$	**b'·(a' + c') + a'·c'+ a·b·c**

Finalmente:

$$\mathbf{F = a'\cdot b' + b'\cdot c' + a'\cdot c'+ a\cdot b\cdot c}$$

Utilizando el mapa de Karnaugh tenemos:

1	1	0	1
1	0	1	0

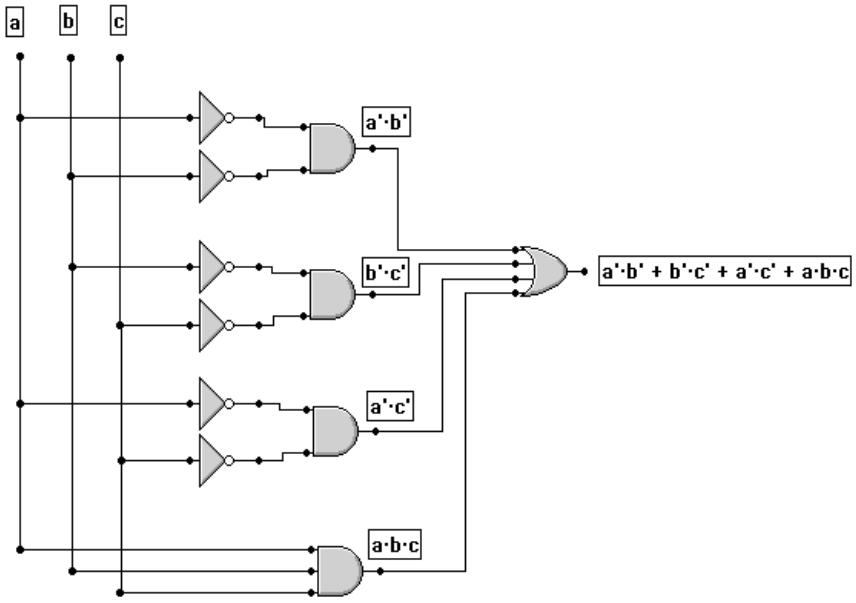
Podemos agrupar:

- m_0 y m_1 de lo cual nos queda el término: **a'·b'**
- m_0 y m_4 que corresponde a: **b'·c'**
- m_0 y m_2 el cual es: **a'·c'**
- Y el término m_7 : **a·b·c**

Por tanto, la función simplificada es:

$$F_s = a'\cdot b' + b'\cdot c' + a'\cdot c' + a\cdot b\cdot c$$

6. Implementando el circuito.

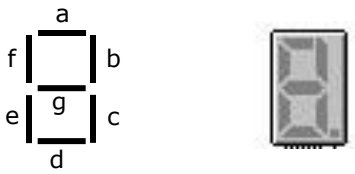


9 Convertidor de binario a decimal

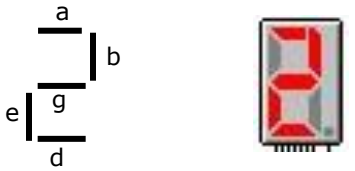
Deseamos construir un circuito que convierta un numero binario a su correspondiente número decimal.

1. Planteamiento del problema.

Para resolver este problema necesitamos de un circuito especial que nos permita ver la información en un formato decimal. Para ello utilizáremos un dispositivo conocido como **Display de 7 Segmentos** representado en las figuras siguientes:



Para formar un número decimal debemos encender los segmentos correspondientes; por ejemplo, para formar el número 2, debemos de encender los segmentos a, b, g, e y d, representándose de la siguiente forma:

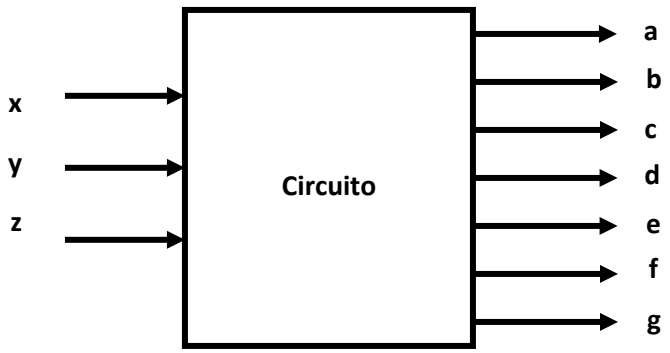


A continuación, vamos a realizar una tabla en donde se especifiquen los segmentos que se deben de encender para formar los números decimales del 0 a 7.

Número decimal	a	b	c	d	e	f	g
0	1	1	1	1	1	1	0
1	0	1	1	0	0	0	0
2	1	1	0	1	1	0	1
3	1	1	1	1	0	0	1
4	0	1	1	0	0	1	1
5	1	0	1	1	0	1	1
6	1	0	1	1	1	1	1
7	1	1	1	0	0	0	0

Para llevar a cabo el circuito correspondiente, a la entrada tendremos un numero en binario que ira de 000 a 111, mientras que en la salida formaremos el numero decimal correspondiente con la ayuda de la tabla anterior y para lo cual tendremos 7 funciones lógicas (a, b, c, d, e f g), cada función lógica encenderá el segmento correspondiente del display

2. Visualización como caja negra.



3. Obtener la tabla de verdad.

De acuerdo al planteamiento del problema:

x	y	z	a	b	c	d	e	f	g
0	0	0	1	1	1	1	1	1	0
0	0	1	0	1	1	0	0	0	0
0	1	0	1	1	0	1	1	0	1
0	1	1	1	1	1	1	0	0	1
1	0	0	0	1	1	0	0	1	1
1	0	1	1	0	1	1	0	1	1
1	1	0	1	0	1	1	1	1	1
1	1	1	1	1	1	0	0	0	0

4. Encontrar la función booleana.

Las funciones booleanas correspondientes son:

a = x'·y'·z' + x'·y·z' + x'·y·z + x·y'·z + x·y·z' + x·y·z

b = x'·y'·z' + x'·y'·z + x'·y·z' + x'·y·z + x·y'·z' + x·y·z

c = x'·y'·z' + x'·y'z + x'·y·z + x·y'·z' + x·y'·z + x·y·z' + x·y·z

d = x'·y'·z' + x'·y·z' + x'·y·z + x·y'·z + x·y·z'

e = x'·y'·z' + x'·y·z' + x·y·z'

f = x'·y'·z' + x·y'·z' + x·y'·z + x·y·z'

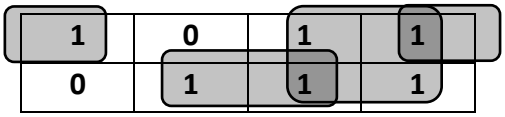
g = x'·y·z' + x'·y·z + x·y'·z' + x·y'·z + x·y·z'

5. Simplificar la función lógica.

Para la función **a = x'·y'·z' + x'·y·z' + x'·y·z + x·y'·z + x·y·z' + x·y·z** vamos a simplificar la función utilizando los postulados y teoremas del álgebra de Boole que se mencionaron anteriormente.

Paso 1. Factorizar el término x' y el término x:	**x'·(y'·z' + y·z' + y·z) + x·(y'·z + y·z' + y·z)**
Paso 2. Factorizar el término y:	**x'·(y'·z' + y·(z' + z)) + x·(y'·z + y·(z' + ·z))**
Paso 3. Aplicando *z' + z = 1* tenemos:	**x'·(y'·z' + y·1) + x·(y'·z + y·1)**
Paso 4. Aplicando *y·1 = y* nos queda:	**x'·(y'·z' + y) + x·(y'·z + y)**
Paso 4. Aplicando *y + y'·z' = y + z' y + y'·z = y + z*	**x'·(y + z') + x·(y + z)**
Paso 5. Eliminado la factorización:	**x'·y + x'·z' + x·y + x·z**
Paso 6. Factorizando y :	**y·(x' + x) + x'·z' + x·z**
Paso 7 Aplicando *x + x' = 1*	**y·1 + x'·z' + x·z**
Paso 7 Aplicando *y·1 = y*	**y + x'·z' + x·z**
Finalmente:	
	a = x·z + x'·z' + y

Utilizando los mapas de Karnaugh, **para la función a**, tenemos:



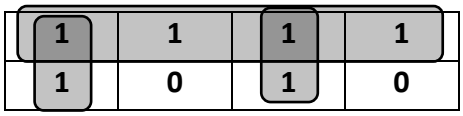
Una vez agrupando y sumando los términos correspondiente nos queda: **a = x·z + x'·z' + y**

Para la función **b = x'·y'·z' + x'·y'·z + x'·y·z' + x'·y·z + x·y'·z' + x·y·z** simplificando con álgebra de Boole:

Paso 1. Factorizar el término z' y el término z: **z'·(x'·y' + x'·y + x·y') + z·(x'·y' + x'·y + x·y)**
Paso 2. Factorizar el término x': **z'·(x'·(y' + y) + x·y') + z·(x'·(y'+ y) + x·y)**
Paso 3. *y + y' = 1* tenemos: **z'·(x'·1 + x·y') + z·(x'·1 + x·y)**
Paso 4. Aplicando *x' ·1 = x'* tenemos: **z'·(x' + x·y') + z·(x' + x·y)**
Paso 5. Aplicando *x' + x·y' = x' + y' x' + x·y = x' + y* **z'·(x' + y') + z·(x' + y)**
Paso 6. Eliminado la factorización: **x'·z' + y'·z' + x'·z + y·z**
Paso 7. Factorizando x' : **x'·(z' + z) + y'·z' + y·z**
Paso 8. Aplicando *z + z' = 1* **x'·1 + y'·z' + y·z**
Paso 9. Aplicando *x'·1 = x'* **x' + y'·z' + y·z**
Finalmente:

b = x' + y'·z' + y·z

Utilizando los mapas de Karnaugh, **para la función b**, tenemos:



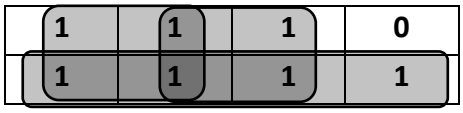
Agrupando nos queda: **b = x' + y'·z' + y·z**

Para la función **c = x'·y'·z' + x'·y'z + x'·y·z + x·y'·z' + x·y'·z + x·y·z' + x·y·z**

Paso 1. Factorizar el término x' y el término x: **x'·(y'·z' + y'·z + y·z + y'·z') + x·(y'·z' + y'·z + y·z' + y·z)**

Paso 2. Factorizar los términos z', z, y', y:	**x'·(z'·(y' + y') + z·(y' + y)) + x·(y'·(z' + z) + y·(z' + z))**
Paso 3. Aplicando *y + y' = 1* y z + z' = 1 tenemos:	**x'·(z'·(y' + y') + z·1) + x·(y'·1 + y·1)**
Paso 4. Aplicando *z·1 = z, y'·1=y', y·1=y* :	**x'·(z'·(y' + y') + z) + x·(y' + y)**
Paso 5. Aplicando *y'+y'=y', y + y' = 1*:	**x'·(z'·y' + z) + x·1**
Paso 6. Aplicando *z + z'·y' = z + y'*; *x ·1 = x* :	**x'·(z + y') + x**
Paso 7. Eliminado la factorización:	**x'·z + x'·y' + x**
Paso 8. Aplicando *x + x'·y' = x + y'*:	**x + y' + x'·z**
Paso 9. Aplicando nuevamente x *+ x'·z = x + z*:	**x + z + y'**
Finalmente:	
	c = x + y' + z

Utilizando los mapas de Karnaugh, **para la función c**, tenemos:



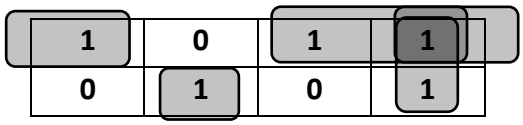
Agrupando (observa que se han agrupado tres bloques con cuatro 1's) nos queda:

c = x + y' + z

Para la función **d = x'·y'·z' + x'·y·z' + x'·y·z + x·y'·z + x·y·z'**

Paso 1. Factorizar el término x':	**x'·(y'·z' + y·z' + y·z) + x·y'·z + x·y·z'**
Paso 2. Factorizar el término z':	**x'·(z'(y' + y) + y·z) + x·y'·z + x·y·z'**
Paso 3. Aplicando *x + y' = 1*:	**x'·(z'·1 + y·z) + x·y'·z + x·y·z'**
Paso 4. Aplicando z'·1=z'	**x'·(z' + y·z) + x·y'·z + x·y·z'**
Paso 5. Aplicando *z' + z·y = z' + y*:	**x'·(z' + y) + x·y'·z + x·y·z'**
Paso 6. Eliminado la factorización:	**x'·z' + x'·y + x·y'·z + x·y·z'**
Paso 6. Factorizar el término z':	**z'·(x' + x·y)+ x'·y + x·y'·z**
Paso 7. Aplicando nuevamente *x' + x·y = x' + y*:	**z'·(x' + y) + x'·y + x·y'·z**
Paso 8. Eliminado la factorización:	**x'·z' + y·z' + x'·y + x·y'·z**
Finalmente:	
	x'·y + y·z' + x'·z' + x·y'·z

Utilizando los mapas de Karnaugh, **para la función d**, tenemos:



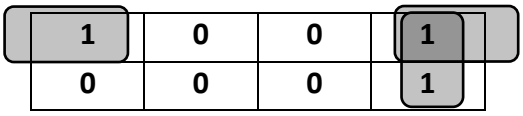
Agrupando nos queda: **d = x'·y + y·z' + x'·z' + x·y'·z**

Para la función **e = x'·y'·z' + x'·y·z' + x·y·z'**

Paso 1. Factorizar el término x': **x'·(y'·z' + y·z') + x·y·z'**
Paso 2. Factorizar el término z': **x'·(z'·(y + y')) + x·y·z'**
Paso 3. Aplicando *y + y' = 1*: **x'·(z'·1) + x·y·z'**
Paso 4. Aplicando z'·1=z' **x'·z' + x·y·z'**
Paso 5. Factorizar el término z': **z'·(x' + x·y)**
Paso 6. Aplicando *x' + x·y = x' + y*: **z'·(x' + y)**
Paso 7. Eliminado la factorización: **z'·x' + z'·y**
Finalmente:

x'·z' + y·z'

Utilizando los mapas de Karnaugh, **para la función e**, tenemos:



Agrupando nos queda: **e = x'·z' + y·z'**

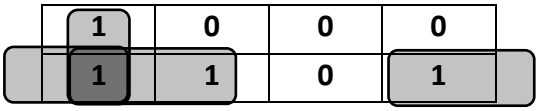
Para la función **f = x'·y'·z' + x·y'·z' + x·y'·z + x·y·z'**

Paso 1. Factorizar el término x: **x·(y'·z' + y'·z + y·z') + x'·y'·z'**
Paso 2. Factorizar el término y': **x·(y'·(z' + z) + y·z') + x'·y'·z'**
Paso 3. Aplicando *z + z' = 1*: **x·(y'·1 + y·z') + x'·y'·z'**
Paso 4. Aplicando *y'·1 = y :* **x·(y' + y·z') + x'·y'·z'**
Paso 5. Aplicando *y' + y·z' = y' + z'*: **x·(y' + z') + x'·y'·z'**
Paso 6. Eliminado la factorización: **x·y' + x·z' + x'·y'·z'**

Paso 7. Factorizar el término y': **y'·(x + x'·z') + x·z'**
Paso 8. Aplicando *x + x'·z' = x + z'*: **y'·(x + z') + x·z'**
Paso 9. Eliminado la factorización: **x·y' + y·z' + x·z'**
Finalmente:

y·z' + x·y' + + x·z'

Utilizando los mapas de Karnaugh, **para la función f**, tenemos:



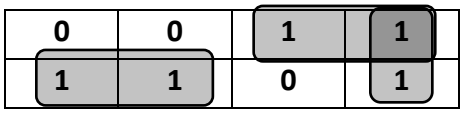
Agrupando nos queda: **f = y'·z' + x·y' + x·z'**

Para la función **g = x'·y·z' + x'·y·z + x·y'·z' + x·y'·z + x·y·z'**

Paso 1. Factorizar el término y; y: **y·(x'·z' + x'·z + x·z') + y'·(x·z' + x·z)**
Paso 2. Factorizar el término x'; x: **y·(x'·(z' + z) + x·z') + y'·(x·(z' + z))**
Paso 3. Aplicando *x + x' = 1*: **y·(x' + x·z') + y'·(x)**
Paso 4. Aplicando *x + x'·y = x + y*: **y·(x' + z') + x·y'**
Finalmente:

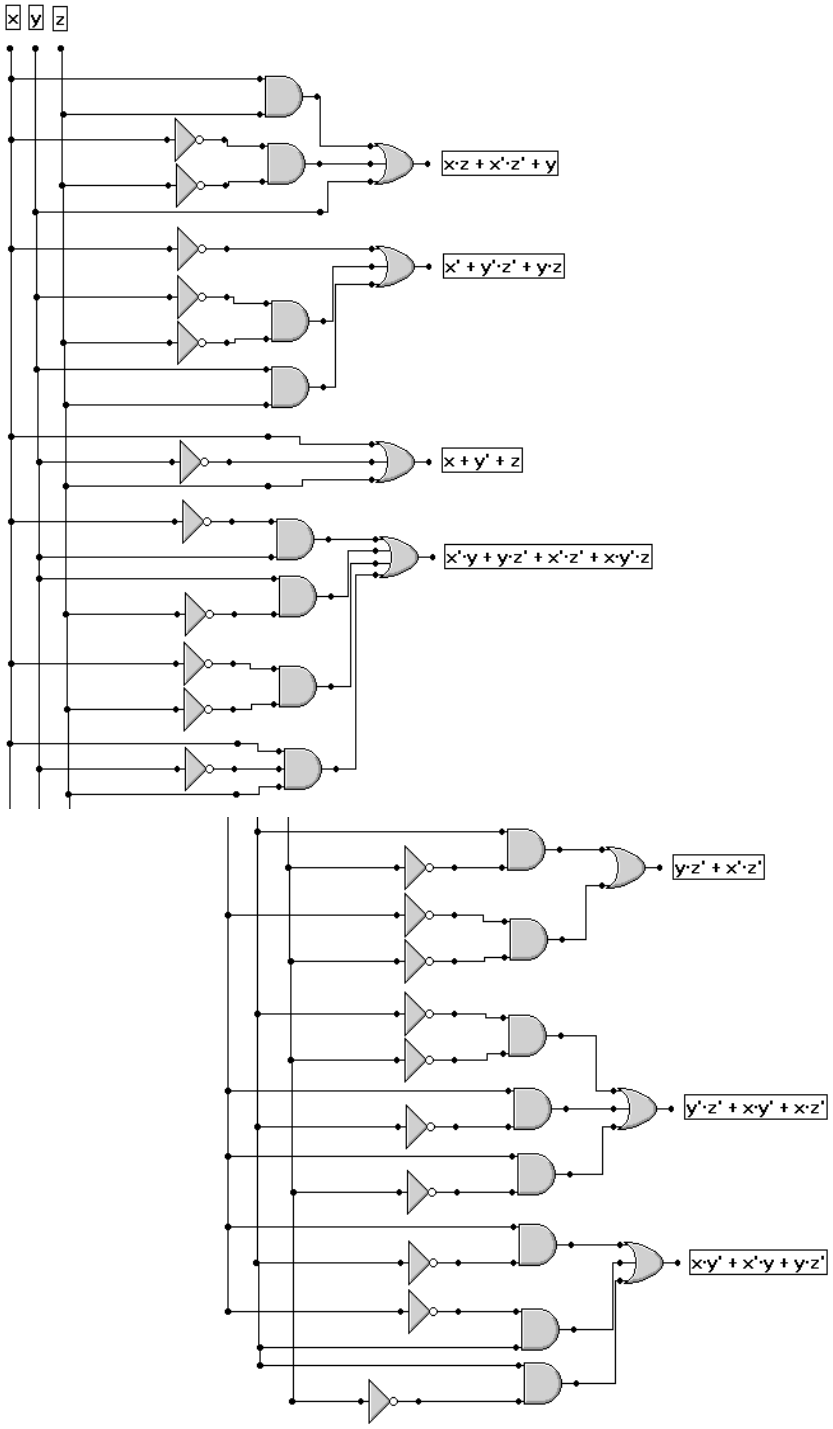
x'·y + y·z' + x·y'

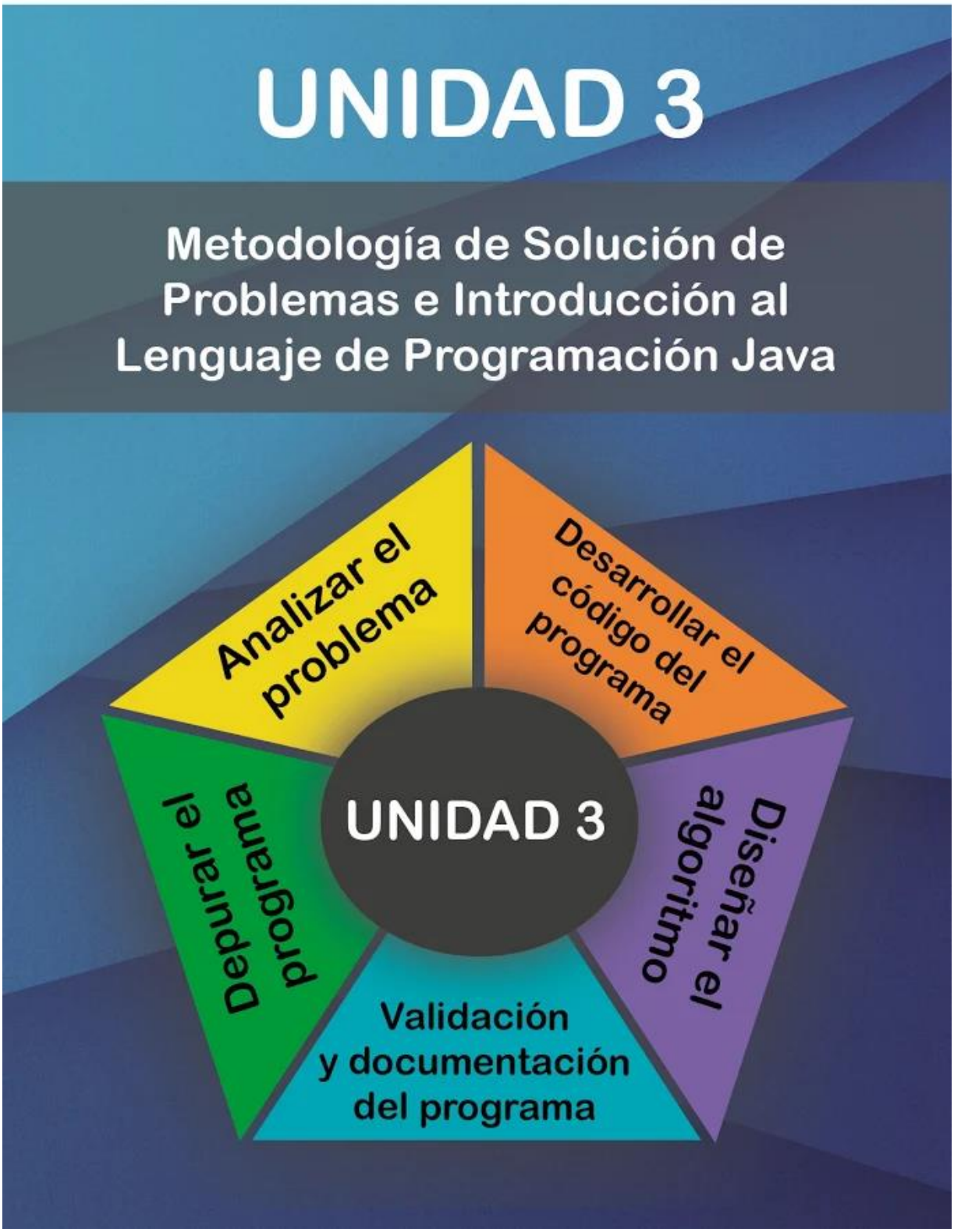
Utilizando los mapas de Karnaugh, **para la función g**, tenemos:



Agrupando nos queda: **g = x·y' + x'·y + y·z'**

6. Implementación del circuito lógico




UNIDAD 3
Metodología de Solución de Problemas e Introducción al Lenguaje de Programación Java
Analizar el problema
Desarrollar el código del programa
Diseñar el algoritmo
Validación y documentación del programa
Depurar el programa
UNIDAD 3

Unidad III. Metodología de solución de problemas e introducción de lenguaje de programación Java

Propósito:

Al finalizar el alumno:

Aplicará la metodología de solución de problemas mediante la construcción de algoritmos y a codificación en el lenguaje de programación Java para tener una visión integral del proceso de solución

Aprendizajes:

- 🕮 Definición de conceptos de problemas.
- 🕮 Identifica los elementos de un problema.
- 🕮 Describe la diferencia entre problemas determinísticos, probabilísticos, secuenciales y condicionales.
- 🕮 Conoce las etapas de la metodología de solución de problemas.
- 🕮 Analiza el resultado de expresiones aritméticas utilizando la jerarquía de las operaciones.
- 🕮 Construye expresiones lógicas utilizando operadores relacionales y lógicos.
- 🕮 Conoce el concepto de algoritmo, diagrama de flujo y pseudocódigo.
- 🕮 Elabora el algoritmo, diagrama de flujo y pseudocódigo para diferentes tipos de problemas.
- 🕮 Conoce la historia del lenguaje de programación Java.
- 🕮 Conoce las características básicas del lenguaje de programación Java.
- 🕮 Conoce el entorno de desarrollo para el lenguaje de programación Java.
- 🕮 Realiza programas empleando el método de salida de datos.
- 🕮 Realiza programas empleando la Clase Scanner para la entrada de datos.
- 🕮 Elaborará el algoritmo, diagrama de flujo y seudocódigo para resolver problemas de estructuras de ciclo
- 🕮 Elaborará el algoritmo, diagrama de flujo y seudocódigo para problemas condicionales.
- 🕮 Construye programas de computadora que resuelvan problemas condicionales.
- 🕮 Elaborará el algoritmo, diagrama de flujo y seudocódigo para problemas condicionales múltiples.
- 🕮 Construye programas de computadora que resuelvan problemas que empleen la sentencia for

- Elaborará el algoritmo, diagrama de flujo y seudocódigo para resolver problemas de ciclo que satisfagan una condición
- Construye programas de computadora que resuelvan problemas que involucren la sentencia while

TEMA 1

Definición y conceptos generales de un problema

Objetivo:

- Definir del concepto de problema

Problema

El término problema, proviene del griego πρόβλημα, y del latín problēma, que significa dificultad. Se define como problema aquel asunto, tema o cuestión que requiere de una solución.

Esta definición ha ido evolucionando a través del tiempo y de las necesidades a resolver por tal motivo si lo enfocamos al área de matemáticas, se puede definir como el conjunto de números y operadores que describen un caso real o un contexto. Es decir, se define como una pregunta que aun esta sin una respuesta inmediata y para la cual queremos encontrar su solución, entonces, se plantean como problemas con preguntas específicas, es decir con las siguientes preguntas podemos plantear un problema a resolver:

1. ¿Cuál es el área de un triángulo?
2. ¿Cuántos números primos hay entre 1 y 100?
3. ¿Cuánto se debe pagar por un viaje a Querétaro, si abordo junto con tres amigos y el precio por persona es de $125 pesos?
4. ¿Cuánta pintura se necesita para pintar una casa de 1200 metros cuadrados, si cada bote de pintura tiene 5 litros y con este se cubren 100 metros cuadrados?

En cada uno de los problemas planteados se encuentra al menos una respuesta, para encontrarla es necesario identificar los elementos del problema que nos permiten dar una solución válida.

TEMA 2

Elementos y relaciones del problema

Objetivo:

- Identificar los elementos de un problema

1 Elementos

Los elementos de un problema se identifican como un modelo de caja negra, este modelo permite representar a un problema desde su planteamiento hasta su solución, permitiendo especificar el camino que lleva a la solución del problema de una forma sistemática y con la solución buscada, representada en uno de sus elementos.

Entrada. Son todos aquellos recursos con los que contamos al momento de tener el problema.

Proceso. Este es el elemento donde se realizan las operaciones o cálculos necesarios para tener una solución al problema planteado. Normalmente no se sabe lo que ocurre en el proceso, solo se sabe que después de este paso se obtiene la solución.

Salida. Son los recursos obtenidos por los cálculos realizados, también, se puede decir que son los resultados obtenidos.

Figura 3.2.1. Modelo de caja negra

La figura 3.2.1, muestra un ejemplo del modelo de caja negra, el cual, permite establecer un panorama claro de todos los elementos útiles para resolver el problema, es decir se establece por separado los elementos de entrada, los cuales estarán vinculados de alguna manera con la salida, con esto, se tiene una idea de cuál debe ser el camino para dar solución al problema planteado.

Ejemplos:

- ¿Cuál es el área de un triángulo?

  Entrada: Los valores de la base y altura del triángulo.
  Proceso: Las operaciones para realizar el cálculo de la altura, A=(base)(altura)/2
  Salida: El valor calculado del área.

- ¿Cuántos números primos hay entre 1 y 100?

  Entrada: Los números del 1 al 100.
  Proceso: Las operaciones para determinar si es número primo o no.
  Salida: Los números que cumplieron con la condición de ser primo.

- ¿Cuánto se debe pagar por un viaje a Querétaro, si abordo junto con tres amigos y el precio por persona es de $125 pesos?

  Entrada: Número de personas que viajan a Querétaro, precio del boleto por persona.
  Proceso: Operaciones para realizar el cálculo de todos los viajeros a Querétaro.
  Salida: Precio total a pagar por los viajeros a Querétaro.

- ¿Cuánta pintura se necesita para pintar una casa de 1200 metros cuadrados, si cada bote de pintura tiene 5 litros y con este se cubren 100 metros cuadrados?

  Entrada: Litros de pintura por bote y superficie total a pintar.
  Proceso: Las operaciones para calcular la cantidad de pintura total a ocupar.
  Salida: Pintura que se necesita para pintar.

TEMA 3

Tipos de problemas

Objetivo:

- Describir las diferencias entre los problemas determinísticos, probabilísticos, secuencias y condicionales

1 Introducción

Existen una variedad de problemas a los que nos enfrentamos todos los días y cada uno de ellos representa un conocimiento que permite enfrentar otros con mejor resultado. Lo que nos ayuda a ser más hábiles en la resolución de problemas de forma cotidiana.

Lo anterior mencionado, permite clasificar a los en: determinísticos, probabilísticos, secuencias y condicionales, donde cada uno de estos permite establecer criterios para abordar con mayor éxito cada problema y poder establecer un camino para buscar la mejor solución.

De esta manera, las diferencias entre estos tipos de problemas consisten en la forma de plantear el problema y por tal motivo, la solución que se encuentre.

2 Determinísticos

Son aquellos problemas donde para determinadas condiciones su solución es única y siempre la misma. Es decir, las mismas entradas de un problema, producirán las mismas salidas. En este tipo de problema se conocen los datos con certeza, por lo tanto, se tiene toda la información necesaria para dar una solución a esté.

Ejemplo: encontrar todas las raíces reales para "x" de la siguiente ecuación: $x^2 + 2x + 1 = 0$

3 Probabilísticos

Son aquellos problemas en donde para los mismos datos de entrada su solución es diferente siempre. Es decir, al menos una de las variables es considerada como un dato al azar y los demás datos son obtenidos de manera probabilística. Para la solución de este tipo de problema

también llamado estocástico, se toma un conjunto de datos por muestreo, con un comportamiento aleatorio.

Ejemplo: Se propone un juego, el cual consiste en lanzar 3 dados al aire, si salen los tres dados con el mismo valor, el jugador en turno gana 7 puntos. En caso de lanzar los dados y ser todas las caras distintas, entonces, pierde el jugador en turno 2 puntos. Los jugadores inician con 50 puntos y gana el jugador que logre quedar con puntos después de que sus competidores pierdan todos sus puntos.

4 Secuenciales

Los problemas secuenciales siguen una trayectoria consecutiva en los pasos para encontrar una respuesta, es decir, una instrucción sigue a otra en secuencia. Las acciones son sucesivas de tal forma que la salida de una es la entrada de otra, de manera que se llega al final del proceso o en su caso a la solución del problema.

Ejemplo: se tiene una calle con sus casas numeradas del 1 al 20, identifica cuantas veces aparece el número dos en los números de las casa.

5 Condicionales

Un problema condicional es aquel que toma como base para dar una respuesta una condición, donde está permite tener una opción de respuesta o más de una. Es decir, se compara una variable con otro valor, en base a esta comparación (condición) se obtiene una solución al problema.

Ejemplo: en el bachillerato, para la asignatura de matemáticas, la evaluación consiste en tres exámenes y que el resultado de la calificación es el promedio de estos, para acreditar la asignatura es necesario que el promedio sea mayor o igual a 6 y 5 en caso contrario.

Ejercicio

Relaciona las siguientes columnas para identificar los conceptos de forma correcta.

( )	1. ¿Cuántos números primos hay entre 1 y 100?	A. Determinístico
( )	2. Elemento donde se realizan las operaciones o cálculos necesarios para tener una solución al problema planteado.	B. Entrada

( ) 3. En este tipo de problema, que para determinadas condiciones su solución es única y siempre la misma.
( ) 4. Para este tipo de problema, los datos de entrada dan una solución aleatoria.
( ) 5. Elemento de un sistema donde se presentan los resultados de un proceso.
( ) 6. Elemento de un sistema que permite alimentarlo con datos.
( ) 7. Se define como una pregunta que aun esta sin una respuesta inmediata y para la cual queremos encontrar su solución.

C. Ejemplo de problema
D. Probabilístico
E. Problema
F. Proceso
G. Salida

TEMA 4

Etapas de la metodología de solución de problemas

Objetivo:

- Conocer las etapas de solución de la metodología de solución de problemas

1 Introducción

La solución de problemas consiste en aplicar una metodología que ayude a la construcción de soluciones a partir de los elementos requeridos para el planteamiento de un problema específico, estableciendo de esta forma que estos elementos son la base para encontrar los recursos que buscamos.

Lo primero que se estables al momento de dar solución a un problema es plantearlo, esto fija con claridad los requerimientos que se necesitan y que elementos tenemos para la solución.

La segunda consideración es establecer el análisis del problema, es decir construir un camino o ruta que lleve a la solución, tomando como base los elementos del propio problema que ya considera para llegar al resultado esperado.

Como tercer punto; es el diseño de solución del problema, en esta etapa se expresa la vía que se debe seguir para resolver el problema, el diseño de la solución se plantea por medio de un algoritmo: la cual puede ser mediante un diagrama de flujo o pseudocódigo.

2 Planteamiento del problema

Al plantear un problema se debe buscar un camino que permita resolverlo y para ello es necesario considerar que este tiene solución y que no es obvio. Para apoyar la solución del problema, es necesario tener claro que dice en él y que busca resolver, esto es el punto más destacado al momento de plantear el problema ya que si no es claro lo que se quiere resolver con base al planteamiento, entonces no será posible encontrar una solución. Para ello se hace necesario revisar varias veces el problema hasta comprenderlo adecuadamente para en caso necesario buscar un mejor resultado.

No se debe suponer que el problema es obvio, es decir, se debe considerar que es importante comprender lo que el problema indica y lo que se solicita para poder ser resuelto. Es claro que, si se tiene la solución inmediata, entonces se considera que no tiene sentido ser planteado porque no hay nada que resolver, es decir si la solución es obvia, entonces no es un verdadero problema.

Debe de existir una definición por escrito del trabajo que se debe hacer, por lo que, al plantear un problema se considera como los antecedentes teóricos del problema los datos que se extraen de esté. Esta información permite ser el objetivo al entender el problema y con ello buscar la solución de lo que se busca resolver. Es decir, concentrarse primero en el *¿Qué?* del problema y no en el *¿Cómo?* se va a resolver.

El comprender el problema permite un mejor planteamiento de lo que debemos encontrar en la solución, para ello es importante tener claro que es lo que queremos resolver, siendo así obligado preguntarse, ¿qué datos tengo?, ¿qué dice el problema?, ¿qué relación hay entre los datos del problema?

Leer varias veces el documento inicial a fin de familiarizarse con el problema a resolver, ayuda tener más claro cuáles son los elementos que intervienen en el problema y como se relacionan, así como encontrar las variables que se relación en el problema.

Presentar el problema en un lenguaje claro y preciso hace posible escribirlo o describirlo en un lenguaje diferente, quitando así la carga de la formalidad, pero conservando los elementos que intervienen y sus relaciones.

2 Análisis del problema

Consiste en especificar un camino que permita la solución del problema, esto es a partir de los elementos que se tienen del problema se describe lo que debe hacerse para alcanzar la solución del problema.

Ese camino a seguir se plantea con varios modelos, en nuestro presente trabajo utilizaremos el modelo de caja negra. Consiste en considerar una caja donde llega uno o varios datos de entrada y al llegar a la caja negra estos datos se procesan dando uno o varios datos que son solución al problema planteado.

El objetivo es determinar las especificaciones del problema, es decir lo que se hará y lo que no se hará. En esta etapa se hace uso del modelo de caja negra en donde se identifica: las entradas de información que el usuario pueda introducir, las fórmulas y procedimientos para obtener resultados y las salidas de información que pueden darse

.

3 Diseño de la solución del problema

En esta etapa expresamos el camino que debemos seguir de forma simbólica para resolver el problema y encontrar la solución. Está se representa con un *Algoritmo*, el cual se acompaña de un diagrama de flujo y pseudocódigo, además de considerar la prueba de escritorio.

Para el diseño de la solución se da mediante la construcción de algoritmos, estos son la base para determinar el resultado del problema, por lo que, con los datos que se tienen de entrada se obtenga la salida esperada.

Los ***algoritmos*** son una lista de pasos a seguir que especifican una sucesión de operaciones necesarias para la solución de cualquier problema, el cual puede ser representado por un diagrama de flujo y pseudocódigo. Los algoritmos deben de contar con las variables de entrada y constantes (estas son definidas al inicio de este), una estructura de control (cuerpo) y salida (resultado esperado).

Representación

La representación de los algoritmos se hace mediante los ***diagramas de flujos***, los cuales son una presentación gráfica de las operaciones y estructuras de control que se utilizan de forma ordenada, para ello se emplea una simbología estándar, considerando un inicio y un final.

El ***Pseudocódigo*** permite, detallar lo más posible las instrucciones a realizar utilizando de es la presentación lo más detallado posible de la solución del problema

Es una representación de la solución a un problema, la base de esta representación es el lenguaje común combinado con un lenguaje de programación. La combinación se da a partir de establecer una idea lógica en los pasos que dan solución al problema a partir de palabras claves que ayudan a comprender el camino que debe seguirse para resolver el problema.

Es una herramienta utilizada para identificar la escritura que seguirá el problema a desarrollar, utilizando palabras del lenguaje común y palabras propias del lenguaje de programación.

Prueba de escritorio. Es un procedimiento que se utiliza para establecer si cada paso del algoritmo fue considerado e implementado adecuadamente y con ello poder resolver el problema. Al iniciar la prueba de escritorio se proponen datos que serán llevados a través de todos los pasos para así evaluar los pasos y saber si llegamos a los valores esperados hasta obtener el resultado final.

La prueba de escritorio es una herramienta útil para entender que hace un determinado algoritmo, o para verificar que un algoritmo cumple con la especificación sin necesidad de ejecutarlo, es como una ejecución a mano del algoritmo, por lo tanto se debe llevar registro de los valores que va tomando cada una de las variables involucradas en el mismo. La prueba de escritorio nos resulta más clara al elaborar una tabla con los valores de las entradas, las salidas y los valores internos del proceso, consiste en ir colocando los valores y cómo van cambiando durante el algoritmo para ver si se cumple con lo esperado.

La prueba de escritorio es útil porque muestra claramente cómo se van generando los valores de los elementos y podemos descubrir en qué punto del proceso puede haber una falla para poder realizar las correcciones correspondientes.

Para profundizar en el tema de la construcción de algoritmos revisar la sección 3.7 así como los temas consecutivos.

Ejercicio

Lee cuidadosamente los siguientes conceptos y relaciónalos:

( ) Etapa en la que no debemos suponer que el problema es obvio.

( ) Etapa en la debemos decir lo que se hará y lo que no se hará del problema.

( ) Etapa en donde debemos elaborar un algoritmo para resolver el problema.

A. Algoritmo

B. Análisis

C. Diagrama de flujo

( ) Conjunto de pasos ordenados que al seguirlos nos llevan a la solución de un problema.

( ) Es una forma de representar con un lenguaje común los pasos para dar solución a un problema.

( ) Es una forma de representar con símbolos la solución de un problema.

D. Diseño

E. Planteamiento

F. Pseudocódigo

TEMA 5

Expresiones y operadores aritméticos

Objetivo:

- Analizar el resultado de expresiones aritméticas utilizando la jerarquía de las operaciones.

1 Introducción

Las expresiones son un conjunto de operandos y operadores los cuales al ser evaluados generan un valor. En donde los operandos son los valores numéricos u constantes, así como las variables y los operadores son representados por los símbolos de las operaciones a realizar las cuales pueden ser aritméticas así como de asignación.

Como mencionan García-Beltrán, Martínez & Jaén (2004) que las expresiones son una parte fundamental en la solución de problemas, ya que sirven para realizar una o varias operaciones para obtener un resultado. Así pues, los operadores definen las operaciones que son realizadas dentro de una expresión.

2 Asignación

El operador de asignación permite determinar a una variable el valor de una expresión, el símbolo que se emplea en una asignación aritmética corresponde al signo de igual "=", al cual se le asigna el valor del lado derecho del igual al contenido que este en el lado izquierdo, teniendo del lado izquierdo la variable o el valor correspondiente.

Ejemplo:

x=6, se asigna el valor de 6 a x.

Aunque la asignación básica corresponde al caso del signo = (igual), la asignación se extiende según el lenguaje y el caso de la asignación con Java tenemos los siguientes casos.

Nombres	Abreviaciones	Significado
Asignación	x = y	x = y
Asignación de adicción	x += y	x = x + y
Asignación de sustracción	x -= y	x = x - y
Asignación de multiplicación	x *= y	x = x * y
Asignación de división	x /= y	x = x / y
Asignación de Resto	x %= y	x = x % y
Asignación de exponenciación	x **= y	x = x ** y
Asignación de desplazamiento a la izquierda	x <<= y	x = x << y
Asignación de desplazamiento a la derecha	x >>= y	x = x >> y
Asignación sin signo de desplazamiento a la derecha	x >>>= y	x = x >>> y
Asignacion AND	x &= y	x = x & y
Asignacion XOR	x ^= y	x = x ^ y
Asignacion OR	x |= y	x = x | y

3 Operadores aritméticos

Los operadores aritméticos son aquellos donde se obtienen datos numéricos (enteros o reales), hay dos tipos de operadores aritméticos: unitarios o binarios. Los unitarios manejan un operando, es decir devuelven el mismo valor o incremento de esté. La tabla muestra los operadores aritméticos unitarios, así como una breve descripción de estos.

Operador	Operación	Ejemplo	Descripción
+	Signo positivo	+5	5 positivo
-	Signo negativo	-4	4 negativo

++	Incremento	++3	Incrementa 3 antes de utilizar el valor
		3++	Incrementa 3 después utilizar el calor
--	Decremento	- - 2	Decrementa 2 antes de utilizar el valor
		2- -	Decrementa 2 después de utilizar el valor

Operadores utilizados en java

Los operadores aritméticos binarios son aquellos donde se manejan dos operandos (constantes o variables), además de los símbolos que representa la operación aritmética a utilizar, cuyo resultado de la operación es numérico.

Dentro de estas operaciones aritméticas básicas se encuentran: la suma, resta, multiplicación, división y modulo, esté último se utiliza para calcular el resto de una división. La tabla muestra una breve descripción de estas operaciones, así como el operador (símbolo) que lo representa.

Operador	Operación	Ejemplo	Descripción
+	Suma	15 + 2= 17	Suma dos números
-	Substracción	6-4 =2	Resta dos números
*	Multiplicación	7 * 3 = 21	Multiplica dos números
/	División	20 / 7 = 2	Divide dos números
%	Modulo	20%7 = 6	Resto de la división entera

Operadores utilizados en java

4 Jerarquía de operadores aritméticos

La jerarquización es un conjunto de reglas que establecen el orden en que deben ser efectuadas las operaciones de una expresión numérica en donde hay varias operaciones a realizar. Entonces la jerarquía de operaciones consiste en dar un orden a dichas operaciones que intervienen en las expresiones. Esté orden da la pauta para que la expresión tenga un solo valor como resultado final al realizar las operaciones que se presenten.

Por lo tanto, los operadores se evalúan de izquierda a derecha, respectando la jerarquía de estos. La tabla 3.5.3, enlista la jerarquía de los operadores aritméticos, en donde los paréntesis “()” dan mayor nivel a las expresiones al momento de ejecutarlas.

Jerarquía	Operador	Operación
1ª	( )	Paréntesis
2 ª	++, --	Incremento y decremento
3 ª	*, /, %	Multiplicación, división y modulo
4 ª	+, -	Suma y resta

Operadores utilizados en java

Ejemplo

Considera la expresión: **7+ (6-9+(4*(2+3)-1)+4)**

- Se resuelve de izquierda a derecha, la operación que se encuentra dentro del tercer paréntesis (recordemos que en java solo se consideran los **()**), por lo tanto se resuelve la suma *2+3* dando como resultado *5 quedando:*

  7+(6-9+(4*(5)-1)+4)

- Eliminados los primeros paréntesis, se realiza por orden de prioridades la multiplicación *4*(5)* dando como resultado *20*

  7+(6-9+(20-1)+4)

- Se resuelve la operación dentro de los 2dos paréntesis *(20-1)* y el resultado es *19*

  7+(6-9+(19)+4)

- Recordando que la operación se resuelve de izquierda a derecha, así como de realizar lo que esta dentro del ultimo paréntesis (6-9+19+4) se obtiene 20

  7+(20)

- Por último tendremos 7+20 de esta operación se tiene como resultado **27**

5 Evaluación de expresiones aritméticas

La evaluación de expresiones aritméticas es el realizar todas las operaciones aritméticas de la expresión y simplificar a su mínima expresión, quedando así un numero como resultado de la evaluación.

Ejemplo: Evaluar la siguiente expresión.

2+ (7-2+(4*(12+3-1)/7)+6)

- Se realiza la suma de 12+3-1 por ser las operaciones que están dentro del paréntesis más interno. Quedando como resultado 14 y entonces la expresión es:

  2+ 87-2+(4*(14)/7)+6)

- La siguiente operación a realizar es la división de (14)/7 que da como resultado 2

  2+ (7-2+(4*(2))+6)

- Se realiza la multiplicación de 4*(2) quedando como resultado 8

  2+ (7-2+(8)+6)

- Multiplicamos el 8 por +1, recuerda que cada vez que hacemos una operación se reducen los resultados haciendo las operaciones o simplificando paréntesis

  2+ (7-2+8+6)

- Se realizan las operaciones que están dentro de la llave

  2+ (19)

- Se multiplica el 19 por +1, quedando

  2+19

- Se realiza la suma de 2+19, teniendo como resultado **21**

TEMA 6

Expresiones y operadores relacionales y lógicos

Objetivo:

- Construir expresiones lógicas utilizando operadores relacionales y lógicas

1 Introducción

Las expresiones y operadores relacionales nos dan la pauta para comparar un conjunto de elementos para tener una referencia de su valor o su posición.

Estas expresiones representan relaciones entre los diferentes casos o fenómenos que nos rodean de forma común y cotidiana, estas relaciones pueden ser para comparar las características de los objetos o elementos como son el color de dos playeras, el tamaño, el precio, la comodidad, el gusto, todas estas características se comparan con la finalidad de tener una referencia entre dos elementos.

Las expresiones y los operadores lógicos buscan establecer un criterio de acción al valorar de verdadero o falso una comparación o una operación.

Apoyando estas comparaciones con un las expresiones y operadores lógicos, nos dan pauta para tomar decisiones de acción.

2 Operadores relacionales

Un operador relacional es un símbolo que se usa para comparar dos valores. El resultado de esa comparación es un valor de verdad como se muestra en la tabla, donde se presenta el operador y un ejemplo.

OPERADOR	NOMBRE	EJEMPLO	SIGNIFICADO
<	menor que	a<b	a es menor que b
>	mayor que	a>b	a es mayor que b
==	igual a	a==b	a es igual a b
!=	no igual a	a!=b	a no es igual a b
<=	menor que o igual a	a<=5	a es menor que o igual a b
>=	mayor que o igual a	a>=b	a es menor que o igual a b
<>	Distinto	A<>b	A es diferente de b

3 Operadores lógicos

Son los símbolos que representan una operación lógica y dan como respuesta un valor de verdad o valor lógico.

Los operadores lógicos son tres: AND, OR y NOT, estos tres operadores permiten que en su entrada haya uno o más valores de verdad o lógicos y den como respuesta después de hacer la operación una respuesta lógica.

El operador lógico And, permite tener dos o más valores lógicos en sus entradas y dar como respuesta un solo valor de salida. El valor de salida es verdadero si todas sus entradas son verdaderas y es falso si una o más entradas son falsas.

El operador lógico OR, permite tener dos valores lógicos en sus entradas y dar como respuesta un solo valor de salida. El valor de salida es verdadero si una o más entradas son verdadero y el falso si todas sus entradas son falsas.

El operador lógico NOT, permite un valor lógico de entrada y da como resultado un valor lógico de salida. El valor de salida es verdadero si la entrada es falsa y el valor de salida es falso si su valor de entrada es verdadero.

4 Jerarquía de operadores lógicos

La jerarquía de los operadores establece que primero se realizan las operaciones con negación (NOT), después las operaciones con multiplicación lógica (AND) y por último la suma lógica (OR).

Ejemplo:
Tenemos en particular el caso de la ecuación lógica S=1*0+0*1' al realizar las operaciones debemos primero considerar la negación teniendo 1'=0 quedando S=1*0+0*0, posterior a esto esta operación realizamos las operaciones de multiplicación lógica, El primer caso es 1*0=0 y el segundo caso es 0*0=0 teniendo la simplificación S=0+0, finalmente hacemos la suma 0+0=0 y nos queda que S= 0

5 Evaluación de expresiones lógicas

La evaluación de las expresiones lógicas consiste en asignar el valor lógico a la variable que corresponda y realizar las operaciones lógicas que corresponda.

Ejemplo:

Tomemos en particular el caso de la ecuación lógica **S=A*B+AB'** y sean los valores de **A = 0** y **B = 1**.

Al sustituir los valores en la ecuación tenemos:

S=A*B+AB'
S=0*1+0*1'
S=0*1+0*0
S=0+0
S=0

Tomando todos los casos posibles tenemos la tabla siguiente:

S=A*B+A*B’

A	B	A*B+A*B’	S
0	0	0*0+0*0’ 0*0+0*1 0+0 0	0
0	1	0*1+0*1’ 0*1+0*0 0+0 0	0
1	0	1*0+0*1’ 1*0+0*0 0+0 0	0
1	1	1*1+1*1’ 1*1+1*0 1+0 1	1

Ejercicio

Lee cuidadosamente las siguientes afirmaciones y clasifica como falso o verdadero, colocando una “V” o una “F” según corresponda.

1. Los operadores aritméticos son símbolos que representan una operación aritmética. ( )
2. Son ejemplos de operadores aritméticos +, -, *, /. ( )
3. 5+(4-(2*1)/2)+1=7 ( )
4. 5+(4-(2*4)/4)+1=7 ( )

5. 5+(4/2-(2*2)+2)+1=7 ( )
6. La expresión a>b es cierta si a=4 y b=6 ( )
7. La expresión a<b es cierta si a=4 y b=6 ( )
8. La ecuación S=A+B, con A=0 y B=1, da como resultado cero ( )
9. La negación de S, siendo S= A*B, con A=0 y B=1, da como resultado cero ( )
10. La operación S=1*0+0*0, da como resultado 1 ( )

TEMA 7

Concepto de algoritmo, diagrama de flujo y pseudocódigo

Objetivo:

- Conocer el concepto de algoritmo, diagrama de flujo y pseudocódigo

1 Introducción

Una computadora, por sí sola, no tiene la capacidad de dar solución a problemas al menos que el usuario de la secuencia adecuada de pasos a realizar, lo que significa, el desarrollo de un algoritmo adecuado, el cual siga los pasos de acuerdo a una metodología de resolución de problemas. El alumno es el usuario, el cual aprenderá y aplicará la metodología de resolución de problemas utilizando como herramienta la computadora, adquiriendo primero la capacidad de resolver problemas de manera sistemática, adecuada y de forma manual, para después construir algoritmos basados en dichas metodologías.

La resolución de problemas, así como la construcción de algoritmos, son utilizados en diferentes áreas de estudio, pero principalmente en la programación (área de computadoras). La palabra Algoritmo es de origen árabe, la cual aparece en el siglo IX y se le atribuye al matemático Al-Khawarizmi, considerando como un método de resolución de problemas matemáticos, este término lo tomaron los alemanes y transformaron en Algoritzmus por su

fácil pronunciación, traduciéndose en los diferentes idiomas, hasta llegar a la palabra Algoritmo.

Los algoritmos son una secuencia de pasos a seguir para resolver un problema determinado, independiente del lenguaje de programación a utilizar, la cual se puede representar de forma gráfica (diagrama de flujo) o descriptiva (pseudocódigo).

2 Algoritmo

"Un Algoritmo se puede considerar como una serie de pasos organizados que describen el proceso que se debe seguir para dar solución a un problema específico". (Hernández, 2010, p.8)

"Un Algoritmo es una secuencia de operaciones detalladas y no ambiguas, que, al ejecutarse paso a paso, conducen a la solución de un problema". (Balderrama, s.f, p. 44)

Por lo tanto, un algoritmo es un conjunto finito de instrucciones definidos, que permite realizar una determina actividad siguiendo una serie de pasos sucesivos. Además de ser una fórmula detallada para la resolución de problemas, donde se tienen un inicio y un fin en el proceso para la resolución del problema, el cual es independiente del lenguaje de programación a utilizar.

Tomado en cuenta que los algoritmos deben ser definidos, precisos y finitos (tiene un inicio y un fin), que representan un modelo de solución a un problema determinado. La secuencia de construcción de un algoritmo se representa en la Figura 3.7.1, donde primero se presenta un problema, a continuación, obtendremos la solución correspondiente mediante la elaboración de un algoritmo: donde esta puede ser de forma gráfica o descriptiva, para después desarrollar un programa en el lenguaje de programación de nuestra elección.

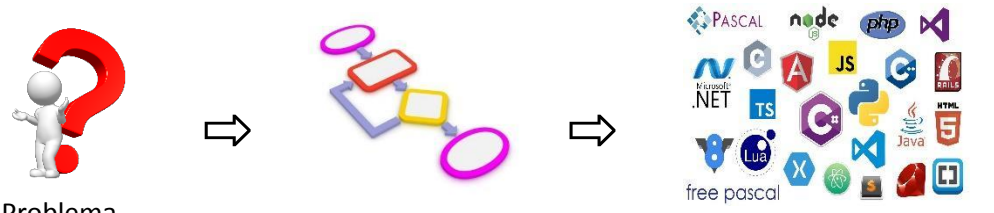


Figura 3.7.1 Construcción de un Algoritmo

3 Diagrama de flujo

Representación gráfica que describe un proceso de forma lógica, para simplificar los algoritmos. Los diagramas de flujo ayudan a la compresión e interpretación de los algoritmos, son una presentación grafica de estos. La figura 3.7.2, muestra un ejemplo de los diagramas de flujo. Algunos autores los llaman diagramas de lógica o flujogramas (muestra visual de un algoritmo).

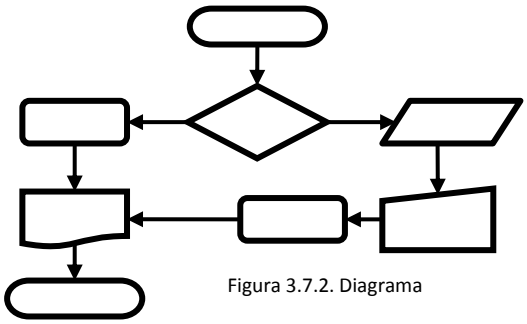


Figura 3.7.2. Diagrama

4 Pseudocódigo

Es considerado código falso, el cual es una forma simple de presentar la solución de un algoritmo, lo más detallado posible para después pasarlo al código del lenguaje de programación a utilizar. Es considerado un método formal en el que se detalla por medio de texto (sintaxis) la solución del problema.

El pseudocódigo es una representación descriptiva o narrativa de los pasos a seguir de un algoritmo, donde se utilizan palabras claras (lenguaje en español) que indican el proceso a seguir en la resolución del problema.

TEMA 8

Elaboración de algoritmos

Objetivo:

- Elaborar el algoritmo, diagrama de flujo y pseudocódigo para diferentes tipos de problemas

1 Introducción

Un algoritmo proporciona la solución general de un problema, este se puede emplear todas las veces que se presente el mismo tipo de problema o similares. Por ejemplo, el algoritmo que da la solución de una tabla de multiplicar, donde ***x*** sea el número a multiplicar se podrá

seguir empleando las veces que sean necesarias de acuerdo a los valores que se le asignen a ***x***.

2 Características de los algoritmos

Los algoritmos cumplen con tres características principales:

- ✓ Preciso: especificar el orden a realizar en cada paso del proceso.
- ✓ Definido: tener las variables definidas, que cumplirán la solución del problema de forma clara.
- ✓ Finito: siguiendo los pasos del algoritmo se llegará a la solución del problema, al fin del algoritmo, es decir, un número determinado de pasos, teniendo un inicio y un fin de este.

La construcción de un algoritmo es suficiente para obtener la solución de un problema, para la elaboración del programa correspondiente en cualquier lenguaje de programación. Si se tienen dos algoritmos que den la solución al mismo problema, se toma el algoritmo más corto, es decir el que tiene menos pasos a seguir para esta solución.

Dentro de los pasos de la resolución de problemas, en ***el paso 3. Generar la solución***: aquí es donde se desarrolla el algoritmo, ya que se crea el pseudocódigo (descripción escrita) o diagrama de flujo (símbolos gráficos), ambos son la secuencia lógica para conseguir la solución del problema. Los algoritmos tienen como fin actuar sobre los datos proporcionados por el usuario, a los que se les aplican procesos con el fin de generar un resultado. El algoritmo es realmente la representación funcional de un sistema.

Definiendo ahora al algoritmo en tres momentos: ***Inicio, Proceso y Fin***. Por ejemplo, en la realización de un jugo de naranja, conociendo la cantidad de jugo necesario, el algoritmo quedaría de la siguiente manera:

- Inicio (entrada de datos): Naranjas, vaso, exprimidor.
- Proceso: lavar las naranjas, cortarlas, (condicionado el jugo para dos personas), exprimir naranjas
- Fin (Salida de datos): Dos vasos de jugo de naranja (resultado esperado).

A este tipo de algoritmos se le conoce como informal, son procesos que se hacen cotidianamente sin tener que usar un programa computacional, procesos que lleva a cabo el ser humano en su vida diaria como: bañarse, preparar el desayuno, realizar labores en casa, ir al colegio, salir de casa, etc.

Al algoritmo que usa cálculos matemáticos para la solución de un problema se le llama algoritmo computacional, puesto que requiere un lenguaje formal y se apoya de gráficos.

Ejemplo: se emplea el lenguaje natural y las tres características principales de los algoritmos: Preciso, definido y finito.

Ejemplo:

- Obtener el promedio de dos valores a y b

Inicio

1. *Obtener el primer valor, a*
2. *Obtener el segundo valor, b*
3. *Sumar los valores de a+b*
4. *El resultado del paso 3 dividirlo entre 2*
5. *Asignar el resultado de promedio = p y visualizarlo*

Fin

Ejercicio

a. *Escribe el algoritmo para el lavado de un auto:*

b. *Escribe el algoritmo para obtener el valor de la resta de dos números:*

3 Clasificación de los algoritmos

Secuenciales

La estructura secuencial es la que lleva los pasos uno tras otro (instrucciones). Las tareas se llevan de tal forma que la salida de las instrucciones es la entrada de la otra y así sucesivamente hasta el término del proceso.

Por ejemplo, ***suma de dos números***: Considerar en el "algoritmo donde de leen dos números y los *suma", el pseudocódigo queda de la siguiente forma:*

Inicio

1. *Leer a*
2. *Leer b*
3. *Asignarle el resultado a s = a+b*
4. *Visualizar resultado "s"*

Fin

Condicionales

Se utilizan para tomar decisiones lógicas, es decir, evalúan condiciones que devuelven como resultado dos valores posibles; verdadero o falso, se ejecutan una serie de instrucciones dependiendo la condición a ejecutar.

La figura 3.8.1 muestra un ejemplo de diagrama de flujo del tipo de algoritmo condicional

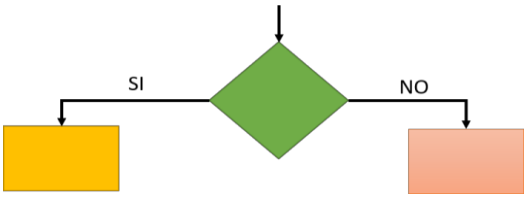


Figura3.8.1. Algoritmo condicional

Existen tres tipos de condicionales: simple, doble o anidada.

- Se quiere conocer el promedio de un alumno y si este es aprobatorio o no, teniendo como referencia que el alumno cursa 3 materias y además que el promedio mínimo aprobatorio es 6.0

Inicio

1. Solicitar las calificaciones del alumno
2. Sumar las 3 calificaciones del alumno
3. El resultado del paso 2 dividirlo entre 3
4. Si el resultado del paso 3 es mayor o igual a 6.0 entonces
    a. Escribir "Alumno aprobado"

Si no

b. Escribir “Alumno Reprobado”
Fin Si

Fin

Repetitivos

También conocidos como cíclicos, para su solución es necesario utilizar un mismo conjunto de acciones, puede ser una cantidad específica de veces. La cantidad puede ser fija (es definida), o puede ser variable (está dada por un dato dentro del mismo algoritmo). Existen tres tipos de estructuras cíclicas:

- ***While*** (mientras): La instrucción que se encuentra dentro del ciclo se ejecuta, siempre y cuando la condición sea verdadera.

Estructura: *Mientras (condición)*
Instrucción a ejecutar
Fin Mientras

Ejemplo: *Algoritmo que cuenta del 1 al 5*

Inicio
n= 1
Mientras n<=5 hacer
Escribir n
n= n + 1
Fin_Mientras
Fin

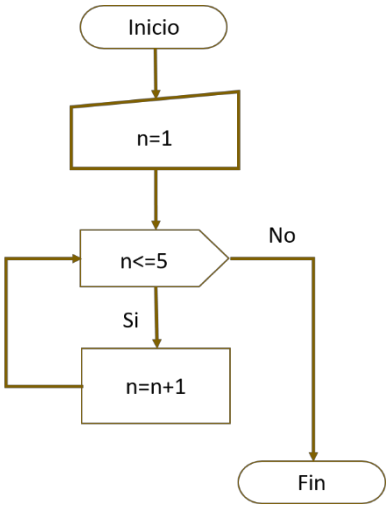


Figura3.8.2. Hacer mientras

- ***Do while*** (Hacer mientras): se repite hasta que la condición sea verdadera

Ejemplo: *algoritmo que escriba un número del 1 al 5, hasta que el número introducido se encuentre fuera del rango.*

Inicio
N: Entero
N=1
Repita
Escribir: "Escriba un número del 1 al 5"
Lea N
Hasta que (N >=1) y (N <=5)
Fin

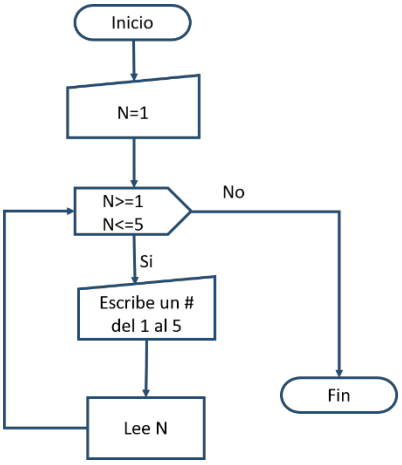


Figura3.8.3. Repetir hasta

- ***For*** (Para). Se repite el número específico de veces que se desea ejecutar las acciones del ciclo, es decir, que incrementa o decrementa hasta que la variable definida es igual a la variable del ciclo.

Inicio
Entero: N, i, Sum
Sum=0
Para(i=1 HASTA 5 Hacer)
Mostrar("Digite un número: ")
Leer(N)
Sum=Sum+N
Fin_Para
Mostrar ("La suma es:", Sum)

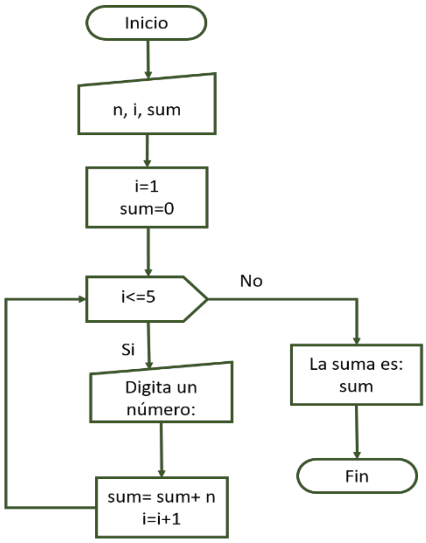


Figura3.8.4. For

Ejercicios

1. *Realiza un algoritmo que determine si un automóvil cometió una infracción por velocidad, si en periférico el límite de velocidad es de 80 km/h.*

TEMA 9

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

Objetivo:

- Elaborar el algoritmo, diagrama de flujo y pseudocódigo para diferentes tipos de problemas

1 Introducción

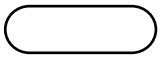
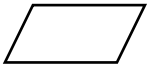
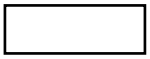
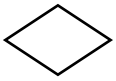
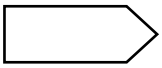
Un algoritmo se puede representar de diferentes formas, desde el lenguaje natural (describirlo verbalmente), apoyado de un pseudocódigo (expresarlo de un lenguaje formal), o diagramas de flujo (representación gráfica), para que finalmente sea presentado en el lenguaje de programación a utilizar. El uso del pseudocódigo y el diagrama de flujo ayudan evitar ambigüedades al utilizar el lenguaje natural, al expresar la solución del problema.

Para representar un algoritmo se consideran datos que representan símbolos numéricos y/o letras que representa una variable o característica particular dentro del algoritmo que dará un significado dentro de la solución del problema.

2 Elementos que componen un diagrama de flujo

Un diagrama de flujo utiliza la siguiente simbología para su interpretación gráfica de un algoritmo. La tabla 1 muestra el listado de símbolos utilizados en la creación de diagramas, normalizados por el instituto norteamericano (ANSI).

Tabla1. Simbología del diagrama de flujo

Símbolo	Nombre símbolo	Descripción
	Inicio o fin	Representa el inicio o fin del diagrama o proceso
	Ingreso de datos	Ingreso de datos (definición de variables)
	Ingresar datos	Entra de datos por teclado, introducción de datos por el usuario.
	Proceso	Representación del proceso u operación a realizar.
	Decisión	Usada para evaluar una condición dentro del proceso, así como darle solución a esta. Generalmente se tienen dos caminos “Si” o “No”
	Conector	Conexión con en diagrama (dentro de la misma página)
	Conector de página	Representa una conexión entre otra hoja donde sigue el diagrama.
	Línea de flujo	Indica la dirección de dónde va el proceso
	Repetición	Proceso que se repite según la condición a realizar.
	Imprimir datos (salida de datos)	Impresión del resultado o datos del proceso.

3 Ejemplos de diagramas de flujos

- Algoritmo que realiza la suma de dos números enteros, introduciendo los datos desde el teclado por el usuario.

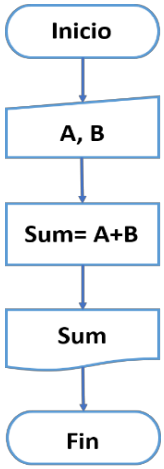
Pseudocódigo	Diagrama
Inicio 1. Ingresar A 2. Ingresa B 3. Realizar Sum=A+B 4. Escribir Sum Fin	

Figura3.9.1. Suma de dos números

Prueba de escritorio

	A	*B*	*A+B*	*Sum*
Ingresar A	7			
Ingresar B		6		
Realizar Sum			7+6	
Sum				13

- Algoritmo que calcula el área de un triángulo

Pseudocódigo	Diagrama
Inicio 1. Ingresar la base, b 2. Ingresa la altura, h 3. Asignar los valores b, h 4. A=(b*h)/2 5. Mostrar: “El área del triángulo es: “ A Fin	

Prueba de escritorio

	Entrada	*Calculo*	*Salida*
Ingresar base	4		
Ingresar altura	3		
Multiplicación		12	
División		12/2	
Área			El área del triángulo es: 6

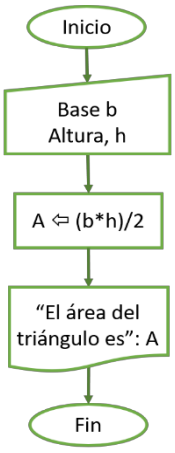


Figura3.9.2. Área de un Triángulo

- Algoritmo para determinar el promedio de un alumno e indicar si aprobó o no el semestre. Se consideran 3 calificaciones obtenidas en el semestre, el alumno aprueba si la calificación es mayor o igual que 6.0.

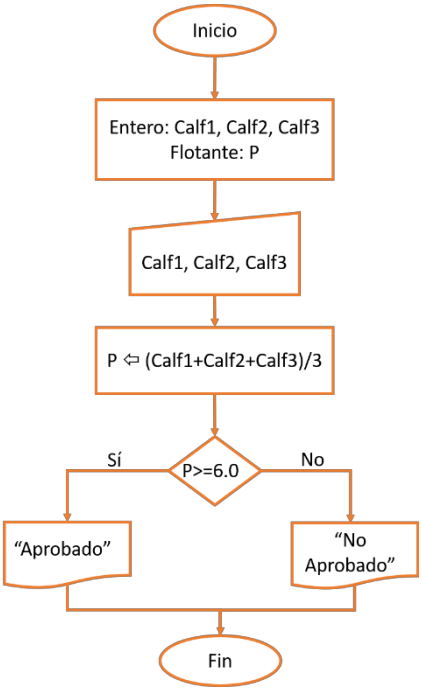
Pseudocódigo	Diagrama
Inicio 1. Entero: Calf1, Calf2, Calf3 2. Flotante: P 3. Introducir Calf1, Calf2, Calf3 4. P =(Calf1+Calf2+Calf3)/3 5. **Si**(P>=6.0) **entonces** Escribir "Aprobó" **6. Sino** Escribir "No Aprobado" **7. Fin_Si** Fin	

Figura3.9.3. Promedio de 3 calificaciones y escribir si aprobó o no

Prueba de escritorio

	Entrada	*Calculo*	*Salida*
Introduce Calf 1	5		
Introduce Calf 2	6		
Introduce Calf 3	5		
Promedio		(5+6+5)/ 3	
P		5.33	

P		5.33<6.0	
Si P >= 6.0		No	No Aprobado

- Algoritmo que determina si una persona es mayor de edad o no.

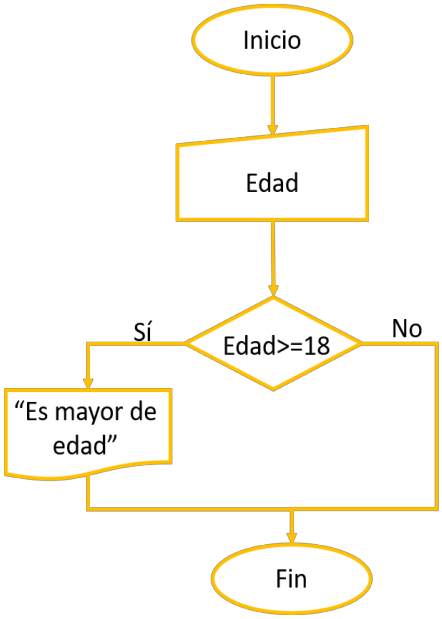
Pseudocódigo	Diagrama
Inicio 8. Entero: Edad 9. Ingresa tú edad: Edad 10. Leer Edad 11. **Si** (Edad>=18) **entonces** Escribir "Es mayor de edad" **12. Fin_Si** Fin	

Figura3.9.4. Mayor de edad

Prueba de escritorio

	Ingresa edad	*Edad*	*Si Edad >=18*
Entrada	16		
Lectura		Edad =16	
Condicional			16<18 Falso sale del programa
Entrada	24		
Lectura		Edad =24	
Condicional			Verdadero Es mayor de edad

Ejercicios

Realiza el algoritmo que calcule el área de una circunferencia, acompañado del pseudocódigo, diagrama de flujo, así como de la prueba de escritorio.

Algoritmo *Pseudocódigo*

Prueba de escritorio

Actividades que refuerzan el aprendizaje

1. Realiza un algoritmo para la elaboración de un pastel, dibuja su diagrama de flujo correspondiente.
2. Realiza el algoritmo para exprimir el jugo de dos limones.
3. Elabora el diagrama de flujo para exprimir el jugo de dos limones.
4. Escribe pseudocódigo que te permita escribir los pasos para salir de viaje a la playa
5. Realiza el algoritmo para las tablas de multiplicar, pidiendo al usuario el número de la tabla de multiplicar a realizar.
6. Realizar el diagrama de flujo de la tabla de multiplicar, así como el pseudocódigo.
7. Realiza pseudocódigo, diagrama de flujo y prueba de escritorio de un algoritmo que calcule el área de un polígono regular.
8. Realiza el algoritmo de la división de dos números enteros.

Ejercicio

1. Son la series de pasos a seguir para la solución de un problema determinado:
   A) Diagrama B) Pseudocódigo C) Algoritmo D)Estrategia

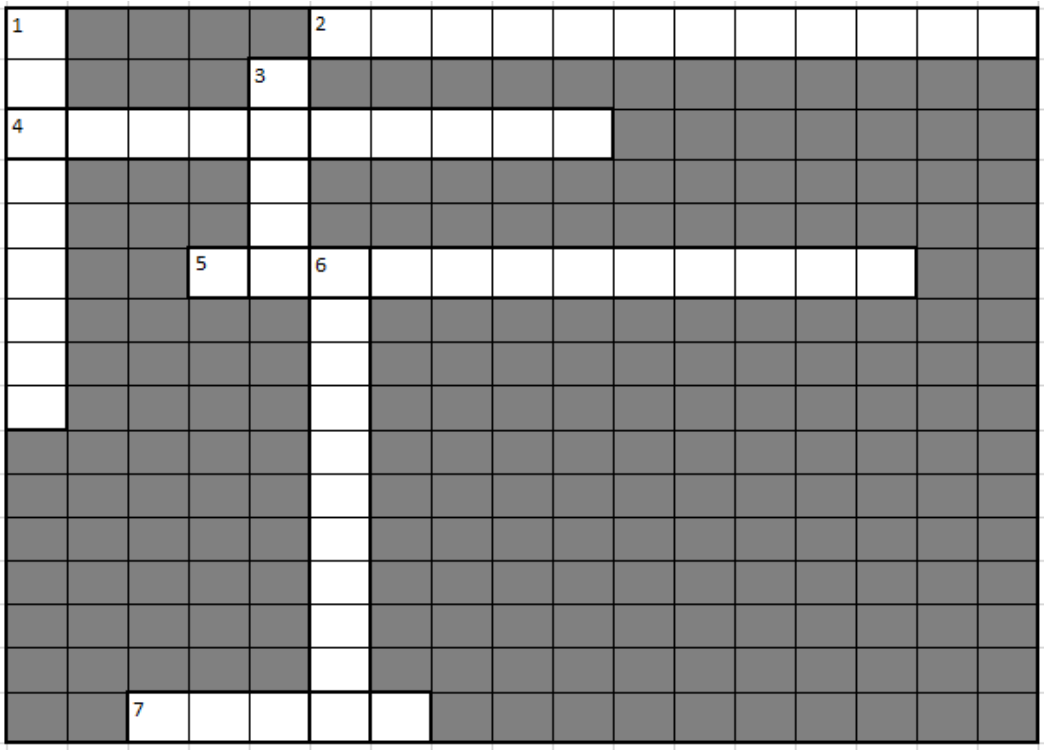
2. También llamado falso código, es una representación de los algoritmos de manera formal lo más detallado posible.
   A) Diagrama B) Algoritmo C) Flujogramas D)Pseudocódigo

3. Es un ejemplo de los algoritmos repetitivos o cíclicos
   A) Hacer mientras B) Secuencial C) Condicionales D)Flujogramas

4. Son parte de la simbología de los diagramas de flujo, **EXCEPTO:**
   A)  B   

5. En que paso de la solución de problemas se encuentra el desarrollo de algoritmos:
   A) Paso 1 B) Paso 4 C) Paso 3 D) Paso 2

Resuelve el siguiente crucigrama



Horizontal

2. Representación descriptiva o narrativa de los pasos a seguir de un algoritmo
4. Conjunto de instrucciones, definida por una serie de
5. Se ejecutan las instrucciones una tras otra.
7. Se ejecuta siempre y cuando la condición sea

Vertical

1. Es al representación grafica de un algoritmo
3. La palabra *algoritmo* es de origen...
6. Devuelve dos posibles resultados: verdadero o falso

TEMA 10

Lenguaje de Programación en Java

Objetivo:

- Resolver problemas elaborando su diseño de solución (algoritmo) y su programa correspondiente en lenguaje de programación Java.

1 Introducción

Java es un lenguaje de programación muy poderoso y nos permite elaborar cualquier tipo de programa. Fue desarrollado por la compañía Sun Microsystems y está orientado para dar soporte y responder a las necesidades de programación actuales.

Una de las características más importantes de Java es que es independiente de la plataforma. Esto significa que no depende de algún procesador específico, esto es, un programa realizado en Java puede ejecutarse en cualquier computadora. Java es un lenguaje muy seguro pues los desarrolladores liberan actualizaciones que corrigen o previenen los posibles problemas conforme se van detectando. Además, permite que otras compañías desarrollen librearías, en esto reside la capacidad y universalidad del lenguaje.

2 Historia del lenguaje

A finales de 1990 Patrick Naughton reclutó a varios colegas entre ellos James Gosling y Mike Sheridan para trabajar sobre un nuevo proyecto conocido como "El proyecto verde". Con la ayuda de otros ingenieros, empezaron a trabajar en una pequeña oficina en Sand Hill Road en Menlo Park, California, y aunque trabajaban para Sun Microsystems, formaron un grupo apartado de la compañía y trabajaron sin descanso durante 18 meses.

El objetivo era desarrollar un lenguaje de programación fácil de utilizar, compatible con la nueva tecnología que permitiera programar la siguiente generación de dispositivos inteligentes, en los que Sun veía un campo nuevo a explorar.

Inicialmente se consideró C++ como el lenguaje a utilizar, pero tanto Gosling como Bill Joy, lo encontraron inadecuado. Gosling intentó varias modificaciones, pero finalmente decidió crear un nuevo lenguaje al que llamo Oak (roble en inglés, por el roble que veía por la ventana de su despacho).

Oak tenía similitudes con C, C++ y Objetive C y no estaba ligado a un tipo de procesador, más tarde, se cambiaría el nombre de Oak a Java, existen diferentes versiones sobre el nombre del lenguaje, una de ellas, supone 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.

Bajo la marca Proyecto verde se desarrollan los prototipos de bajo nivel del sistema, incluyendo sistema operativo, Green OS; el lenguaje Oak, las librerías, alguna aplicación básica y el hardware, hasta que el 3 de septiembre de 1992 se termina el desarrollo y con ello el Proyecto Verde.

De 1993 a 1994, el equipo de Patrick Naughton se lanzó en busca de nuevas oportunidades en el mercado, La incipiente subsidiaria fracasó en sus intentos, por lo que el equipo concluyó que el mercado para sus productos no era apropiado. La subsidiaria Proyecto verde fue liquidada por la compañía Sun a mediados de 1994.

El cierre del Proyecto Verde coincidió con la aparición de la WEB, Sin embargo, el trabajo desarrollado que habían realizado era compatible con este nuevo ambiente.

Patrick Naughton procedió al desarrollo del lenguaje de programación Java, así el 29 de septiembre de 1994 se termina el desarrollo del prototipo de HotJava. Después de la demostración a los ejecutivos de Sun, se reconoce el potencial de Java y se acepta el proyecto.

Una de las características de HotJava fue su 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.

El 23 de mayo de 1995, en la conferencia SunWorld `95, John Gage, de Sun Microsystems, y Marc Andreessen, cofundador y vicepresidente de Netscape, anuncian la versión alpha de Java, que 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. Como parte de su estrategia de crecimiento mundial, para favorecer la promoción de la nueva tecnología, Java Soft otorgó permisos a otras compañías para que pudieran tener acceso al código fuente y al mismo tiempo mejorar sus navegadores. También les permitía crear herramientas de desarrollo para programación Java y los facultaba para acondicionar máquinas virtuales Java (JVM), a varios sistemas operativos.

En poco tiempo prestigiosas firmas como: IBM, Microsoft, Symantec, Silicon Graphics, Oracle, Toshiba y Novell gozaban de licencias o permisos de Java.

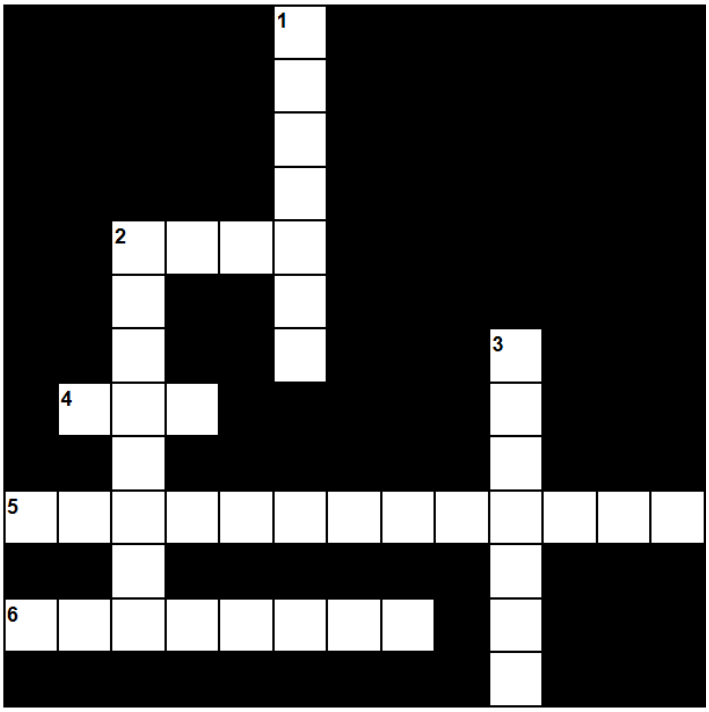
Los apples Java (basados en JDK 1.02) fueron apoyados por los dos navegadores más populares en ese momento, Nestcape Navigator 3.0 y Microsoft Internet Explorer 3.0.

Los nuevos proyectos de Java son copatrocinados por cientos de millones de dólares en capital disponible de recursos tales como la Fundación Java, un fondo común de capital formado por 11 compañías, incluyendo Cisco Systems, IBM, Netscape y Oracle.

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 en Wall Street hasta juegos de uso doméstico y tarjetas de crédito: Java está en todas partes.

Ejercicio

Instrucciones. Crucigrama. Revisa las definiciones, escribe su respuesta en el número correspondiente de forma vertical u horizontal. Las respuestas que cuentan con más de una palabra escríbelas seguidas (sin espacio).



Horizontales

2 Nombre del lenguaje de programación que se deriva de James Gosling, Arthur Van y Andy Bechtolsheim.

4 Primer lenguaje de programación como antecedente de Java.

5 Nombre del proyecto generado por Sun Microsystem para generar un nuevo lenguaje de programación compatible con la nueva tecnología (sin espacio).

6 Primer Navegador en el que se incorpora Java.

Verticales

1 Primer prototipo que soporta applets (sin espacio).

2 Nombre de la empresa creada para dedicarse exclusivamente para crear aplicaciones y herramientas para aumentar las capacidades del lenguaje (sin espacio).

3 Sistema Operativo desarrollado bajo el mismo proyecto (sin espacio).

3 Características de Java

Características de Java

Java es un lenguaje de programación de propósito general orientado a objetos, fue creado con la finalidad de eliminar la complejidad de otros lenguajes de programación, esto lo hace sencillo, pero a la vez es robusto; por la máquina virtual que provee, se vuelve multiplataforma, además permite realizar diferentes tipos de aplicaciones, esto permite que se pueda hacer uso de un amplio conjunto de bibliotecas. Asimismo, es interpretado y compilado a la vez.
A continuación, se describen cada una de sus principales características.

(Tipos de aplicaciones) Propósito general. Este lenguaje de programación permite realizar diferentes tipos de programas. Los programas en Java se pueden clasificar en:

- Applets. Son pequeños programas que se incorporan en una página Web y que, por lo tanto, necesitan de un Navegador Web compatible con Java para poder ejecutarse. Los applets se descargan junto con una página HTML desde un Servidor Web y se ejecutan en la máquina cliente.

- Aplicaciones. Son programas standalone (Programa que puede ser ejecutado en cualquier computadora sin necesidad de ser instalado) de propósito general que normalmente se ejecutan desde la línea de comandos del sistema operativo.

- Servlets. Al contrario de los applets son programas que están pensados para trabajar en el lado del servidor y desarrollar aplicaciones Web que interactúen con los clientes.

Arquitectura Neutral. Java es un lenguaje de arquitectura neutral sería porque puede ejecutarse en cualquier tipo de plataforma, esto lo vuelve ideal para desarrollar aplicaciones

en internet porque los programas podrían ejecutarse en todos los tipos de computadoras conectadas a la red.

Las aplicaciones en Java pueden ser ejecutadas en diferentes plataformas sin necesidad de realizar cambios en el código fuente y sin necesidad de volver a compilar el programa, esto es, "compila una vez y ejecuta en cualquier plataforma, a esta característica se le conoce como multiplataforma. El compilador de Java genera bytecodes, código intermedio entre el código fuente y el código máquina, lo que permite transportar el código eficientemente a múltiples plataformas de hardware y software. El resto de los problemas los soluciona el intérprete de Java.

Orientado a objetos. Java fue diseñado como un lenguaje orientado a objetos, trabaja con sus datos como objetos y con interfaces a ellos. Los objetos agrupan en estructuras encapsuladas tanto sus datos como los métodos (funciones) que manipulan esos datos; soporta tres características de los lenguajes orientados a objetos encapsulamiento, herencia y polimorfismo.

Las plantillas de objetos son llamadas clases y tienen sus propias instancias. Las clases en java tienen una representación en tiempo real (runtime)**.**

Sencillo. Con el uso de este lenguaje se reducen errores comunes en la programación orientada a objetos, porque se dejan de utilizar las referencias, macros, definición de tipos y además se libera memoria.

Robusto. Java fue diseñado para crear software altamente fiable, puede realizar verificaciones en busca de problemas en compilación y en tiempo de ejecución. La comprobación de tipos en java ayuda a detectar errores en el ciclo de desarrollo y la recolección de basura elimina la necesidad de liberación explícita de memoria.

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

Disponibilidad de un amplio conjunto de bibliotecas. La programación en Java se basa no solo en el empleo de instrucciones que componen el lenguaje, sino, fundamentalmente, en la posibilidad de utilizar el amplio conjunto de clases (plantillas para la creación de objetos) que Sun y su comunidad pone a disposición del programador y con las cuales se pueden realizar cualquier tipo de aplicaciones.

Interpretado y compilado a la vez. Java puede ser compilado e interpretado en tiempo real, ya que cuando se construye el código fuente este se transforma en una especie de código de máquina. Java es compilado, en la medida en que su código fuente se transforma en código de máquina. Por otra parte, es interpretado, ya que los bytecodes se pueden ejecutar directamente sobre cualquier máquina, a la cual se ejecuten el intérprete y el sistema, en tiempo real (run-time).

Ejercicios

Instrucciones. Relaciona las columnas. Coloca dentro del paréntesis el número correspondiente del concepto a su definición.

(____) Programas que se incorporan en una página Web. Estos se ejecutan en la máquina del cliente.
(____) Programa que se ejecuta desde el servidor para interactuar con (los clientes) la computadora del usuario.
(____) Por esta característica las aplicaciones Java se pueden ejecutar desde cualquier plataforma (SO)
(____) De esta característica permite hacer uso del encapsulamiento, herencia y polimorfismo.
(____) Con ello se apreció la reducción de errores y liberación de memoria.
(____) Por esta característica se crea software altamente fiable.
(____) Establece o acepta conexiones con los servidores o clientes remotos.
(____) Es posible hacer uso de clases que Sun y su comunidad realizan.
(____) Se realiza cuando el código fuente de un programa se transforma en código máquina.
(____) Por esta característica es ejecutado directamente en cualquier máquina.

1. Applets
2. Arquitectura nuetral
3. Compilado
4. Disponibilidad de bibliotecas
5. Distribuido
6. Interpretado
7. Orientado a Objetos
8. Robusto
9. Sencillo
10. Servlets

4 Entorno de desarrollo del lenguaje de programación Java

Para la escritura de aplicaciones y applets de Java se necesitan herramientas de desarrollo como el Java Development Kit o JDK, este es un tipo de software que provee herramientas de desarrollo para la creación de programas en Java, es un conjunto de herramientas (programas y librerías) que permiten desarrollar (compilar, ejecutar, generar documentación, etc.) programas en lenguaje Java. Existe una versión disponible del JDK para los diferentes sistemas operativos, su distribución es gratuita.

IDE. El entorno de desarrollo integrado, (IDE, por sus siglas en inglés: Integrated Development Environment) de java se encuentra en diferentes programas como BlueJ. jGrasp, Eclipse, NetBeans, entre otros, los cuales permiten editar, compilar y ejecutar aplicaciones de java.

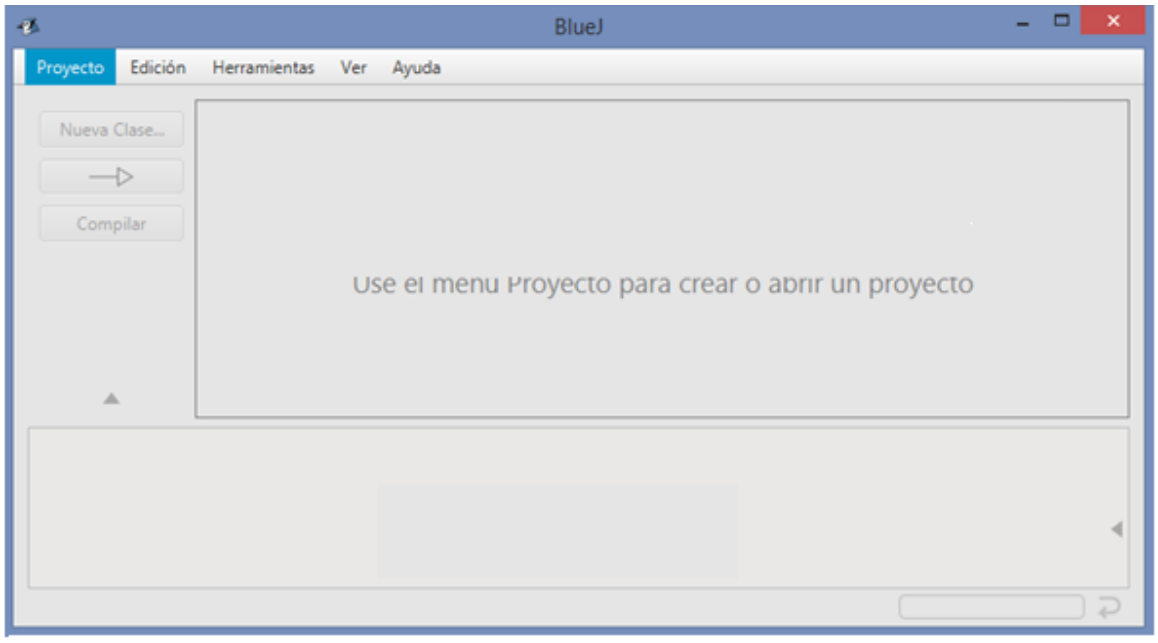
En este material se explicará el entorno de trabajo de BlueJ.

BlueJ es un sencillo entorno de programación exclusivamente diseñado para la enseñanza y el aprendizaje de Java. Se trata de un proyecto nacido en el seno de un grupo de investigación universitario integrado por miembros británicos y australianos.

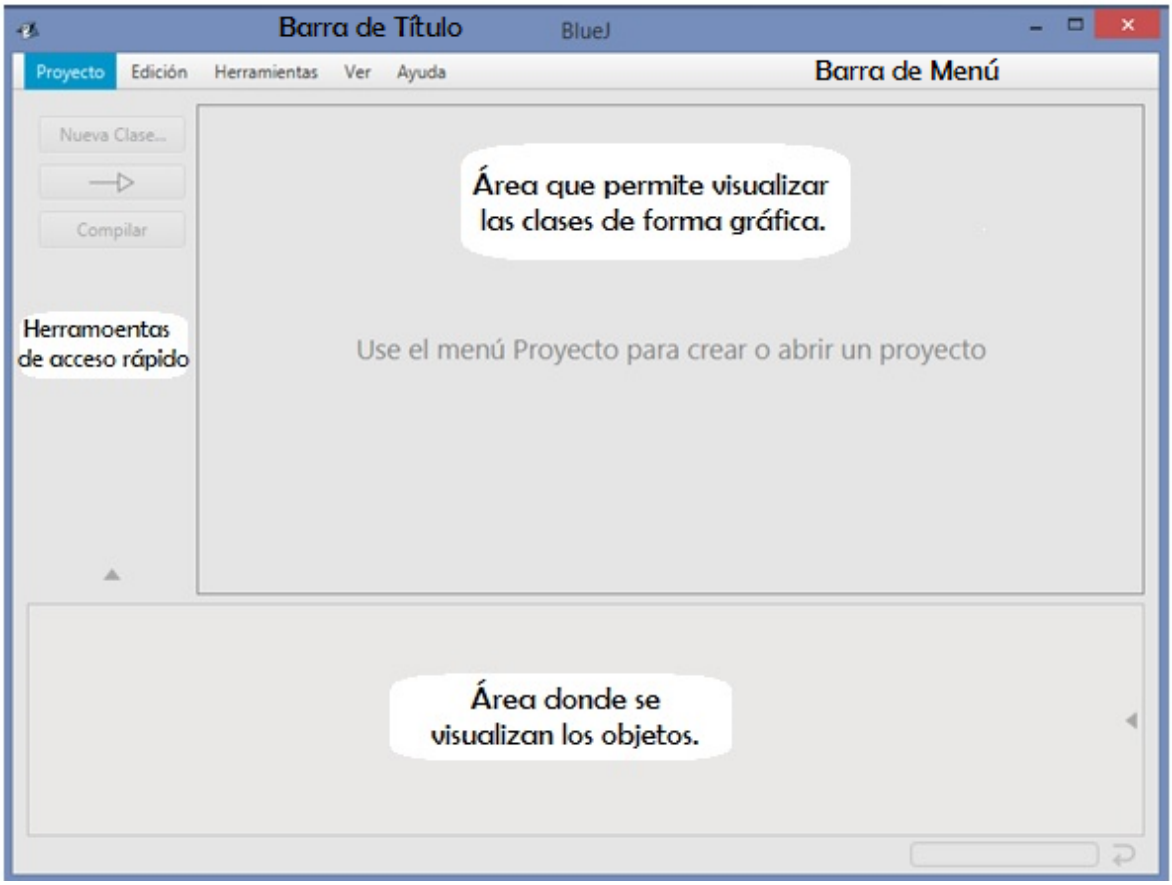
Se puede descargar de la página https://www.bluej.org/

Pantalla de BlueJ

Al abrir por primera vez este programa se visualiza la siguiente ventana.



En donde encontramos los siguientes elementos.

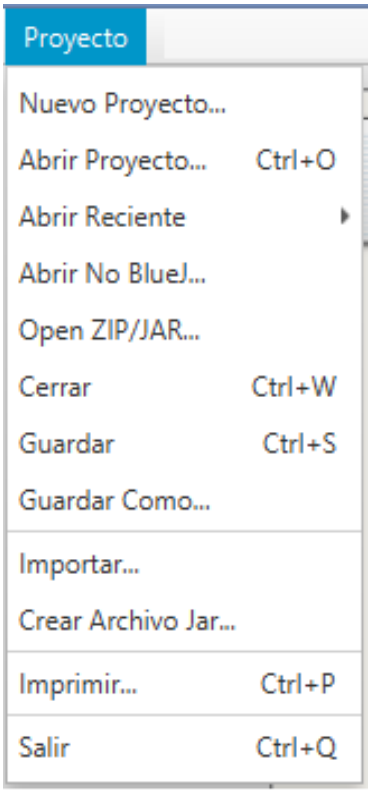


Barra de Título

En esta barra se presenta el nombre del programa (BlueJ) y el nombre del proyecto.

Barra de Menú

Esta barra contiene los diferentes menús a los que se puede tener acceso. A continuación, se explican brevemente, los comandos que se utilizarán en este apartado de cada menú.



Menú Proyecto - Comandos

Nuevo Proyecto. Al seleccionar este comando se crea un nuevo proyecto al cual se le asigna un nombre, esto generará una carpeta con ese nombre 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.

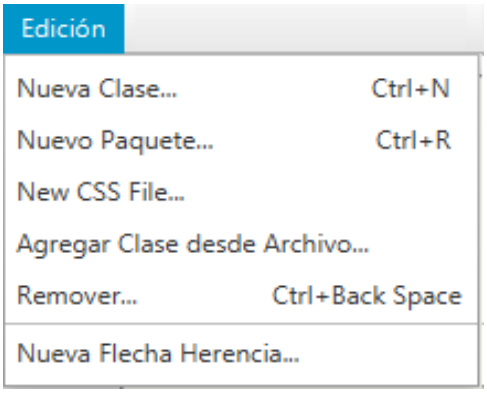
Abrir Reciente. Permite abrir un proyecto de una lista de los proyectos usados recientemente.

Cerrar. Cierra el proyecto en uso.

Guardar. Almacena la última versión del proyecto en la misma localidad donde se encuentra el original.

Guardar como. Almacena una copia del proyecto con otro nombre.

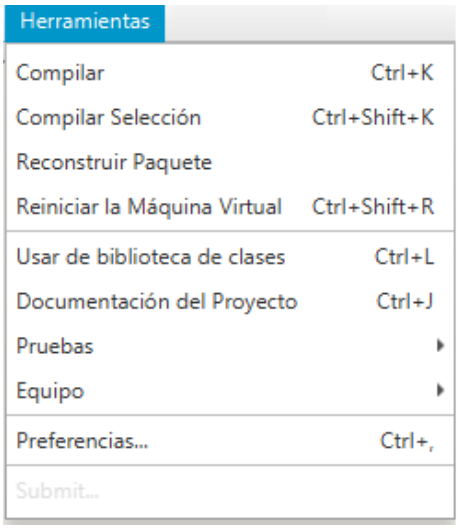
Salir. Cierra el proyecto en uso y termina el programa BlueJ.



Menú Edición - Comandos

Nueva clase. Este comando permite crear una Nueva clase solicitando el nombre que se asignará al archivo java

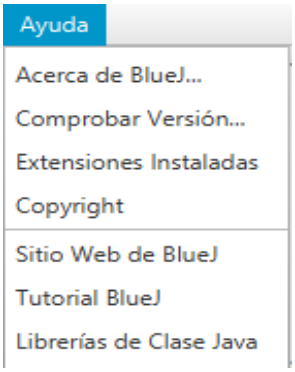
Remover. Este comando permite eliminar el elemento seleccionado.



Menú Herramientas – Comandos

Compilar. Este comando compila el programa fuente.

Compilar selección. Con este comando se compila sólo la selección

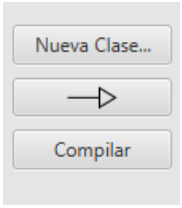


Menú Ayuda - Comandos

Acerca de BlueJ. Mostrará la información del programa y su versión.

Tutorial BlueJ. Abre la documentación de BlueJ en Internet.

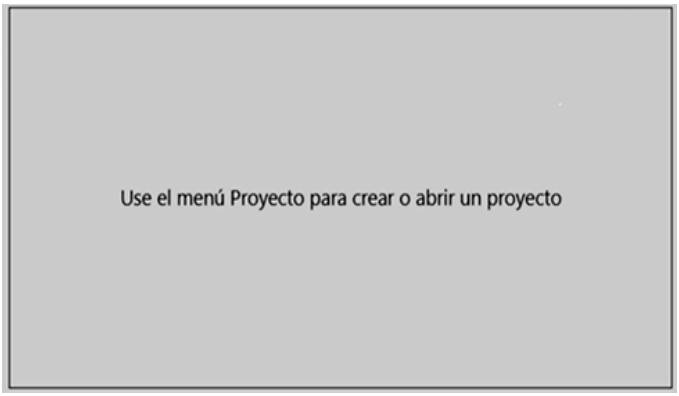
Herramientas de acceso rápido



En esta área contamos con botones que nos permiten crear una Nueva Clase, Insertar una relación de herencia y compilar el programa.

Área que permite visualizar las clases de forma gráfica

Al abrir por primera vez BlueJ el aspecto de esta área es:



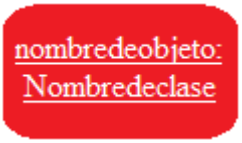
Al crear un proyecto, lo primero que aparece en esta área es un icono que representa su documentación.

Al generarse una nueva clase aparece el icono correspondiente en esta área.

Resultando.

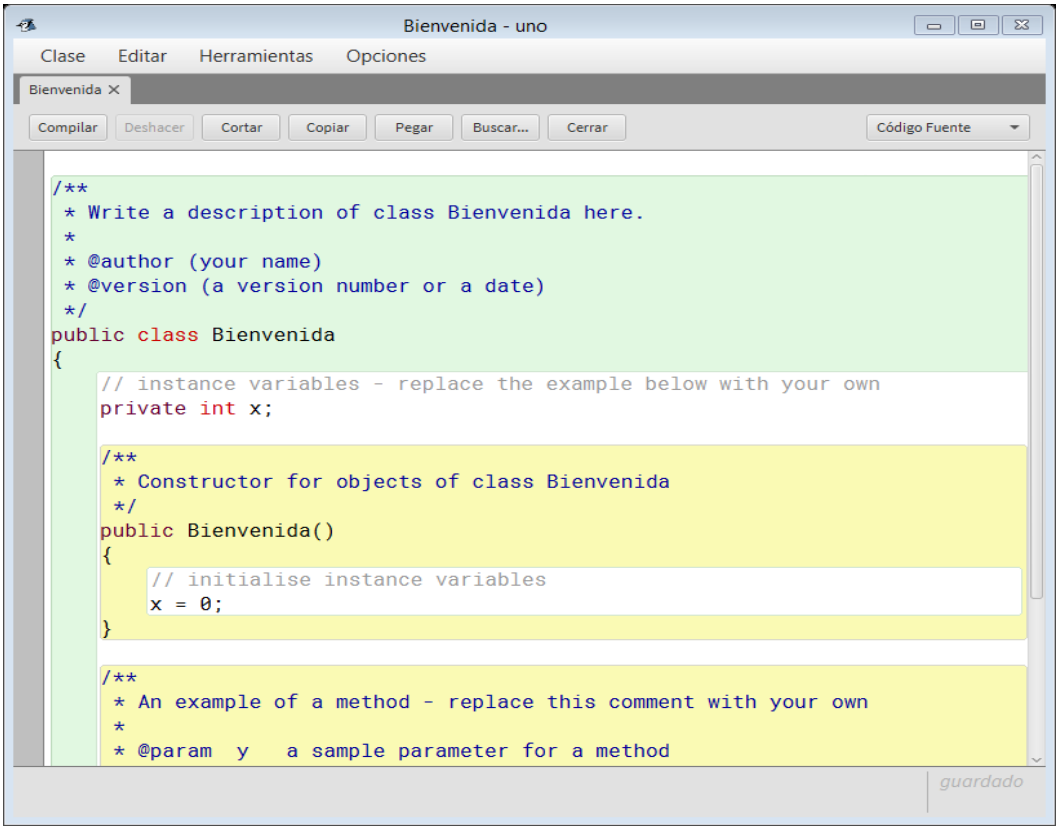
Área donde se visualizan los objetos

Esta área refleja los objetos que se crean con su icono correspondiente.



Ventana editora de código

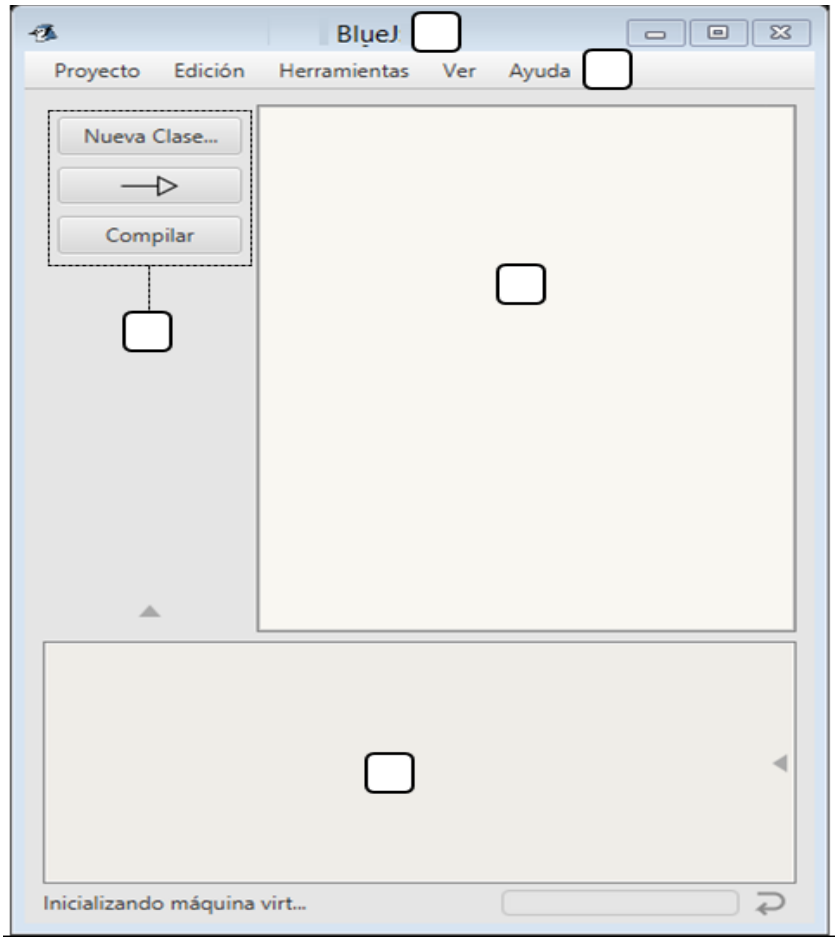
Para empezar a editar el código de un programa, al icono de la nueva clase creada o a la clase ya creada se le puede dar doble clic, o clic derecho y se selecciona el comando Abrir, a continuación, se visualiza la siguiente ventana:



La cual contiene el código predefinido por el programa, este código se puede editar si es necesario.

Ejercicio

I. **Instrucciones.** Identifica las partes del IDE de BlueJ, coloca la letra correspondiente al elemento de la ventana.



a) Área para visualizar las clases de forma gráfica
b) Barra de menú
c) Herramientas de acceso rápido
d) Barra de título
e) Área para visualizar los objetos

II. **Instrucciones.** Selecciona la respuesta correcta.

1. Conjunto de herramientas (programas y librerías) que permiten compilar, ejecutar, generar documentación, etc., de código de programas de computadora.

a. ( ) Compilador
b. ( ) IDE
c. ( ) Constructor
d. ( ) Editor

2. De las siguientes opciones, ¿cuál NO corresponde a un entorno de desarrollo de Java?

a. ( ) BlueJ
b. ( ) jGrasp
c. ( ) Eclipse
d. ( ) Xcode

3. _____ es el proceso de traducción de un código fuente (escrito en un lenguaje de programación de alto nivel) a lenguaje máquina (código objeto) para que pueda ser ejecutado por la computadora.

a. ( ) Interpretar
b. ( ) Ejecutar
c. ( ) Compilar
d. ( ) Importar

4. _____ se compone de archivos .java, archivos .class y documentación.

a. ( ) Un programa
b. ( ) Un sistema
c. ( ) Un proyecto
d. ( ) Una Clase

5. BlueJ es el nombre de un:

a. ( ) IDE
b. ( ) Compilador
c. ( ) Editor de texto
d. ( ) Interprete

TEMA **11**

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

Objetivo:

- Conocer el ambiente de trabajo y realizarás tus primeros programas.

1 Introducción

El objetivo de este tema es dar a conocer los elementos básicos que componen un programa desarrollado en el lenguaje de programación Java, iniciaremos con un programa muy sencillo, en donde el objetivo es que el alumno aprenda a editar y compilar un programa utilizando el editor Bluej, después se mostrarán las sentencias para leer datos, los tipos de variables, las sentencias condicionales y finalmente los ciclos repetitivos.

2 Declaración de la clase

En Java, se representan conceptos de la vida real como objetos con propiedades o atributos (características que describen al objeto) y métodos (acciones asociadas al objeto, es decir, operaciones que puede realizar y a las que puede responder el objeto).

El código de un programa Java se puede editar desde cualquier editor de texto, el block de notas, por ejemplo. Para trabajar con Java existen varios IDE's, nosotros utilizaremos BlueJ, siendo una herramienta dedicada especialmente para la enseñanza, además es libre, esto es, no tiene costo.

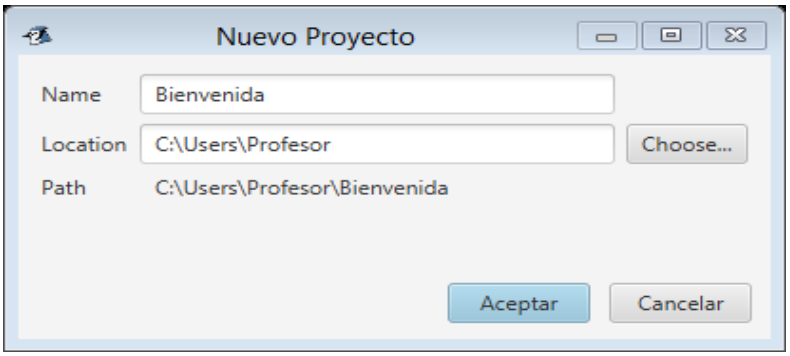
A continuación, se describe el procedimiento para crear el primer programa en Java con BlueJ. Se describirá con el ejemplo de la salida del mensaje "Bienvenidos al mundo de Java"

Ejemplo 1. ***Mi primer programa en Java***

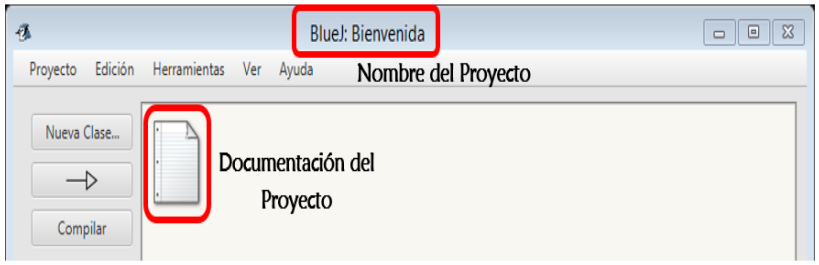
Creación de un proyecto

1. Se abre el programa BlueJ, nos dirigimos al menú **Proyecto** y seleccionamos *Nuevo Proyecto*, enseguida aparecerá la siguiente ventana, en donde nos solicita el nombre del proyecto

   En este ejemplo escribiremos *Bienvenida,* que corresponde al nombre del proyecto y en *Location* se indica la ubicación.



Se da clic en el botón Aceptar, y en la ventana aparecen los siguientes cambios.



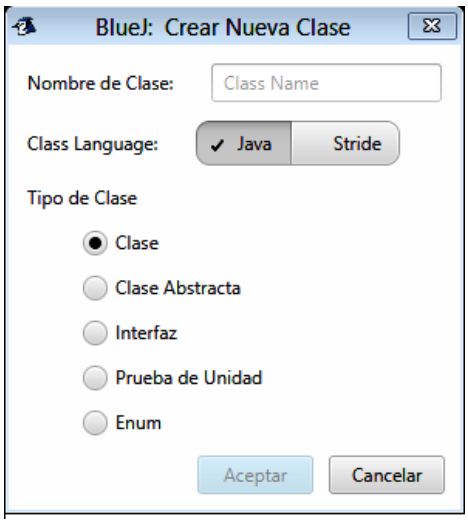
Creación de una clase

Una aplicación se compone, al menos, de un archivo.class (una clase) y el mismo debe contener como mínimo el punto de entrada de la aplicación, el método main().

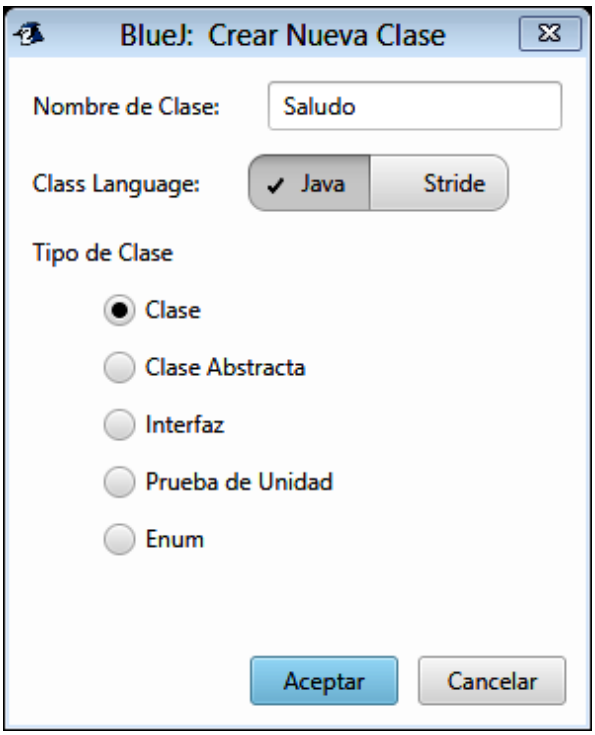
Por lo anterior, se debe crear una clase, esto se puede realizar de dos formas.

a. Dirigirse al menú Proyecto y dar clic en Nueva Clase
b. Dar clic en el botón Nueva Clase... de la Barra de herramientas de Acceso Rápido

Se muestra la siguiente ventana:

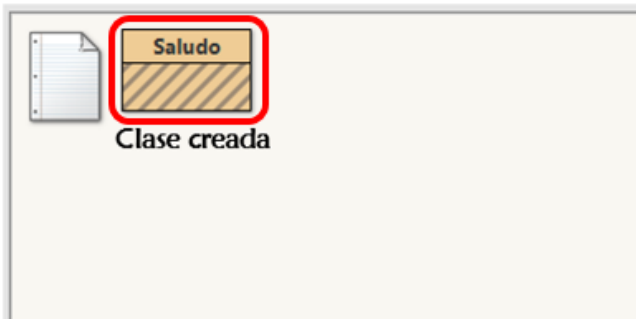


En el recuadro *Nombre de Clase*, escribimos Saludo



En tipo de Clase se deja la que está seleccionada y se da clic en el botón Aceptar.

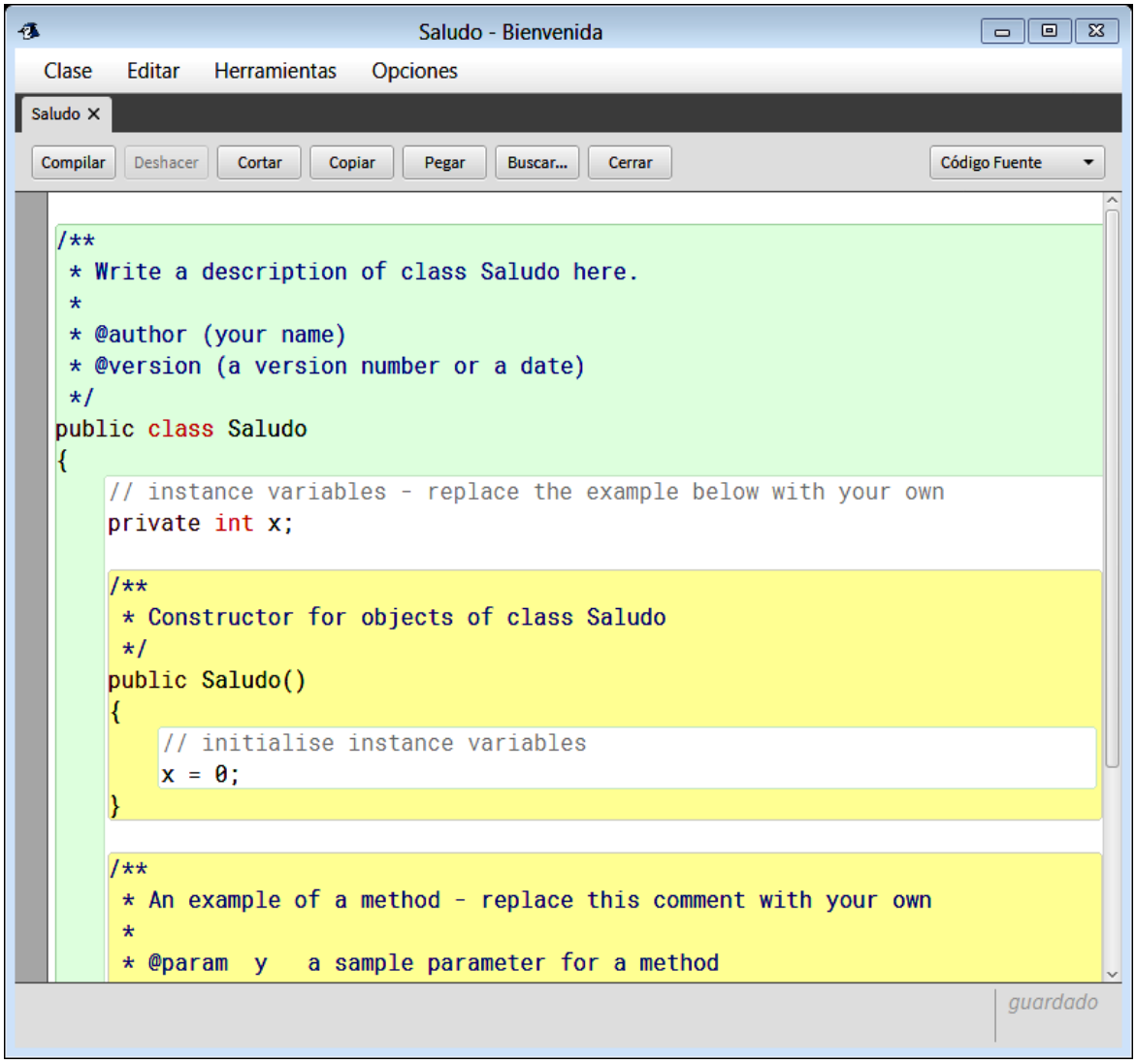
A continuación, se muestra la pantalla de la vista gráfica de clases.



Enseguida se editará el programa. Para entrar al Editor de Código se pueden realizar cualquiera de las dos acciones siguientes:

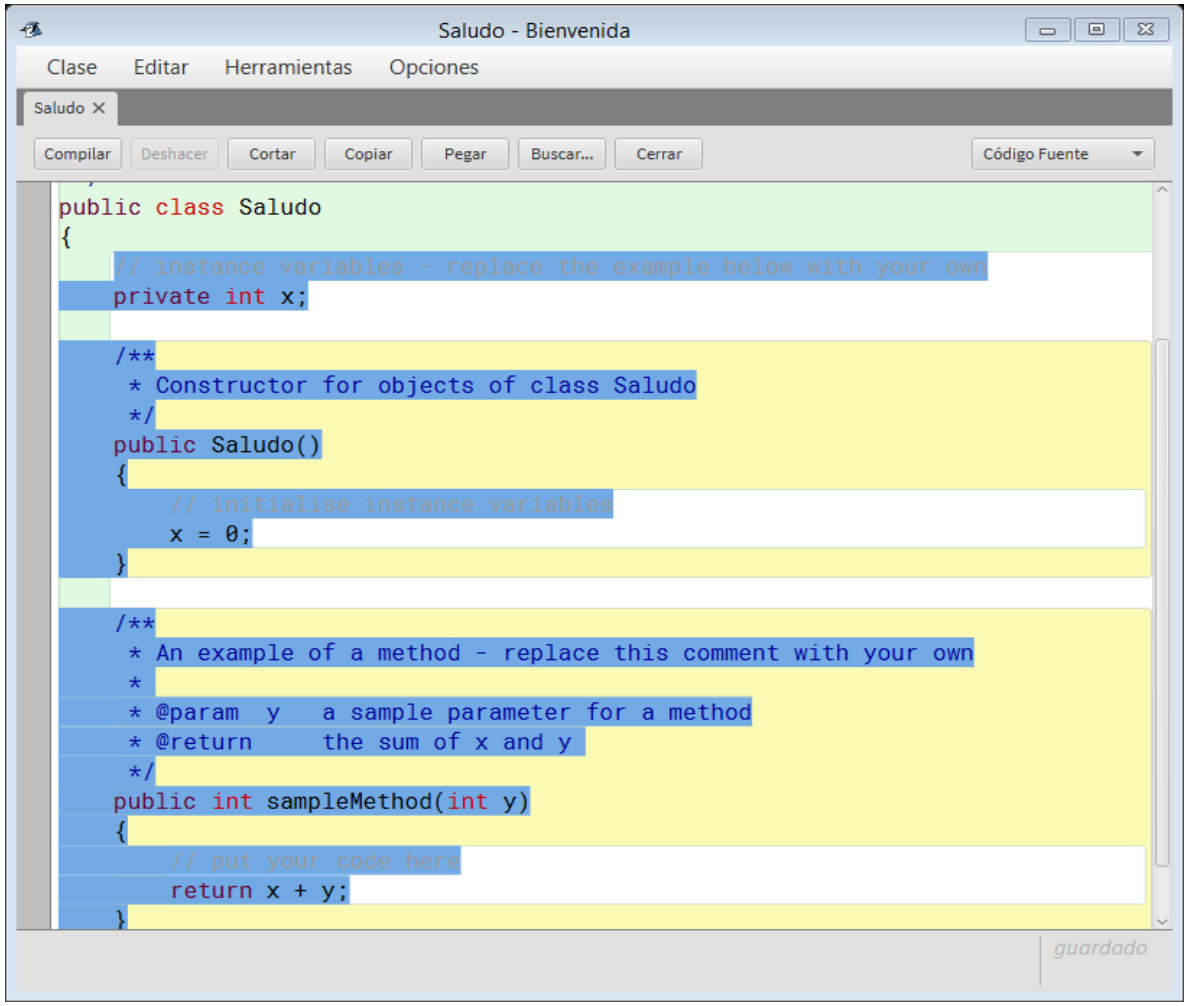
a. Dar doble clic al icono de la Clase
b. Dar clic derecho sobre la Clase y seleccionar Abrir Editor de Código

Ahora, se visualizará la siguiente ventana.

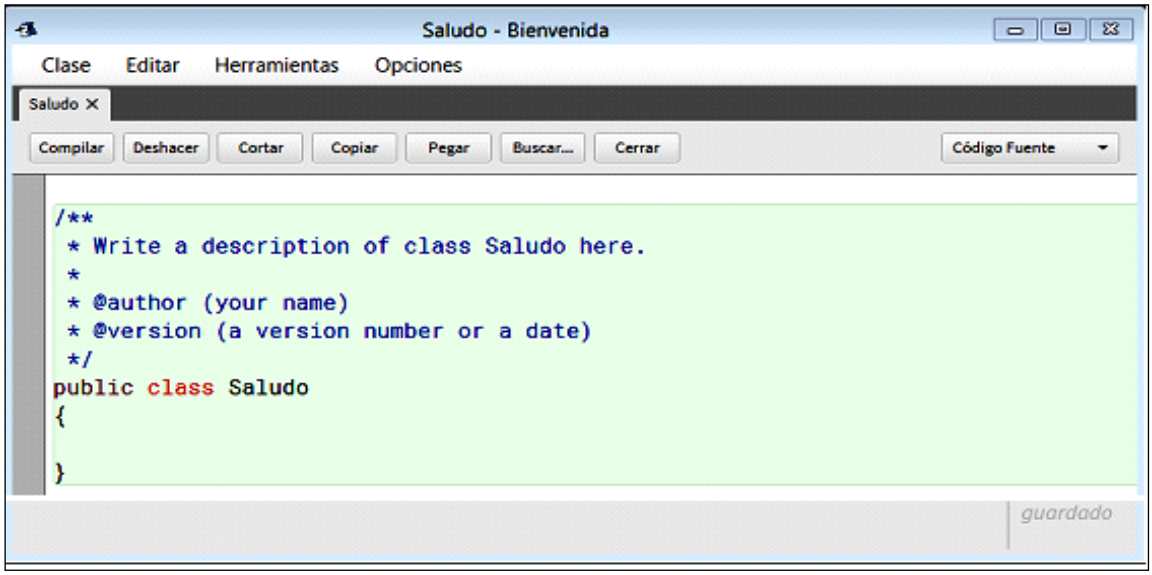


En ella se puede ver el código predefinido por BlueJ.

Por el momento, eliminaremos el código que no se va a utilizar. Selecciona el código que está dentro de la clase Saludo, como se muestra en la siguiente imagen.



Al eliminar el código que no utilizaremos, nos queda la siguiente pantalla.



En donde sólo se muestra la ***clase*** Saludo.

Para iniciar la captura del código debes tomar en cuenta las siguientes consideraciones:

- El símbolo que representa el término de una instrucción es (**;**) un punto y coma.
- Java es sensible a las mayúsculas y minúsculas, por lo que los nombres de los métodos, variables y clases deben estar escritas de la forma correcta o resultará en un error.
- Los bloques de código o instrucciones que son parte de un método o bucle deben estar rodeados por llaves (**{ }**).

2 Método main

Un método, es el nombre que reciben los módulos en Java. A continuación, se va a generar el método principal llamado main, para ello escribimos el siguiente código:

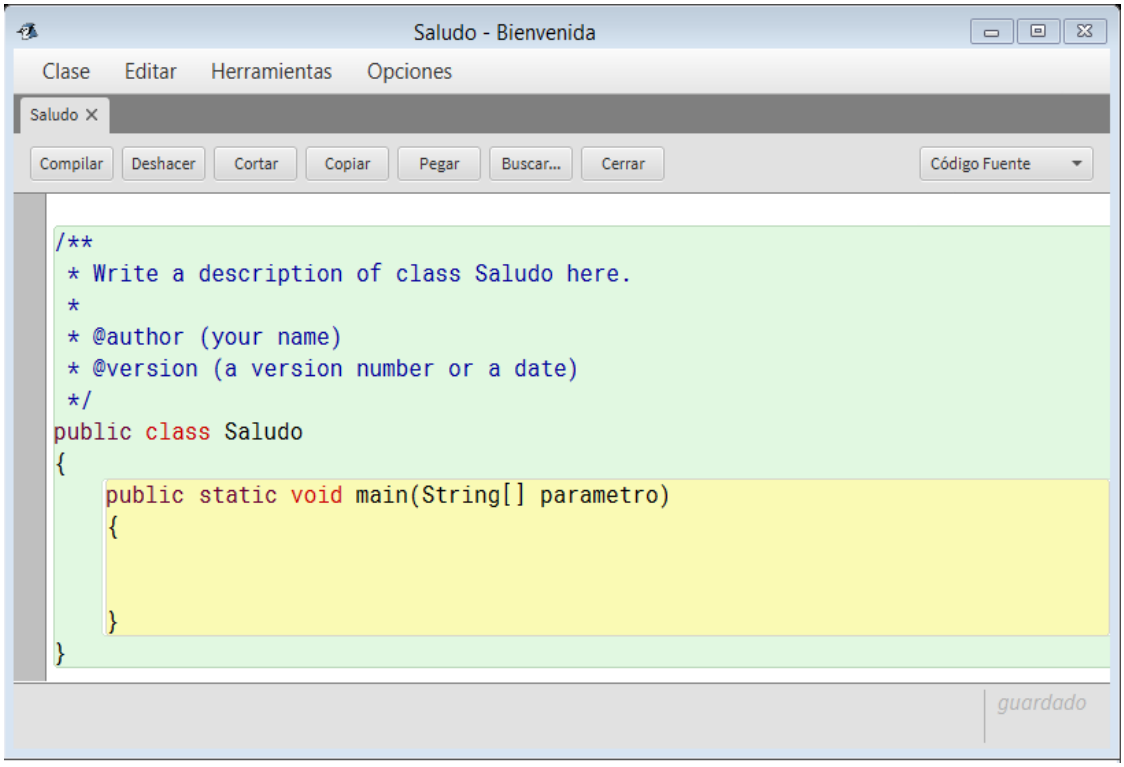
public static void main(String [] args)

Es el primer método que se ejecutará al iniciar el programa. Éste tendrá la misma declaración en todos tus programas de Java.
El cuerpo de este método debe contener las instrucciones necesarias para el arranque de la aplicación, es decir, la creación de instancias de clase, donde una instancia es un miembro de una clase con atributos, la inicialización de variables y la llamada a métodos. De manera que este método puede contener una sola instrucción.

En donde:

public	Modificador de acceso (acceso público) utilizado para hacer que el método sea accesible al conjunto de otras clases y objetos de la aplicación.
static	Modificador de acceso utilizado para definir el método main() como método de la clase.
void	Palabra clave utilizada para indicar que el método es un procedimiento que no devuelve valor.
main	Identificador del método.
String []	Parámetro del método, es una cadena de caracteres. Este parámetro se utiliza para pasar argumentos en línea de comando al ejecutar la aplicación.

Considerando lo anterior tendremos la pantalla que siguiente:



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

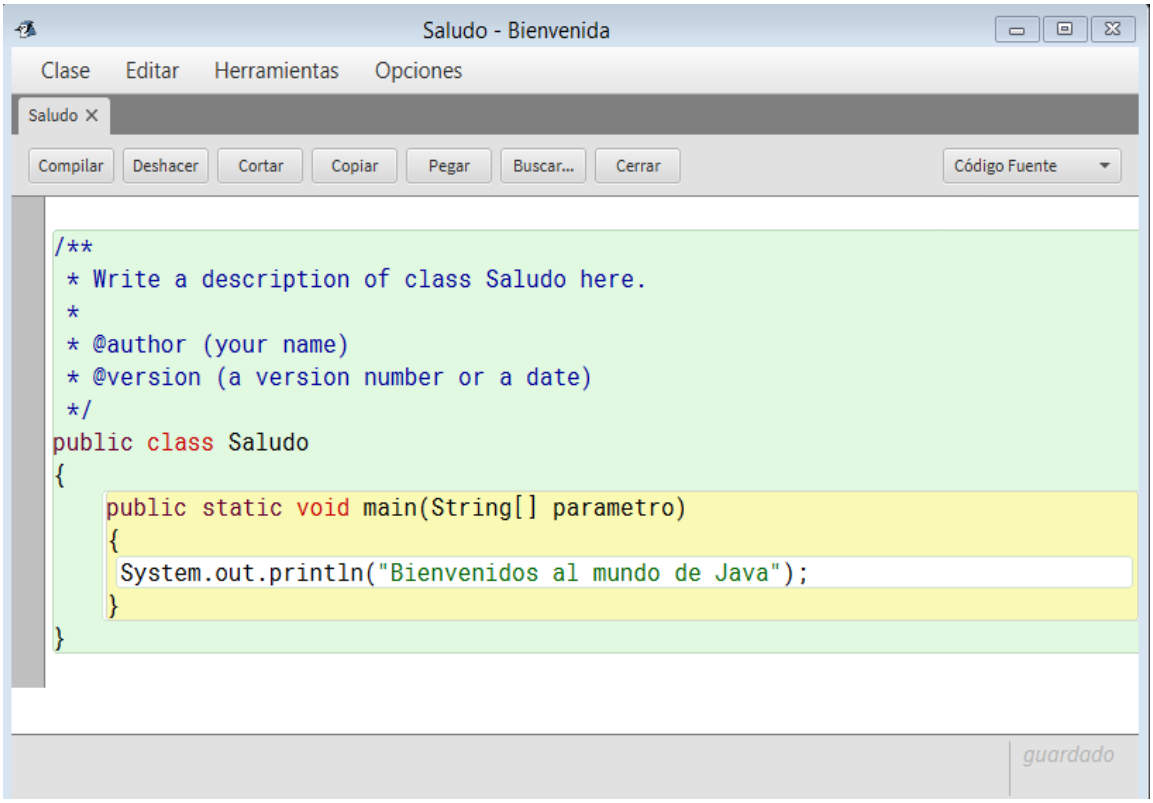
El método que se utiliza para la salida de datos es *System.out.println* en donde:

System	Le indica al sistema que debe hacer algo.
out	Le indica al sistema que se llevará a cabo algún tipo de operación de salida.
Print y Println	El primero imprime en la consola el valor del argumento que le pasamos. El segundo hace lo mismo, pero agrega un salto de línea al final.

Para imprimir el mensaje de salida ***Bienvenidos al mundo de Java*** en nuestro ejemplo, escribimos la siguiente línea:

System.out.println("Bienvenidos al mundo de Java");

Los paréntesis alrededor de ("***Bienvenidos al mundo de Java*** ") indican que el método System.out.println() recibe un parámetro, que en este caso es la cadena o String "*Bienvenidos al mundo de Java.*"



4 Errores sintácticos y lógicos

En esta fase se revisará la sintaxis del programa para ello es necesario **compilar** el programa, esto significa que al elegir la opción Compilar, se ejecutará un programa *compilador* que traducirá el código fuente o código objeto a código máquina, para ello es necesario que el compilador no detecte ningún error de sintaxis en dicho código fuente, en caso de ocurrir nos indicará la o las líneas donde estén los errores y la compilación se llevará a cabo hasta que se hayan corregido todos.

Los tipos de errores que se pueden presentar son:

- Errores sintácticos: Estos se presentan en la etapa de compilación, ocurren cuando se escribe una sentencia que no va de acuerdo con las reglas de escritura del lenguaje de programación.

- Errores lógicos: Estos se observan hasta la ejecución del programa, ocurren a causa de un mal diseño del programa. Esto es, una línea de código puede cumplir con las reglas de sintaxis del lenguaje, pero el código tenga una lógica equivocada y no devuelva el resultado adecuado.

.

Compilando el programa

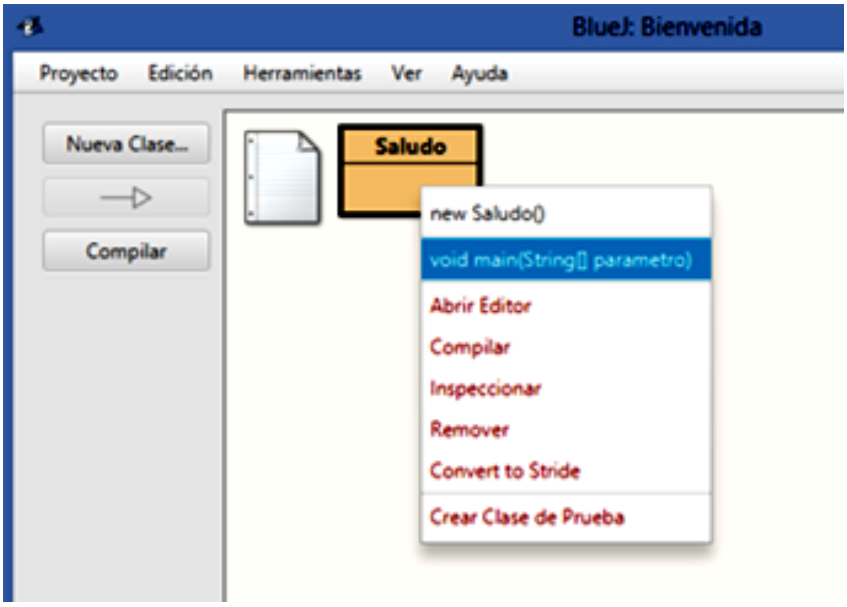
Al finalizar la edición del código se cierra la ventana del editor y se da clic en el Botón Compilar de las Barra de herramientas de Acceso rápido; o bien, seleccionamos el menú Herramientas y dar clic en la opción Compilar, o también desde el teclado, el teclado basta oprimir Ctrl + K.

Al terminar de compilar se visualizará en la pantalla el mensaje *Compilando… Listo*

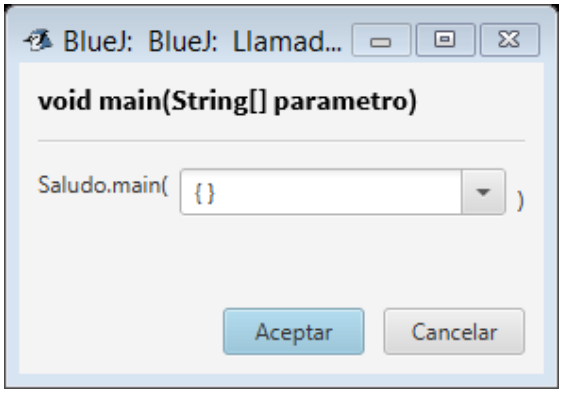
En este caso el ejemplo es muy sencillo, pero puede ocurrir que en programas más complejos se presente la situación de que no existan errores de sintaxis, y sin embrago el programa no realice adecuadament la tarea para la cual fue diseñado, se tiene que revisar el diseño del programa porque tenemos errores lógicos.

5 Ejecución del programa

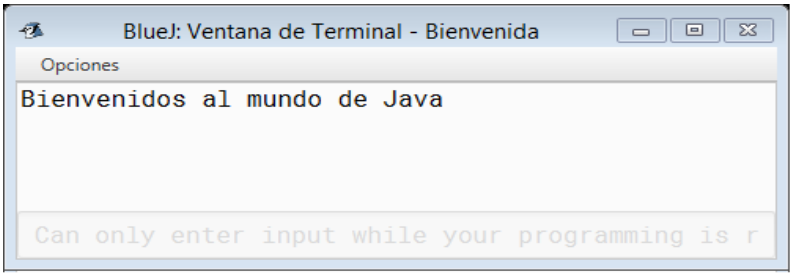
Para ejecutar el programa se da clic derecho sobre la clase y en el menú contextual seleccionar el comando *void main(String[] parametro).*



Aparecerá la siguiente ventana, dar clic en el botón Aceptar.



A continuación, se muestra el resultado:



Ejercicio

I. Modifica el programa del ejemplo anterior, para que además del saludo: "Bienvenidos al mundo de Java.", escriba en dos líneas adicionales tu nombre, grupo y número de cuenta.

II. **Instrucciones.** Selecciona la opción correcta de las siguientes cuestiones.

1. _____ es un paquete o fragmento de código Java que permite crear al menos una instancia (objeto).

a. ( ) Un proyecto
b. ( ) Una clase
c. ( ) Un método
d. ( ) Un atributo

2. En JAVA existe un método especial llamado _____que será el punto de partida de nuestro programa, dentro de este método ya se podrán hacer las llamadas a todas las rutinas que compondrán nuestro programa.

a. ( ) public
b. ( ) void
c. ( ) static
d. ( ) main

3. En la sentencia ***public static void main(String [] args)*** public se refiere a:

a. ( ) Modificador de acceso (acceso público) utilizado para hacer que el método sea accesible al conjunto de otras clases y objetos de la aplicación.
b. ( ) Modificador de acceso utilizado para definir el método main() como método de la clase.
c. ( ) Palabra clave utilizada para indicar que el método es un procedimiento que no devuelve valor.
d. ( ) Identificador del método.

4. El método que se utiliza para la salida de datos:

a. ( ) System.out.println
b. ( ) public static void main

c. ( ) Class Saludo
d. ( ) println

5. Un _____ ocurre cuando el programa no realiza adecuadamente la tarea para la cual fue diseñado.

a. ( ) Error de sintaxis
b. ( ) Error lógico
c. ( ) Error matemático
d. ( ) Error de compilación

TEMA **12**

Introducción de datos desde el teclado

Objetivo:

- Realizar programas empleando la Clase Scanner para la entrada de datos.

1 Introducción

La clase Scanner va a permitir la interacción de un programa con el usuario porque a través de ella el usuario podrá introducir datos , para ello se requiere utilizar un objeto de tipo Scanner y el método System.in los cuales permitirán el registro de la información introducida. Asimismo, se diferenciará tipo de información por medio de los diferentes tipos de datos numéricos y alfanuméricos

2 La clase Scanner

Java proporciona en el paquete java.util una clase llamada Scanner que nos permitirá leer datos, estos se pueden leer utilizando métodos para leer valores de entrada de varios tipos, los valores de entrada pueden venir de varias fuentes, incluyendo valores que se entren por el teclado o datos almacenados en un archivo.

Para importar la clase Scanner, se agrega la siguiente línea de código al principio de nuestro programa.

import java.util.Scanner;

Para utilizarla, debemos crear un objeto nuevo del tipo Scanner a los campos correspondientes podemos asignar valores y podemos utilizar sus métodos.

3 Definición del objeto de la Clase Scanner

Para crear un objeto **Scanner** y conectarlo con System.in, se utiliza el siguiente código:

Scanner teclado = new Scanner(*System.in****);***

Esto quiere decir:

Scanner teclado :	Declara una variable llamada teclado. El tipo de dato de esta variable es **Scanner**. Como Scanner es una clase, entonces la variable *teclado* es un objeto de la clase **Scanner**.
=	Este símbolo indica que se está asignando un valor a la variable teclado.
new Scanner	La palabra reservada ***new*** crea un nuevo objeto en la memoria, el tipo de objeto que creará es **Scanner(**System.in**)**, básicamente está reservando memoria en la computadora para que se pueda guardar la información de System.in.
new Scanner(*System.in****);***	El valor es la información que el usuario introduzca por el teclado.

4 Método System.in

Java tiene un método llamado System.in, el cual permite obtener información proporcionada por el usuario. Sin embargo, Sytem.in sólo lee la información en bytes y en programación se hace uso de diferentes tipos de variables (char, string, int, float, etc). Para solucionar este problema se hace uso de la clase **Scanner,** que está diseñada para leer los valores tipo byte y convertirlos en valores primitivos (int, long, double, float, boolean, etc) o en valores String.
Para utilizar la clase Scanner tenemos que crear primero un objeto de ella, esto es, un objeto del tipo de la clase, esto nos permitirá utilizar sus métodos de lectura.

La siguiente declaración crea un objeto de la clase Scanner que lee valores de entrada del teclado.

Scanner teclado = new Scanner(System.in);

Los objetos de tipo Scanner permitirán leer datos de cualquier tipo (String, enteros, reales, etc), mediante la invocación de distintos métodos con diversas posibilidades de separadores. Asimismo, permitirá leer un archivo de texto por línea, guardando cada línea en un objeto de tipo String.

5 Tipos de datos

Un tipo de datos define un conjunto de valores y las operaciones que se pueden realizar sobre ellos. La mayoría los lenguajes de programación permiten la declaración de los tipos de datos. Actualmente la mayoría de los lenguajes de programación permiten al programador definir tipos de datos personalizados. Por ejemplo, podría requerirse un nuevo tipo de dato llamado "Alumno", las variables de tipo Alumno podrían almacenar valores referentes a la matrícula y el nombre.

Antes de describirá las características de los datos o variables describiremos un elemento importante en el desarrollo de un programa, como son los comentarios.

Comentarios

Un comentario es una referencia u observación que se hace en el código del programa y que solo sirve para el programador, no afecta al código del programa.

En Java existen tres tipos de comentarios:

1. // comentarios para una sola línea
2. /* comentarios de una o más líneas */
3. /** comentario de documentación, de una o más líneas */

En el desarrollo de los ejemplos veremos la implementación de los comentarios.

Variables

Las variables son una de las características fundamentales de los lenguajes de programación, permiten acceder a la memoria para almacenar y recuperar los datos con los que nuestros programas van a trabajar. Son por tanto el mecanismo que los lenguajes nos proveen para asignar el nombre a las variables que se utilizan en un programa, a partir de ese momento el compilador traducirá de forma automática ese nombre en un acceso a memoria. Por ejemplo:

```
edad = 25        //Almacenamos un dato en memoria referenciado por el nombre edad
edad = 25 + 1    //Recuperamos el dato almacenado y lo modificamos
```

Podemos definir variables en cualquier parte del código simplemente indicando el tipo de datos y el nombre de la variable, el nombre asignado a una variable se denomina identificador, este debe comenzar con una letra o guion bajo y no puede llevar espacios ni caracteres especiales, por ejemplo:

- Identificadores o nombre de variables válidos son:
  - fecha
  - iFecha
  - fechaNacimiento
  - fecha_nacimiento
  - fecha3
  - _fecha
- Identificadores NO válidos son:
  - 3fecha
  - fecha-nacimiento
  - fecha+nacimiento
  - -fecha

Tipos de variables

Java es un lenguaje tipado y nos obliga a declarar nuestras variables antes de poder hacer uso de ellas, con esta declaración le indicamos al compilador el espacio en memoria que debe de reservar para almacenar la información del tipo especificado. Por ejemplo:

```
String Alumno;
```

El tipo de dato **String** permite introducir una cadena de texto.

Aquí estamos reservando memoria para una variable de tipo **String** y la identificamos con el nombre “Alumno”. De ahora en adelante si en el programa hablamos de Alumno, estamos haciendo referencia a esa porción de memoria y al valor que contiene.
Se puede asignar un valor en el momento de declarar una variable. Por ejemplo:

String Alumno =”Daniel González”;

Aquí reservamos memoria para una cadena de caracteres y le asignamos el valor "Daniel González". También se puede declarar la variable y en otro lugar del programa asignarle un valor un valor:

String alumno;

//Sigue el programa

Alumno = “Daniel González”;

La sintaxis para declarar una variable es:

Tipo_de_Dato Nombre_de_Variable;	Cuando se espera un valor por el teclado o se asigna un valor después
Tipo_de_Dato Nombre_de_Variable [= Valor_inicial];	Asignando un valor de inmediato

En la siguiente tabla se presentan los tipos de datos primitivos en java.

Tipo de dato	Representación	Tamaño (Bytes)	Rango de Valores	Valor por defecto	Clase Asociada
byte	Numérico Entero con signo	1	-128 a 127	0	Byte
short	Numérico Entero con signo	2	-32768 a 32767	0	Short
int	Numérico Entero con signo	4	-2147483648 a 2147483647	0	Integer
long	Numérico Entero con signo	8	-9223372036854775808 a 9223372036854775807	0	Long
float	Numérico en Coma flotante de precisión simple Norma IEEE 754	4	$\pm$ 3.4x10^{-38} a $\pm$ 3.4x10^{38}	0.0	Float

Tipo de dato	Representación	Tamaño (Bytes)	Rango de Valores	Valor por defecto	Clase Asociada
double	Numérico en Coma flotante de precisión doble Norma IEEE 754	8	$\pm$ 1.8x10^{-308} a $\pm$ 1.8x10^{308}	0.0	Double
char	Carácter Unicode	2	\u0000 a \uFFFF	\u0000	Character
boolean	Dato lógico	-	true ó false	false	Boolean
void	-	-	-	-	Void

En Java, cualquier número con punto flotante automáticamente se considera double. Para que sea considerado float se agrega una letra "f" o "F" al final del valor.

```
double d = 22.50;
float f = 22.50F;
```

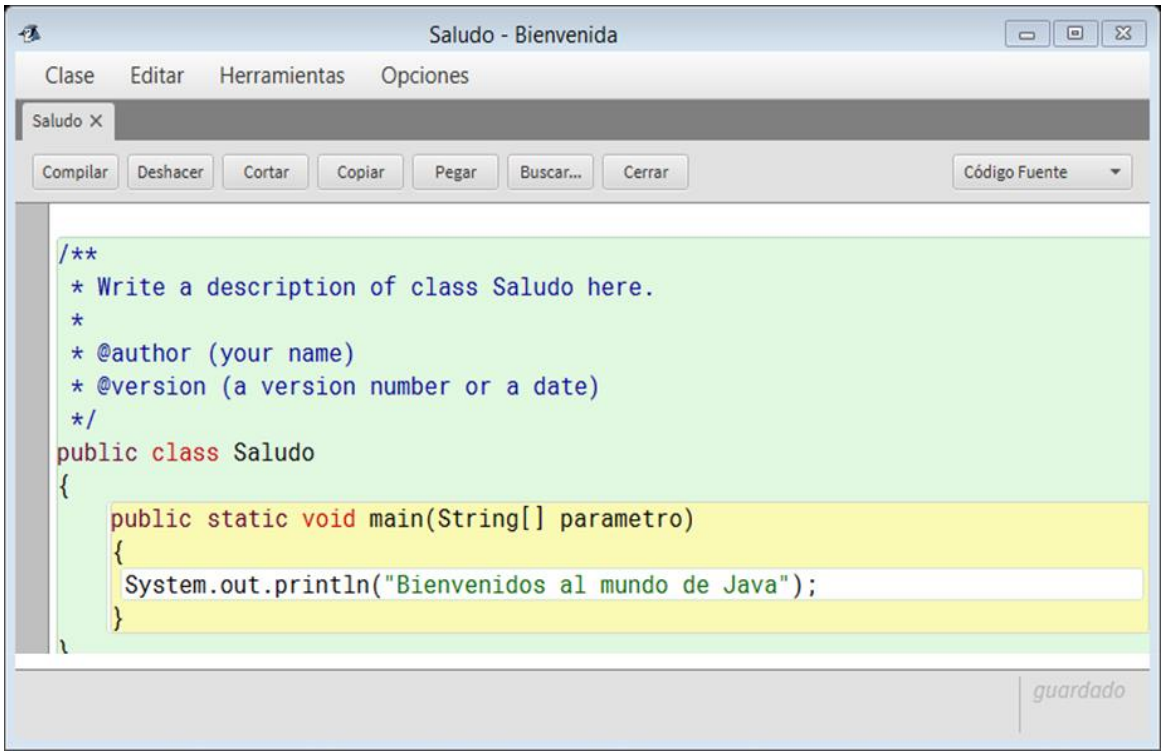
Para ingresar el valor de una variable se hace uso de un método específico según el tipo de variable. A continuación, se muestran los métodos.

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

El método **nextByte** no significa que va a reconvertir la información en bytes, sino que va a transformar la información en el valor **byte** la cual puede ser un número del -128 al +127.
A continuación, se desarrollan varios ejemplos en los que se aplican los conceptos anteriormente descritos.

Ejemplo 2.
Continuando con el proyecto anterior *Bienvenida*, se solicitará el nombre al usuario para mostrar una bienvenida personalizada.

Para ello, se abre el proyecto creado del ejercicio anterior y nos dirigimos a la ventana del Editor de código (recuerda que, para ello, se da doble clic al icono que representa la clase Saludo) observando la siguiente ventana.



A este código se le agregarán las instrucciones necesarias para solicitar el nombre del usuario y mostrar como salida como la que se muestra a continuación:

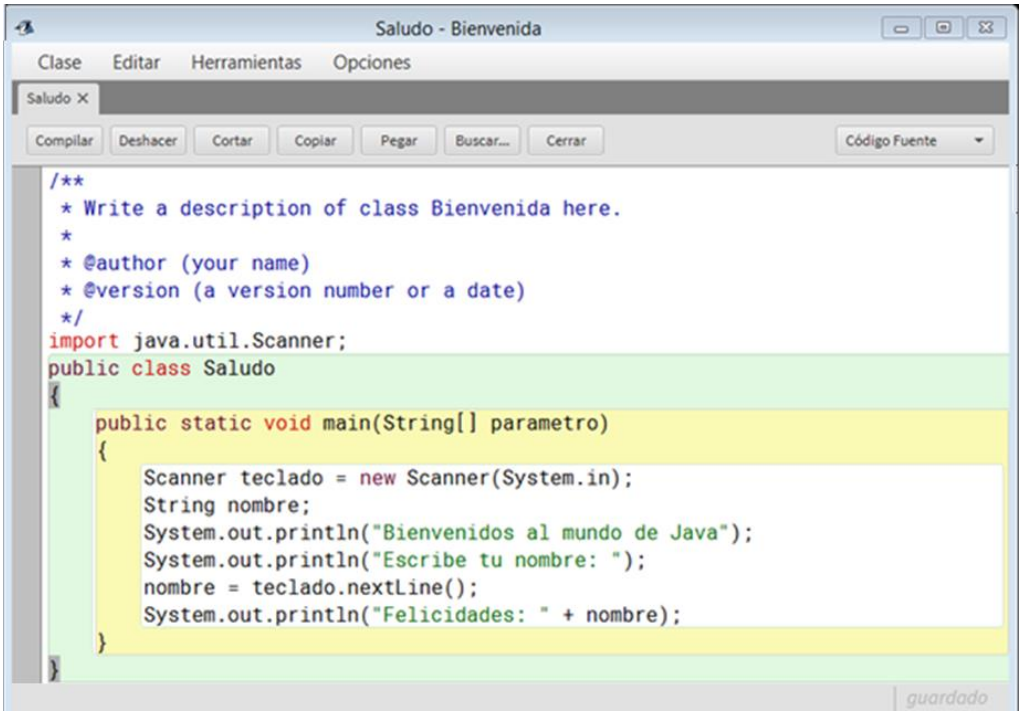
Bienvenidos al mundo de Java
Felicidades *nombre*.

El código es el siguiente:

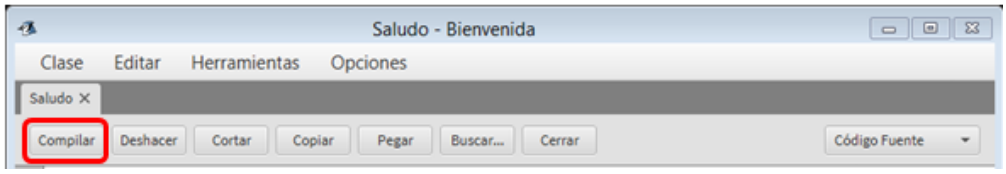
```
import java.util.Scanner;              // Instrucción para hacer uso de la clase Scanner
public class Saludo
{
   public static void main(String[] parametro)
   {
//Creación del objeto teclado de tipo Scanner
Scanner teclado = new Scanner(System.in);
String nombre;    //Declaración de la variable nombre de tipo Cadena - Texto
      System.out.println("Bienvenidos al mundo de Java");
      System.out.println("Escribe tu nombre: ");
      nombre = teclado.nextLine();  //Asignación de lo escrito por el teclado a la variable nombre
```

```
        System.out.println("Felicidades: " + nombre);
    }
}
```

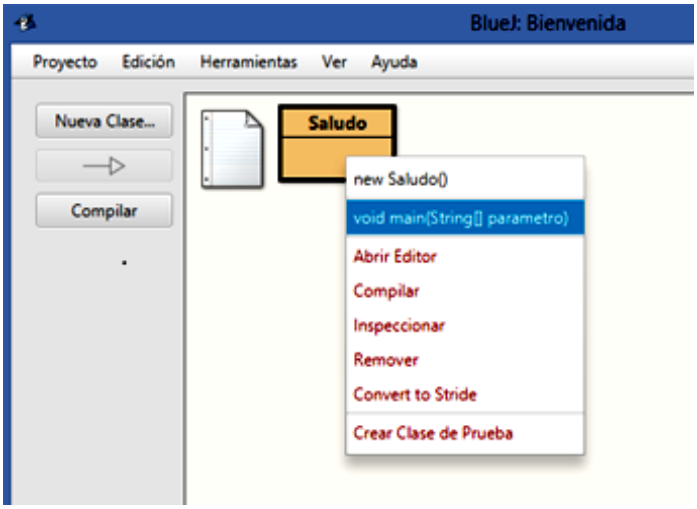
Ahora se muestra cómo debe aparecer la ventana del editor de código.

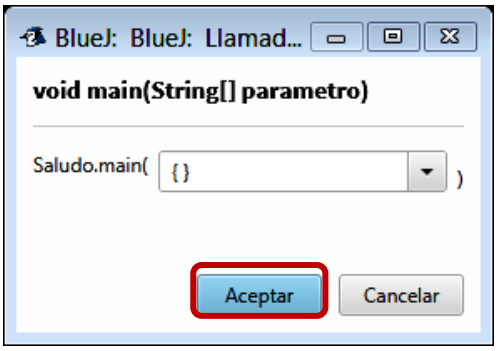


Compilamos el programa y verificamos que no existan errores. Si se presentan errores habrá que verificar la escritura del código para corregirlos.

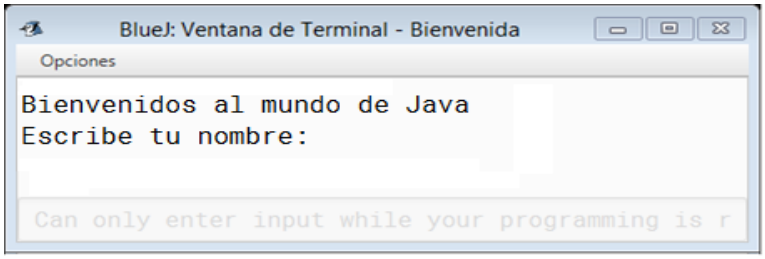


Al no existir errores. Se ejecuta el programa, para ello cerramos la ventana del editor de código y presionamos botón derecho sobre la clase y en el menú contextual seleccionamos el comando *void main* y a la ventana que aparece se le da clic en Aceptar, a continuación, se muestra el procedimiento.

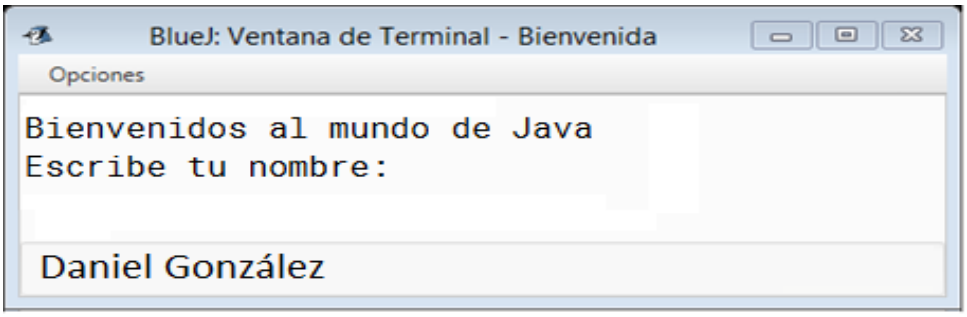




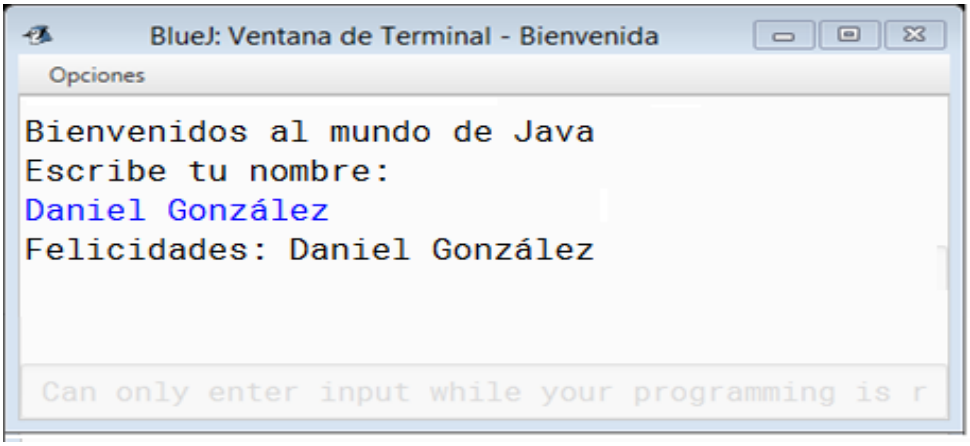
Se visualizan los resultados la ventana de terminal:



El usuario debe ingresar un valor, por ejemplo: ***Daniel González***



Al oprimir *enter* se visualizará:



Ejemplo 3.

Realizar un programa que sume dos números introducidos desde el teclado.
Aprovecharemos los ejemplos para describir los pasos a seguir en el desarrollo de un programa.

Planteamiento. En esta fase se define el problema en términos específicos se define qué es lo que queremos lograr, en este caso, pretendemos que el programa a desarrollar despliegue una pantalla en la que se mostrarán mensajes solicitando dos números a sumar.

Algoritmo. Son los pasos que seguir para resolver el problema, en este caso tenemos el siguiente:

0. *Inicio*
1. *Declarar variables*
2. *Leer el primer número a*
3. *Leer el segundo número b*
4. *Suma=a+b*
5. *Escribir ("La suma es : "+ suma);*
6. *Fin*

Seudocódigo. Recordemos que es el programa con base en un lenguaje de programación. Pero hecho por el programador en su propio idioma, para describir un algoritmo y poder comprender mejor la estructura de dicho programa. Para nuestro ejemplo, tenemos el siguiente:

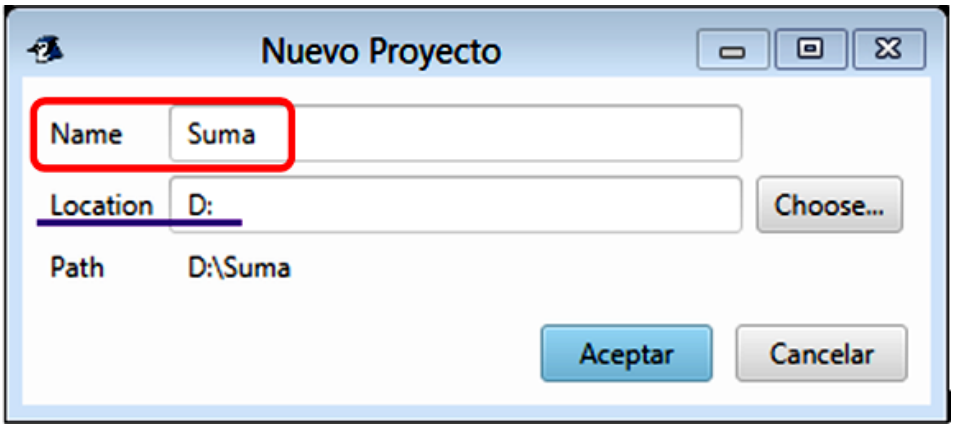
Seudocódigo

```
public class Adicion
{
    Inicio del método principal
```

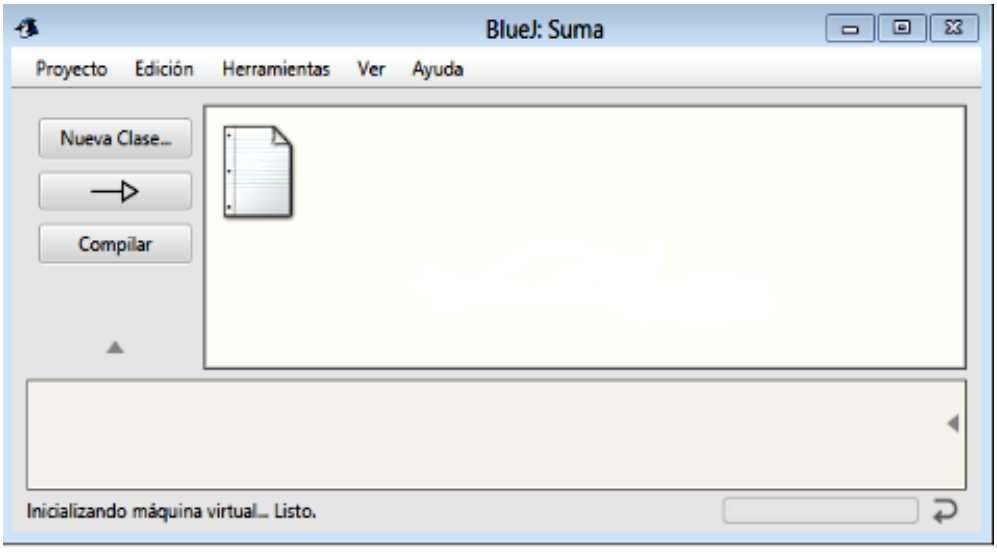
```
{
   //Declaración de las variables de tipo entero
   int a,b, suma;
   escribir("El primer número a sumar : ");
   a = teclado.nextInt();
   escribir("El segundo número a sumar : ");
   b = teclado.nextInt();
   suma=a+b;
   escribir("El resultado de la suma es : " + suma);
}
```

Desarrollo del programa

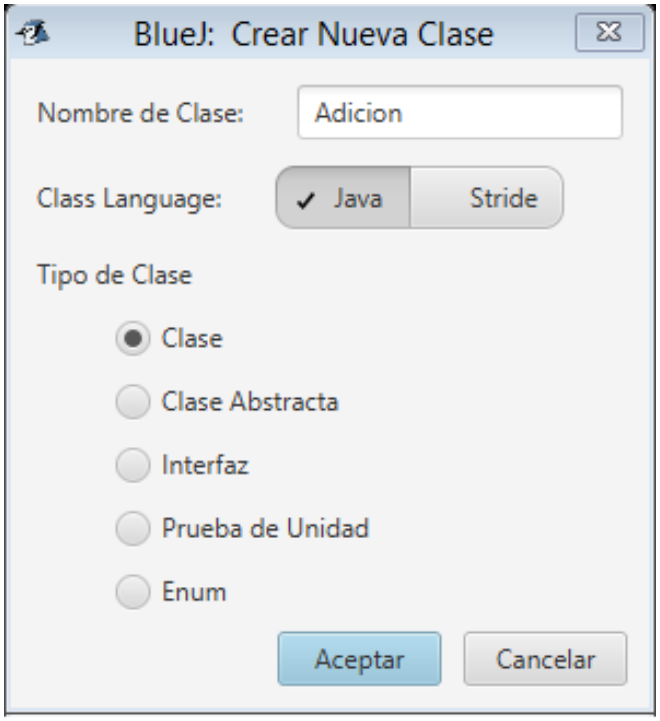
En BlueJ, se crea un nuevo Proyecto con el nombre Suma, estará ubicado en la unidad D, se da clic en Aceptar.



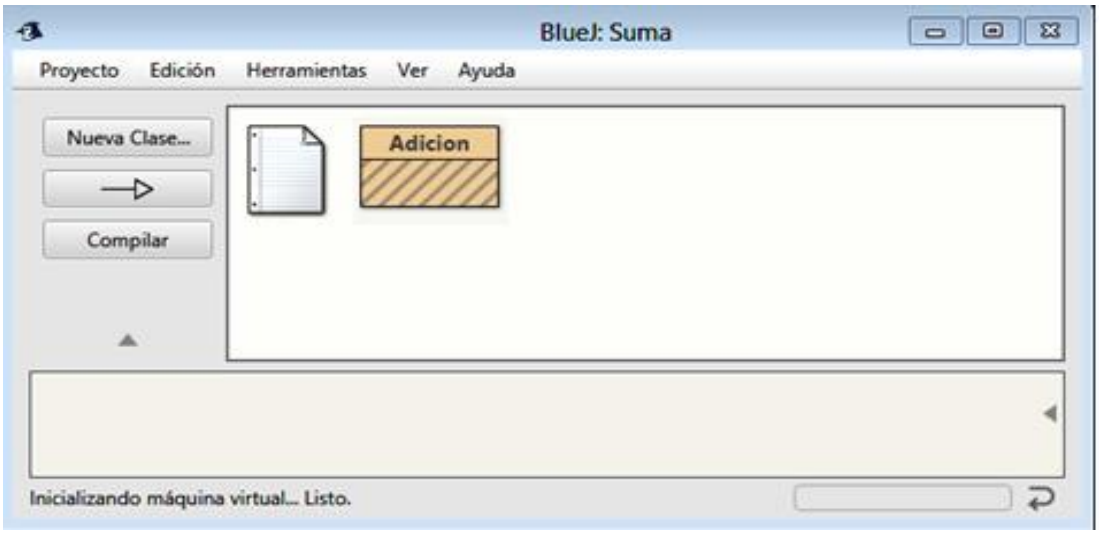
Después aparece la siguiente ventana:



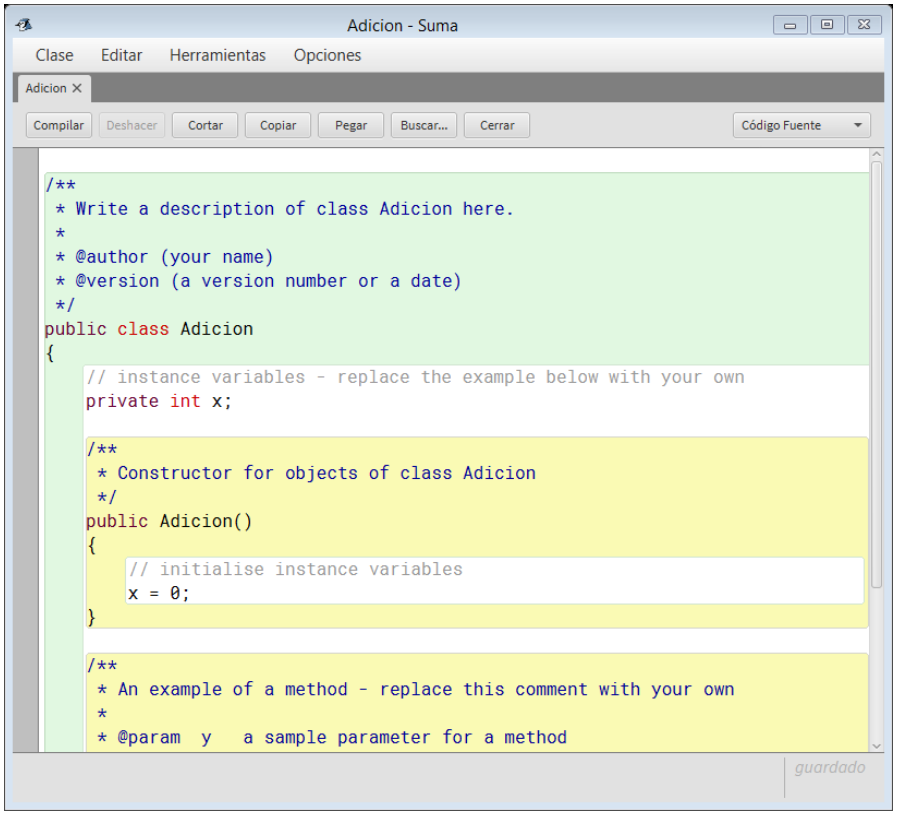
Crearemos una nueva clase y le asignaremos el nombre de Adición.



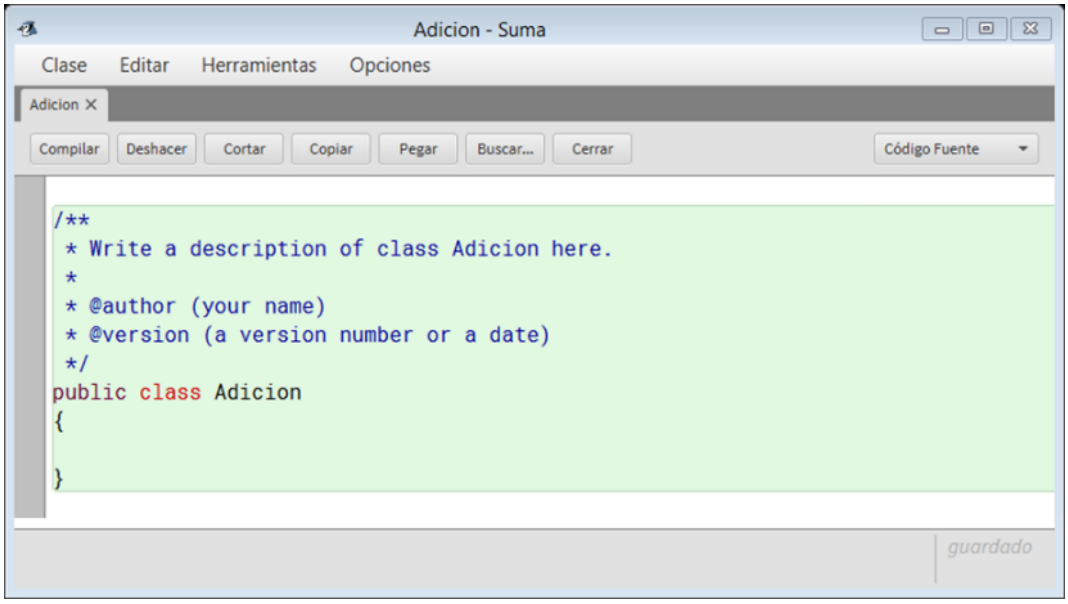
Ahora el ambiente de trabajo nos queda:



Se da doble clic a la clase Adición para abrir la ventana Editor de código, se muestra la siguiente ventana:



Eliminamos el código que se tiene demás y sólo dejamos el código de que corresponde a la creación de la clase principal Adición, quedando la ventana como sigue:



Ahora escribimos el *código fuente* que corresponde al programa

import java.util.Scanner; // Instrucción para hacer uso de la clase Scanner

public class Adicion

```
{
   //Creación del método principal
   public static void main(String[] parametro)
   {
     //Creación del objeto teclado de tipo
      Scanner teclado = new Scanner(System.in);
      //Declaración de las variables de tipo entero
      int a,b, suma;
      System.out.println("Escribe el primer número a sumar : ");
     //Asignación del valor a la variable a
      a = teclado.nextInt();
     System.out.println("Escribe el segundo número a sumar : ");
      //Asignación del valor a la variable b
      b = teclado.nextInt();
      suma=a+b;
      System.out.println("El resultado de la suma es : " + suma);
   }
}
```

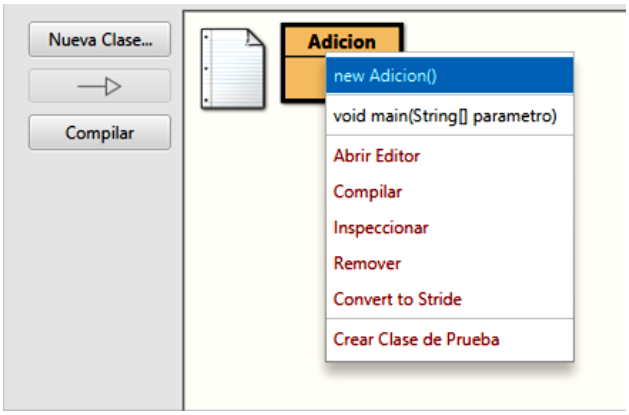
Después de capturar el código, tendremos la siguiente pantalla:

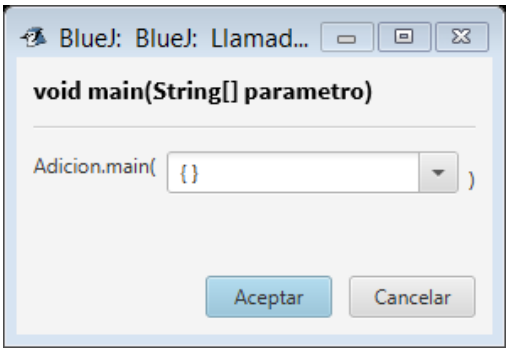
```
import java.util.Scanner;
public class Adicion
{
     public static void main(String[] parametro)
    {
        Scanner teclado = new Scanner(System.in);
        int a,b, suma;
        System.out.println("Escribe el primer número a sumar : ");
        a = teclado.nextInt();
        System.out.println("Escribe el segundo número a sumar : ");
        b = teclado.nextInt();
        suma=a+b;
        System.out.println("El resultado de la suma es : " + suma);;


    }
}
```

Ejecución.

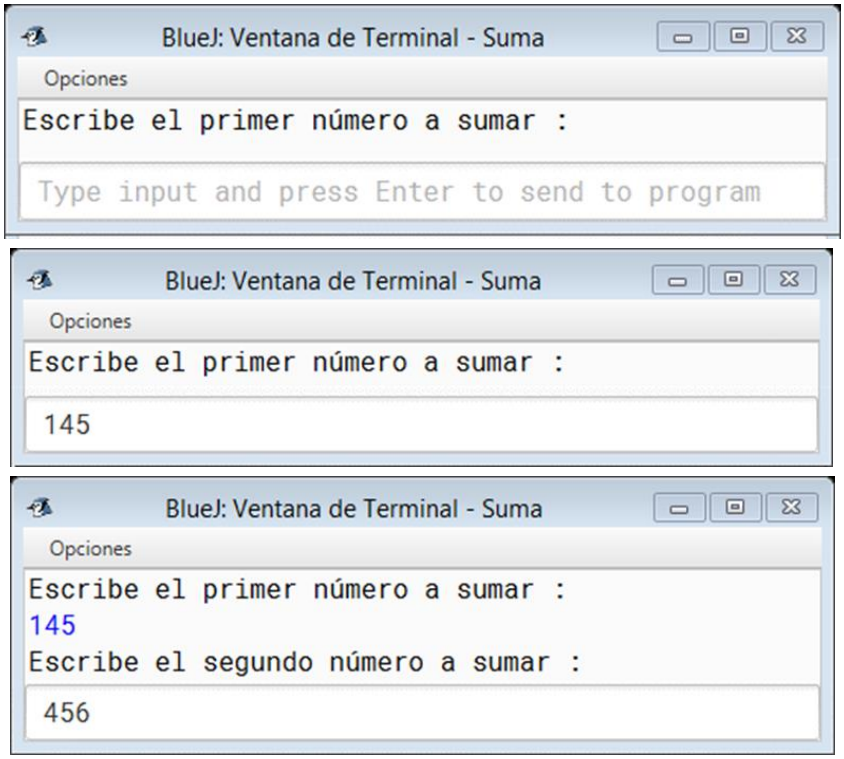
Se compila el programa y se verifica que no existan errores de sintaxis, si se presentan habrá que verificar la escritura del código para omitirlos.

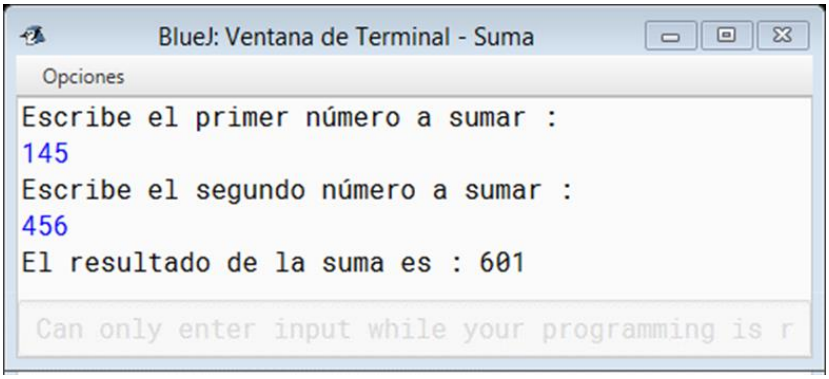
Para ejecutar el programa, se cierra la ventana del código y se da clic derecho a la clase *Adicion*, se selecciona el comando void main, en la ventana que aparece se da clic en Aceptar. Las siguientes figuras muestran el proceso:





Aparece la ventana de terminal - Suma





Ejemplo 4.

Elaborar un programa que calcule el área de un triángulo.

Planteamiento. Se debe desarrollar un programa que solicite al usuario los valores correspondientes a la base y la altura de un triángulo, como resultado se desplegará en la pantalla el valor del área del triángulo.

Algoritmo.

0. *Inicio*
1. *Declarar variables*
2. *Leer el valor de la base*
3. *Leer el valor de la altura*
4. *a=(b*h)/2*
5. *Escribir el valor del área*
6. *Fin*

Seudocódigo

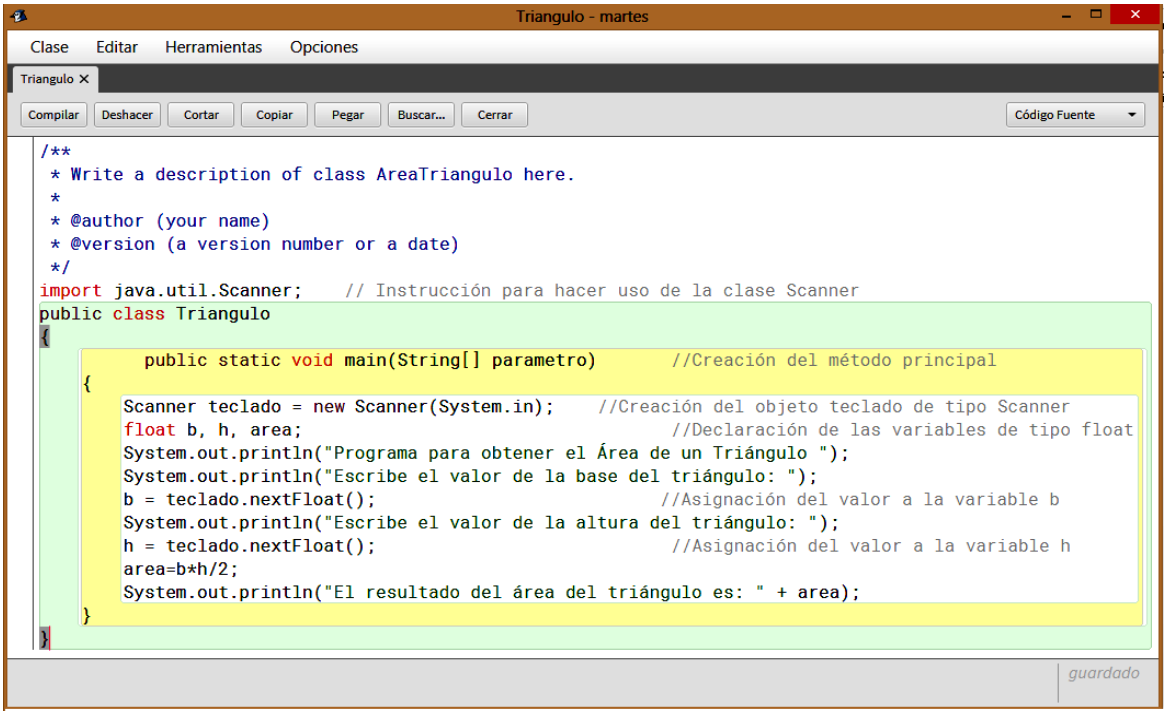
```
public class Triangulo
{
      Inicia método principal
  {
    Declaración de variables
     float b, h, area;    //float por ser números reales
     Escribir ("Escribe el valor de la base del triángulo: ");
     b = teclado.nextFloat();  //Asignación del valor a la variable b
    Escribir("Escribe el valor de la altura del triángulo: ");
     h = teclado.nextFloat();      //Asignación del valor a la variable h
     area=b*h/2;
     escribir("El resultado del área del triángulo es: " + area);
  }
}
```

Desarrollo del programa.

Para generar este programa se crea un nuevo proyecto, al que nombraremos Área y crearemos una clase que se llame Triangulo que estará formada por el siguiente código:

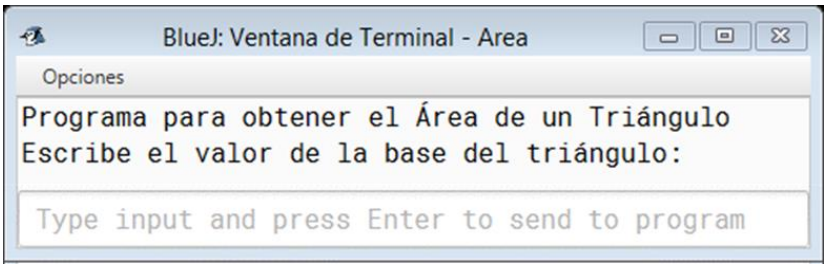
```
import java.util.Scanner;    // Instrucción para hacer uso de la clase Scanner
public class Triangulo
{
      public static void main(String[] parametro) //Creación del método principal
  {
     Scanner teclado = new Scanner(System.in); //Creación del objeto teclado de tipo Scanner
     float b, h, area;   //Declaración de las variables de tipo float
     System.out.println("Programa para obtener el Área de un Triángulo ");
     System.out.println("Escribe el valor de la base del triángulo: ");
     b = teclado.nextFloat();   //Asignación del valor a la variable b
     System.out.println("Escribe el valor de la altura del triángulo: ");
     h = teclado.nextFloat();   //Asignación del valor a la variable h
     area=b*h/2;
     System.out.println("El resultado del área del triángulo es: " + area);
  }
}
```

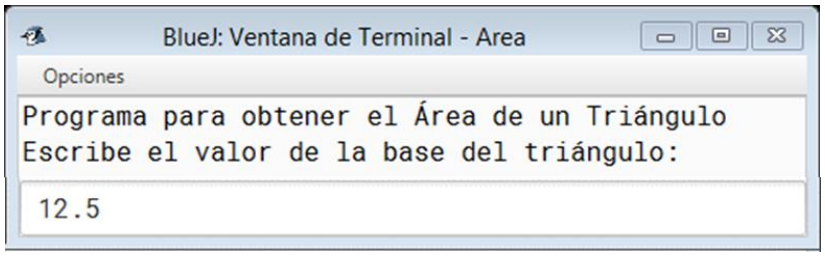
En el código anterior se hace distinción sobre la declaración de las variables tipo float y el registro de cada una de ellas. A continuación, se muestra la pantalla con el código del programa fuente ya capturado:

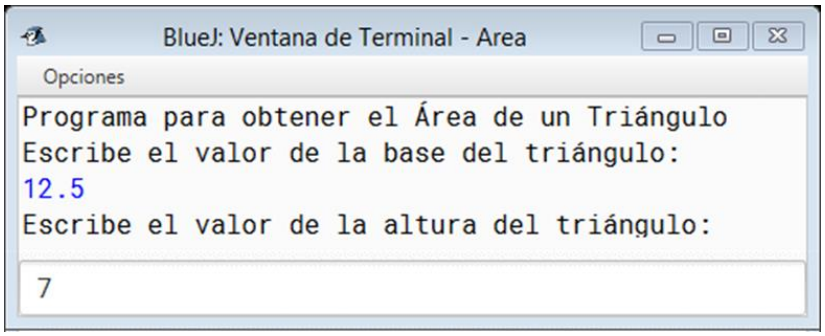


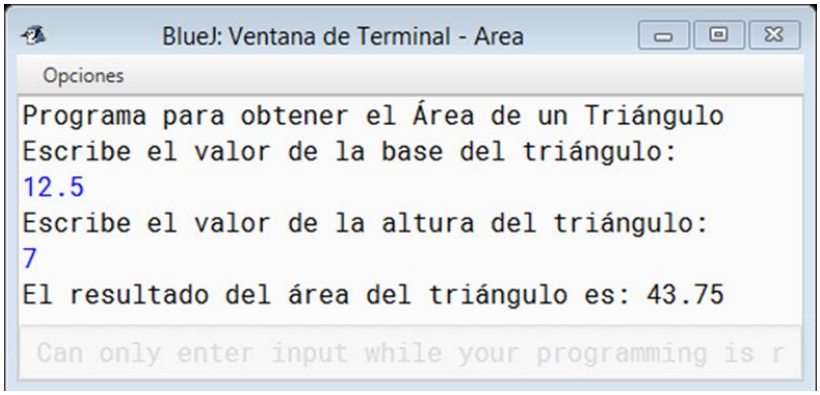
Ejecución

Al compilar y ejecutar el código anterior se tiene:









Ejercicio

- Elabora un programa que lea cuatro números introducidos desde el teclado y obtenga el promedio de ellos.

TEMA **13**

Estructuras Condicionales

Objetivos:

- Elaborar el algoritmo, diagrama de flujo y seudocódigo para problemas condicionales.
- Construir programas de computadora que resuelvan problemas condicionales.
- Elaborar el algoritmo, diagrama de flujo y seudocódigo para problemas condicionales múltiples.

1 Introducción

Las sentencias condicionales permiten modificar el flujo de ejecución de las instrucciones de un programa; de acuerdo con una condición, ejecutar un grupo u otro de sentencias (sentencia If..Else); así también con el valor de una variable, ejecutar un grupo u otro de sentencias (sentencia Swtich).

2 Operadores Relacionales

Los operadores relacionales comparan dos operandos y dan como resultado de la comparación verdadero o falso.

Los operadores relacionales en java son:

Operador	Nombre	Ejemplo: Si a = 7, b = 9, c = 7	
		Operación	Resultado
<	Menor que	b < c	false
>	Mayor que	b > c	true
<=	Menor o igual	a <= b	true
>=	Mayor o igual	a >=c	true
!=	Distinto	a != c	false
==	Igual	a==b	false

Nota. Los operandos tienen que ser de tipo primitivo.

2 Operadores Lógicos

Los operadores lógicos se utilizan con operandos de tipo booleano, es decir el resultado de dos comparaciones. Se utilizan para construir expresiones lógicas, cuyo resultado es de tipo true o false.

Los operadores lógicos en Java son:

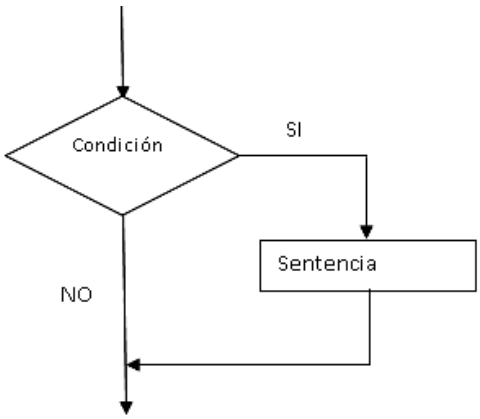
Operador	Nombre	Descripción
&&	AND	El resultado es verdadero si los dos operandos son verdaderos. El resultado es falso en caso contrario.
||	OR	El resultado es falso si los dos operandos son falsos. Si uno es verdadero el resultado es verdadero.
!	NOT	Se aplica sobre un solo operando. Cambia el valor del operando de verdadero a falso y viceversa.

2 Sentencias Condicionales

Sentencia If simple

La sentencia condicional ***if*** es utilizada para tomar decisiones lógicas, esto quiere decir que se evalúa una condición y en función del resultado se realizan una o varias instrucciones. Las condiciones se especifican utilizando expresiones lógicas.

El diagrama de flujo correspondiente es el siguiente:



La sintaxis es la siguiente

```
If (condicion)
{
   //sentencias si la condición es verdadera
}
```

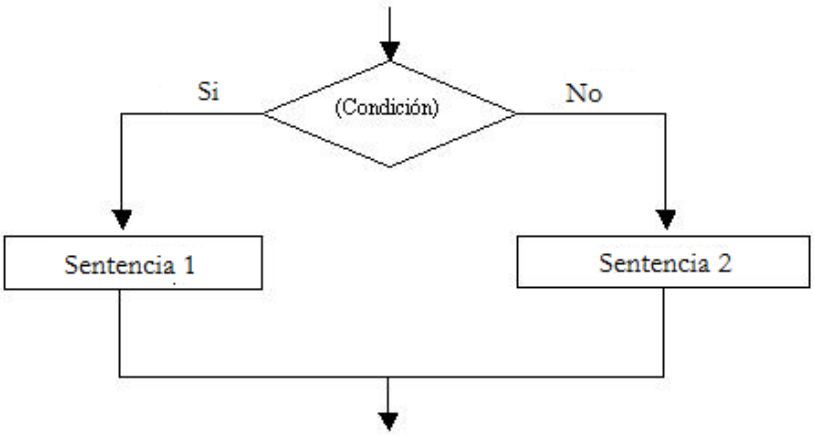
Sentencia *If … else*

La sentencia condicional If … else nos permite evaluar una condición y obtener un valor verdadero o falso y con base en ello tomar una decisión dentro del programa.

La condición es una expresión booleana que puede ser verdadera o falsa. Una expresión booleana se forma comparando valores de las expresiones utilizando operadores relacionales y operadores lógicos.

Si la condición se cumple se ejecutará un determinado bloque de instrucciones, en caso contrario podemos optar por ejecutar otro bloque distinto de instrucciones o no ejecutar ninguna.

Podemos representar su funcionamiento con el siguiente diagrama de flujo:



La sintaxis de la instrucción *if…else* es la siguiente:

```
If (condicion)
{
   //sentencias si la condición es verdadera
}
else
{
   //sentencias si la condición es falsa
}
```

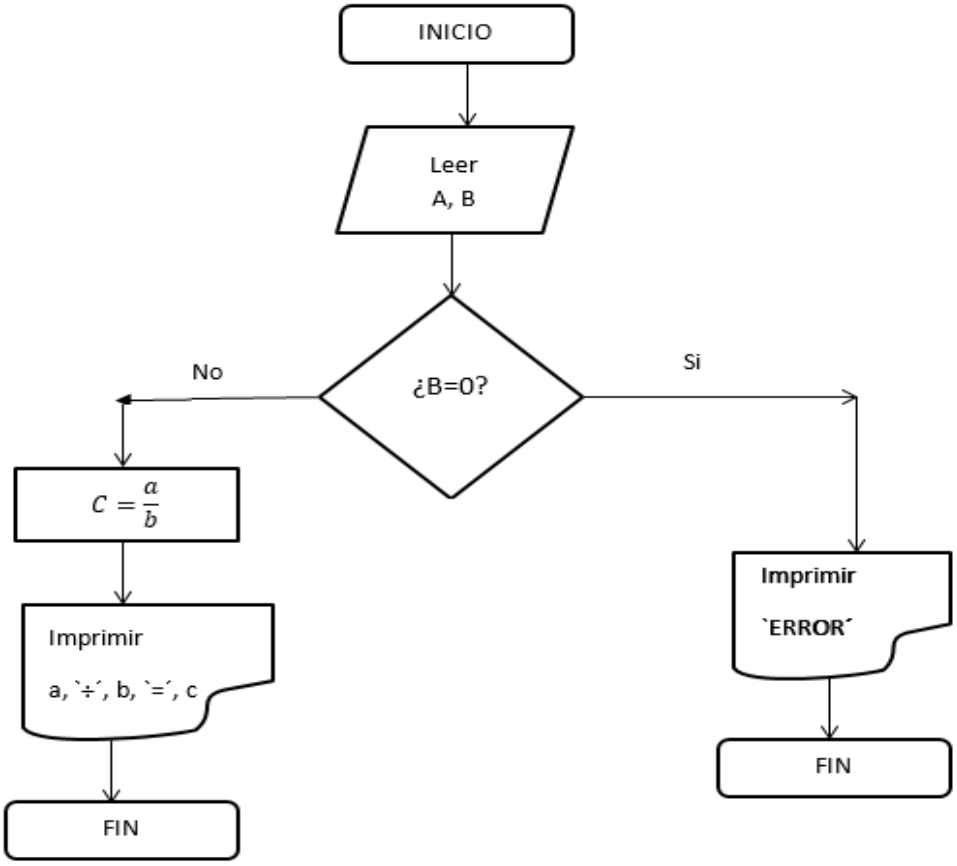
La sentencia *else* es opcional. Si no se utiliza el programa seguirá con la ejecución en la siguiente instrucción después del cierre del if. A continuación, se desarrollan ejemplos con la aplicación de esta sentencia.

Ejemplo 5. Realizar un programa para dividir dos números introducidos desde el teclado por el usuario.

Planteamiento

El programa solicitará al usuario dos números, primero el dividendo y después el divisor, cuando ya conoce los valores realizará la división, tomando en cuenta que cuando el divisor valga cero, no se realizará la división y se debe enviar un mensaje de error.

El ***diagrama de flujo*** es el siguiente:



Algoritmo

De acuerdo con el diagrama de flujo, el algoritmo es el siguiente:

Algoritmo

1.- Inicio
2.-Leer A y B
3.- ¿b=0?
    Sí= imprime “error y termina el programa
    No=Ir a paso 4
4.-“Imprimir la división de”, A, ‘÷’, B, “es”, C.
5.-Fin

Seudocódigo

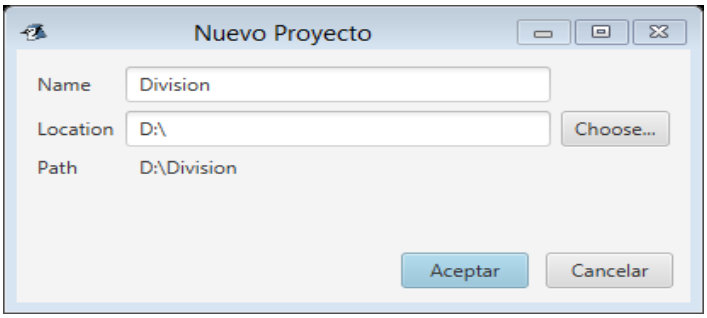
A continuación, pasamos del algoritmo a seudocódigo, destacando la forma en que utilizamos la sentencia condicional.

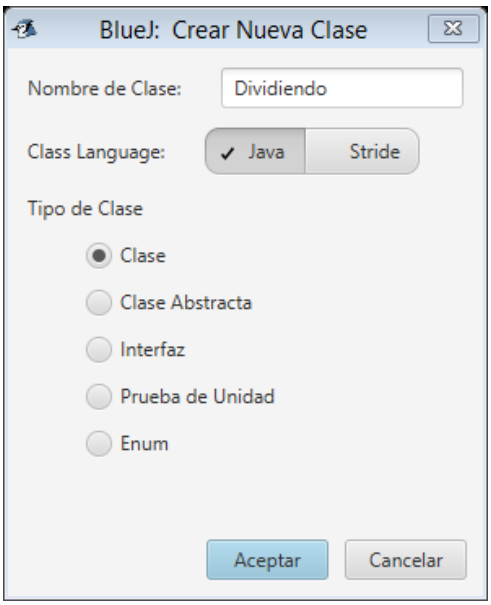
Inicio
Declarar Variables
Leer A,B y C

```
if(B==0) {
    Escribir ("Operación invalida");
}
Else{
   Escribir ("C= " + A/B);
}
Fin
```

Desarrollo del programa

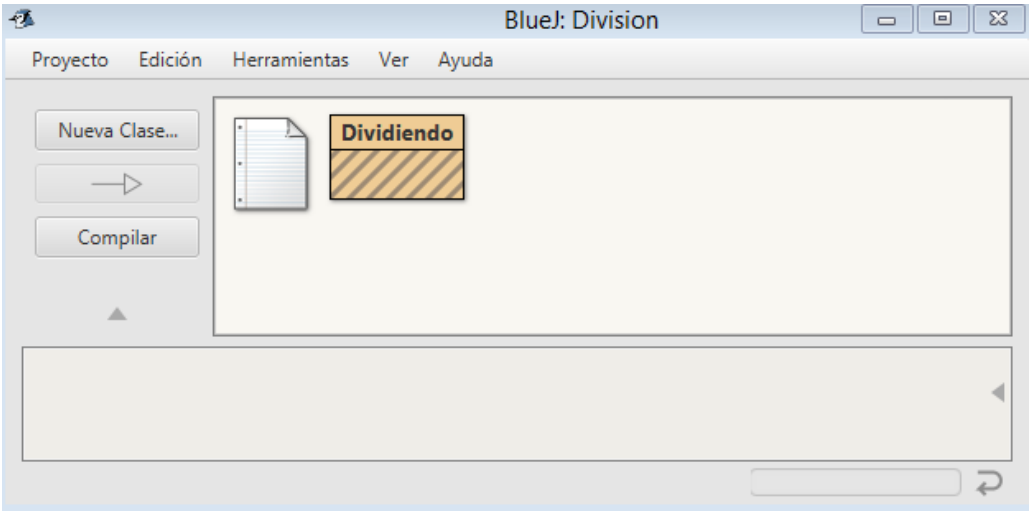
Accesamos a BlueJ, abrimos un nuevo proyecto y creamos la clase dividiendo





Creamos la clase dividiendo

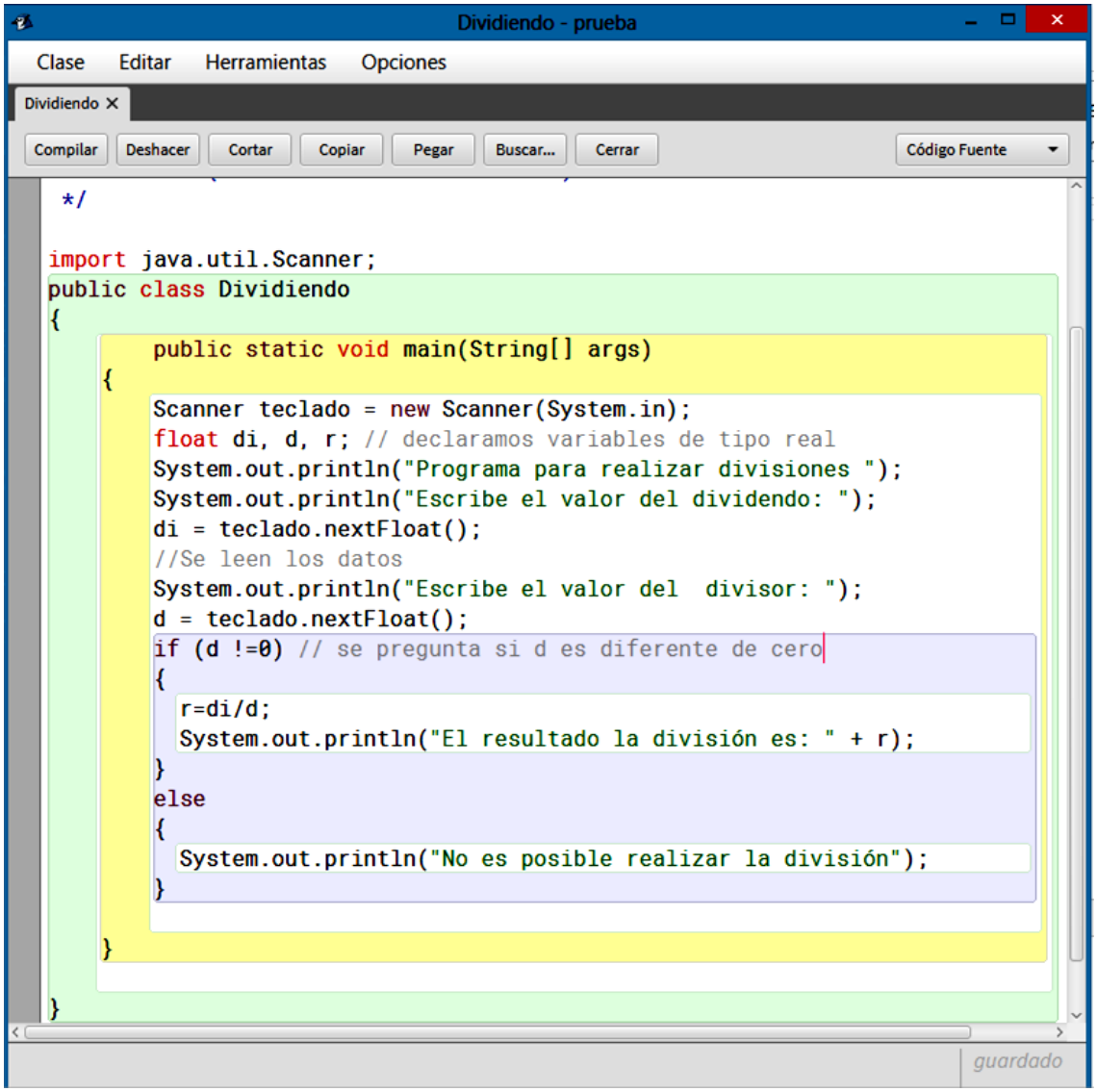
Se genera la siguiente pantalla:



Capturamos el código correspondiente al programa

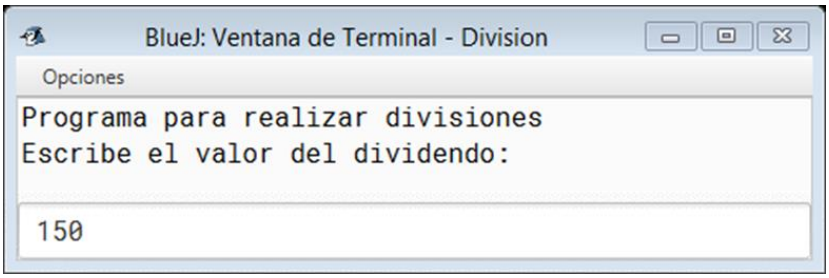
```
import java.util.Scanner;
public class Dividiendo
{
        public static void main(String[] args)
  {
    Scanner teclado = new Scanner(System.in);
    float di, d, r;
    System.out.println("Programa para realizar divisiones ");
    System.out.println("Escribe el valor del dividendo: ");
    di = teclado.nextFloat();
    System.out.println("Escribe el valor del  divisor: ");
    d = teclado.nextFloat();
    if (d !=0)
    {
     r=di/d;
     System.out.println("El resultado la división es: " + r);
    }
    else
    {
     System.out.println("No es posible realizar la división");
    }
  }
}
```

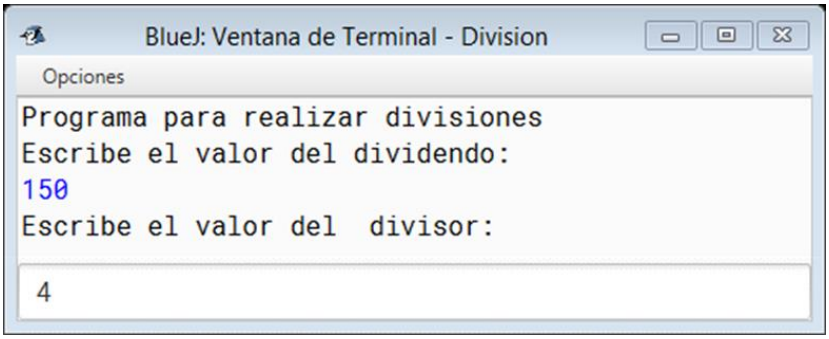
La pantalla con el código capturado es la siguiente:

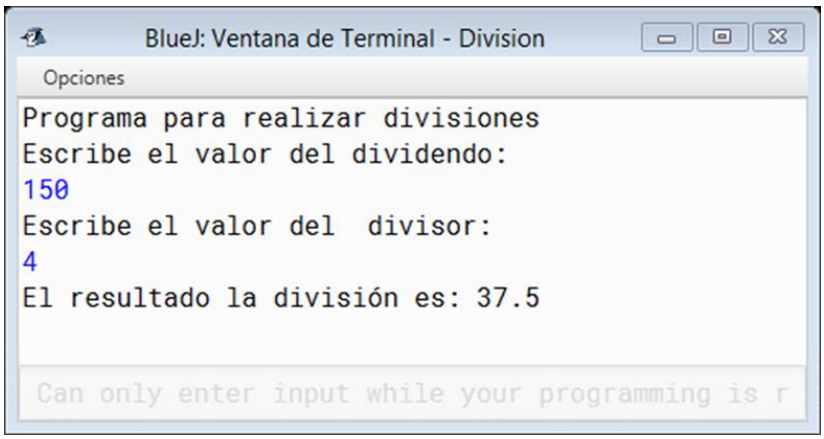


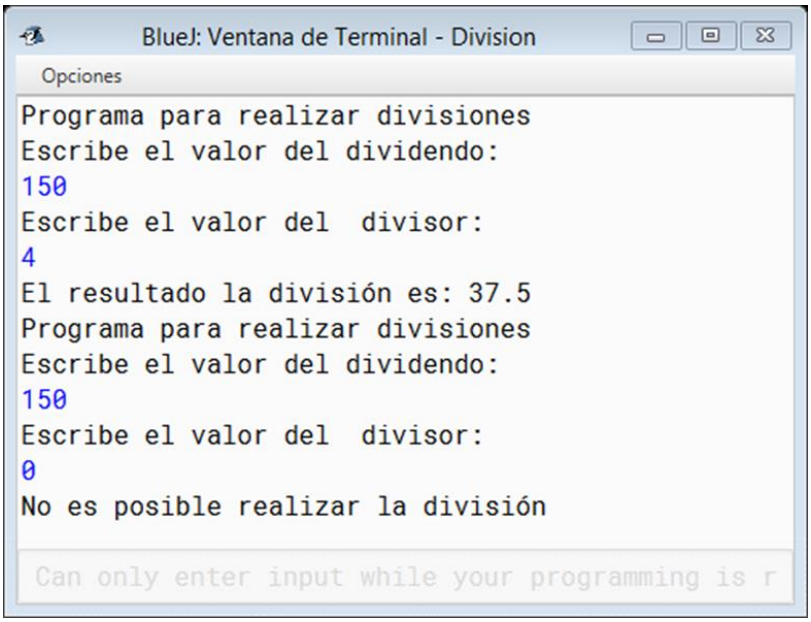
Ejecución de programa

Después de compilar el programa, como lo hemos hecho en los ejemplos anteriores, lo ejecutamos, las pantallas correspondientes son las siguientes:




BlueJ: Ventana de Terminal - Division
Opciones
Programa para realizar divisiones
Escribe el valor del dividendo:
150
Escribe el valor del divisor:
4


BlueJ: Ventana de Terminal - Division
Opciones
Programa para realizar divisiones
Escribe el valor del dividendo:
150
Escribe el valor del divisor:
4
El resultado la división es: 37.5
Can only enter input while your programming is r


BlueJ: Ventana de Terminal - Division
Opciones
Programa para realizar divisiones
Escribe el valor del dividendo:
150
Escribe el valor del divisor:
4
El resultado la división es: 37.5
Programa para realizar divisiones
Escribe el valor del dividendo:
150
Escribe el valor del divisor:
0
No es posible realizar la división
Can only enter input while your programming is r

3

Sentencias Condicionales Anidadas

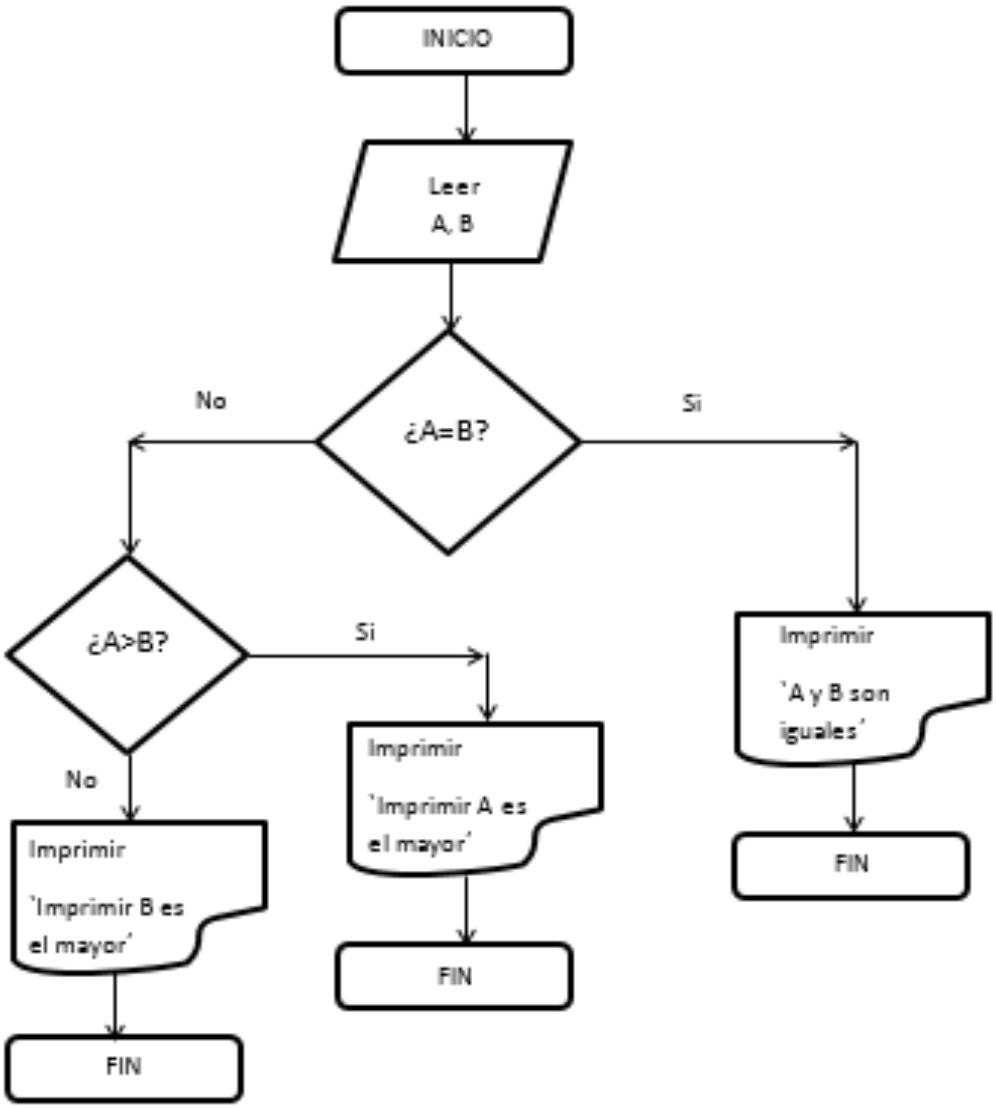
La sentencia if es anidada si la sentencia verdadera o la falsa es a su vez una sentencia if, ésta sentencia implementa decisiones que tienen una o dos alternativas, se usa comúnmente cuando se requiere implementar la toma de decisiones con varias alternativas.

Ejemplo 6.

Elaborar un programa que compare dos números enteros introducidos desde el teclado e imprima cual es el mayor o si son iguales.

El planteamiento: Desarrollaremos un programa que pida al usuario introducir dos números enteros desde el teclado, el programa tiene que determinar, en base a comparaciones, cuál de los dos números es el mayor de los dos o si son iguales y finalmente imprimir el resultado.

El ***diagrama de flujo*** es el siguiente:



El algoritmo es el siguiente:

Algoritmo

0. *Inicio*
1. *Leer A, B*
2. *¿A es igual a B?*
    a. *Si: Imprimir "A y B son números iguales", Ir a Fin.*
    b. *No: ir al paso 3.*
3. *¿A es mayor a B?*
    a. *Si: Imprimir "A es el mayor", Ir a Fin.*
    b. *No: Imprimir "B es el mayor", Ir a Fin.*
4. *Fin*

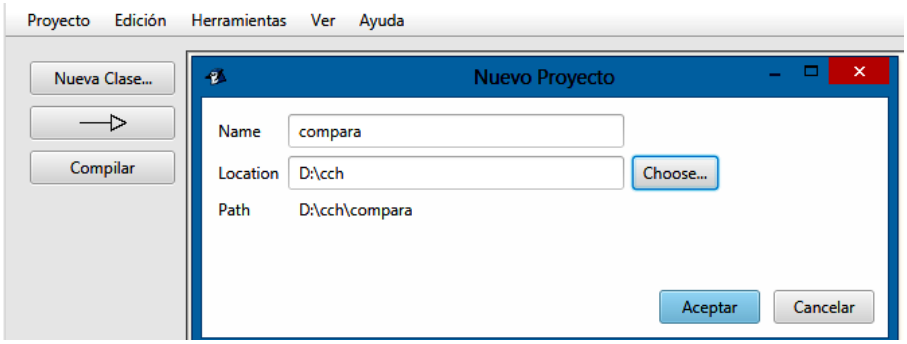
Seudocódigo

A partir del algoritmo realizado, escribimos el seudocódigo, en donde escribiremos la sentencia *if..else*, tal como la utilizaremos en el programa a desarrollar.

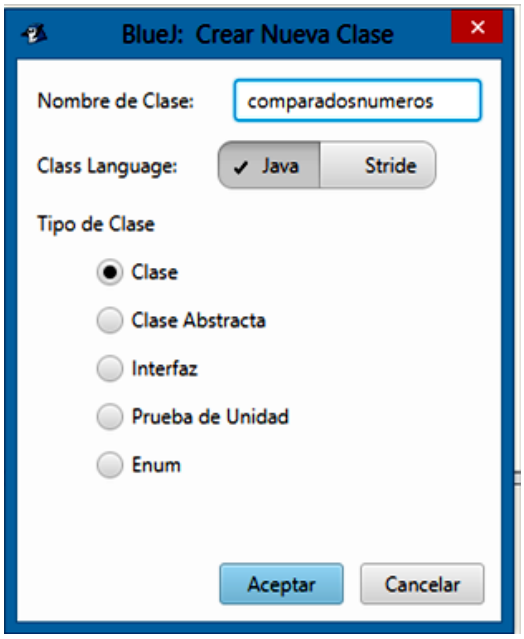
```
Inicio
Declarar variables;
Solicitar entrada de las variables;
If (A = B){
Escribir ““A y B son números iguales”
}
else{
        If (A > B)
        {
        Escribir “A es el mayor”
        }
        else{
        Escribir “B es el mayor”
        }
}
Fin
```

Desarrollo del programa

Primero Abrimos un nuevo proyecto:

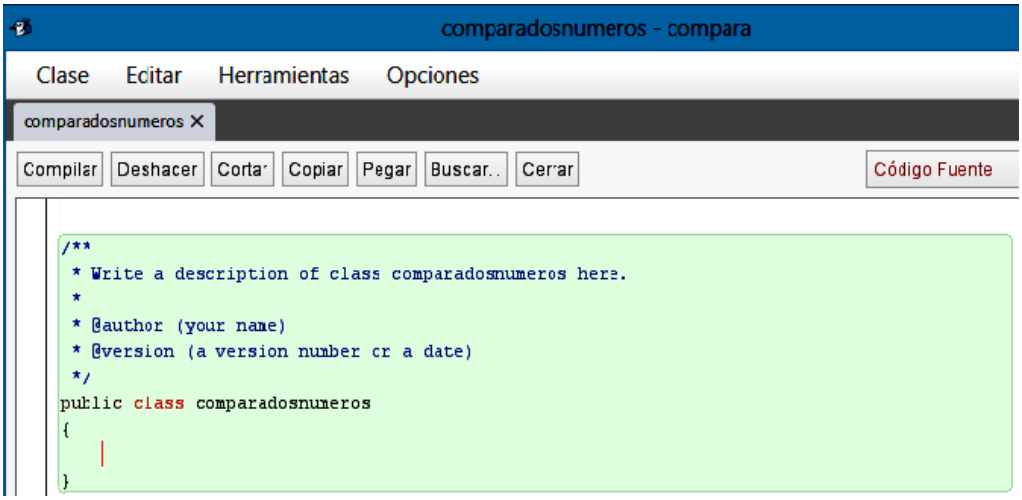


Creamos una nueva clase:



Seleccionamos aceptar y aparecerá la siguiente pantalla:

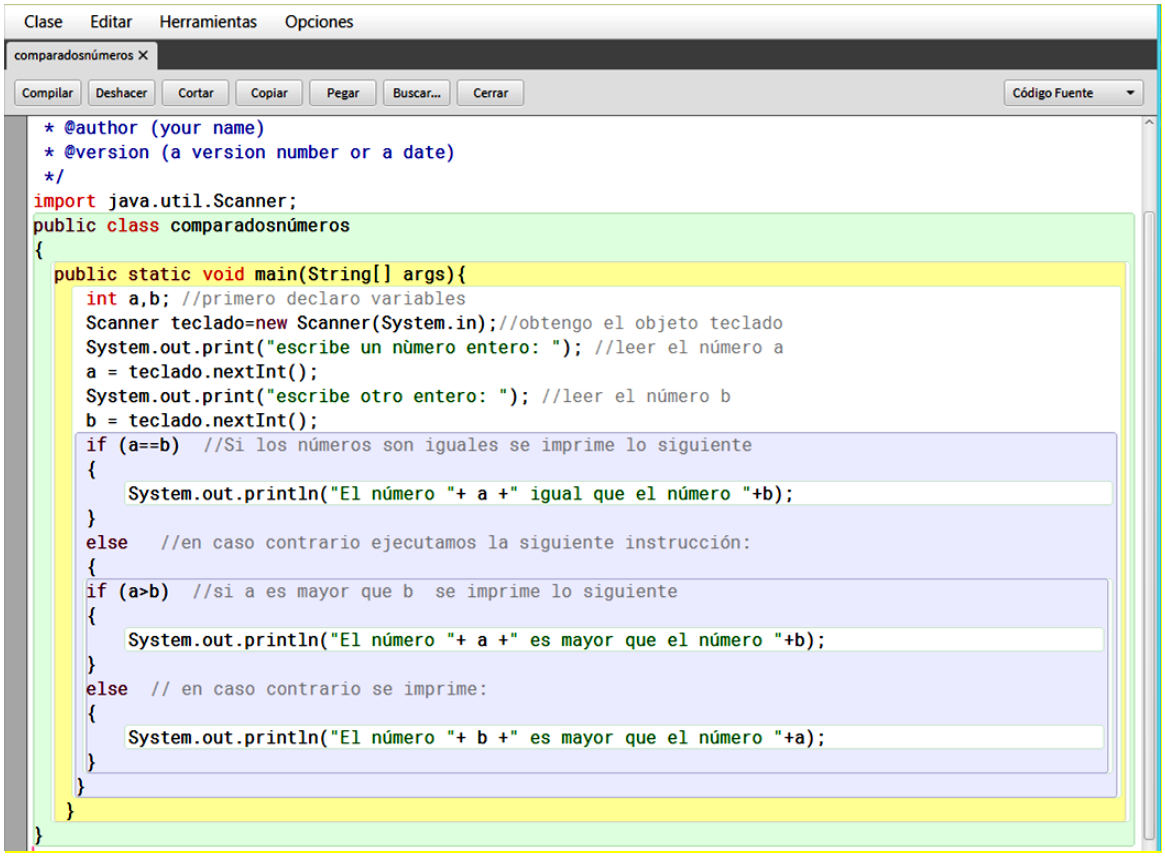
Damos doble clic sobre el icono de la clase y quitamos el código que no necesitamos y obtenemos la siguiente ventana:



Ahora escribiremos el código fuente de nuestro programa:

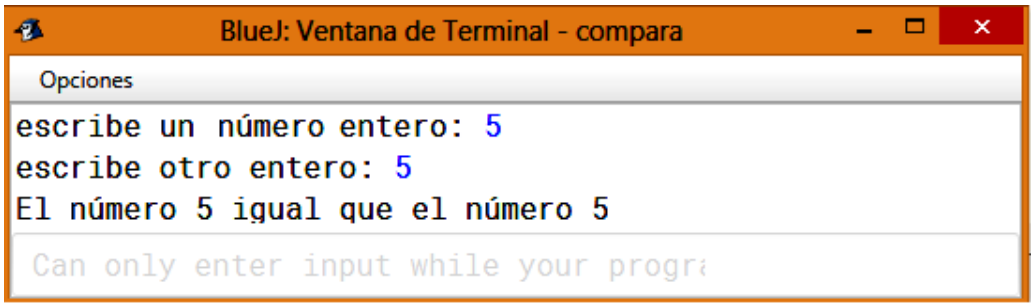
```
import java.util.Scanner;
public class comparadosnúmeros
{
 public static void main(String[] args){
   int a,b; //primero declaro variables
   Scanner teclado=new Scanner(System.in); //obtengo el objeto teclado
   System.out.print("escribe un nùmero entero: "); //leer el número a
   a = teclado.nextInt();
   System.out.print("escribe otro entero: "); //leer el número b
   b = teclado.nextInt();
   if (a==b)  //Si los números son iguales se imprime lo siguiente
   {
     System.out.println("El número "+ a +" igual que el número "+b);
   }
   else   //en caso contrario ejecutamos la siguiente instrucción:
   {
   if (a>b)  //si a es mayor que b  se imprime lo siguiente
   {
     System.out.println("El número "+ a +" es mayor que el número "+b);
   }
   else  // en caso contrario se imprime:
   {
     System.out.println("El número "+ b +" es mayor que el número "+a);
   }
  }
 }
}
```

La pantalla con el código capturado es:

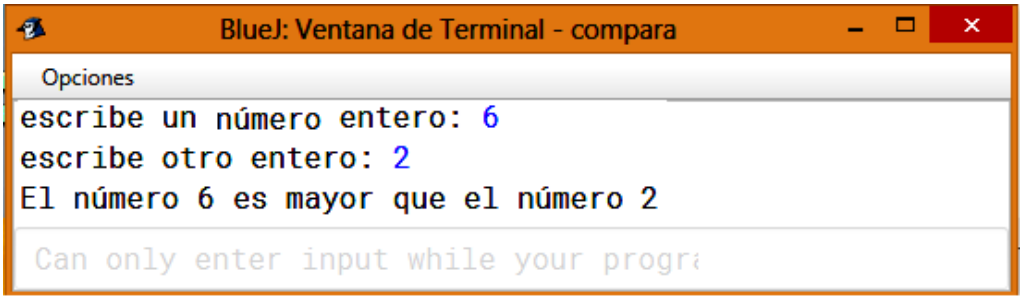


Después de compilar y ejecutar nuestro programa, como lo hemos hecho en los ejemplos anteriores, se mostrarán los siguientes resultados:

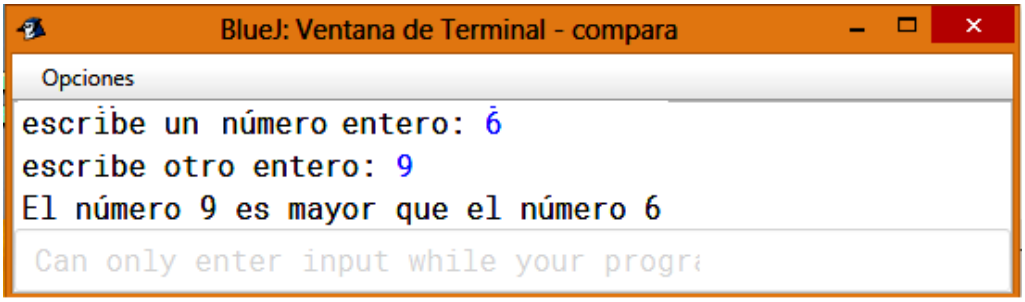
Cuando los números son iguales:



Cuando el primer número es mayor



Cuando el segundo número es mayor

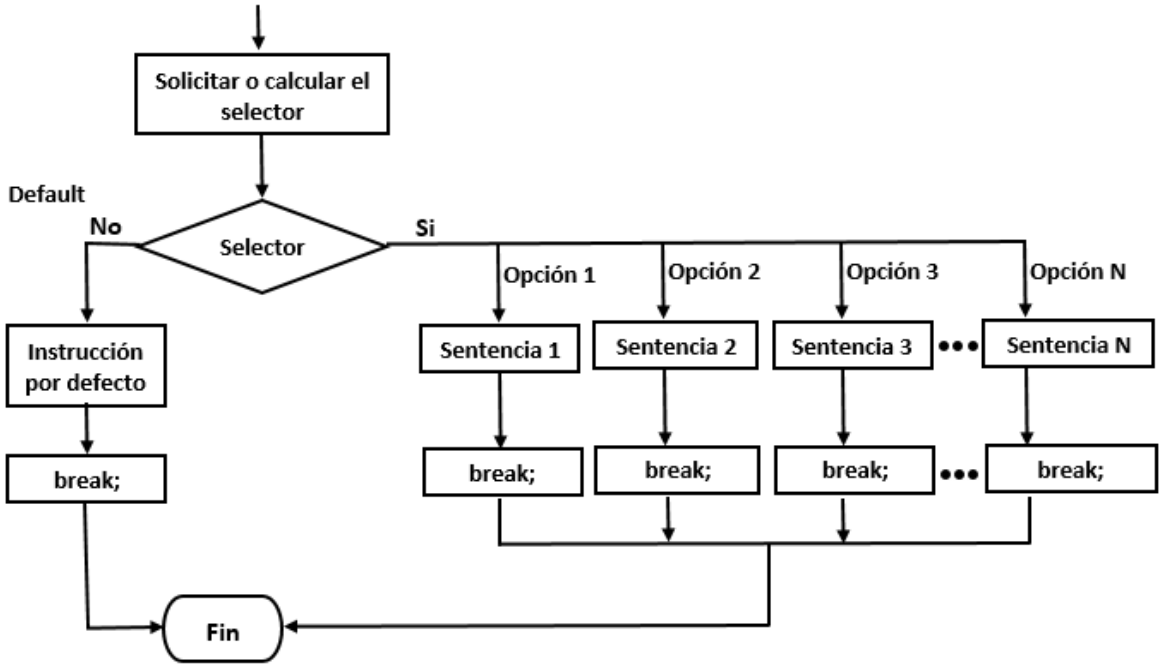


Sentencia Switch

La sentencia *switch* permite ejecutar diferentes bloques de instrucciones en función del resultado de la evaluación de una expresión.

Switch es una sentencia que nos permite evaluar un valor y funciona muy parecido a la sentencia If, sin embargo con el If nos podemos evaluar una o más condiciones y tomar uno o dos caminos. En el caso del Switch nos permite evaluar el valor de una variable para poder tomar varios caminos, por lo tanto podemos decir que esta sentencia es de selección múltiple, esto es, permite elegir una opción de varias.

Diagrama de flujo



La sintaxis es la siguiente:

switch (<expresión>) {
case <valor>:
    <bloque de instrucciones>;
    break;
case <valor>:
    <lista de sentencias separadas por punto y coma>;
    break;
…
default:
    <lista de sentencias separadas por punto y coma>;
}

En donde:

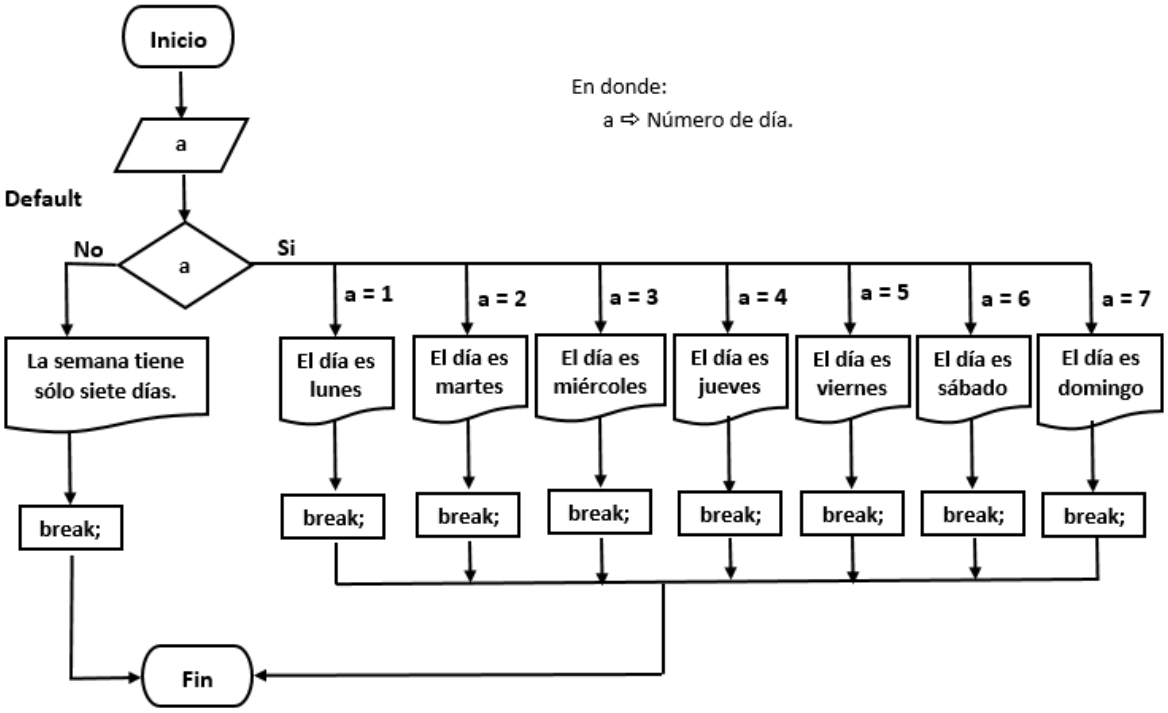
- ✓ El tipo de variable de la <expresión> debe ser de tipo entero (int) o caracter (char).
- ✓ El tipo de variable de la <expresión> y el <valor> deben coincidir.
- ✓ La cláusula *default* es opcional. Puede haber tantas cláusulas *case* como se requiera.
- ✓ El <valor> no puede ser una expresión, sólo puede ser un literal.

Al llegar a la sentencia *switch*, se evalúa la expresión y el resultado se compara con cada <valor> consecutivamente, hasta encontrar uno que coincida, cuando coincide un <valor>, se ejecutan las sentencias de esa cláusula.

La sentencia *default* es opcional se utiliza en caso de que ninguno de los valores coincida con el proporcionado, esto es, si algún valor introducido por el usuario no se encuentra en ninguno de los casos.

Ejemplo 7. Crear un programa que solicite un número que represente el día de la semana y devuelva el nombre del día.

Planteamiento: Se mostrará un mensaje en pantalla solicitando al usuario que escriba un número entre uno y siete, a continuación, se escribirá el nombre del día de la semana que corresponda a ese número. Si se escribe un número que no corresponda a un día de la semana, se desplegará un mensaje de error.
El **diagrama de flujo** es el siguiente:



Algoritmo.

0. *Inicio*
1. *Leer día*
    1: *Imprimir "Lunes"*

2: Imprimir "Martes"
3: Imprimir "Miércoles"
4: Imprimir "Jueves"
5: Imprimir "Viernes"
6: Imprimir "Sábado"
7: Imprimir "Domingo"
Otro valor: imprimir "La semana sólo tiene siete dìas"

2. *Fin*

Seudocódigo

El seudocódigo es el siguiente:

```
Inicio
Leer "día"
Switch (día) {
   case 1:Escribir ("Lunes");
   break;
   case 2: Escribir ("Martes");
   break;

   case 3:Escribir ("Miércoles");
   break;
   case 4: Escribir ("Jueves");
   break;
   case 5:Escribir ("Viernes");
   break;
   case 6: Escribir ("Sábado");
   break;
   case 7: Escribir ("Domingo");
   break;
default
      Escribir ("La semana tiene sólo siete días");
  break;
Fin
```

Como se puede observar en las diferentes formas de solución presentadas, se lee la variable *dia* y puede tomar 7 valores posibles, entre 1 y 7, en caso de que no se proporcione alguno de estos, se imprimirá la palabra *Error*, cada vez que se cumpla la condición la sentencia *break* hará que el programa salga del *Switch.*

Desarrollo del programa

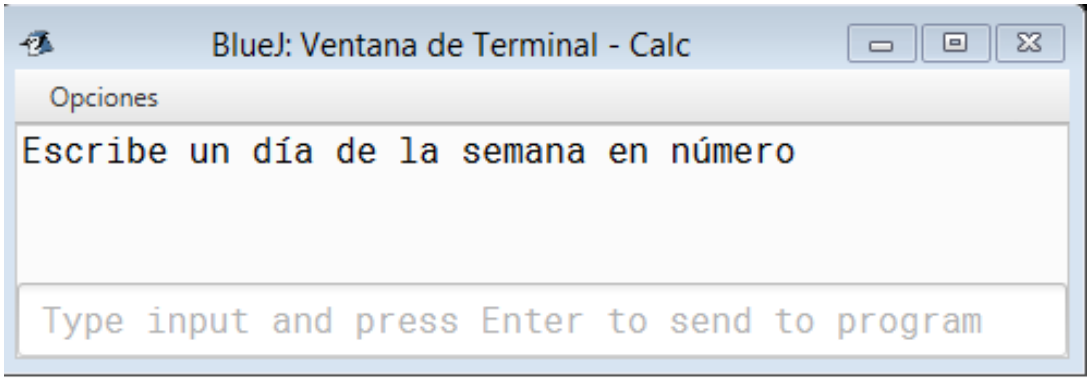
A continuación, escribimos el código del programa fuente. El código es el siguiente:

```
import java.util.Scanner;
public class Dìa
{
    public static void main (String [] args){
    int a;
    Scanner leer=new Scanner(System.in);
    System.out.println("Escribe un día de la semana en número");
    a=leer.nextInt();
    switch (a)
    {
        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");
        break;
      }

   }
}
```

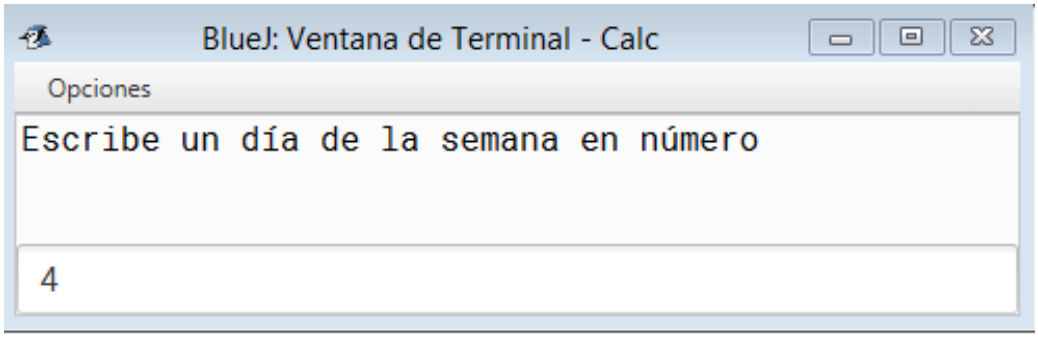
El código se muestra en la siguiente pantalla:

```
Clase  Editar  Herramientas  Opciones
Dia ×
Compilar  Deshacer  Cortar  Copiar  Pegar  Buscar...  Cerrar  Código Fuente
 */
import java.util.Scanner;
public class Dia
{
    public static void main (String [] args){
    int a;
    Scanner leer=new Scanner(System.in);
    System.out.println("Escribe un día de la semana en número");
    a=leer.nextInt();
    switch (a)
    {
        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("Ese día no existe");
        break;
      }

    }
}
guardado
```

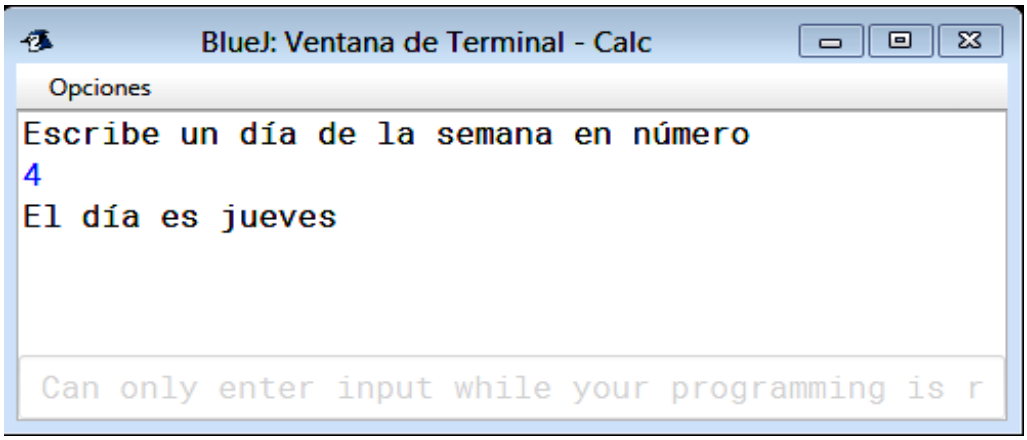
Ejecución del programa. Compilamos y ejecutamos el programa como se ha hecho en los ejemplos anteriores. A continuación, se muestran las pantallas correspondientes:
Se pide al usuario un número



El usuario escribe el número 4



El programa imprime *jueves*



Ejemplo 8. Realiza un programa que permita elegir una de las cuatro operaciones básicas.

Planteamiento. Desarrollaremos un programa que permita al usuario elegir alguna de las 4 operaciones básicas: suma, resta, multiplicación y división, después que lea dos números introducidos desde el teclado y ejecute la operación elegida.

Para facilitar el desarrollo de este programa es necesario comprender como expresamos las operaciones básicas en código de java:

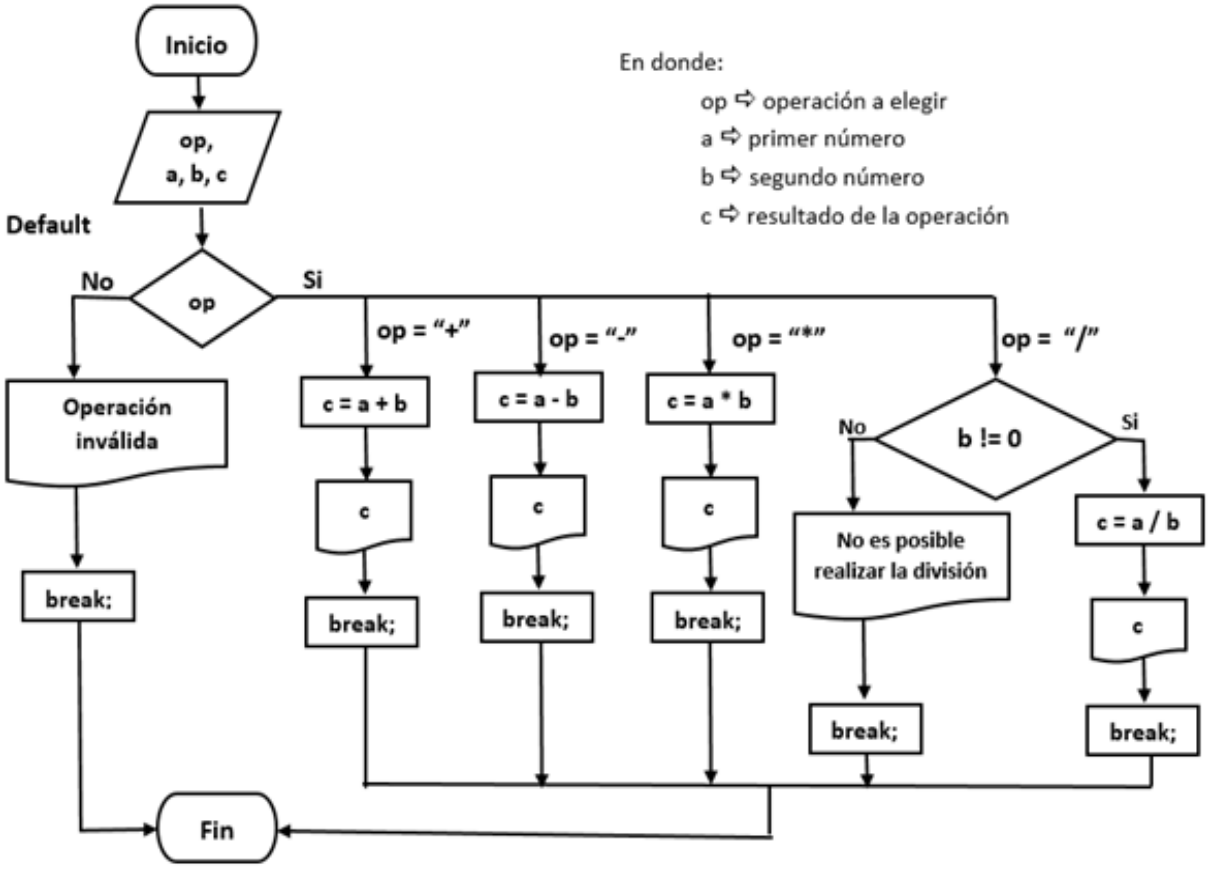
Suma: c= a+b

Resta: c=a-b

*Multiplicación: c=a*b*

División: c=a/b

El diagrama de flujo de la solución del problema es el siguiente:



Algoritmo.

El algoritmo es el siguiente:

0. *Inicio*
1. *Leer variable de operación.*
   "+": Suma. Ir al paso 2.
   "-": Resta. Ir al paso 2.
   "": Multiplicación. Ir al paso 2.*
   "/": División. Ir al paso 2.
   5: Ir a Fin.
2. *Leer el número a*
3. *Leer el número b*
4. *Realizar la operación seleccionada*
5. *Imprimir resultado de la operación.*
6. *Fin*

Seudocódigo

Inicio

Declarar variables;
Solicitar variable de seleccion; (operacion)
Leer los números a y b;

Switch(operacion)
{
case "+":
c = a+b;
Escribir ("La suma de: " a "y" b "es: " c);
break;

case "-":
c = a-b;
Escribir ("La resta de: " a "y" b "es: " c);
break;

case "*":
c = a*b;
Escribir ("La multiplicación de: " a "y" b "es: " c);
break;

case "/":
c = a/b;
if(b!=0){
c=a/b;
Escribir("La división es: "+ c);
}
else
{
Escribir ("No es posible realizar la división");
}
Escribir ("La división de: " a "y" b "es: " c);

break;
default:
Escribir ("Operación invalida");
break;
}
Fin.

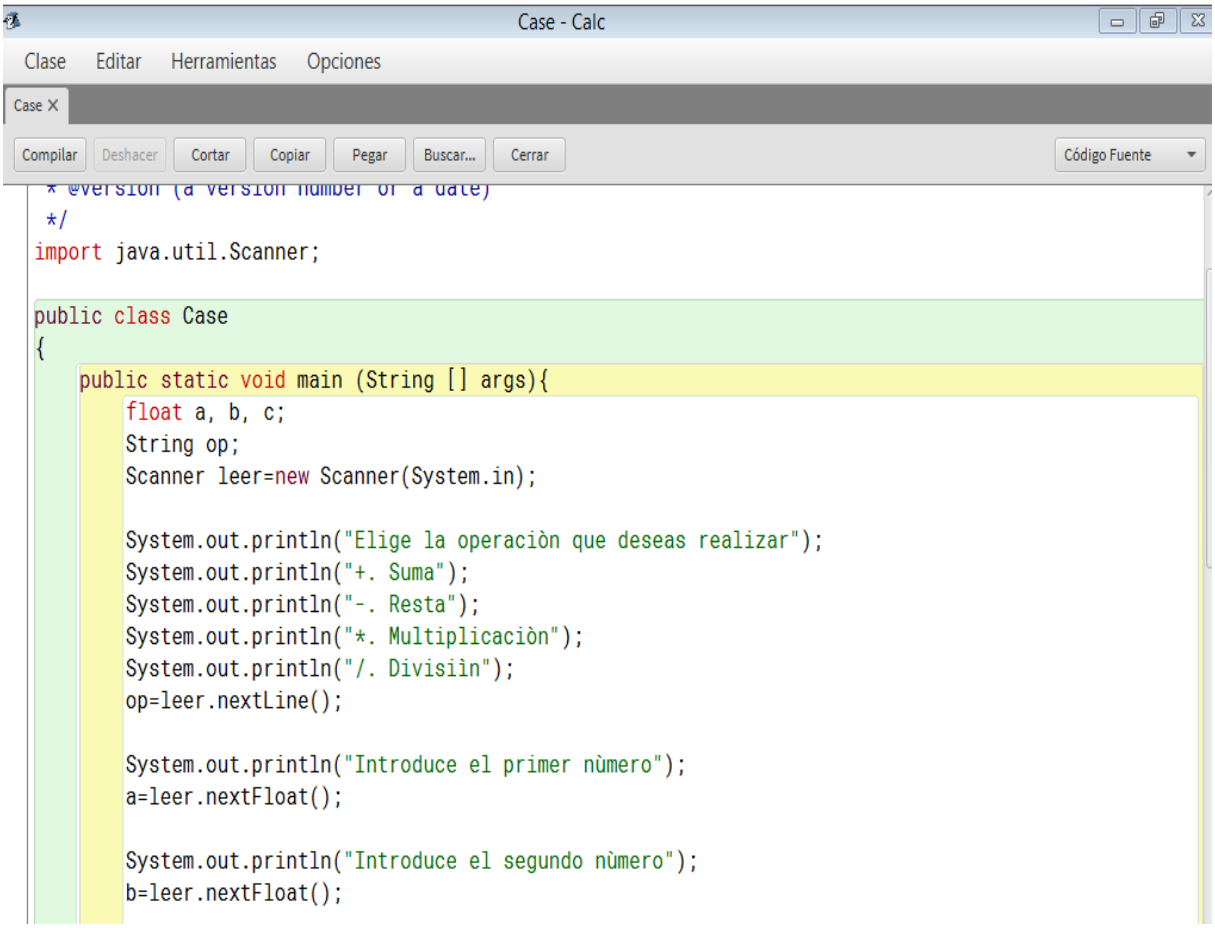
Desarrollo del programa.

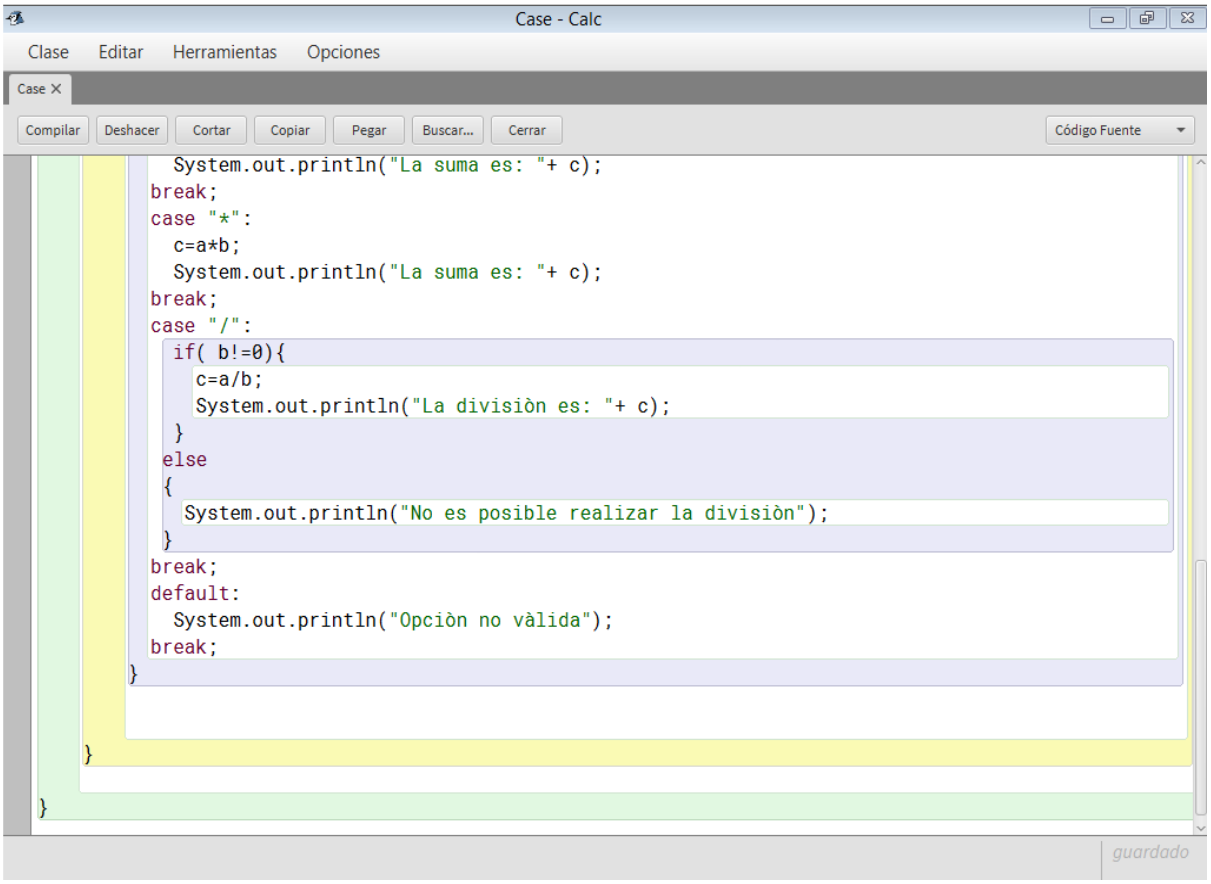
A continuación, se muestra el código del programa fuente.

```
import java.util.Scanner;
public class Case
{
   public static void main (String [] args){
      float a, b, c;
      String op;
      Scanner leer=new Scanner(System.in);

      System.out.println("Elige la operación que deseas realizar");
      System.out.println("+. Suma");
      System.out.println("-. Resta");
      System.out.println("*. Multiplicación");
      System.out.println("/. División");
      op=leer.nextLine();

      System.out.println("Introduce el primer número");
      a=leer.nextFloat();

      System.out.println("Introduce el segundo número");
      b=leer.nextFloat();

      switch (op)
      {
       case "+":
         c=a+b;
         System.out.println(String.format("%.2f", c));
         System.out.println("La suma es: "+ c);
       break;
       case "-":
         c=a-b;
         System.out.println("La resta es: "+ c);
       break;
       case "*":
         c=a*b;
         System.out.println("La multiplicación es: "+ c);
       break;
       case "/":
         if( b!=0){
           c=a/b;
```

```
          System.out.println("La división es: "+ c);
         }
        else
        {
          System.out.println("No es posible realizar la división");
        }
       break;
       default:
         System.out.println("Opción no válida");
       break;
      }

   }

}
```

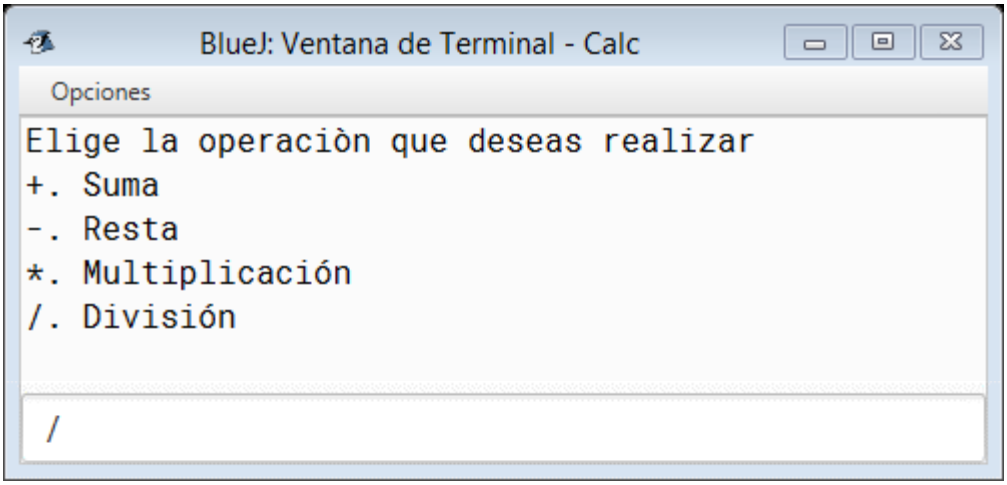
Capturamos el código y a continuación se muestran las pantallas correspondientes:

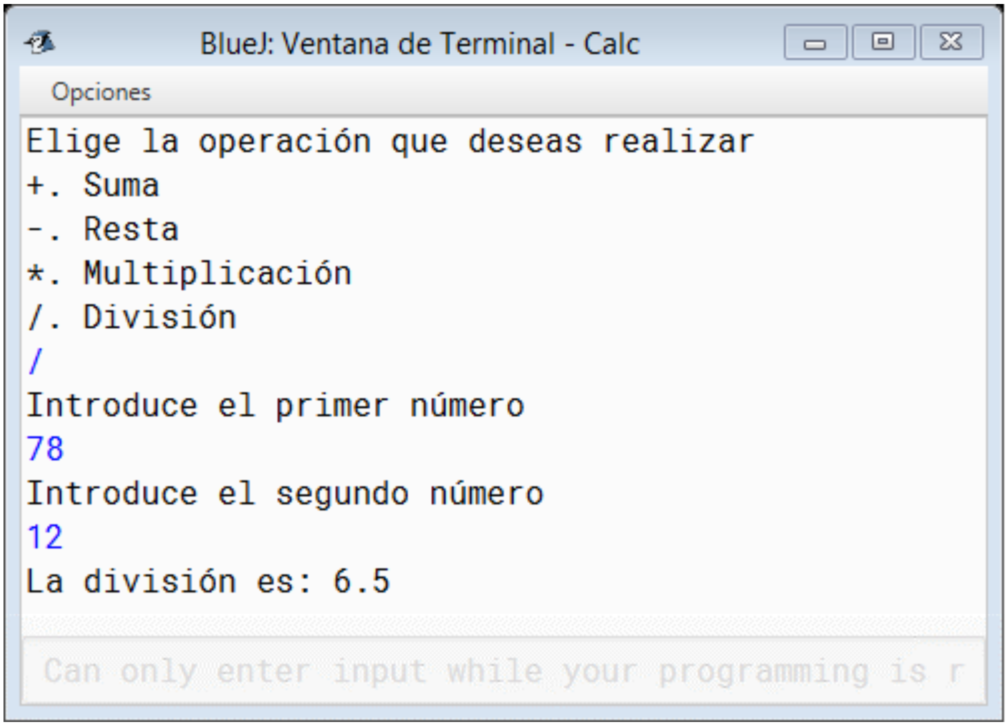




Ejecución del programa.

Después de compilar el programa los ejecutamos, a continuación se muestran las pantallas correspondientes:





Ejercicio

I. **Instrucciones.** Evalúa en Verdadero (V) o Falso (F) las siguientes expresiones:

1. La sentencia *switch* permite ejecutar diferentes bloques de instrucciones en función del resultado de la evaluación de una expresión. --- ( )
2. Una expresión booleana se forma comparando valores utilizando operadores relacionales operadores lógicos. ------------------- ( )
3. El tipo de variable de la <expresión> y el <valor> pueden ser diferentes. ( )
4. La cláusula *default* es opcional. ----------------------------- ( )
5. Existe un número limitado de cláusulas *case* . ------------------ ( )

II. **Instrucciones.** Elige la opción correcta a las siguientes cuestiones.

1. La estructura de control if .. else permite:

a. Elegir una alternativa entre dos o más opciones para que un conjunto de órdenes se ejecute.
b. Elegir una alternativa entre a lo más dos opciones para que un conjunto de órdenes se ejecute.

c. Que una secuencia de órdenes se ejecute independiente del valor de la condición.
d. Ejecutar un programa el número de veces que lo que se desee.

2. ¿Cuál es la condición correcta, para que un programa determine si la edad de una persona es mayor a 18 años y menor a 60? Considera la variable E como edad.

a. (E<18 && E>60)
b. (E>18 || E<60)
c. (E>18 && E<60)
d. (E<18 || E>60)

3. ¿Qué operador lógico se debe utilizar si se requiere que una de dos condiciones se cumpla?

a. &&
b. ||
c. !
d. ==

4. Esta sentencia es opcional para la sentencia *switch:*

a. default
b. case
c. break
d. expresión

5. ¿Cuántas alternativas de clausula case puede tener la sentencia *switch*?

a. 7
b. 12
c. 10
d. Ilimitado

TEMA **14**

Estructuras de ciclos

Objetivos:

- Elaborar el algoritmo, diagrama de flujo y seudocódigo para resolver problemas de estructuras de ciclo
- Construir programas de computadora que resuelvan problemas que empleen la sentencia for
- Elaborar el algoritmo, diagrama de flujo y seudocódigo para resolver problemas de ciclo que satisfagan una condición
- Construir programas de computadora que resuelvan problemas que involucren la sentencia while

1 Introducción

Las sentencias de control repetitivas son sentencias iterativas, también se les conoce como bucles o ciclos, se utilizan frecuentemente, cuando se requiere ejecutar más de una vez un bloque de sentencias. Es decir, se utilizan para evitar programas largos que contienen código repetido. Las sentencias cíclicas son *while, do – while, for.*

2 Sentencia for

El ciclo repetitivo *for* sirve para ejecutar un código o bloque de instrucciones un número conocido de veces, esto es, se conoce el valor inicial y el final, así como el incremento.

Se establece el número o valor inicial desde el que va a empezar ciclo, la condición que mientras se cumpla, permitirá que se ejecuten las sentencias o bloque de instrucciones, finalmente el incremento o decremento, normalmente se maneja el nombre de incremento, pero en caso de ser necesario puede ser decremento.

La sintaxis es:

For (Valor inicial;condición;incremento)

{Instrucciones que vamos a ejecutar mientras se cumpla la condición };

Ejemplo 9.
Desarrollar un programa que imprima la tabla de multiplicar de un número n.

Planteamiento.
El programa debe pedir al usuario un número entero n y después realizar las multiplicaciones de ese número por los números del 1 al 10, esto es:

n x 1=n
n x 2=2n
n x 3=3n
.
.
.
n x 10=10n

Finalmente, el programa debe escribir las operaciones y el resultado correspondientes. El algoritmo correspondiente se muestra a continuación.

Algoritmo

0. Inicio
1. Leer n
2. Desde que x=1 y hasta x<=10
3. m = n * x.
4. escribir la multiplicación y el resultado
5. Fin

Seudocódigo

A continuación, se muestra el seudocódigo, en donde hacemos énfasis en la instrucción ***for***, que es la adecuada para este procedimiento.

Inicio
Declarar variables
Limpiar pantalla
Escribir (“Valor a multiplicar”);
Leer n;
for (x=1;x<=10;x++)// x++ indica que x se incrementa en uno
{
//multiplicamos n por x=1, hasta x=10
m=n*x;
Escribir (n+”*”+x+”=”+m);
}
Escribir (“Programa terminado”);
Fin.

Desarrollo del programa

El *programa fuente* correspondiente al algoritmo y seudocódigo desarrollados anteriormente, es el siguiente:

```
import java.util.Scanner;
public class Tabla_mult
{
   public static void main(String[] args){
     //Declaramos las variables
      int x,n,m;
      Scanner teclado=new Scanner(System.in);
      // Limpia pantalla
     System.out.print('\u000C');
      System.out.print("Valor a multiplicar: ");
      //Se asigna un valor a n desde el teclado
      n=teclado.nextInt();
      System.out.println("La tabla del número "+n+" es:");
      for (x=1;x<=10;x++)
      {
         m=n*x;
         System.out.println(n+"*"+x+"="+m);
      }

   System.out.println("");
   System.out.println("Programa terminado");
}
}
```

Después de capturar el programa veremos la siguiente pantalla:

Tabla_mult - prueba

Clase Editar Herramientas Opciones

Tabla_mult

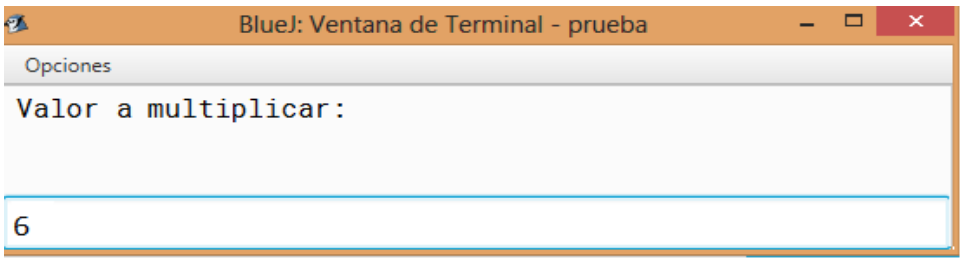
Compilar Deshacer Cortar Copiar Pegar Buscar... Cerrar Código Fuente

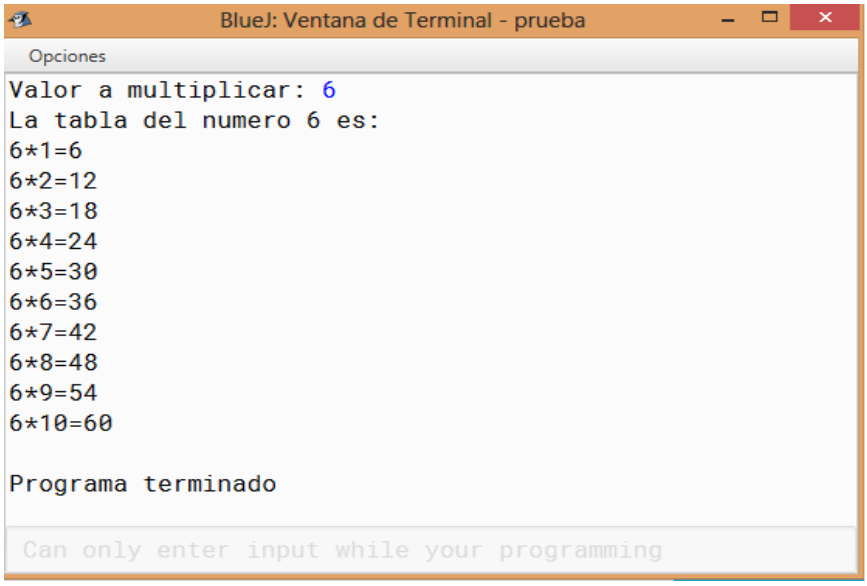
```
import java.util.Scanner;
public class Tabla_mult
{
    public static void main(String[] args){
       //Declaramos las variables
        int x,n,m;
        Scanner teclado=new Scanner(System.in);
        // Limpia pantalla
       System.out.print('\u000C');
        System.out.print("Valor a multiplicar: ");
        //Se asigna un valor a n desde el teclado
        n=teclado.nextInt();
        System.out.println("La tabla del número "+n+" es:");
        for (x=1;x<=10;x++)
        {
            m=n*x;
            System.out.println(n+"*"+x+"="+m);
        }

    System.out.println("");
    System.out.println("Programa terminado");
}
}
```

guardado

Ejecución del programa.

Después de compilar y ejecutar el programa se obtendrán los siguientes resultados:





2 Sentencia while

La sentencia *while* ejecuta un bloque de sentencias mientras una condición sea verdadera. Su sintaxis es:

```
while (expresión) {
    sentencia(s);
}
```

La sentencia *while* evalúa una expresión, mientras esta se cumpla, es decir, siempre que devuelva un valor verdadero, las sentencias o bloque de instrucciones se ejecutarán, el ciclo se detiene cuando la expresión evaluada tenga un valor falso.

Ejemplo 10. Programa que escriba en pantalla los números del 1 al 20 utilizando la instrucción while.

Planteamiento. Es un procedimiento sencillo, se tienen que escribir los números, uno por línea, iniciando con el número uno y terminando con el 20.

Algoritmo

Inicio
Declarar variables
2. i=1;
3. ***mientras*** **(**i<=20)
4. Escribir (i)
5. I=i+1
6. Regresa al paso 3

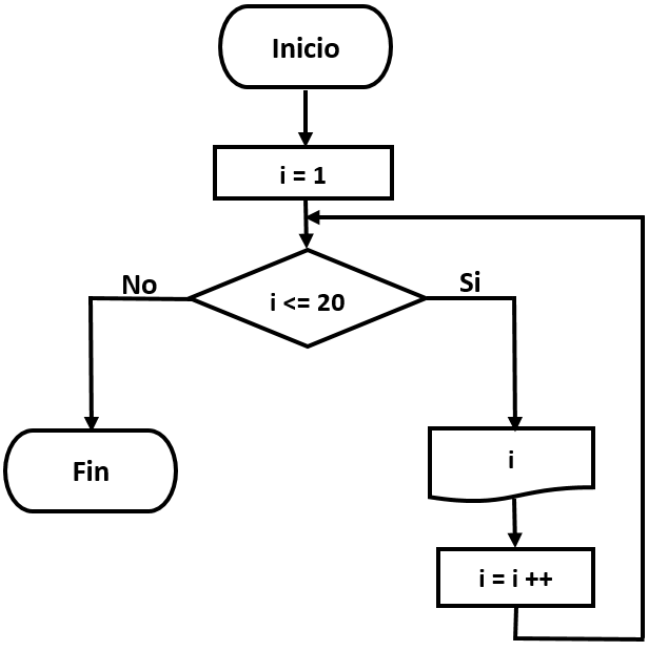
Fin

Seudocódigo

Procedemos a representar el algoritmo anterior con el siguiente seudocódigo:

```
Inicio
      int i=1;
      while(i<=20) {
            System.out.println(i);
            i++; // se ejecuta un autoincremento en i
      }
 Fin
```

Diagrama de flujo

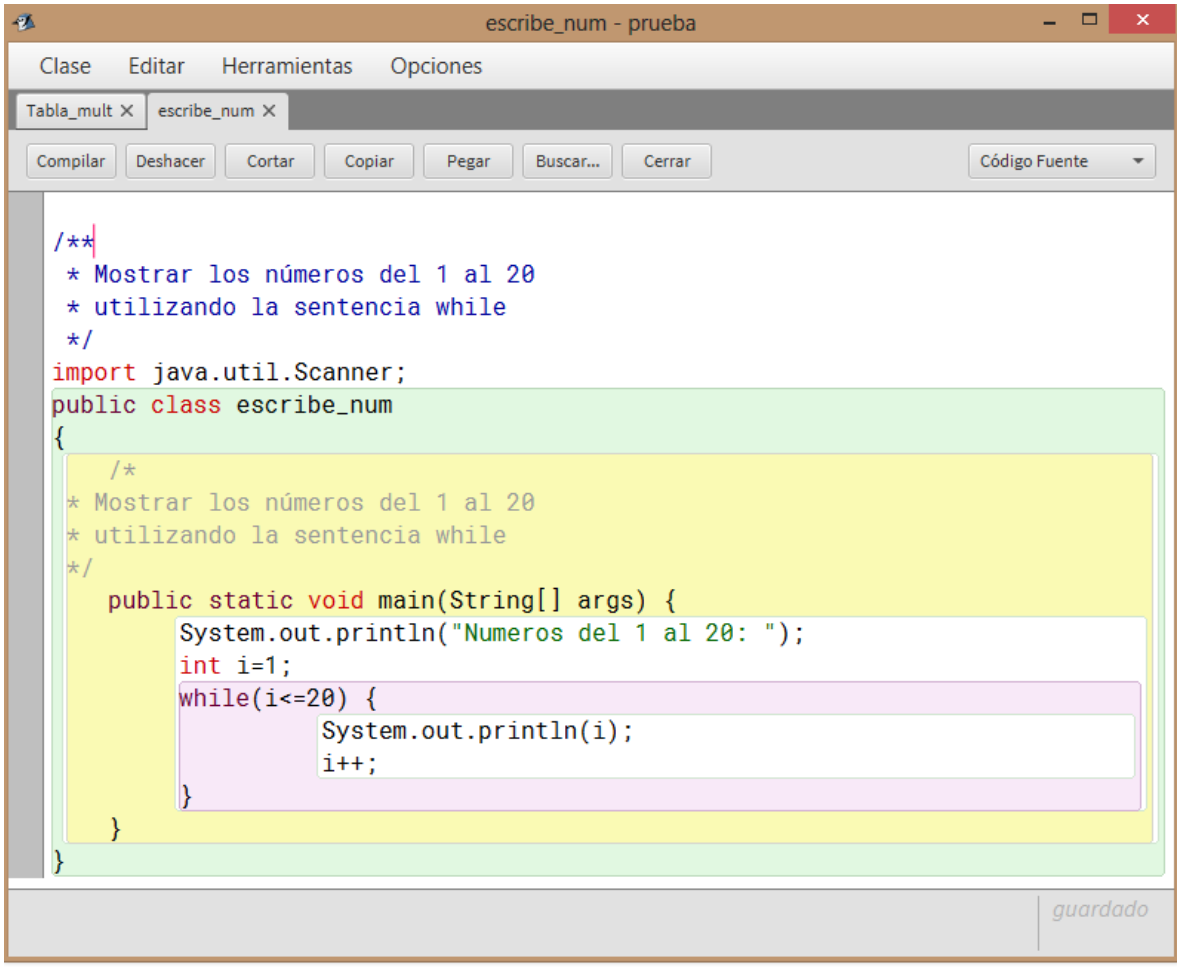


El *programa fuente* es el siguiente:

```
import java.util.Scanner;
public class escribe_num
{
   /*
 * Mostrar los números del 1 al 20
 * utilizando la sentencia while
 */
   public static void main(String[] args) {
       System.out.println("Numeros del 1 al 20: ");
       int i=1;
```

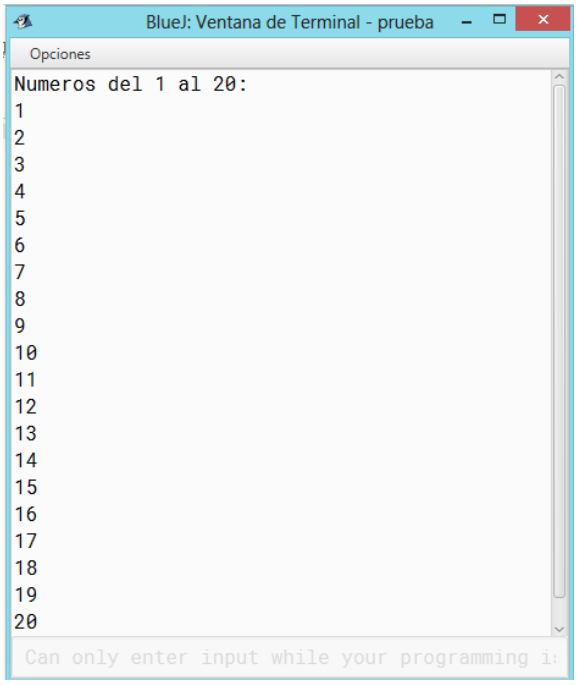
```
        while(i<=20) {
                System.out.println(i);
                i++;
        }
    }
}
```

Las pantallas generadas después de capturar el código son las siguientes:



Ejecución del programa.

Después de compilar y ejecutar el programa obtendremos la siguiente pantalla:



Sentencia do - while

El ciclo *do while* es prácticamente igual al *while*, pero con la diferencia de que el código del ciclo se ejecutará al menos una vez ya que la comprobación se hace después de cada iteración y no antes como en el caso del *while*.

La sintaxis de la sentencia *do-while*, es:

```
do {
    sentencia(s);
} while (expresión);
```

Como se observa la expresión se evalúa al final del ciclo, por lo tanto las sentencias o bloque de instrucciones se ejecutan al menos una vez, esto es en *do – while* el ciclo se ejecuta al menos una vez, después se evalúa la expresión, el ciclo se ejecutará mientras la expresión sea verdadera.

Ejemplo 11.
Desarrollar un programa que realice la pregunta: ¿Desea continuar? Y que termine cuando la respuesta sea n.
Planteamiento. Se le pedirá al usuario que responda si desea continuar, las opciones serán los caracteres *s* o *n*, si el usuario responde *s* se volverá a escribir la pregunta, el programa se detendrá hasta que la respuesta sea diferente de *s.*

Algoritmo.

El algoritmo es el siguiente:

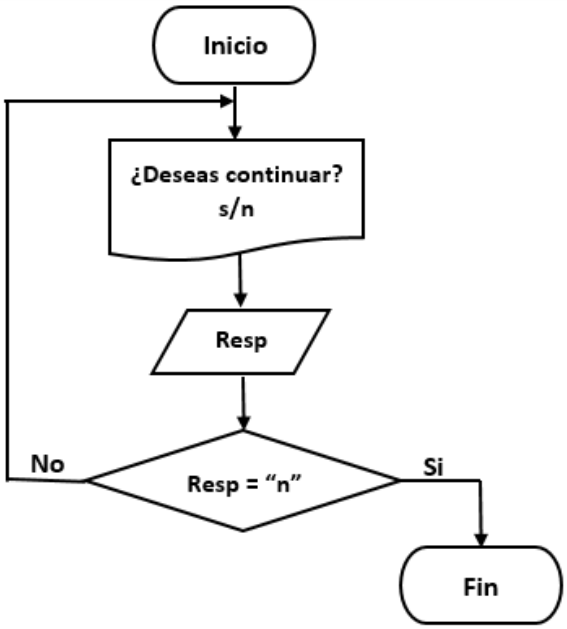
0. **Inicio**
1. Declarar variables
2. Resp= “s”
3. Limpiar pantalla
4. Preguntar ¿Desea continuar s/n?
5. Leer Resp
   ¿Resp=s?:
   a. Si: Ir al paso 3.
   b. No: Imprimir “Programa Finalizado”, Ir a Fin.
6. **Fin**

Seudocódigo.

Procedemos a representar el algoritmo anterior con el siguiente seudocódigo:

```
Inicio
Declarar variables;
Inicializar variable Resp=’s’;
do
{
Limpiar pantalla
Escribir “Desea continuar? s/n”
Leer Resp
    while(Resp = 'n');
        System.out.println("Programa Finalizado...");
}

Fin.
```

Daigrama de flujo

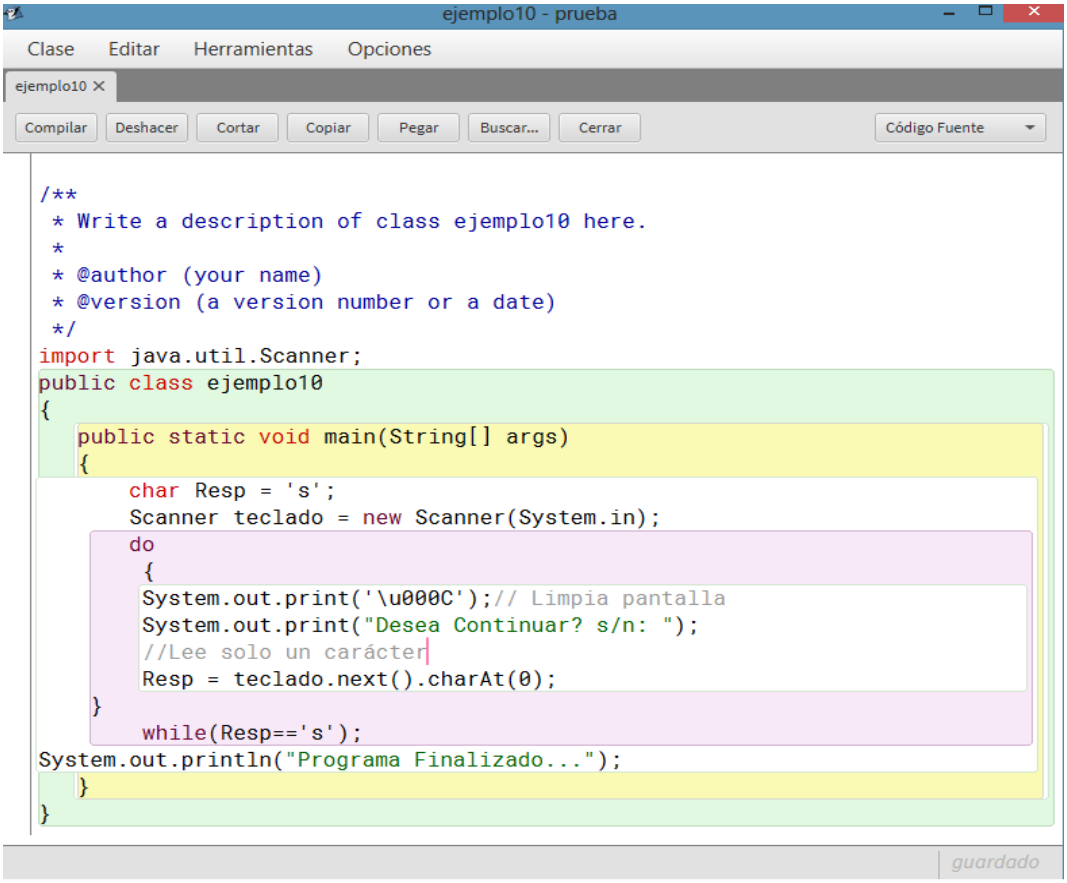


Desarrollo del programa

El *programa fuente* es el siguiente:

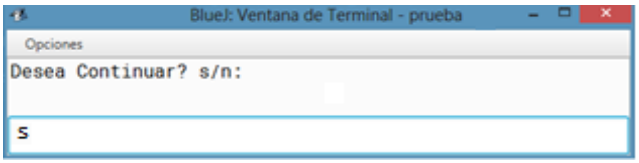
```
import java.util.Scanner;
public class ejemplo10
{
  public static void main(String[] args)
  {
     char Resp = 's';
     Scanner teclado = new Scanner(System.in);
     do
      {
      System.out.print('\u000C');// Limpia pantalla
      System.out.print("Desea Continuar? s/n: ");
      //Lee solo un carácter
      Resp = teclado.next().charAt(0);
   }
      while(Resp=='s');
System.out.println("Programa Finalizado...");
  }
}
```

Después de capturar el programa visualizaremos la siguiente pantalla:

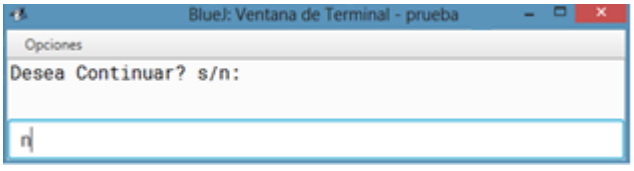


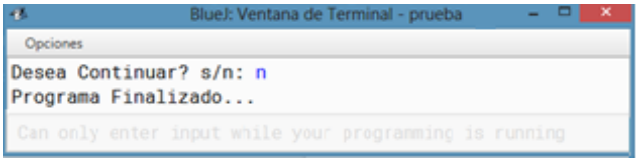
Ejecución del programa.

Después de compilar y ejecutar el programa se obtendrán los siguientes resultados:



Cuando la respuesta es diferente de *s*:





Ejercicio

Instrucciones. Selecciona la opción correcta a lo que se te pide.

1. **Selecciona la opción correcta que corresponda a una de las características de la sentencia for y while**
   a. El ciclo for puede no ejecutarse, pero el while se ejecuta al menos una vez.
   b. El ciclo for se ejecuta un número determinado de veces y el while depende de una condición.
   c. El ciclo while puede funcionar de forma similar al ciclo for pero no al contrario.
   d. El ciclo while permite conocer el valor inicial y el final, pero el ciclo for no.

2. **De acuerdo con la sintaxis del ciclo while:**

```
while (condición) {
        sentencias
      }
```

¿Cuál de las siguientes opciones no corresponde a su definición?

   a. ( ) La ***condición*** es una variable booleana.
   b. ( ) La ***condición*** sólo se evalúa sólo una vez al principio de la ejecución del ciclo.
   c. ( ) Si la ***condición*** es verdadera, se ejecuta el bloque de sentencias, y se vuelve al principio del ciclo.
   d. ( ) Si la ***condición*** es falsa, no se ejecuta el bloque de sentencias.

3. **¿Cuál es el resultado que se obtiene después de ejecutar el siguiente código en java?**

```
int n = 0;
do {
  n++;
  if (n%2 == 0) {
        System.out.println('El número es: '+n);
    }
} while (n < 101);
```

a. ( ) Se imprimen diferentes de 2.

b. ( ) Se imprimen los números diferentes de 2 que sean menores o iguales a 100.

c. ( ) Se imprimen los múltiplos de 2 menores de 100.

d. ( ) Se imprimen los múltiplos de 2 menores o iguales a 100.

4. De acuerdo con la sintaxis del ciclo do-while:

```
do{
   sentencias
} while (condición)
```

¿Cuál de las siguientes opciones no corresponde a su definición?

a. ( ) Si *condición* es verdadera, entonces el ciclo se sigue ejecutando.

b. ( ) El ciclo se ejecuta una vez y después de valida *condición.*

c. ( ) Si *condición* es falsa, el ciclo se ejecuta al menos una vez.

d. ( ) Si *condición* es falsa, el ciclo no se ejecuta.

5. De acuerdo con la sintaxis del ciclo for:

```
for (inicialización ; condición ; actualización) {
  sentencias
}
```

Una de las siguientes opciones no es correcta, ¿Cuál es?

a. ( ) La *inicialización* se realiza cada vez que se ejecuta al ciclo de sentencias.

b. ( ) La *condición* se comprueba cada vez que se ejecuta al ciclo de sentencias.

c. ( ) La *actualización* se realiza siempre al terminar de ejecutar las sentencias.

d. ( ) La *inicialización, condición y actualización* son elementos necesarios para que se ejecute el for.

6. ¿Cuál de estas características de las sentencia “break” no es correcta?

a. ( ) La sentencia break es una etiqueta asociada a la sentencia switch.

b. ( ) El uso de la sentencia break rompe la ejecución de los ciclos repetitivos (while, do-while, for).

c. ( ) La sentencia break termina la ejecución de una sentencia de bifurcación (case) que componen a la sentencia switch.

d. ( ) Termina la ejecución de una sentencia dentro del switch y transfiere el control del programa a la siguiente sentencia.

7. Indicar el valor de la variable *j* después de ejecutar del siguiente programa:

```
public class Buclefor; {
public static void main (String [] args) {
int j=1;
for (int i= -13; i<=-10; i++) {
j++;
}
System.out.println(j);
}
}
```

a. ( ) -4
b. ( ) 4
c. ( ) 5
d. ( ) 6

8. Indica los parámetros de la sentencia for, para la variable de control i, para que el siguiente programa imprima los números enteros múltiplos de dos se encuentran entre 2 y 20.

```
public class MultiplosDos {
    public static void main(String[] args) {
        for (int i =___; i < ___; i += ___) {
            System.out.println("Número: " + i);
        }
    }
}
```

a. ( ) 2,21,2
b. ( ) 0,21,2
c. ( ) 2,20,2
d. ( ) 1,21,2

Solución a los Ejercicios

1 Unidad 1

La Cibernética

Pág. 17

1. Cibernética
2. Sistema dinámico complejo
3. Norbert Wiener
4. Teoría de la información
5. Neurofisiología
6. W. Harvey
7. Blaise Pascal
8. Matemáticas
9. Robótica
10. Biónica

Pág. 25

1.

2. c	3. b	5. a	7. a	9.d
3. a	4. d	6. b	8. d	10.b

Pág. 31

Horizontales

1. Entrada
7. Sistema
10. Sistemas cerrados

Verticales

2. Sistemas abiertos
3. Sistemas abstractos
4. Entropía
5. Salida

6. Sistemas aislados
8. Ambiente
9. Sistemas físicos

Pág. 35

1. V 2. V 3. V 4. F 5. F

Pág. 41

1. b 2. f 3. E 4. d 5. a 6. c

2 Unidad 2

Circuitos Lógicos

Pág. 57

1. R= 1111
2. R= 10111
3. R= 101101
4. R= 1100111
5. R= 11101011
6. R= 111001001
7. R= 1110100110
8. R= 10000000000
9. R= 100100110100
10. R= 1000111010111

Pág. 58

1. R= 5
2. R= 9
3. R= 13
4. R = 23
5. R = 25
6. R = 29
7. R = 54
8. R = 43
9. R = 77
10. R = 93

Pág. 62

1. R = 14
2. R = 42
3. R = 143
4. R = 232
5. R = 352
6. R = 1103
7. R = 3705
8. R = 10727
9. R = 17333
10. R = 23420

Pág. 63

1. R = 10101
2. R = 100110
3. R = 111100
4. R = 1100110
5. R = 11110010
6. R = 111110101
7. R = 1011100101
8. R = 100010101110
9. R = 110101100011
10. R = 1010101100110

Pág. 67

a.

1. R = 22
2. R = 61
3. R = 9C
4. R = 100
5. R = 3DB
6. R = 420
7. R = D80
8. R = 2177
9. R = 268D
10. R = 306B

b.

1. R = 46
2. R = 90
3. R = 1163
4. R = 3021

5. R = 3326
6. R = 11937
7. R = 24546
8. R = 32730
9. R = 65505
10. R = 111341

Pág. 76

1. R = 1100
2. R = 1011
3. R = 10110
4. R = 11011
5. R = 11011
6. R = 110011
7. R = 111000
8. R= 101101
9. R = 1100011
10. R = 1101101

Pág. 80

1. R = 11
2. R = - 1
3. R = 10
4. R = - 1
5. R = 11
6. R = - 1
7. R = 110
8. R = -111
9. R = 1011
10. R = - 11

Pág. 82

1. R = 11100
2. R= 101010
3. R = 11000
4. R = 11010010
5. R= 1011000
6. R = 1100000
7. R = 1111110
8. R = 1010000
9. R = 1111000
10. R = 10110110

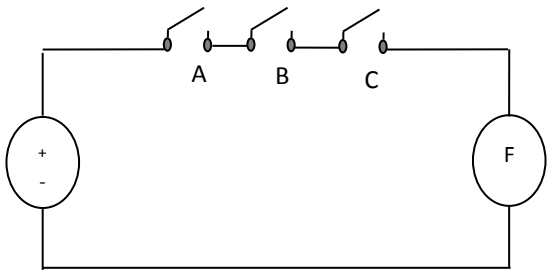
Pág. 86

1. R = 11
2. R = 11
3. R = 111
4. R = 11
5. R = 100

Pág. 92

Realiza la tabla de verdad y la función lógica de los siguientes circuitos eléctricos:

a)

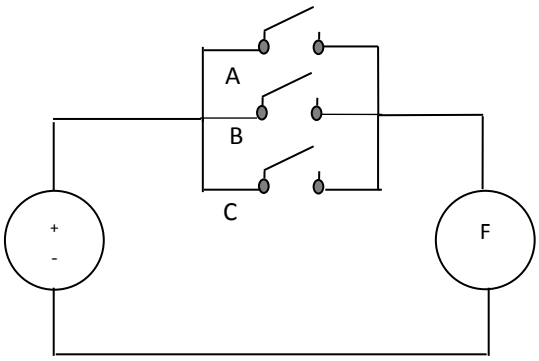


R=

A	B	C	F
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	0
1	1	1	1

F = A · B · C

b)

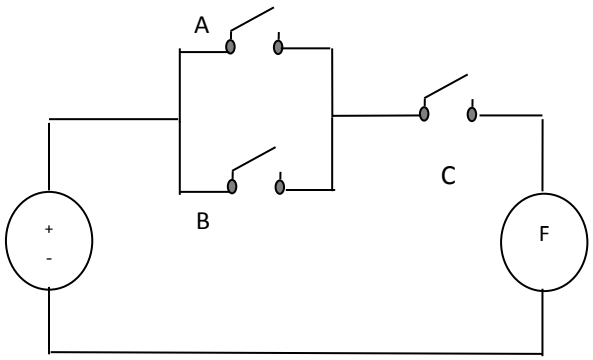


R=

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

F = A + B + C

c)



R=

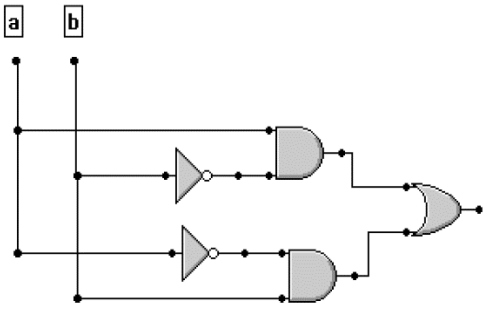
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

F = (A + B) · C

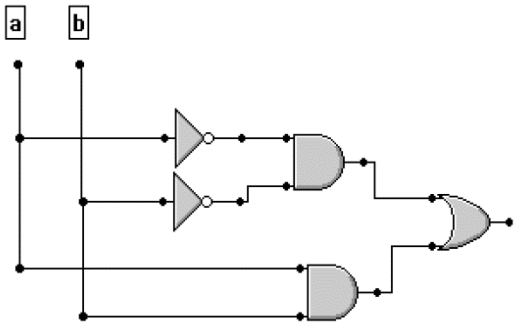
Pág. 99

Realiza el circuito lógico de las siguientes funciones:

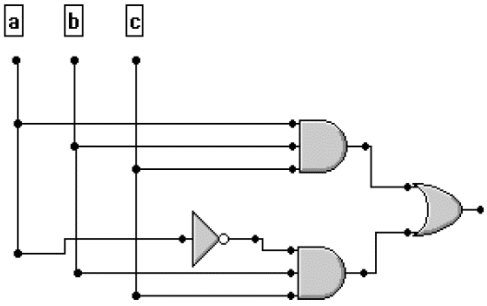
1) F = a·b’ + a’·b



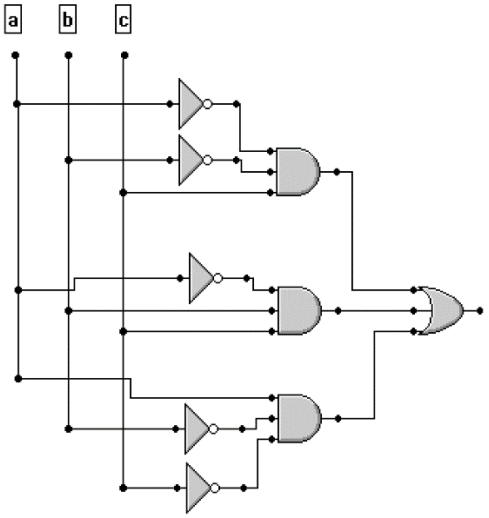
2) F = a’·b’ + a·b



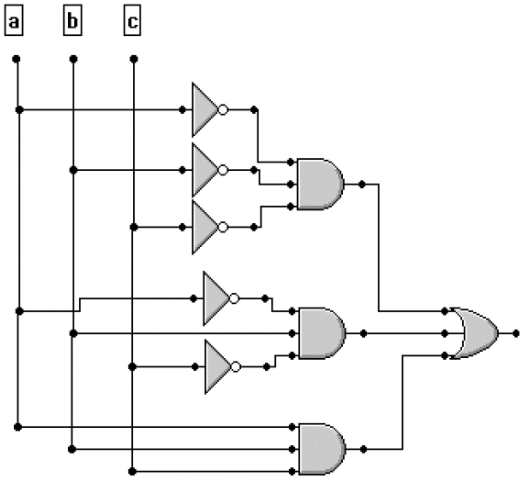
3) F = a·b·c + a’·b·c



4) F = a'·b'·c + a'·b·c + a·b'·c'



5) F= a'·b'·c' + a'·b·c' + a·b·c



Pág. 110

Encuentra la tabla de verdad de las siguientes funciones booleanas:

1. F_1 = x'·y + x·y'

x	y	F_1
0	0	0
0	0	1
0	1	0
0	1	1

2. F_2= a·b + a'·b' + (a·b)'

a	b	F_2
0	0	1
0	0	1
0	1	1
0	1	1

3. F_3= x·y·z + x'·y'·z'

x	y	z	F_3
0	0	0	**1**
0	0	1	**0**
0	1	0	**0**
0	1	1	**0**
1	0	0	**0**
1	0	1	**0**
1	1	0	**0**
1	1	1	**1**

4. F_4= x'·y·z' + x·y'·z + x·y·z

x	y	z	F_3
0	0	0	**0**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

5. F_5= = x·y'·z + x'·y·z' + x·y·z + x'·y'·z'

x	y	z	F_3
0	0	0	**1**
0	0	1	**0**
0	1	0	**1**
0	1	1	**0**
1	0	0	**0**
1	0	1	**1**
1	1	0	**0**
1	1	1	**1**

Pág. 115

Encuentra las funciones booleanas de las siguientes tablas de verdad.

a	b	c	F_1	F_2	F_3	F_4	F_5
0	0	0	**0**	**1**	**1**	**1**	**0**
0	0	1	**0**	**0**	**0**	**1**	**1**
0	1	0	**0**	**0**	**0**	**0**	**1**
0	1	1	**0**	**0**	**1**	**0**	**0**
1	0	0	**0**	**0**	**1**	**1**	**1**
1	0	1	**0**	**0**	**1**	**0**	**1**
1	1	0	**1**	**1**	**0**	**0**	**0**
1	1	1	**1**	**1**	**0**	**1**	**0**

R=
$F_1 = a \cdot b \cdot c' + a \cdot b \cdot c$
$F_2 = a' \cdot b' \cdot c' + a \cdot b \cdot c' + a \cdot b \cdot c$
$F_3 = a' \cdot b' \cdot c' + a' \cdot b \cdot c + a \cdot b' \cdot c' + a \cdot b' \cdot c$
$F_4 = a' \cdot b' \cdot c' + a' \cdot b' \cdot c + a \cdot b' \cdot c' + a \cdot b \cdot c$
$F_5 = a' \cdot b' \cdot c + a' \cdot b \cdot c' + a \cdot b' \cdot c' + a \cdot b' \cdot c$

Pág. 133

Encuentra la simplificación de las siguientes funciones booleanas:

6. $x' \cdot y \cdot z' + x' \cdot y \cdot z + x \cdot y' \cdot z' + x \cdot y' \cdot z$
   $R = x \cdot y' + x' \cdot y$

7. $x' \cdot y' \cdot z' + x \cdot y' \cdot z + x \cdot y \cdot z + x' \cdot y \cdot z$
   $R = x \cdot z + y \cdot z + x' \cdot y' \cdot z'$

8. $x \cdot y' \cdot z' + x \cdot y' \cdot z + x' \cdot y' \cdot z + x' \cdot y \cdot z' + x' \cdot y \cdot z$
   $R = x \cdot y' + x' \cdot z + x' \cdot y$

9. $x' \cdot y' \cdot z' + x' \cdot y' \cdot z + x' \cdot y \cdot z' + x \cdot y' \cdot z' + x \cdot y \cdot z$
   $R = x' \cdot y' + y' \cdot z' + x' \cdot z' + x \cdot y \cdot z$

10. $x' \cdot y' \cdot z' + x \cdot y' \cdot z + x \cdot y \cdot z + x' \cdot y \cdot z' + x' \cdot y \cdot z$
    $R = x' \cdot z' + x' \cdot y + x \cdot z$

3 Unidad 3

Metodología de solución de problemas e introducción de lenguaje de programación Java

Pág. 165

1	2	3	4	5	6	7
C	F	A	D	G	B	E

Pág. 169

1	2	3	4	5	6
E	B	D	A	F	C

Pág. 177

1	2	3	4	5	6	7	8	9	10
V	V	F	F	F	F	V	F	F	V

Pág. 182

a.

1. Inicio: jabón, trapo, bolsa para basura , aspiradora, agua, cepillo de escoba
2. Proceso:
    a. Levantar la basura de interior del auto
    b. Prepara agua con jabón
    c. Humedecer el trapo con el jabón
    d. Tallar el auto con el trapo
    e. Tallar llantas con el cepillo
    f. Enjuagar el auto con abundante agua el coche
    g. Secar el auto
    h. Aspirar los interiores del auto
3. Final: Auto lavado

b.

Inicio
Introduce el valor de, A
Introduce el valor de, B
Restar el valor de A a B
El resultado de la resta es: res
Final

Pag. 186

Inicio

2. Escribir: "Límite de velocidad"
3. Ingresa la velocidad del auto: vel
4. Leer vel
5. **Si** (vel <=80) **entonces**

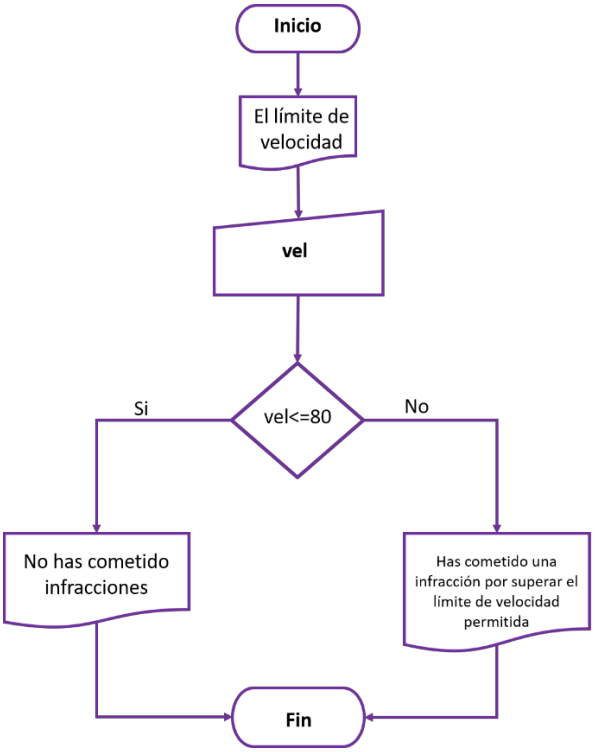
a. Escribir "Sin infracciones cometidas"

Si no

b. Escribir "Has cometido una infracción por superar el límite de velocidad permitida"

Fin Si

Fin



Pag. 191

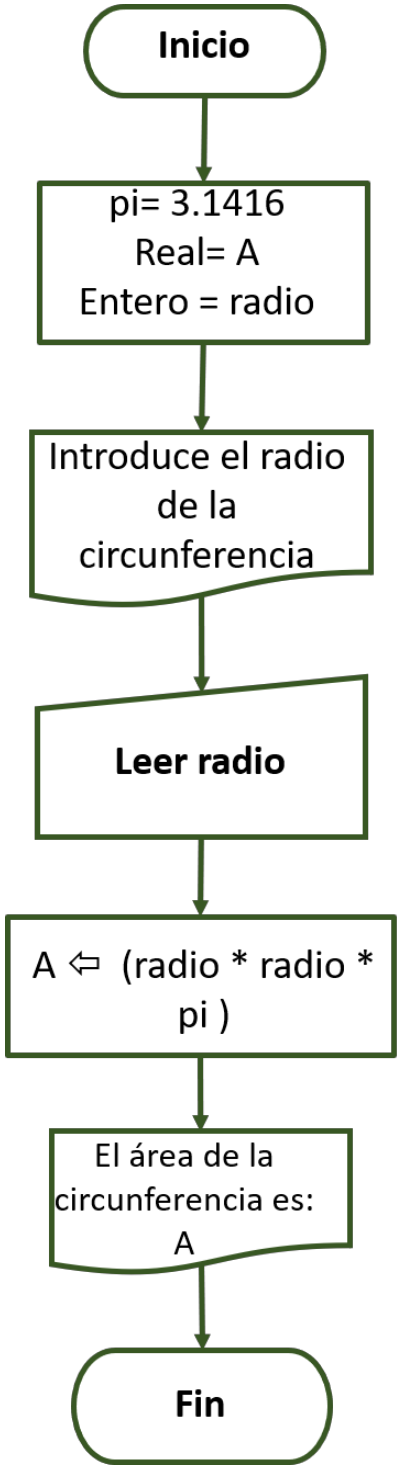
Pseudocódigo ***Diagrama de flujo***

Inicio
Constante: pi=3.1416
Variables
Entero: radio
Real: A
Imprime: "Introduce el radio de la circunferencia: "

Leer radio
A = radio * radio * pi
Imprime: “El área de la circunferencia es: “ A
Fin

Prueba de escritorio

Datos de entrada	*Fórmula*	*Cálculo*	*Dato de salida*
pi, radio	$A= \pi r^2$	A= 3.1416*3*3	El área de la circunferencia es 28.2744
Pi= 3.1416			
Introduce radio: 3			



Pag. 192

1. Inicio

Entrada de datos:

Harina para pastel, leche, huevo, polvo para hornear, bicarbonato, vainilla, azúcar, mantequilla

Proceso:

1. En un recipiente mezclar: harina para pastel, 1 tz. de azúcar, 1 cda. de polvo para hornear, 1 cda de bicarbonato y ¼ de cucharadita de sal.
2. Agregar ¾ tz. de leche, ¼ de tz. de mantequilla y ½ cda. de vainilla, batir por dos minutos, agregar 1 huevo. Batir por 2 minutos más. Verter en un molde redondo previamente engredado y enharinado.
3. Hornear a 200° C por 30 0 35 minutos, sacar del molde y dejar enfriar
4. Decorar

Fin (salida de datos): Pastel

2.

Inicio (entrada de datos):

2 limones, cuchillo, recipiente, exprimidor

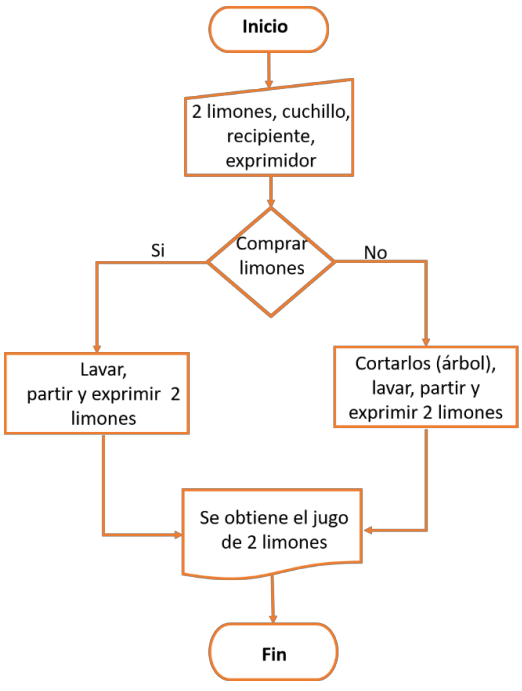
Proceso:

1. Cortarlos o comprar los limones
2. Lavarlos
3. Cortar los limones
4. Exprimir los limones
5. Obtener el jugo de limón

Fin (salida de datos)

Tener el jugo de dos limones

3.



4.

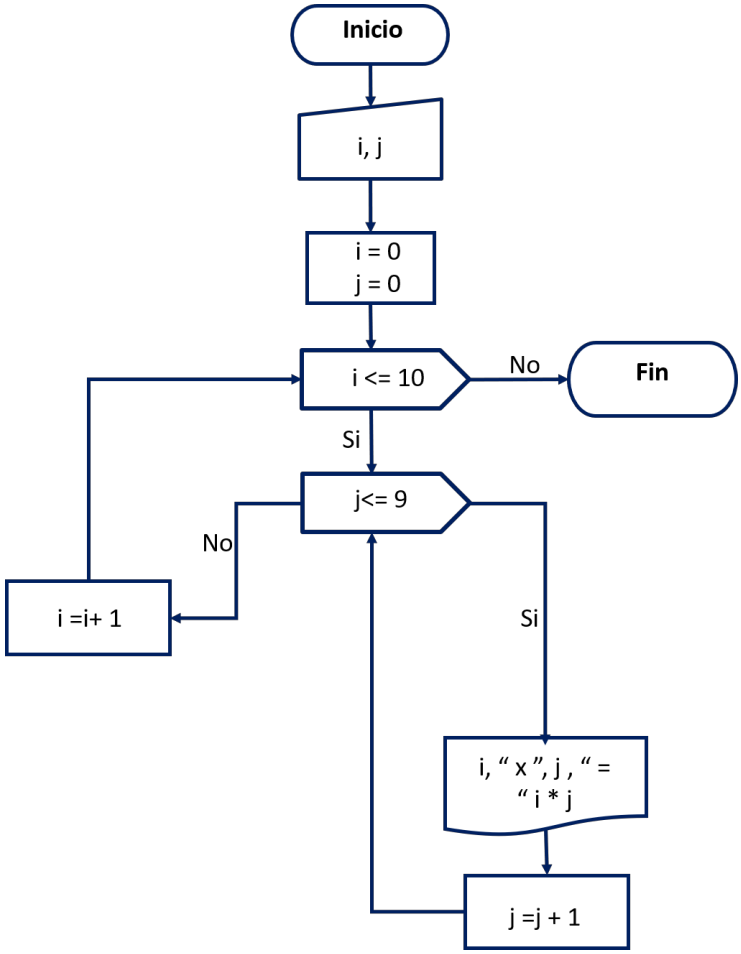
Inicio
Revisar presupuesto
Decidir playa destino
Realizar itinerario
Fechas adecuadas para el viaje
Reservar hotel
Comprar boletos de avión
Viajar
Disfrutar el viaje y broncearse
Fin

5.

Inicio
Entero: num, i
Imprime: " Introduce el número de tabla de multiplicación a resolver: "
Leer num
Para (i=1 ***hasta*** 10 ***hacer***)
Escribir num, " * " , i , " = " i * num
Fin para
Fin

6.

Inicio
Entero: i, j
Para (i=1 ***hasta*** 10 ***hacer***)
Para (k=1 ***hasta*** 9 ***hacer***)
Escribir i, " x ", j , " = " i * j
Fin para
Fin para
Fin

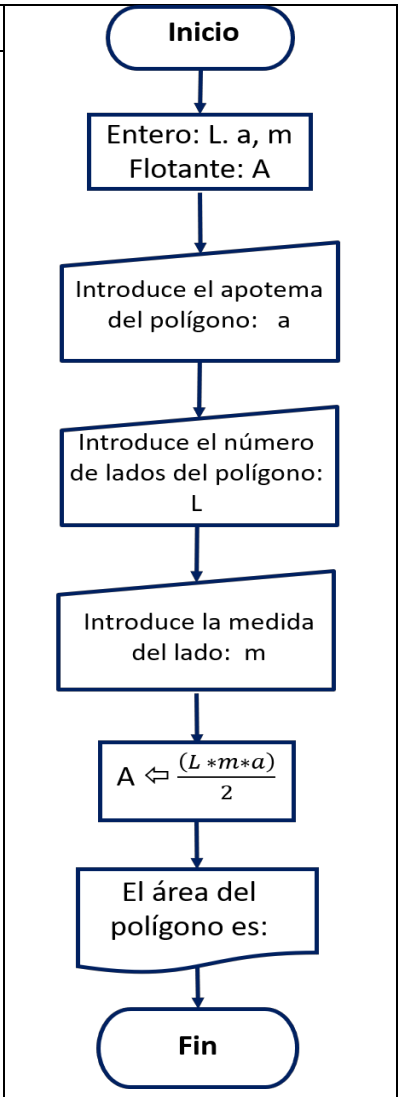


7.

Pseudocódigo	Algoritmo
Inicio Entero: L, a, m Flotante: A Escribir: "Introduce la apotema del polígono" Leer a Escribir: "Introduce el número de lados del polígono: " Leer L Escribir "Introduce la medida del lado: " Leer m A= (L*m*a)/2 Imprimir: "El área del polígono es: " A **Fin**	

Prueba de escritorio

	Entrada	*Calculo*	*Salida*
Introduce apotema	6		
Introduce lados	8		
Introduce media de lado	13		
Multiplicación		(6*8*13)	
División		$\frac{624}{2}$	
Área			El área del polígono es: 312



8.

Inicio

Entero: Dividendo, divisor

Flotante: Div

Escribir: "Ingrese un número entero: "

Leer Dividendo

Escribir: "Introduce un segundo número entero: "

Leer dividendo

Div = Dividendo / divisor

Imprimir: "la división es: " Div

Fin

Pag. 192

1.C 2. D 3. A 4. B 5. C

Crucigrama

Verticales	*Horizontales*
6. Pseudocódigo 4. Algoritmos 5. Secuenciales 7. While	1. Diagramas 3. Árabe 6. Condicionales

Pág. 196

Hotizontales

2. Java
4. Oak
5. ProyectoVerde
6. Netscape

Verticales

1. HotJava
2. JavaSoft
3. GreenOs

Pág. 199

1. 1 3. 2 5. 9 7. 5 9. 3

2. 10 4. 7 6. 8 8. 4 10. 6

Pág. 206

I. d b c a e

II. 1. b 2. d 3. c 4. c 5. a

Pág. 219

II. 1. c 2. d 3. a 4. a 5. b

Pág, 265

I.	1. V	2. V	3. F	4. V	5. F
II.	1.b	2. c	3. b	4.a	5. d

Pág, 278

1. b	3. d	5. a	7. c
2. b	4. d	6. a	8. a

Fuentes de información

1 Unidad 1

- ✓ Ciberluan87. (2009). La Cibernética.Consultado el 12 de septiembre de 2017. Recuperado de http://ciberluan87.blogspot.es/2
- ✓ INTRODUCCIÓN A LA INFORMÁTICA. Consultado el 25 de mayo de 2018. Recuperado de http://kame-sennin.blogspot.com/2009/03/entradas-procesos-y-salidas.html
- ✓ iDave Barrientos. (2012). Historia de la Cibernética. Consultado el 27 de octubre de 2017.Recuperado de http://historiadelaciberneticacchsur.blogspot.mx/
- ✓ Salvador Olivares. (2018). Concepto de la cibernética. SCRIBD.Consultado el 21 de noviembre de 2017.Recuperado de https://www.scribd.com/doc/95475194/CONCEPTO-DE-CIBERNETICA-1
- ✓ Euston96. (2017). Euston. Recuperado de https://www.euston96.com/herman-hollerith/
- ✓ Biografías y Vidas. (2004-2018). La enciclopedia bibliográfica en línea. Consultado el 17 de enero de 2018.Recuperado de https://www.biografiasyvidas.com/biografia/t/turing.htm
- ✓ Evelio Martínez. (2013). Claude Shannon: el padre de la teoría de la información. Consultado el 17 de enero de 2018.Recuperado de http://www.eveliux.com/mx/Claude-Shannon-el-padre-de-la-teoria-de-la-informacion.html
- ✓ Los orígenes del arte cibernético. Consultado el 12 de febrero de 2018.Recuperado de https://www.infoamerica.org/documentos_pdf/wiener2.pdf
- ✓ Sergio Rajsbaum y Eduardo Morales, editores huéspedes. (2016). Presentación Norbert Wiener y el origen de la cibernética. Consultado el 12 de febrero de 2018.Recuperado de http://www.revistaciencia.amc.edu.mx/images/revista/67_1/PDF/Presentacion.pdf
- ✓ Jramoi, A., (1968). *Introducción e Historia de la Cibernética*. México: Grijalbo, S.A.
- ✓ Salido, J., (2010).*Cibernética Aplicada*. México: Alfaomega.
- ✓ Ross. W., (1977). *Introducción a la Cibernética*. México: Libreros Mexicanos.

2 Unidad 2

- ✓ Mano, M. (1997). *Lógica digital y diseño de computadores*. México: Prentice-Hall Hispanoamericana.
- ✓ Tocci, R. (1996). *Sistemas digitales*. México: Prentice Hall-Hispanoamericana.
- ✓ Floyd, T. (2011). *Fundamentos de sistemas digitales*. México: Pearson Educación de México, S.A. de C.V.
- ✓ Hein, K. (1973). *Algebra de los circuitos lógicos*. Madrid: Dossat.
- ✓ Bignell, J. and Donovan, R. (1999). *Electrónica digital*. México: Continental.
- ✓ Markus, J. and Alatorre Miguel, E. (1994). *Circuitos digitales*. México: McGraw-Hill.
- ✓ YouTube. (2018). *DECIMAL A BINARIO* [en línea] Disponible en: https://www.youtube.com/watch?v=M6rgWijW4Gw [Consultado 27 enero 2018].
- ✓ YouTube. (2018). *DECIMAL A OCTAL* [en línea] Disponible en: https://www.youtube.com/watch?v=BAqeZ8N8snM [Consultado 28 enero 2018].
- ✓ YouTube. (2018). *DECIMAL A HEXADECIMAL* [en línea] Disponible en: https://www.youtube.com/watch?v=kljKnagetg8 [Consultado 28 enero 2018].
- ✓ YouTube. (2018). *SUMA BINARIA* [en línea] Disponible en: https://www.youtube.com/watch?v=2WtqivPA4tk [Consultado 27 enero 2018].
- ✓ YouTube. (2018). *RESTA BINARIA*. [en línea] Disponible en: https://www.youtube.com/watch?v=X-bgT3tjImE [Consultado 27 enero 2018].
- ✓ YouTube. (2018). MULTIPLICACIÓN BINARIA [en línea] Disponible en: https://www.youtube.com/watch?v=mhXqEIHDJRY [Consultado 27 enero 2018].

3 Unidad 3

- ✓ Aguilar, L., (2013), Fundamentos generales de programación, C.D. México, México: Mc Graw – Hill
- ✓ Bembibre, V., (2009), Diagrama de flujo. [Sitio web], Recuperado de: https://www.definicionabc.com/comunicacion/diagrama-de-flujo.php
- ✓ Correa, G. (2011), Desarrollo de Algoritmos y sus aplicaciones, McGrawHill
- ✓ Hernández, M.L., (2010), Diseño estructurado de algoritmos, Diagramas de flujos y pseudocódigos.

- ✓ García-Beltrán, A., Martínez, R. y Jaén, J.A. (2004) Fundamentos de programación, 2ª edición, Madrid: Bellisco
- ✓ Mena, C., (2018), Jerarquía de Operadores, Programación de computadoras, [File PDF], Recuperado de: https://es.scribd.com/document/88334727/Jerarquia-de-Operadores
- ✓ Kölling, Michael, Rosenberg J. (2017). BlueJ (Software) . Versión 2.
- ✓ Pinales, F.J, Velázquez, C.E., (2014), Algoritmos resueltos con diagramas de flujo y pseudocódigos, México, Aguascalientes: textos universitarios
- ✓ Rodriguez, J. (9 de marzo de 2012), Símbolos diagramas de Flujo (SlideShare), Recuperado de: https://es.slideshare.net/AliniuZizRguezT/simbolos-diagrama-de-flujo
- ✓ Mejía, E. Z. (s.f.). *Programación Java*. (F. d. Ingeniería, Editor) Recuperado el 18 de enero de 2018, de http://profesores.fi-b.unam.mx/carlos/java/indice.html
- ✓ SIN/AUTOS. (s.f.). *Principales Características de JAVA.* Recuperado el 18 de enero de 2018, de http://personales.upv.es/rmartin/cursoJava/Java/Introduccion/PrincipalesCaracteristicas.htm
- ✓ WIKILIBROS. (15 de marzo de 2016). *Programación en Java/Caracteristicas del lenguaje*. Recuperado el 18 de enero de 2018, de https://es.wikibooks.org/wiki/Programaci%C3%B3n_en_Java/Caracter%C3%ADsticas_del_lenguaje
- ✓ García, H. Enrique (2018). Tipos de datos en Java. Recuperado el 24 de Enero de http://puntocomnoesunlenguaje.blogspot.mx/2012/04/tipos-de-datos-java.html
- ✓ Salas, Ivan (2018). Bucles for, while y do while en java. Recuperado el 14 de marzo de 2018, de http://profesores.fi-b.unam.mx/carlos/java/java_basico1_4.html