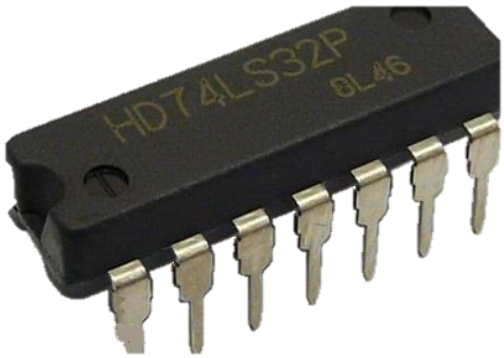
Universidad Nacional Autónoma de México

Colegio de Ciencias y Humanidades

Plantel Naucalpan

Programa de Apoyo al Egreso

Cibernética y Computación I



PLAN 2016

Elaboraron:

Salvador Gómez Moya

Jacobo López Suárez

Marbella Denisse Díaz Moya

Líber A. Mass Zúñiga

Antonio Pavón García

José Ignacio Ortiz Cervantes

Coordinador:

Salvador Gómez Moya

Contenido

Unidad I

La cibernética

Unidad I: La cibernética

En esta unidad se presentará una definición de cibernética, así como a su fundador, Norbert Wiener; se abordará a la cibernética como una ciencia interdisciplinaria, donde intervienen disciplinas como las matemáticas, la física, la ingeniería e incluso la filosofía. Posteriormente, se introducirá el concepto de sistema; se expondrán los tipos de sistemas, sus componentes, y sobre todo ejemplos prácticos. El tema de sistemas está estrechamente ligado al siguiente, el de retroalimentación, es decir, al de los sistemas abiertos y cerrados. Se describirán los sistemas que usan sensores (cerrados) y los que no (abiertos). En esta misma unidad, se abordará el tema de modelado, su definición, los tipos de modelos existentes y su utilidad.

Como **propósito** de la presente 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.

A su vez, los **aprendizajes** de la presente unidad son los siguientes:

El alumno:

1. Comprende la influencia de la cibernética en el desarrollo de la ciencia.
2. Describe el trabajo científico sobre la cibernética de Norbert Wiener, Arturo Rosenblueth, Claude Shannon, entre otros.
3. Comprende los componentes de un sistema.
4. Comprende los componentes esenciales de un sistema de control.
5. Comprende el concepto y la importancia del modelo.
6. Desarrolla el modelo de un sistema.
7. Explica cómo construyó el modelo del sistema, las partes que lo conforman y su funcionamiento.

Norbert Wiener y el origen de la cibernética

Norbert Wiener (Estados Unidos, 1894) fue el creador de la cibernética. Wiener se formó principalmente en las matemáticas, sin embargo, necesitó de profundos conocimientos en física, ingeniería, computación, estadística, medicina, fisiología, e incluso filosofía para desarrollar su nueva ciencia. Es por ello, que se dice que la cibernética es una ciencia interdisciplinaria; y es que Wiener intentó aplicar la teoría de la cibernética en máquinas y seres vivos por igual.

En los años en los que la cibernética era sólo un esbozo en la mente de Wiener (aproximadamente los años 30 y 40 del siglo pasado), apareció un mexicano para ayudar a Wiener a materializar su idea. Arturo Rosenblueth era un fisiólogo docente en la Escuela de Medicina de Harvard que organizaba mensualmente cenas en las que se discutían temas científicos. Norbert y Arturo se hicieron amigos. Les parecía interesantes los paralelismos que existían entre el funcionamiento de las máquinas y los animales, especialmente en el campo de la retroalimentación. Durante una larga estancia en la Ciudad de México, Wiener escribió el libro que contendría los fundamentos de su nueva ciencia. Este sería el acta de nacimiento de la cibernética. Wiener escogió este nombre para denominar a la ciencia dedicada al control de máquinas y seres vivos que estaba por dar a la luz, debido a sus amplios conocimientos del griego. Cibernética viene de la pronunciación inglesa (cybernetics) del vocablo griego kubernetes que significa timonero, el timón de un barco.

Cibernética o el control y comunicación en animales y máquinas se publicó en 1948 y pronto se convirtió en lectura obligatoria en círculos inclusive no científicos. En tiempo vertiginoso Norbert Wiener alcanzó la fama. En 1951 la UNAM le otorgó un doctorado honoris causa. La cibernética se diseminó alrededor del mundo. Personajes ahora de sobra reconocidos como Enrico Fermi en Italia o Alan Turing en Inglaterra, formaron o participaron en círculos dedicados a la cibernética.

Con el tiempo, y el avance de la ciencia y la tecnología, la cibernética se dividió en muchas ramas, de las cuales, las más notables actualmente son la inteligencia artificial y la ingeniería de control.

Actividad 1.1

Elabora un mapa conceptual de todas las ciencias que intervienen en la cibernética y coméntalo en grupo con tu profesor.

El diccionario de la Real Academia Española define a la cibernética como la 'Ciencia que estudia las analogías entre los sistemas de control y comunicación de los seres vivos y los de las máquinas'. Esta definición es correcta como primer acercamiento. Sin embargo, como ciencia interdisciplinaria, la cibernética se ocupa de conceptos e incluso teorías completas que, en apariencia disímbolas como la filosofía y la ingeniería, y en gran medida la física y la medicina, forman una ciencia que, en su definición más general, es aquella dedicada al control o regulación de máquinas y seres vivos.

En 1958, Norbert Wiener publicó el libro *Cibernética y sociedad* para explicar en qué consiste la cibernética al público no especializado. En el prólogo, Wiener hace una reseña histórica sobre la historia de la física, en ésta menciona que, en la física clásica, desarrollada por Newton, el mundo es determinista, es decir, existen leyes que lo describen todo en cualquier momento. Con las leyes de Newton se podía predecir el movimiento tanto de cuerpos terrestres como celestes. Con el tiempo esta percepción cambió, dando paso a la física relativista, en la que el universo es contingente, es decir, ya no había leyes ni predicciones, sino probabilidades de mundos posibles. En esta física más contemporánea, se introduce un nuevo concepto que Norbert Wiener utilizaría para establecer una definición alternativa de cibernética. Se trata de la entropía. La entropía es la tendencia del universo al desorden. De acuerdo con Wiener, la cibernética procura regular la entropía mediante la retroalimentación.

Con el tiempo la cibernética se ha diversificado en varias ramas que por sí solas se han establecido como ciencias por derecho propio, una de ellas es la ingeniería de control. En su vertiente *ingenieril*, la cibernética es la ciencia dedicada al control de máquinas por medio de la retroalimentación. La retroalimentación es un concepto que se explicará a detalle en las siguientes secciones.

Actividad 1.2

Investiga en algún medio digital qué es la entropía y los múltiples significados que tiene en varios campos de la física. Relaciónalo con la definición de cibernética dada anteriormente.

Actividad 1.3

Investiga la biografía de Norbert Wiener y elabora una línea del tiempo con los acontecimientos más importantes de su vida.

Actividad 1.4

Investiga las biografías de Arturo Rosenblueth y Claude Shannon y comenta en plenaria con tus compañeros y profesor cuáles fueron sus aportaciones a la cibernética.

Sistemas

Un sistema es un grupo de componentes interrelacionados que trabajan juntos hacia un fin en común, aceptando entradas y produciendo salidas en un proceso de transformación. La siguiente imagen muestra el diagrama más simple de los componentes de un sistema.

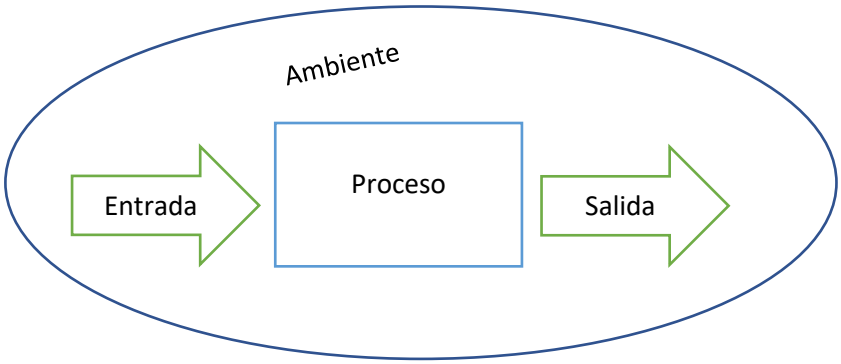


Figura 1.1. Diagrama de los elementos de un sistema.

Los componentes de un sistema son los siguientes:

Entradas. Son los recursos o insumos requeridos para que funcione el sistema.

Procesos. En esta etapa la entrada se transforma y se produce un resultado.

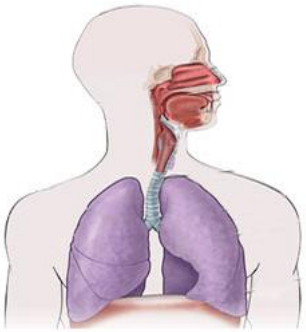
Salidas. Son los resultados que se obtienen de procesar las entradas.

Ambiente. Es el conjunto de elementos que sin formar parte del sistema posee propiedades relevantes que tiene efectos sobre el sistema.

Así pues, se necesita de una entrada para que el sistema se ponga en marcha. La entrada, al hacer funcionar al sistema, sufre una transformación, lo que la convierte en salida.

Ejemplo 1.1

Describe las características del sistema respiratorio.



Componentes: nariz, tráquea y pulmones

Entradas: oxígeno

Proceso: el oxígeno es llevado a la sangre; es necesario para la vida

Salidas: dióxido de carbono

Ambiente: aire (nitrógeno, oxígeno y otros elementos)

Ejemplo 1.2

Describe las características del sistema electoral.

Componentes: INE, políticos

Entrada: electores

Proceso: votación

Salida: ganadores de la elección

Ambiente: no hay (tal vez medios de comunicación)

Clasificación de los sistemas

Los sistemas pueden clasificarse de acuerdo con su interacción con el entorno, o su origen, como se muestra a continuación.

Sistemas cerrados. Son aquellos que no tienen ambiente ni contexto. No presentan intercambio con el medio que lo rodea, son herméticos a cualquier influencia ambiental. Un ejemplo es una olla de presión en la que la temperatura ambiente no interviene en lo que se cuece en su interior.

Sistemas abiertos. Son aquellos que tienen ambiente, es decir, sí interactúan con su entorno. Intercambian energía y materia con el medio ambiente. Un ejemplo es una taza de café, la cual se ve modificada cuando se enfría por la temperatura ambiente o se le agrega azúcar o leche.

Sistemas naturales. Nacen en respuesta a fenómenos físicos, químicos y biológicos y se crean por la naturaleza. El sistema solar es un ejemplo de ello.

Sistemas artificiales. Son aquellos que fueron logrados por la intervención directa de la actividad humana. Un ser humano participó de manera activa en su diseño, manejo, control y ejecución. Un ejemplo de ellos son los automóviles o los robots.

Modelos

Los modelos son esquemas teóricos, abstracciones de un ente real, generalmente en forma matemática, de un sistema o de una realidad compleja, como la evolución económica de un país, que se elabora para facilitar su comprensión y el estudio de su comportamiento. Los modelos son representaciones de sistemas, y frecuentemente se utilizan simulaciones computacionales, ecuaciones matemáticas, diagramas, maquetas o una mezcla de estas.

Gracias a los modelos, se pueden hacer cálculos necesarios antes de construir un objeto real. Por ejemplo, en la arquitectura de elaborar maquetas para presentaciones; en el

diseño de automóvil se hacen simulaciones computaciones; las ecuaciones matemáticas son muy útiles en el diseño de circuitos electrónicos o de edificios.

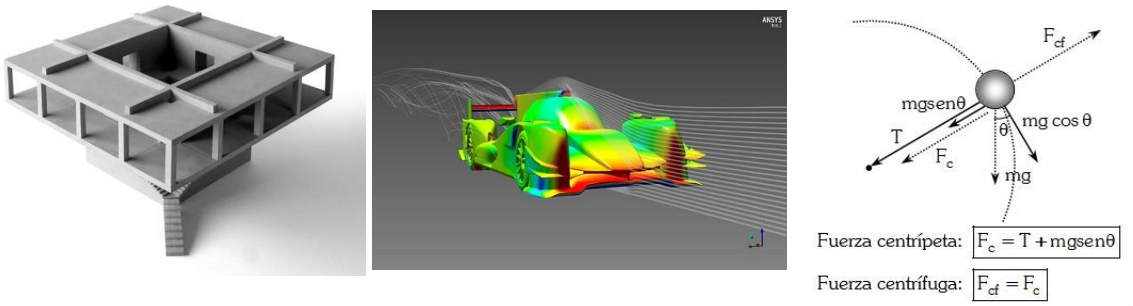


Figura 1.2. A la izquierda una maqueta usada en la arquitectura. Al centro una simulación computacional para calcular la resistencia al viento de un automóvil. A la derecha las ecuaciones matemáticas que representan el movimiento de un objeto.

Sistemas de control

Un sistema de control es un arreglo de componentes conectados de tal manera que el arreglo se puede comandar, dirigir o regular a sí mismo o a otro sistema. Es decir, es un sistema al que se le agrega un controlador para que tenga cierto funcionamiento deseado. Existen dos tipos de sistemas de control, de acuerdo con su autonomía.

Sistemas de control de lazo abierto

Son sistemas de control en los que la salida o resultado del proceso no tiene ningún efecto sobre la acción de control, es decir, en un sistema de control abierto la salida no se mide.

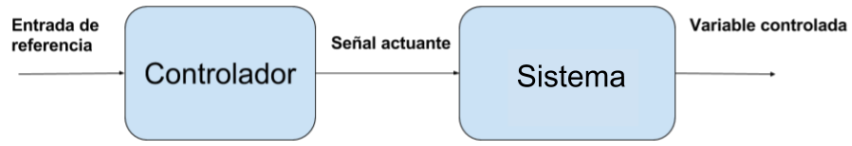


Figura 1.3. Diagrama de un sistema de control en lazo abierto.

Sistema de control de lazo cerrado

Son aquellos en los que la señal de salida tiene efecto sobre la acción de control, es decir, es un sistema de lazo abierto en el que la salida se “retroalimenta”. Por lo tanto, el sistema se “cierra”. Estos sistemas corresponden a las máquinas automáticas. La retroalimentación o medición de la salida se realiza por medio de un sensor.

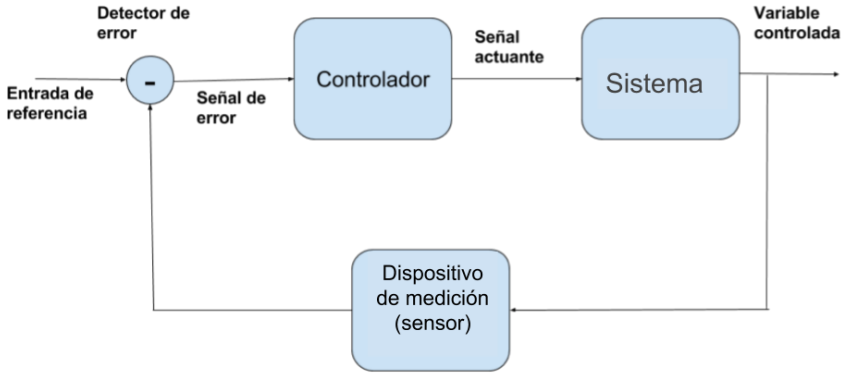


Figura 1.4. Diagrama de un sistema de control en lazo cerrado.

Atención
En general, el *Controlador*, el *Sistema* y el *Dispositivo de medición* son componentes físicos, tangibles; la *Entrada de referencia*, *Señal de error*, *Señal actuante* y *Variable controlada* son valores, acciones, datos, flujos, etc.

Para mayor claridad, a continuación se presentan las definiciones de los elementos de componen los sistemas de control cerrados y abiertos.

Sistema. Componentes interrelacionados con un fin en común.

Dispositivo de medición. Es el sensor, también conocido como transductor; mide una señal o variable.

Entrada de referencia. También conocida como *setpoint*, es lo que deseamos del sistema. Es el valor en el que queremos que se regule el sistema. Este valor es el que recibe el controlador.

Señal de error. Es la comparación entre lo que se desea del sistema (entrada de referencia) y lo que se está recibiendo (variable controlada).

Controlador. Es el dispositivo que hace que el sistema funcione como deseamos. Comúnmente se trata de un circuito electrónico. El controlador recibe la señal de error y actúa en consecuencia.

Señal actuante. Es lo que el controlador manda al sistema como entrada para que funcione de acuerdo con la entrada de referencia.

Variable controlada. Es lo que mide el sensor para saber cómo está funcionando el sistema; es lo que se recibe del sistema, su salida.

Actividad 1.5

Investiga por algún medio digital qué sensores existen y cuáles utilizas cotidianamente. Comenta tus resultados con tus compañeros.

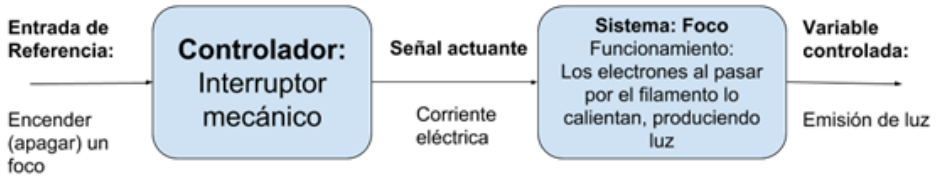
Ejemplo 1.3

Realiza el diagrama de control de un foco simple.

Sistema: foco simple

Tipo de sistema: lazo abierto

Objetivo de control: apagar o encender un foco



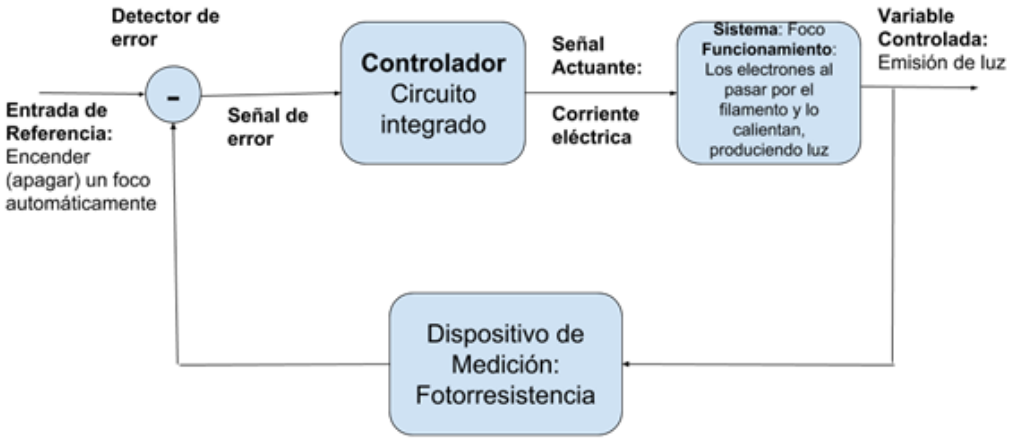
Ejemplo 1.4

Realiza el diagrama de control de un foco automático.

Sistema: foco automático

Tipo de sistema: lazo cerrado

Objetivo de control: apagar y encender un foco de forma automática


Detector de error
Entrada de Referencia: Encender (apagar) un foco automáticamente
–
Señal de error
Controlador Circuito integrado
Señal Actuante:
Corriente eléctrica
Sistema: Foco
Funcionamiento: Los electrones al pasar por el filamento y lo calientan, produciendo luz
Variable Controlada: Emisión de luz
Dispositivo de Medición: Fotorresistencia

Ejercicios

1. De acuerdo con las definiciones abordadas anteriormente, sintetiza la información en una sola definición de cibernética.
2. Escribe cinco datos importantes sobre la vida de Norbert Wiener.
3. De los siguientes sistemas, completa la siguiente información: una lavadora, un foco o una estufa.
    - Nombre del sistema
    - Cuáles son los elementos del sistema: componentes, entradas, proceso, salidas y ambiente
    - Explicación de su funcionamiento
    - Si es cerrado o abierto (por qué)
    - Si es natural o artificial (por qué)
4. Dibuja y describe el diagrama de los siguientes sistemas de control abierto: parrilla eléctrica, ventilador y un radio.
5. Dibuja y describe el diagrama de los siguientes sistemas de control cerrado: un invernadero, un refrigerador con regulador de temperatura y un coche autónomo.

Unidad II
Circuitos lógicos

Unidad II: Circuitos lógicos

Los circuitos lógicos son la base de los sistemas computacionales. Resulta de gran importancia el aprender a diseñarlos. El primer paso para diseñar un circuito lógico es convertir números de la base que usamos cotidianamente, la decimal, a bases octales, hexadecimales y principalmente la binaria, pues los circuitos lógicos realizan operaciones con ceros y unos. Una vez que se dominan la conversión entre bases, se introducen las operaciones aritméticas con números binarios. Con esta base se presenta el álgebra de Boole, esto es, tablas de verdad y funciones lógicas con los que podremos resolver prácticos.

Como **propósito** de la presente 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.

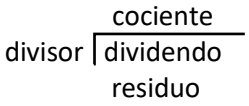
A su vez, los **aprendizajes** de la presente unidad son los siguientes:

El alumno:

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

Conversión de decimal a binario

Antes de comenzar con el procedimiento de conversión, es útil recordar cómo está compuesta una división.



Hecho esto, al algoritmo para convertir de un número en base decimal a base binaria, consiste en dividir el número entre dos, hasta que el cociente sea cero. Finalmente, los residuos se acomodan del último al primero. En los ejemplos se mostrará más claramente este proceso. Observa que un número en base decimal sólo puede estar expresada en ceros y unos.

Ejemplo 2.1

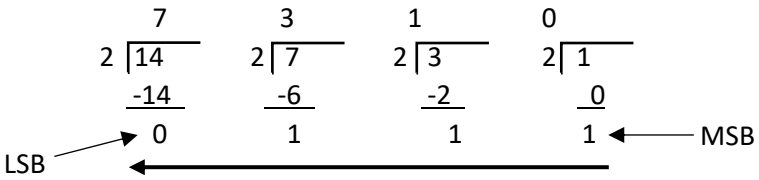
Los siguientes números están expresados en base 10, conviértelos a base binaria.

a) $14_{10} =$_____$_2$

b) $121_{10} =$_____$_2$

c) $1357_{10} =$_____$_2$

a) $14_{10} =$_____$_2$



Al último residuo se le llama bit más significativo (MSB, del inglés most significant bit), y al primer residuo bit menos significativo (LSB). Se llama MSB porque se colocará

primero, y se seguirá el orden hasta llegar al LSB, de derecha a izquierda, como indica la flecha de arriba.

Por lo tanto, 14, en decimal, es **1110** en binario.

b) 121_{10} =_____$_2$

60	30	15	7	3	1	0
2 | 121	2 | 60	2 | 30	2 | 15	2 | 7	2 | 3	2 | 1
-120	-60	-30	-14	-6	-2	0
1	0	0	1	1	1	1

Nuevamente, debemos de tomarlos del último al primero.

121, en decimal, es **1111001** en binario.

c) 1357_{10} =_____$_2$

678	339	169	84	42	21	10	5	2	1	0
2 | 1357	2 | 678	2 | 339	2 | 169	2 | 84	2 | 42	2 | 21	2 | 10	2 | 5	2 | 2	2 | 1
-1356	-678	-338	-168	-84	-42	-20	-10	-4	-2	0
1	0	1	1	0	0	1	0	1	0	1

En este caso, 1357, en decimal, es **10101001101** en binario.

La conversión de una base a otra es un algoritmo, por lo tanto, para memorizarse, se recomienda hacer muchos ejercicios.

Actividad 2.1

Convierte los siguientes números decimales a base binaria. Comprueba la solución.

a) 45_{10} b) 103_{10} c) 384_{10} d) 756_{10}

Solución: a) 101101_2 b) 1100111_2 c) 110000000_2 d) 1011110100_2

Conversión de base decimal a cualquier otra

El procedimiento para convertir un número en base decimal a una representación en cualquier otra base es igual que el anterior, de decimal a binario, solo que en vez de dividir entre dos hasta que el cociente sea cero, se divide entre la base de interés. Los siguientes ejemplos muestran este procedimiento para distintas bases.

Ejemplo 2.2

Los siguientes números están expresados en base 10, conviértelos a la base que se indica.

a) 96_{10} =_____$_3$
b) 535_{10} =_____$_8$
c) 1020_{10} =_____$_{16}$

a) 96_{10} =_____$_3$

32	10	3	1	0
3 | 96	3 | 32	3 | 10	3 | 3	3 | 1
-96	-30	-9	-3	-0
0	2	1	0	1

Como en conversión a binario, tomamos del último al primero.

96, en decimal, es **10120** en base 3.

b) 535_{10} =_____$_8$

66	8	1	0
8 | 535	8 | 66	8 | 8	8 | 1
-528	-64	-8	0
7	2	0	1

Siguiendo el mismo principio, 535 en decimal, es **1027** en octal.

c) 1022_{10} =_____$_{16}$

63	3	0
16 | 1022	16 | 63	16 | 3
-96	-48	- 0
62	15	3
-48		
14		

La base hexadecimal es un caso especial, porque contiene diez números y cinco letras, tal como se muestra en la siguiente tabla. Así pues, si el residuo resulta entre 10 y 15, estos números deben ser cambiados a su letra correspondiente.

Decimal	Hexadecimal
0	0
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	A
11	B
12	C
13	D
14	E
15	F

Entonces, si tomamos los residuos del último al primero, y cambiamos los números a letras como corresponden, el 1022 decimal es **3FE** en hexadecimal.

Conversión con números decimales

Cuando un número decimal tiene una parte entera y otra decimal, y se desea convertir a binario, primero se convierte la parte entera como se hay hecho hasta ahora (dividir entre dos); luego la parte decimal se multiplica por 2, se toma la primera cifra entera de ese producto, la parte decimal resultante (sin la parte entera) se vuelve a multiplicar por 2, así hasta los decimales que se desean obtener.

Ejemplo 2.3

Convierte los siguientes números de decimal a la base que se indica.

a) 41.6875_{10}=_____2

b) 153.513_{10}=_____8

a) 41.6875_{10} =_____$_2$

Primero convertimos la parte entera

20	10	5	2	1	0
2 | 41	2 | 20	2 | 10	2 | 5	2 | 2	2 | 1
- 40	-20	-10	-4	-2	0
1	0	0	1	0	1

Ahora convertimos la parte decimal.

0.6875 X 2 = 1.375

Se toma solo la parte decimal y se vuelve a multiplicar por 2.

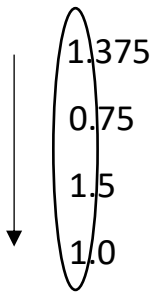
0.375 X 2 = 0.75

Se repite el mismo procedimiento.

0.75 X 2 = 1.5

0.5 X 2 = 1.0

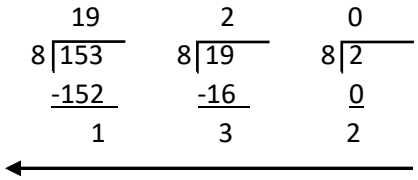
Tomamos solo la parte entera de los resultados de arriba hacia abajo.



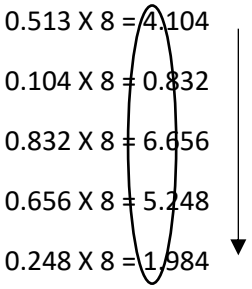
Entonces, **41.6875** en decimal es **101001.1011**.

b) $153.513_{10} =$_____$_8$

Primero convertimos la parte entera



Para la parte decimal



Finalmente, 153.513 en decimal es **231.40651** en octal.

Actividad 2.2

Convierte de base a base según se indica. Comprueba la solución.

a) $611_{10} =$ _____$_3$ b) $401_{10} =$ _____$_6$ c) $95_{10} =$ ____$_{16}$

b) d) $23.295_{10} =$ _____$_2$

Solución: a) 211122_3 b) 1505_6 c) $5F_{16}$ d) 10111.01110_2

Conversión de cualquier base a decimal

Un número en decimal, como el 2890.567, debido a que es posicional, se puede escribir de la siguiente manera:

$$2x10^3+8x10^2+9x10^1+0x10^0+5x10^{-1}+6x10^{-2}+7x10^{-3} = 2890.567$$

Es decir, toda cantidad en decimal se escribe como una combinación de la base (del cero al nueve) por potencias de 10, estas potencias se van incrementando de uno en uno de derecha a izquierda a partir del punto decimal, y se decrementan de izquierda a derecha a partir del punto decimal.

Siguiendo la lógica anterior, el número en binario 1011.101, también posicional, se puede expresar como:

$$1x2^3+0x2^2+1x2^1+1x2^0+1x2^{-1}+0x2^{-2}+1x2^{-3} = 116.625,$$

Donde 116.625 está en base decimal.

Lo visto hasta ahora se puede generalizar como sigue. Un número expresado en alguna base *r*, consiste en una suma de coeficientes que se multiplican por potencias de r.

$$a_n \cdot r^n + a_{n-1} \cdot r^{n-1} + \ldots + a_2 \cdot r^2 + a_1 \cdot r + a_0 + a_{-1} \cdot r^{-1} + a_{-2} \cdot r^{-2} + \ldots + a_{-m} \cdot r^{-m}$$

Los coeficientes a_j se encuentran dentro del conjunto formado por la base *r*, esto es, número entre 0 y *r*-1.

Ejemplo 2.4

Convierte de la base que se indica a decimal.

a) 101110_2=_____$_{10}$

b) 1101.01_2=_____$_{10}$

c) 4021.201_5=_____$_{10}$

d) $B6F_{16}$=_____$_{10}$

a) 101110_2

Se le asigna a cada coeficiente su posición, de derecha a izquierda.

Coeficiente	1	0	1	1	1	0
Posición	5	4	3	2	1	0

$$1x2^5+0x2^4+1x2^3+1x2^2+1x2^1+0x2^0 = 46$$

Comprobación

Se aplica el método realizado anteriormente para convertir de decimal a otra base.

```
 23    11    5    2    1    0
2|46  2|23  2|11  2|5  2|2  2|1
 0     1     1    1    0    1
```

101110

b) 1101.01_2

1	1	0	1	.	0	1
3	2	1	0	.	-1	-2

$1x2^3+1x2^2+0x2^1+1x2^0+0x2^{-1}+1x2^{-2} = 13.25$

c) 4021.201_5

4	0	2	1	.	2	0	1
3	2	1	0	.	-1	-2	-3

$4x5^3+0x5^2+2x5^1+1x5^0+2x5^{-1}+0x5^{-2}+1x5^{-3} = 511.408$

```
  102   20   4   0
5|511 5|102 5|20 5|4
   1     2    0   4
```

4021

d) B6F

B	6	F
2	1	0

$11x16^2+6x16^1+15x16^0 = 2927$

Conversión entre números binarios, octales y hexadecimales

Debido a que $2^3= 8$, cada dígito en octal corresponde a tres dígitos en binario. Lo mismo sucede en hexadecimal; en este caso, debido a que $2^4 = 16$, cada dígito en hexadecimal corresponde a 4 dígitos en binario. Así pues, el procedimiento para convertir un número binario consiste en agrupar de tres en tres dígitos hacia la izquierda a partir del punto, y tres en tres hacia la derecha a partir del punto. Para convertir de binario a hexadecimal el procedimiento es el mismo, solo que se toman cuatro dígitos cada vez.

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

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

Ejemplo 2.5

Convierte los siguientes números a la base que se indica.

a) 1111110101110011_2 = _____$_8$ y _____$_{16}$

b) 101111.10111_2=_____$_8$

c) $19AF_{16}$ = _____$_2$

d) 327.45_8=_____$_2$

a) 1111110101110011_2= _____$_8$ y _____$_{16}$

Se dividen grupos de tres, de derecha a izquierda.

1,111,110,101,110,011

Se encuentra el equivalente en octal de acuerdo con la tabla correspondiente.

1	111	110	101	110	011
1	7	6	5	6	3

El número en octal es 176563

En hexadecimal se agrupa de 4 en 4.

1111,1101,0111,0011

1111	1101	0111	0011
F	D	7	3

El número en hexadecimal es FD73

b) 101111.10111_2=_____$_8$

101,111.101,11

101	111	.	101	11
5	7	.	5	6

El número en octal es 57.56

c) $19AF_{16}$ = _____$_2$

En este caso solo se busca el equivalente hexadecimal a binario en la tabla correspondiente.

1	9	A	F
0001	1001	1010	1111

El número en binario es 0001100110101111 o 1100110101111.

d) 327.45_8=_____$_2$

Como en el caso anterior, sólo se busca el número correspondiente en la tabla de octal.

3	2	7	.	4	5
011	010	111	.	100	101

El número en binario es 011010111.100101 o 11010111.100101.

Actividad 2.3

Convierte los siguientes números a decimal. Comprueba la solución.

a) 1011110101_2 = ___$_{10}$ b) 10120_3 = ___$_{10}$ c) 1720.40651_8 = ___$_{10}$
b) d) $14E1_{16}$ = ___$_{10}$

Solución: a) 757_{10} b) 96_{10} c) 976.513_{10} d) 5345_{10}

Actividad 2.4

Convierte los siguientes números entre binario, octal y hexadecimal según se indica. Comprueba la solución.

a) $5AC_{16}$ = ___$_2$ b) 567_8 = ___$_2$ c) 1011011011.11111010_2 = ___$_{16}$
b) d) 111111000000.101011_2=___$_8$

Solución: a) 010110101100_2 b) 101110111_2 c) $2DB.FA_{16}$ d) 7700.53_8

Suma en binario

La primera operación matemática en binario es la suma; para esto debemos seguir las siguientes reglas básicas.

0 + 0 = 0	La suma es 0 y el acarreo es 0
0 + 1 = 1	La suma es 1 y el acarreo es 0
1 + 0 = 1	La suma es 1 y el acarreo es 0
1 + 1 = 10	La suma es 0 y el acarreo es 1

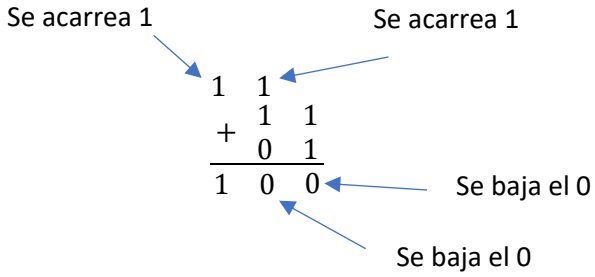
Como se observa en la tabla anterior las operaciones son muy similares a lo que se hace en decimal, sin embargo, se debe tener cuidado en 1+1, pues esta no es 2, es 10, pero esta cifra en realidad quiere decir que 'se baja cero y se lleva uno', como se muestra en los siguientes ejemplos.

Ejemplo 2.6

Realiza las siguientes sumas en binario.

a) $\begin{array}{r} 1\ \ 1 \\ +\ 0\ \ 1 \\ \hline \end{array}$

En la columna de la derecha, en 1 + 1 se baja el 0 y se acarrea 1. En la columna de la izquierda hay un acarreo de la suma anterior, se suma con el otro 1, por lo que se vuelve a obtener un 10, es decir se baja el cero y acarrea otro uno, el cual baja sin sumarse al no haber otro número.



El mismo procedimiento se utiliza en los siguientes incisos.

b)
$$\begin{array}{r} 0\ 1\ 1 \\ +\ 1\ 1\ 1 \\ \hline 1\ 0\ 1\ 0 \end{array}$$

c)
$$\begin{array}{r} 1\ 0\ 0\ 1 \\ +\ 1\ 0\ 1\ 1 \\ \hline 1\ 1\ 1\ 1 \\ \hline 1\ 0\ 0\ 0\ 1\ 1 \end{array}$$

Actividad 2.5

Realiza las siguientes sumas en binario. Comprueba la solución.

a) 1 0 + 1

b) 1 0 + 1 1 1

c) 1 0 1 + 1 0 1

Solución: a) 1 1 b) 1001 c) 1111

Multiplicación en binario

Al igual que la suma, la multiplicación se realiza de acuerdo con una tabla, que se muestra a continuación.

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

El procedimiento de la multiplicación en binario es igual a la multiplicación en decimal. Se realiza cada multiplicación parcial y el resultado se va recorriendo a la izquierda, una vez realizada cada multiplicación, se hace la suma, como se muestra en el siguiente ejemplo.

Ejemplo 2.7

A continuación se muestran dos multiplicaciones en binario.

a)

Sumas parciales

```
          1  1  0  1
        X 1  0  1
     ------------------
          1  1  0  1
       1  1  0  1
    1  1  0  1
  ---------------------
1 0  0  0  0  0  1
```

b)

```
  1  0  1  1  1   1  0
     X        1  1  0  1
  ---------------------
1 0 0 1 1 0 0 0 1 1 0
```

Actividad 2.6

Realiza las siguientes multiplicaciones en binario. Comprueba la solución.

a) 1 0 1 1 X 1 1

b) 1 0 1 1 1 1 X 1 0 0 1

Solución: a) 100001 b) 110100111

Resta en binario

Las siguientes, son las cuatro reglas básicas de la resta en binario.

0 – 0 = 0
1 – 0 = 1
1 – 1 = 0
0 – 1 = Se pide una base (2) y a la siguiente columna se lleva un acarreo negativo (-1)

En los siguientes ejemplos se explica el procedimiento de resta binaria.

Ejemplo 2.8

Realiza las siguientes restas en binario.

a) 110 6
-100 -4
010 2

En la resta anterior se usaron las primeras tres reglas, las cuales no requieren ningún acarreo.

b) 101
-010

$$\begin{array}{cccc} -1 & +2 & & \\ & 1 & 0 & 1 \\ -0 & 1 & 0 & \\ \hline & 0 & 1 & 1 \end{array} \qquad \begin{array}{r} 5 \\ -2 \\ \hline 3 \end{array}$$

En la columna de la derecha, la resta es directa, la regla dice que 1-0 es 1; en la columna central, 0 - 1 necesita pedir una base (+2), ahora se puede restar 2-1, cuyo resultado es 1, pero se lleva un acarreo negativo a la columna izquierda; como -1+1 es cero, entonces se baja el 0. El mismo procedimiento se utiliza en los siguientes incisos.

c) $\begin{array}{cccc} 1 & 1 & 0 & 1 \\ - & 1 & 1 & 1 \\ \hline 0 & 1 & 1 & 0 \end{array}$ d) $\begin{array}{ccccc} 1 & 0 & 0 & 1 & 0 \\ - & 1 & 0 & 1 & 1 \\ \hline 0 & 0 & 1 & 1 & 1 \end{array} \qquad \begin{array}{r} 18 \\ -11 \\ \hline 7 \end{array}$

Actividad 2.7

Realiza las siguientes restas en binario. Comprueba la solución.

a) $\begin{array}{r} 1110 \\ -111 \\ \hline \square \end{array}$ b) $\begin{array}{r} 11011111 \\ -101001 \\ \hline \square \end{array}$

Solución: a) 0111 b) 10110110

División en binario

El procedimiento de la división en binario es similar a la división en decimal; se busca cuántas veces cabe el divisor en el dividendo, sólo que en binario sólo hay dos posibilidades, cabe una o cero veces. En el siguiente ejemplo se muestra este procedimiento.

Ejemplo 2.9

Realiza las siguientes divisiones en binario.

a)

```
      1 0 0 1                    9
  11 | 1 1 1 0 1          3 |  29
     -11                        2
       1 0 1
       - 1 1
         1 0
```

b)

```
          111110                   38
 101 | 10111110            5 |  190
             0                     0
```

Actividad 2.8

Realiza las siguientes divisiones en binario. Comprueba la solución.

a) $10\overline{)101}$ b) $111\overline{)11111}$

Solución: a) 10, residuo 1 b) 100, residuo 11

Diseño de circuitos lógicos

Compuertas lógicas

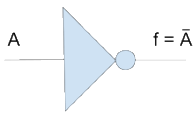
En 1854, George Boole inventó un tipo de álgebra con solo ceros y unos. Es lo que ahora se llama álgebra booleana. C. E. Shannon demostró que esta álgebra podía utilizarse para diseñar circuitos lógicos.

Podemos ver una compuerta lógica como un sistema que recibe entradas binarias, realiza una operación y se obtiene una salida también binaria. Las compuertas lógicas básicas y sus respectivas tablas de verdad se muestran a continuación.

Compuerta NOT (inversor, negado o complemento)

La salida de esta compuerta es el valor contrario al que entra. Su símbolo es una comilla (‘) o un gorrito (¯).

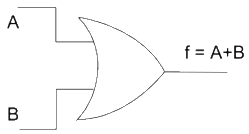
A	f
0	1
1	0



Compuerta OR

La salida de esta compuerta es uno si cualquiera de las entradas es uno, en caso contrario es cero. Esta compuerta corresponde a la suma (+).

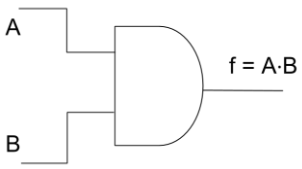
A	B	f
0	0	0
0	1	1
1	0	1
1	1	1



Compuerta AND

Para que la salida sea uno, ambas entradas deben ser uno, en caso contrario es cero. Esta compuerta corresponde a la multiplicación y su símbolo es el (·), aunque se puede omitir.

A	B	f
0	0	0
0	1	0
1	0	0
1	1	1



Para realizar correctamente las operaciones, es necesario aplicar la siguiente jerarquía.

1. Paréntesis
2. NOT
3. AND
4. OR

Finalmente, para saber cuántas filas tendrá una tabla de verdad, hay que contar el número de variables de la función booleana, que llamaremos n, luego hacemos la operación 2^n. El número de columnas corresponde al número de variables.

Ejemplo 2.10

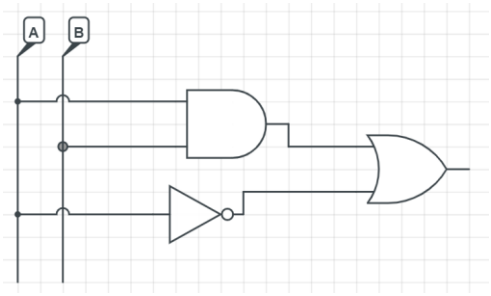
Encuentra la tabla de verdad y el circuito lógico de las siguientes funciones booleanas.

a) $f = AB + \bar{A}$
b) $f = x + \bar{y}z$
c) $z = A\bar{B} + \overline{(C + D)}$

a) $f = AB + \bar{A}$

Como se trata de dos variables, la tabla tendrá cuatro filas, pues $2^2 = 4$. Por jerarquía de operaciones, primero se escribe la columna con la operación $\bar{A}$ y luego la columna con la multiplicación AB. Finalmente se escribe la columna de f, que es el resultado de la suma de AB con $\bar{A}$.

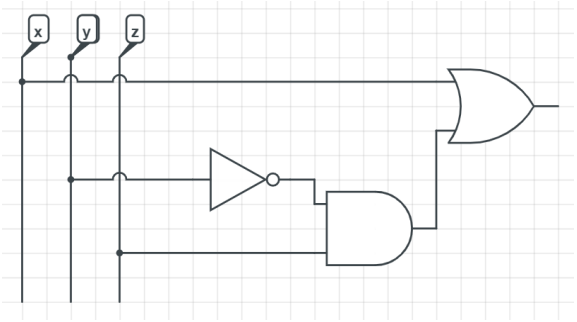
n	A	B	$\bar{A}$	AB	$f = AB + \bar{A}$
1	0	0	1	0	1
2	0	1	1	0	1
3	1	0	0	0	0
4	1	1	0	1	1



El mismo procedimiento se emplea en los siguientes incisos.

b) $f = x + \bar{y}z$

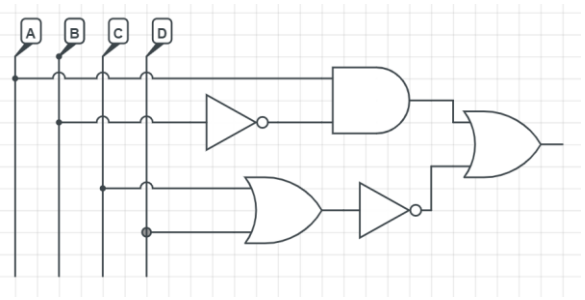
n	X	Y	Z	f
1	0	0	0	0
2	0	0	1	1
3	0	1	0	0
4	0	1	1	0
5	1	0	0	1
6	1	0	1	1
7	1	1	0	1



8	1	1	1	1

c) $z = A\bar{B} + \overline{(C + D)}$

n	A	B	C	D	Z
1	0	0	0	0	1
2	0	0	0	1	0
3	0	0	1	0	0
4	0	0	1	1	0
5	0	1	0	0	1
6	0	1	0	1	0
7	0	1	1	0	0
8	0	1	1	1	0
9	1	0	0	0	1
10	1	0	0	1	1
11	1	0	1	0	1
12	1	0	1	1	1
13	1	1	0	0	1
14	1	1	0	1	0
15	1	1	1	0	0
16	1	1	1	1	0



Actividad 2.9

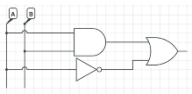
Encuentra la tabla de verdad y el circuito lógico de las siguientes funciones booleanas.

a) $z = \bar{A} + AB$ b) $f = XY + X\bar{Y}Z$

Solución:

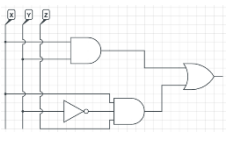
a)

n	A	B	Z
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	1



b)

n	X	Y	Z	f
1	0	0	0	0
2	0	0	1	0
3	0	1	0	0
4	0	1	1	0
5	1	0	0	0
6	1	0	1	1
7	1	1	0	1
8	1	1	1	1



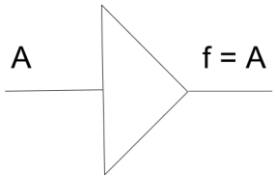
Otras compuertas lógicas

Las siguientes son las compuertas lógicas complementarias a las básicas.

Buffer

La salida es igual a la entrada.

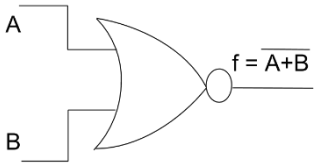
A	f
0	0
1	1



NOR

Es el negado de la compuerta OR

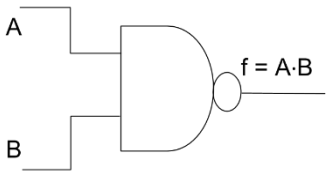
A	B	f
0	0	1
0	1	0
1	0	0
1	1	0



NAND

Es el negado de la compuerta AND

A	B	f
0	0	1
0	1	1
1	0	1
1	1	0

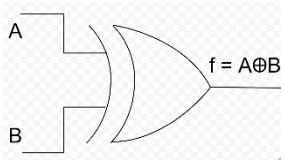


Hay que recordar que $\overline{(\boldsymbol{A}+\boldsymbol{B})} \neq \overline{\boldsymbol{A}}+\overline{\boldsymbol{B}}$

XOR

Si las entradas son distintas, la salida es uno; si las entradas son iguales, la salida es cero

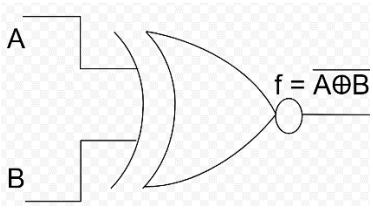
A	B	f
0	0	0
0	1	1
1	0	1
1	1	0



XNOR

Es el negado de XOR

A	B	f
0	0	1
0	1	0
1	0	0
1	1	1

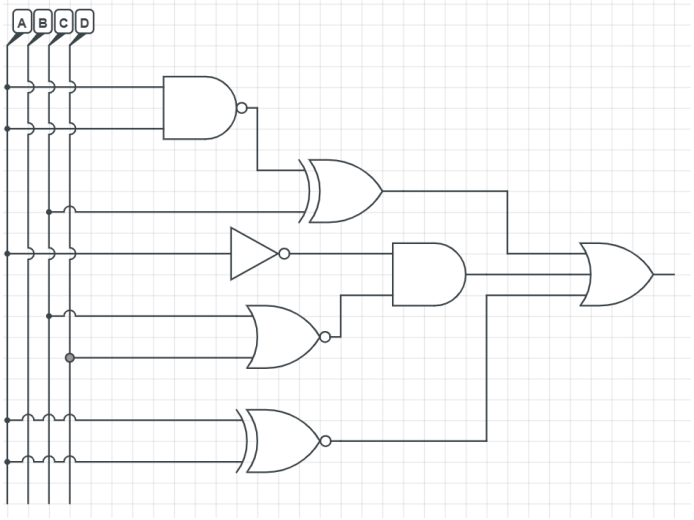


Ejemplo 2.11

Encontrar la tabla de verdad y el circuito de la siguiente función booleana.

$$f = \overline{AB} \oplus \mathrm{C} + \bar{A}\,\overline{(C + D)} + \overline{A \oplus \mathrm{B}}$$

Si se siguen las reglas antes mencionadas, el resultado final es el siguiente.



n	A	B	C	D	f
1	0	0	0	0	1
2	0	0	0	1	1
3	0	0	1	0	1
4	0	0	1	1	1
5	0	1	0	0	1
6	0	1	0	1	1
7	0	1	1	0	0
8	0	1	1	1	0
9	1	0	0	0	1
10	1	0	0	1	1
11	1	0	1	0	0
12	1	0	1	1	0
13	1	1	0	0	1
14	1	1	0	1	1
15	1	1	1	0	1
16	1	1	1	1	1

Actividad 2.10

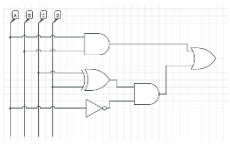
Encuentra la tabla de verdad y el circuito lógico de la siguiente función booleana.

a) $z = AB + (C \oplus \mathrm{D})\bar{A}$

Solución:

a)

n	A	B	C	D	Z
1	0	0	0	0	0
2	0	0	0	1	1
3	0	0	1	0	1
4	0	0	1	1	0
5	0	1	0	0	0
6	0	1	0	1	1
7	0	1	1	0	1
8	0	1	1	1	0
9	1	0	0	0	0
10	1	0	0	1	0
11	1	0	1	0	0
12	1	0	1	1	0
13	1	1	0	0	1
14	1	1	0	1	1
15	1	1	1	0	1
16	1	1	1	1	1



Obtención de funciones lógicas y su simplificación

Hasta ahora se ha obtenido la tabla de verdad y el circuito lógico a partir de una función booleana. En la práctica, es más útil el procedimiento inverso, es decir, a partir de una tabla de verdad o circuito lógico, se obtiene la función booleana.

El procedimiento consiste en tomar sólo las combinaciones donde la salida es uno; en estos casos, se escriben las entradas como una multiplicación usando una compuerta AND. Con el siguiente ejemplo se muestra con mayor claridad ese procedimiento.

Ejemplos 2.12

Encuentra la función booleana de las siguientes tablas de verdad.

n	x	y	F
1	0	0	1
2	0	1	1
3	1	0	0
4	1	1	0

El primer paso consiste en ubicar dónde la salida es 1. Como se observa en la tabla, la función F es 1 en el renglón 1 y 2.

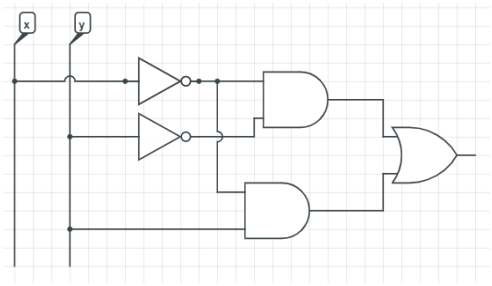
El segundo paso es escribir las entradas como una multiplicación tal que se cumpla que el producto sea 1.

En el renglón 1, debido que x y y son cero, la multiplicación quedaría $\overline{x}\,\overline{y}$. En el segundo renglón, como x es cero y y es uno, la multiplicación sería $\overline{x}y$.

En el tercer paso se toman las multiplicaciones y se suman, es decir, la función f quedaría como

$$F = \overline{x}\,\overline{y} + \overline{x}y$$

El cuarto paso es construir el circuito lógico como se ha hecho anteriormente.



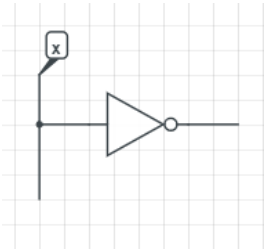
Para simplificar, se factoriza el termino común.

$$F = \bar{x}(\bar{y} + y)$$

Por la regla número 6 de la Tabla 2.1, la función es

$$F = \bar{x}$$

El circuito de esta función simplificada es el siguiente. Como se puede observar, la cantidad de compuertas y de cables bajó significativamente. Esto es importante, pues significa que nos ahorraremos tiempo, dinero y la probabilidad de tener algún error baja.



La Tabla 2.1 muestra las reglas para simplificar funciones.

1	A+0=A
2	A+1=1
3	A*0=0
4	A*1=A
5	A+A=A
6	A+A’=1
7	A*A=A
8	A*A’=0
9	A’’=A
10	A+AB=A
11	A+A’B=A+B
12	A+BC=(A+B)(A+C)

Tabla 2.1. Reglas de simplificación.

Ejemplo 2.13

Encuentra la función booleana de las siguientes tablas de verdad.

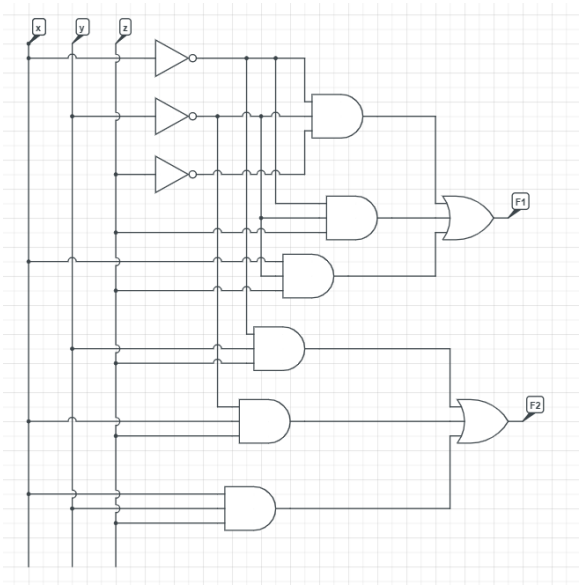
x	y	z	F1	F2
0	0	0	1	0
0	0	1	1	0
0	1	0	0	0
0	1	1	0	1
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	1

Las funciones boolenas son las siguientes.

$$F1 = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + x\bar{y}z$$

$$F2 = \bar{x}yz + x\bar{y}z + xyz$$

El circuito de las funciones anteriores es el siguiente.



La función F1 se simplifica de la siguiente manera.

Se factorizan los términos comunes más convenientes.

$$F1 = \bar{x}\bar{y}\,(\bar{z} + z) + x\bar{y}z$$

Se usa la propiedad 6.

$$F1 = \bar{x}\bar{y} + x\bar{y}z$$

Esta nueva función se puede factorizar $\bar{y}$.

$$F1 = \bar{y}(\bar{x} + xz)$$

Por la propiedad 12, la función queda

$$F1 = \bar{y}(\bar{x} + x)(\bar{x} + z)$$

$$\boldsymbol{F1 = \bar{y}(\bar{x} + z)}$$

Ahora se simplifica la función F2.

$$F2 = \bar{x}yz + x\bar{y}z + xyz$$

En esta función pueden simplificarse yz o xz. Se escoge la primera opción, las dos son aceptables.

$$F2 = yz(\bar{x} + x) + x\bar{y}z$$

$$F2 = yz + x\bar{y}z$$

Por la propiedad 6

$$F2 = yz + x\bar{y}z$$

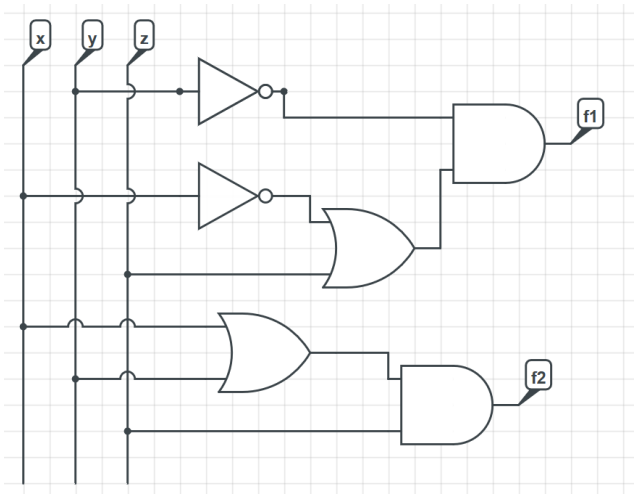
Se factoriza z.

$$F2 = z(y + x\bar{y})$$

Por la propiedad 11

$$\boldsymbol{F2 = z(y + x)}$$

El circuito simplificado es el siguiente.



Atención
Un error frecuente es el siguiente: $\bar{A}\bar{B} + AB \neq \overline{AB} + AB$. Las expresiones de la izquierda y derecha no son equivalentes.

Actividad 2.11

De la siguiente tabla de verdad, encuentra:

La función booleana, la función simplificada, el circuito lógico de la función original y de la función simplificada.

x	y	z	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	1
1	1	1	1

Solución:

De la función original

$$F = \bar{x}\bar{y}\bar{z} + \bar{x}\bar{y}z + xy\bar{z} + xyz$$

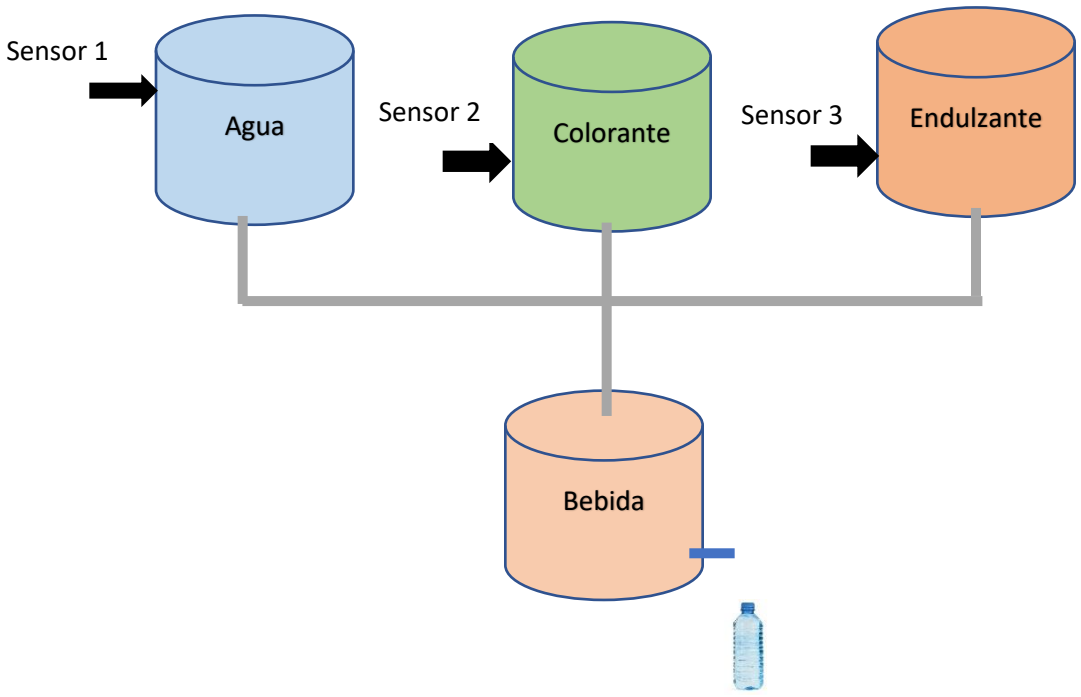
De la función simplificada

$$F = \bar{x}\bar{y} + xy$$

Problemas que se resuelven con circuitos lógicos

Ejemplo 2.14

Una fábrica de bebidas embotelladas desea automatizar su proceso. La fábrica cuenta con tres tanques: de agua, de colorante líquido y de endulzante líquido. Cada tanque cuenta con un sensor de nivel para saber si hay suficiente líquido en cada caso, tal como se muestra en la figura.



La planta realiza dos tipos de bebidas.

1. Se llama bebida dietética cuando solo hay agua, o a cuando hay agua y colorante.
2. Se llama refresco a la mezcla de agua, colorante y endulzante.

Diseña el circuito lógico para que la planta funcione como se indica.

El primer paso consiste en identificar las entradas y nombrarlas. En este caso son:

Entrada	Nombre
Sensor 1	S1
Sensor 2	S2
Sensor 3	S3

Ahora se identifican las salidas y se les asigna un nombre; estas corresponden a las mezclas anteriormente indicadas, como se muestra en la siguiente tabla.

Salida	Nombre
Bebida dietética	Dieta
Refresco	Refresco

El segundo paso consiste en realizar la tabla de verdad con esta información.

S1	S2	S3	Dieta	Refresco
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	1	0
1	0	1	0	0
1	1	0	1	0
1	1	1	0	1

El tercer paso es encontrar la función booleana.

$$Dieta = S1\overline{S2}S3 + S1S2\overline{S3}$$

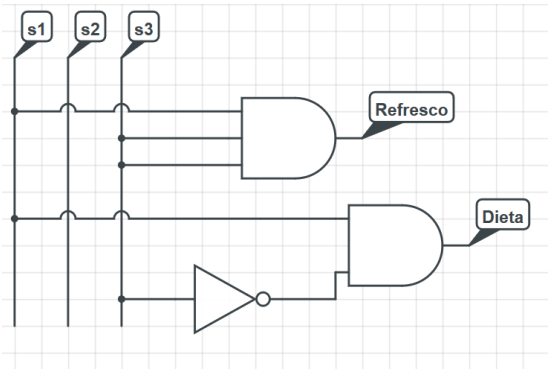
$$Refresco = S1S2S3$$

Cuarto paso. Hacer alguna simplificación si es posible.

Sólo la función dieta se puede simplificar.

$$Dieta = S1\overline{S3}$$

El circuito lógico es el siguiente.



Actividad 2.12

Una puerta automática tiene una cerradura electrónica de tres botones (A, B y C). La puerta tiene solo dos usuarios; estos escogieron la siguiente combinación para abrir: cuando se presiona A y C y cuando se presionan los tres botones. Además, los usuarios acordaron que cuando se presionan los botones B y C suene una alarma.

Solución:

$$Abrir = A\bar{B}C + ABC$$
$$Abrir = AC$$
$$Alarma = \bar{A}BC$$

Ejercicios

1. Convierte los siguientes números decimales a base binaria.

a) 1573_{10} = _____$_2$

b) 5731_{10} = _____$_2$

c) 8355_{10} = _____$_2$

d) 12001_{10} = _____$_2$

e) 1000000_{10} = _____$_2$

2. Convierte de base a base según se indica.

a) 59_{10} = _____$_5$
b) 401_{10} = _____$_6$
c) 88_{10} = _____$_8$
d) 214_{10} = _____$_9$
e) 332_{10} = _____$_{16}$
f) 1234_{10} = _____$_{16}$
g) 321.936_{10} = _____$_2$
h) 43.74_{10} = _____$_7$

3. Convierte los siguientes números a decimal.

a) 11110100001001000000_2 = _____$_{10}$
b) 1111.10111_2 = _____$_{10}$
c) $2A6_{16}$ = _____$_{10}$

4. Realiza las siguientes operaciones en binario.

a)
```
 1 1 1 1
 1 1 1 1
 1 1  1 1
+1 1 1 1
```

b)
```
 1 1 1 0 1
 1 0 1 1 1
 1 1 1 0 1
 1 1 1 1 1
+1 0 0 0 1
```

c)
```
 1 1 1 1 1
X 1 1 1 1 1
```

d)
```
 10101010
 −11111
```

e)
```
 1000000
−111111
```

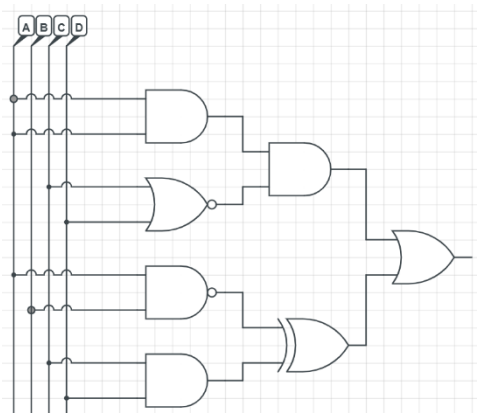
f) $1000\overline{)10001000}$

g) $110\overline{)101110110}$

5. Encuentra la tabla de verdad y el circuito lógico de la siguiente función booleana.

$$z = AB + \bar{C}(\overline{D + B})$$

6. Del siguiente circuito lógico, encuentra la tabla de verdad y la función booleana.



7. De las siguientes tablas de verdad, encuentra:

a) Función booleana.

b) Función simplificada.

c) Circuito lógico de la función original y de la función simplificada.

7.1)

x	y	z	F1	F2	F3
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	0	0
0	1	1	0	0	1
1	0	0	0	1	1
1	0	1	0	1	1
1	1	0	1	1	0
1	1	1	0	1	0

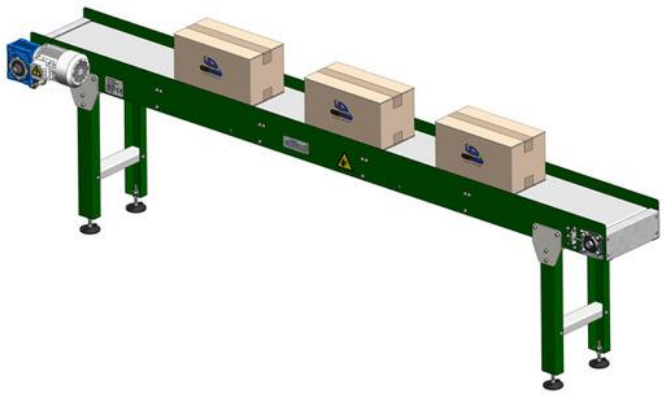
7.2)

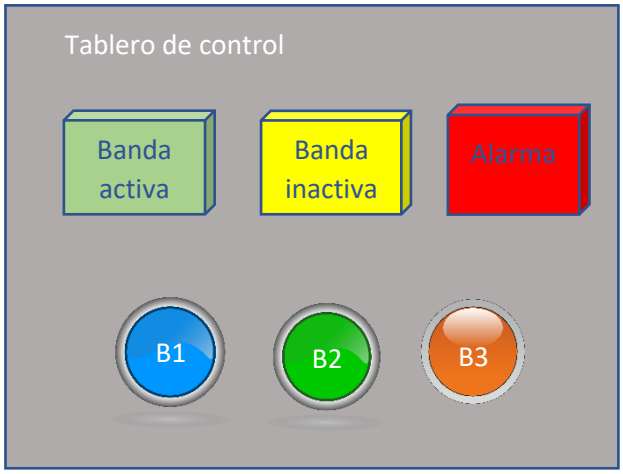
A	B	C	D	F1	F2
0	0	0	0	0	1
0	0	0	1	0	1
0	0	1	0	0	0
0	0	1	1	0	0
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	0	1	1	1
1	0	1	0	0	0
1	0	1	1	1	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	0

8. Una fábrica de chocolates necesita automatizar su línea de producción. Para empaquetar sus productos adquirió una banda transportadora que se mueve por medio de un motor y que cuenta con un tablero de control con tres botones y tres indicadores como se observa en la imagen.
   - El motor funciona con tres velocidades distintas. Cuando se presiona B1 la banda se mueve lentamente; si presiona B1 y B2 la velocidad es media; si se presionan los tres botones al mismo tiempo la banda se mueve rápidamente. En cualquiera de estos tres casos se enciende el indicador de *Banda activa*.
   - Si ningún botón está presionado, el indicador de *Banda inactiva* deberá estar encendido.

- Si se presiona sólo el botón B2, o el botón B1 junto con B3, se encenderá el indicador de *Alarma*.

Diseña un circuito lógico para que las entradas (botones) y salidas (indicadores) funcionen como se mencionó en los puntos anteriores.





Unidad III

Metodología de solución de problemas e introducción al lenguaje de programación Java

Unidad III: Metodología de solución de problemas e introducción al lenguaje de programación Java

En la presente unidad, se adecuará el concepto de sistemas (entrada-proceso-salida), correspondiente a la primera unidad, a la metodología de resolución de problemas, pues se abordarán los problemas como un sistema que recibe datos, se procesan y se obtiene una salida con el resultado. De esta manera, se articularán todos los contenidos del temario en una sola línea secuencial que tendrá más sentido para el estudiante.

De igual manera, en esta unidad se sugerirá un método para resolver problemas. Se mostrará la forma de representar una solución por medio de algoritmos, específicamente con diagramas de flujo y pseudocódigos.

Más adelante se abordarán conceptos básicos de Java, esto es, las características, lo que lo distingue de otros lenguajes de programación, los identificadores, los tipos de datos, las variables, las operaciones aritméticas y de comparación, y entrada y salida de datos.

Como **propósito** de la presente unidad, el alumno:

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

A su vez, los **aprendizajes** de la presente unidad son los siguientes:

El alumno:

1. Define el concepto de problema.
2. Identifica los elementos de un problema.

3. Describe la diferencia entre problemas determinísticos, probabilísticos, secuenciales y condicionales.
4. Conoce las etapas de la metodología de solución de problemas.
5. Analiza el resultado de expresiones aritméticas utilizando la jerarquía de las operaciones.
6. Construye expresiones lógicas utilizando operadores relacionales y lógicos.
7. Conoce el concepto de algoritmo, diagrama de flujo y pseudocódigo.
8. Elabora el algoritmo, diagrama de flujo y pseudocódigo para problemas secuenciales.
9. Conoce la historia del lenguaje de programación Java.
10. Conoce las características básicas del lenguaje de programación Java.
11. Conoce el entorno de desarrollo para el lenguaje de programación Java
12. Realiza programas empleando el método de salida de datos.
13. Realiza programas empleando la Clase Scanner para la entrada de datos.
14. Elabora el algoritmo, diagrama de flujo y pseudocódigo para problemas condicionales.
15. Construye programas de computadora que resuelvan problemas condicionales.
16. Elabora el algoritmo, diagrama de flujo y pseudocódigo para problemas condicionales múltiples.
17. Construye programas de computadora que resuelvan problemas que involucren la toma de decisiones múltiples.
18. Elabora el algoritmo, diagrama de flujo y pseudocódigo para resolver problemas de estructura de ciclo.
19. Construye programas de computadora que empleen la sentencia for.
20. Elabora el algoritmo, diagrama de flujo y pseudocódigo de problemas de ciclo (cíclicos) que satisfagan una condición.
21. Construye programas de computadora que involucren la sentencia while.

Definiciones y conceptos generales de un problema

Específicamente, las computadoras resuelven problemas en los que se reciben datos, luego estos son procesados y finalmente se obtiene una salida o resultado con la solución. En la Figura 3.1 visualizamos esto en un diagrama de bloques.

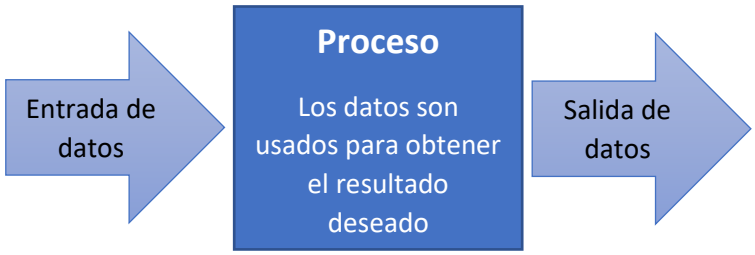


Figura 3.1. Esquema que muestra cómo se resuelve un problema con computadoras.

Básicamente, se siguen las siguientes fases para resolver un problema con computadoras; a su vez se resumen en una pregunta que debe ser contestada para poder continuar.

1. **Análisis del problema**. Consiste en comprender el problema y buscar métodos, operaciones o acciones que lleven a la solución. ¿Cuál es el mejor procedimiento para llegar a la solución?

2. **Entradas requeridas**. Consiste en buscar los datos que se necesitan procesar para encontrar la solución. ¿Cuáles son las entradas que necesito para que el procedimiento arroje la respuesta deseada?

3. **Salidas deseadas**. Una vez que se propone un procedimiento de solución, y se tiene claro qué entradas se requieren, se verifica que efectivamente la salida sea la esperada. ¿Es la salida la correcta?

Si la salida es efectivamente la correcta, entonces la solución se guarda, en caso contrario, estas etapas se repiten, se ajustan, o tal vez incluso se buscan nuevos procedimientos y entradas para encontrar la salida deseada.

Algoritmos, diagramas de flujo y pseudocódigo

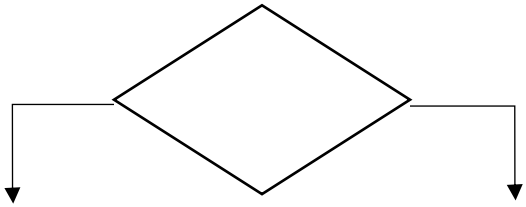
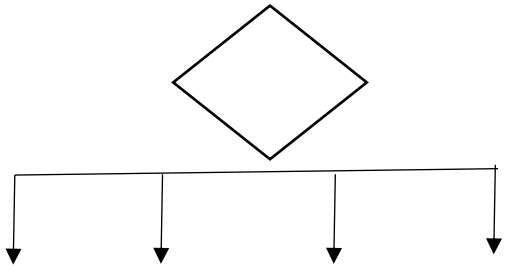
Las computadoras son máquinas que siguen instrucciones. Estas instrucciones deben ser muy precisas y seguir un orden lógico y claro, sin ambigüedades. Por eso, antes de escribir un programa que será ejecutado por una computadora, es necesario escribir *a mano* la solución con los pasos a seguir por la computadora. Estos pasos a seguir, o instrucciones, que nos llevan a la solución de un problema dado, es lo que llamamos algoritmo, y se caracterizan por ser:

1. Precisos: los pasos a seguir se deben indicar sin ambigüedad, o lo que es lo mismo, con claridad.
2. Definidos: las salidas del algoritmo deben ser las mismas ante las mismas entradas.
3. Finitos: deben terminar en algún momento.

Definición. Un *algoritmo* es un conjunto ordenado de instrucciones que llevan a la solución de un problema.

Los algoritmos tienen diferentes representaciones, dos de las más comunes son los diagramas de flujo y pseudocódigos.

Los **diagramas de flujo** son una de las herramientas más utilizadas para representar un algoritmo, pues gracias a que utilizan figuras y flechas, visualmente son muy intuitivos y fáciles de seguir. Para lograr esto, cada figura tiene una función en particular. En la Tabla 3.1 se muestran las figuras, mejor conocidas como símbolos, y su significado. La utilización de estos, y solos estos símbolos, es muy importante, ya que responden una estandarización internacional, es decir, si representas una solución con ellos podrá ser entendido por cualquier programador en cualquier parte del mundo.

Símbolo	Significado
	Marca el inicio y el fin del diagrama de flujo
	Indica la entrada o lectura de datos
	Indica que se realiza un proceso o una operación
	Representa una decisión. Dentro del rombo se escribe una condición o pregunta, si la respuesta es verdadera se toma un camino, en caso contrario se toma el otro
	Representa una decisión múltiple. Dentro del rombo se escriben varias opciones, dependiendo de la que se elija se toma un camino.

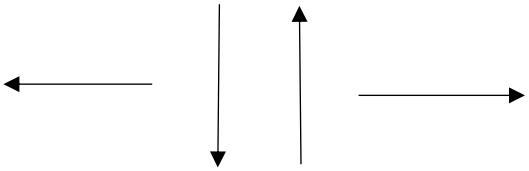
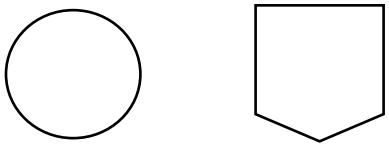
	Simboliza impresión de un dato o resultado. A diferencia del bloque de lectura, este es de escritura. Representa lo que va a aparecer en pantalla
	Estas flechas indican el flujo a seguir en el diagrama de flujo (nótese que son verticales y horizontales, no pueden estar inclinadas o cruzadas)
	Estos conectores se utilizan cuando el diagrama de flujo excede el espacio disponible. El de la izquierda, el círculo, se usa para saltos en una página, y el de la derecha para saltos entre páginas.

Tabla 3.1. Significado de los símbolos que componen un diagrama de flujo.

Los diagramas de flujo (y pseudocódigos) utilizan variables, constantes y operadores para procesar los datos de entrada y mostrar otros datos (la solución) a la salida.

Usamos variables y constantes para hacer operaciones como multiplicación, división, suma y resta, pero también comparaciones, como comparar si un número es mayor a otro. Para acceder o establecer una referencia a las constantes o variables usamos identificadores. La Tabla 3.2 muestra todas las operaciones y comparaciones permitidas.

Definición. Las *constantes* son objetos que nunca cambian mientras se ejecuta el algoritmo.

Definición. Las *variables* son objetos que cambian su valor mientras se ejecuta un programa.

Definición. Un *identificador* es el nombre que se le da a una variable o constante.

Símbolo	**Operación**
+	Suma
-	Resta
*****	Multiplicación
/	División
^	Potencia
>	Mayor que
<	Menor que
=	Igual que
>=	Mayor o igual que
<=	Menor o igual que
<>	Diferente que

Tabla 3.2. Operadores permitidos en un diagrama de flujo o pseudocódigo.

Veamos con un ejemplo cómo funciona un diagrama de flujo. Digamos que se nos pide diseñar un algoritmo que calcule el área de un círculo. Sigamos las fases para resolver un problema como se mencionó anteriormente.

1. **Análisis del problema**. Necesitamos calcular el área de un círculo. Esta es una fórmula conocida; en algún libro de geometría, una búsqueda en internet o seguramente de memoria, llegaremos a la fórmula $A = \pi r^2$.

2. **Entradas requeridas**. Al analizar la fórmula, vemos que para calcular el área de círculo son necesarias dos cosas, la longitud del radio y el número π, pero éste es una constante, que tomaremos como 3.1416, y tendrá por identificador 'pi' (es decir pi=3.1416), por lo que la única entrada necesaria es el radio. Usaremos el identificador 'r' para referirnos al radio, y como será un valor que será ingresado por el usuario, se tratará de una variable

3. **Salidas deseadas**. La solución al problema es el área, así pues, ésta es la salida deseada. Como es un valor que depende del radio, también es una variable, que en este caso llamaremos *'A'*. Para saber si el procedimiento es el adecuado, necesitamos proponer un diagrama de flujo y ver si funciona correctamente.

En la Figura 3.2 se muestra el diagrama de flujo que obtiene el área de un círculo; aunque no es parte propiamente del diagrama de flujo, del lado derecho, en los recuadros punteados, se explica cada paso.

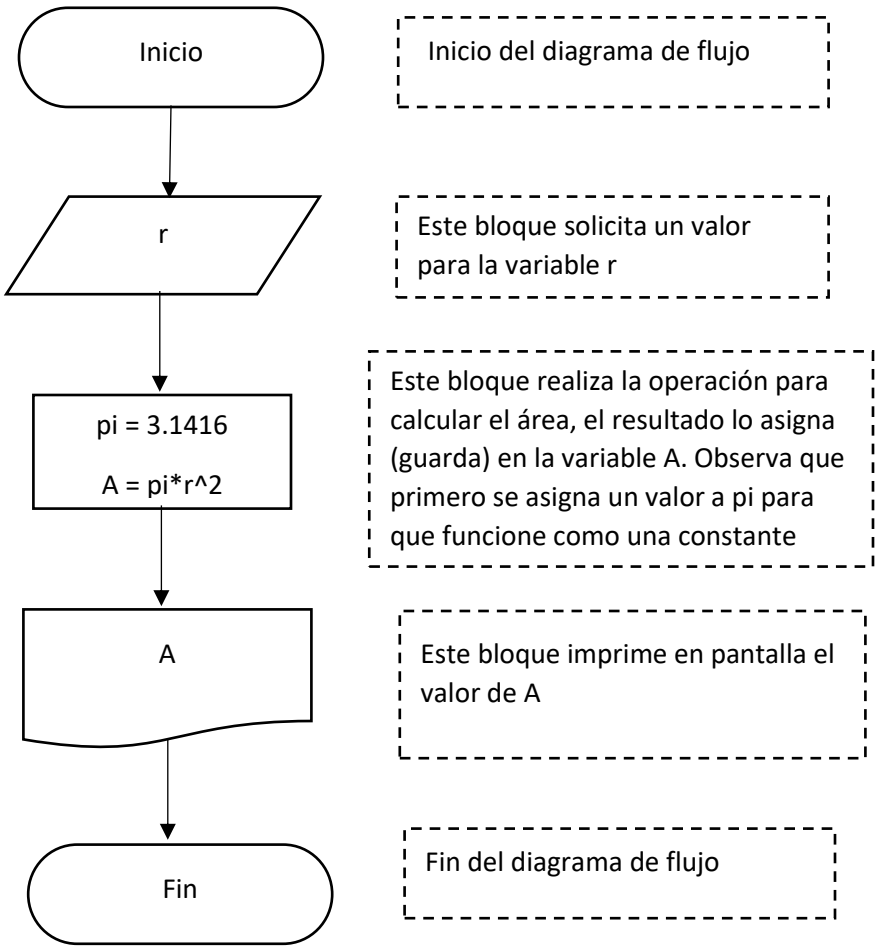


Figura 3.2. Diagrama de flujo que obtiene el área de un círculo.

Primeros pasos en Java

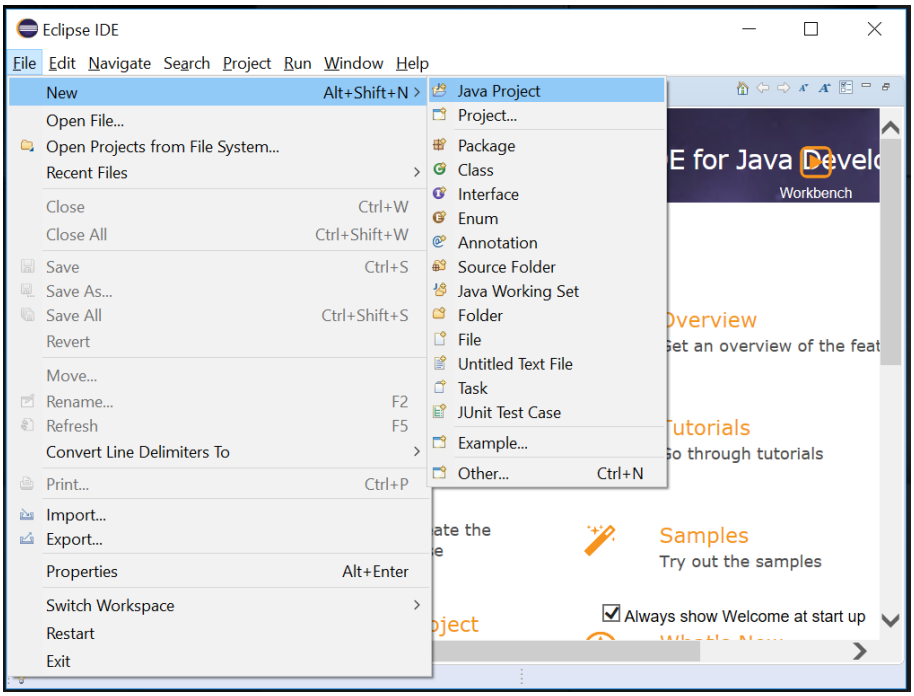
Para escribir un programa en Java, la mejor opción es contar con un entorno de desarrollo integrado, más conocido como IDE, por sus siglas en inglés. En un IDE los programas se pueden 1) escribir 2) compilar y 3) ejecutar.

Eclipse es un IDE muy popular porque, además de ser gratuito, contiene muchas herramientas útiles para los programadores.

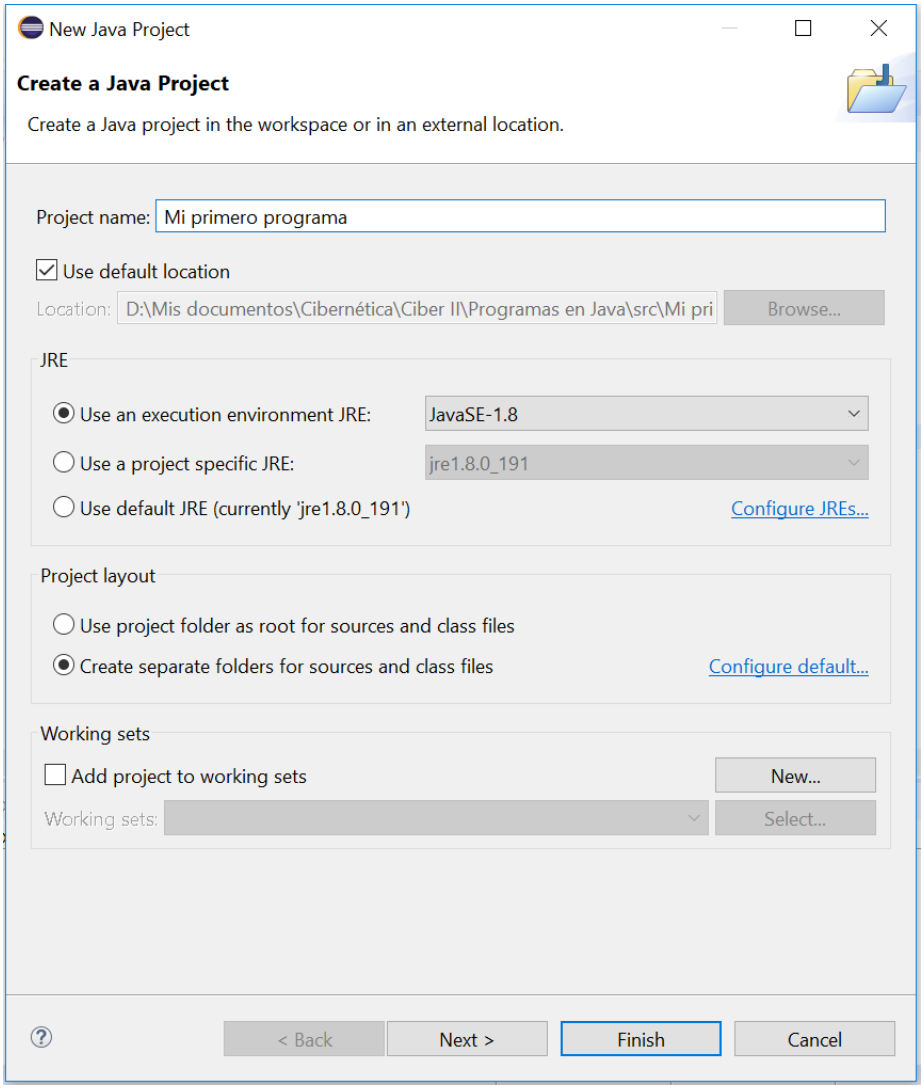
Sigue los siguientes pasos para escribir un programa en lenguaje Java con Eclipse.

1. Abre Eclipse. Te saldrá una ventana como la siguiente. Si es la primera vez que o usas cierra la ventana de bienvenida (Welcome).

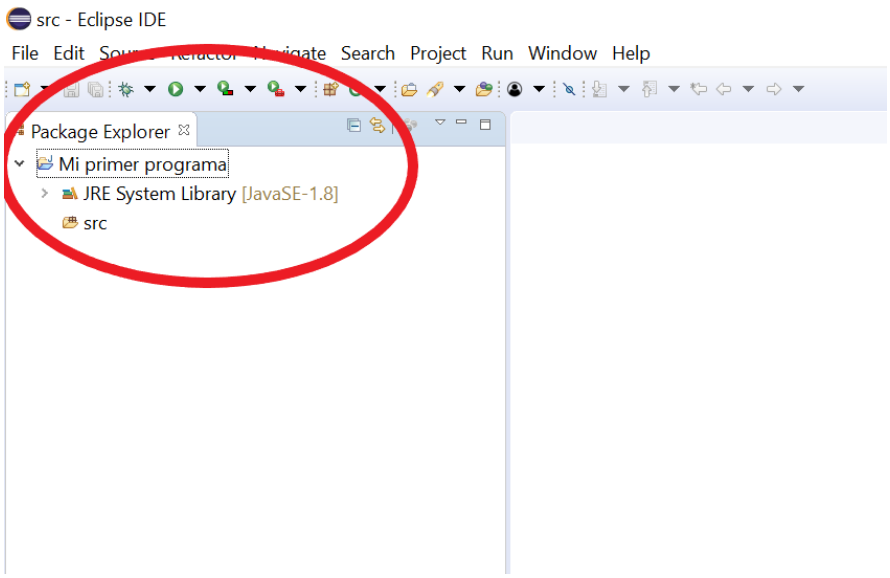
2. Ve al menú *File*, luego a *New* y finalmente a *Java Proyect*.



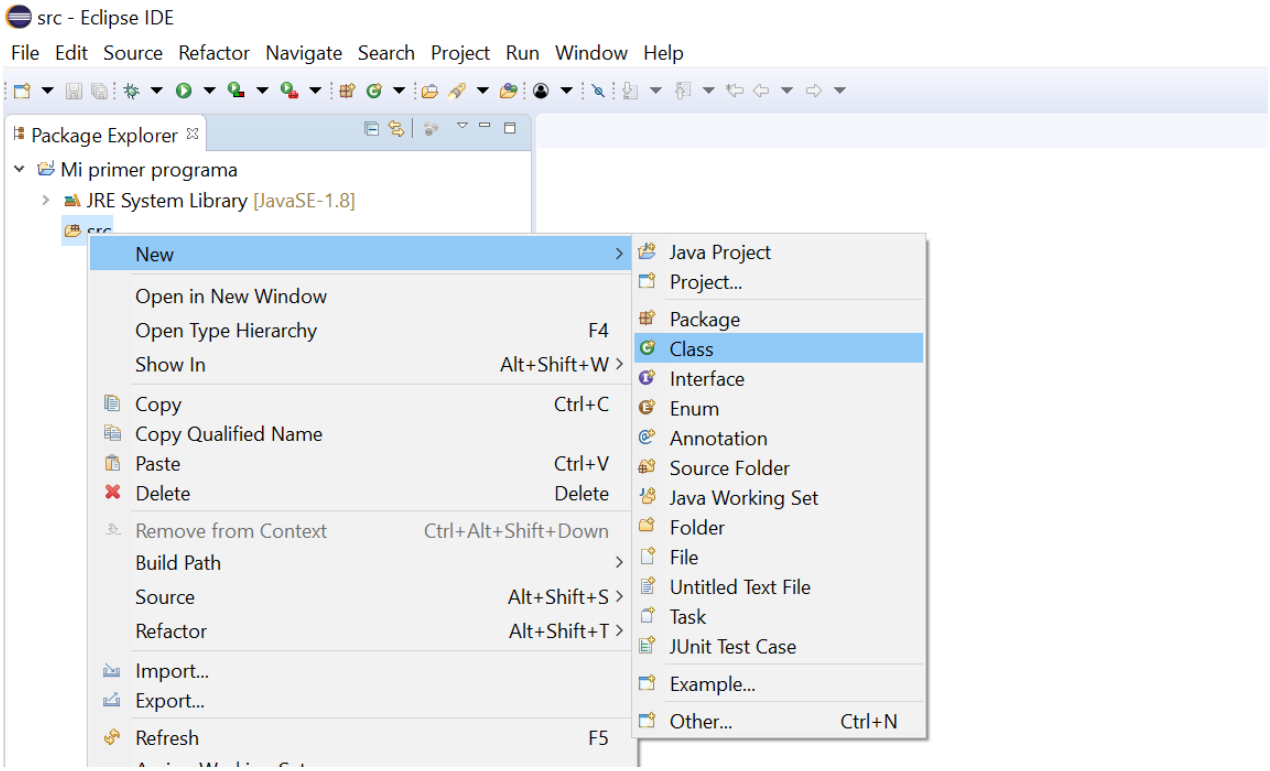
3. En la siguiente ventana escribe un nombre para tu proyecto. Quita la paloma de *Use default location* y busca la carpeta donde prefieres guardar tu proyecto. Finalmente da click en *Finish*.



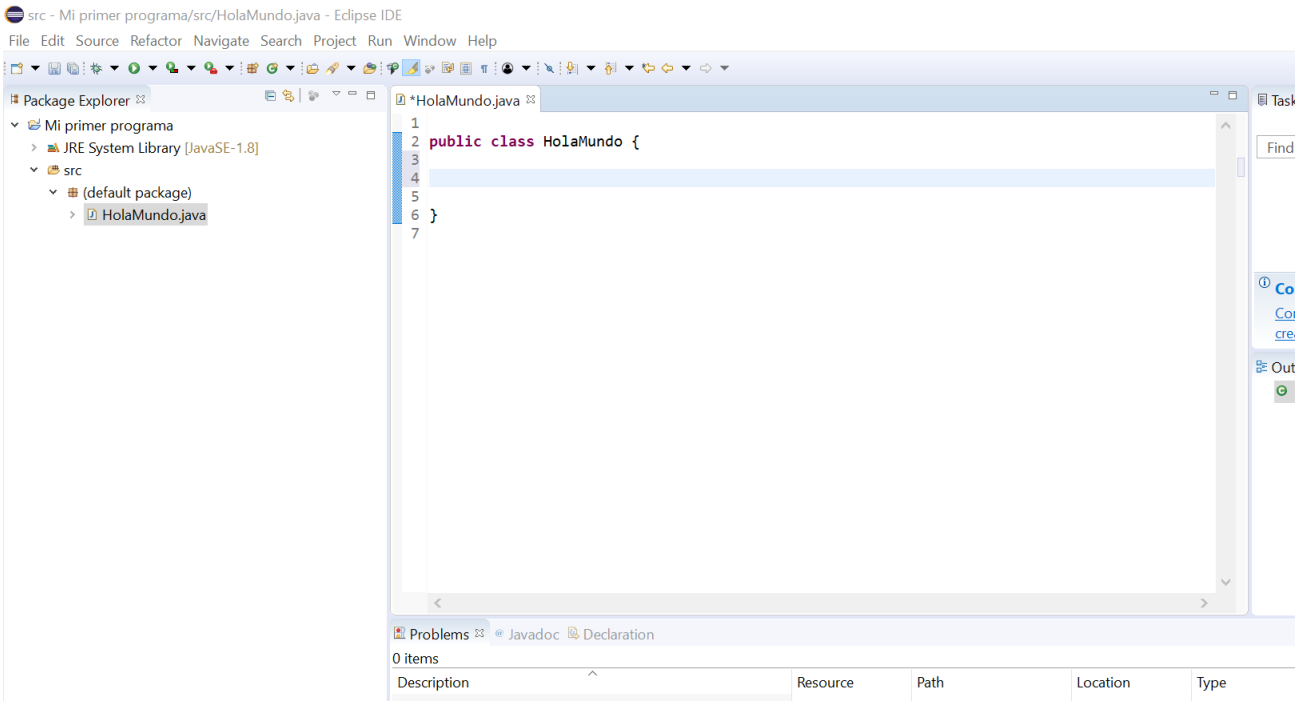
4. En la parte izquierda, hay un directorio de las carpetas (*Package Explorer*) creadas al haber hecho un nuevo proyecto. Hay que hacer clic en la flecha que aparece a un lado del nombre del proyecto para desplegar su contenido.



5. Da clic derecho en la carpeta *scr* (en ella se guardarán todas las clases o programas de nuestro proyecto), luego en *New* y escoger *Class*.



6. Emergerá una ventana en la cual hay que escribir el nombre de la clase o programa. Importante: Este nombre siempre se debe escribir con mayúscula, y si se usan más palabras deben ir también con mayúscula y sin espacio. Por ejemplo, este programa se llamará HolaMundo. El nombre de la clase debe coincidir con el del archivo creado originalmente.



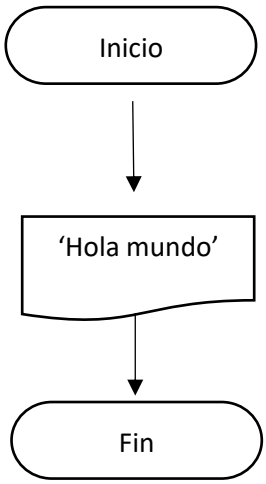
Los pasos anteriores son para comenzar a escribir un programa en Java. El programa en sí necesita una clase main, o principal, para funcionar. Es decir, todo programa debe tener la siguiente estructura.

```
public class NombreClase {

            public static void main(String[] argumentos){


            }

}
```

Dentro del *main* se escriben las sentencias del programa.

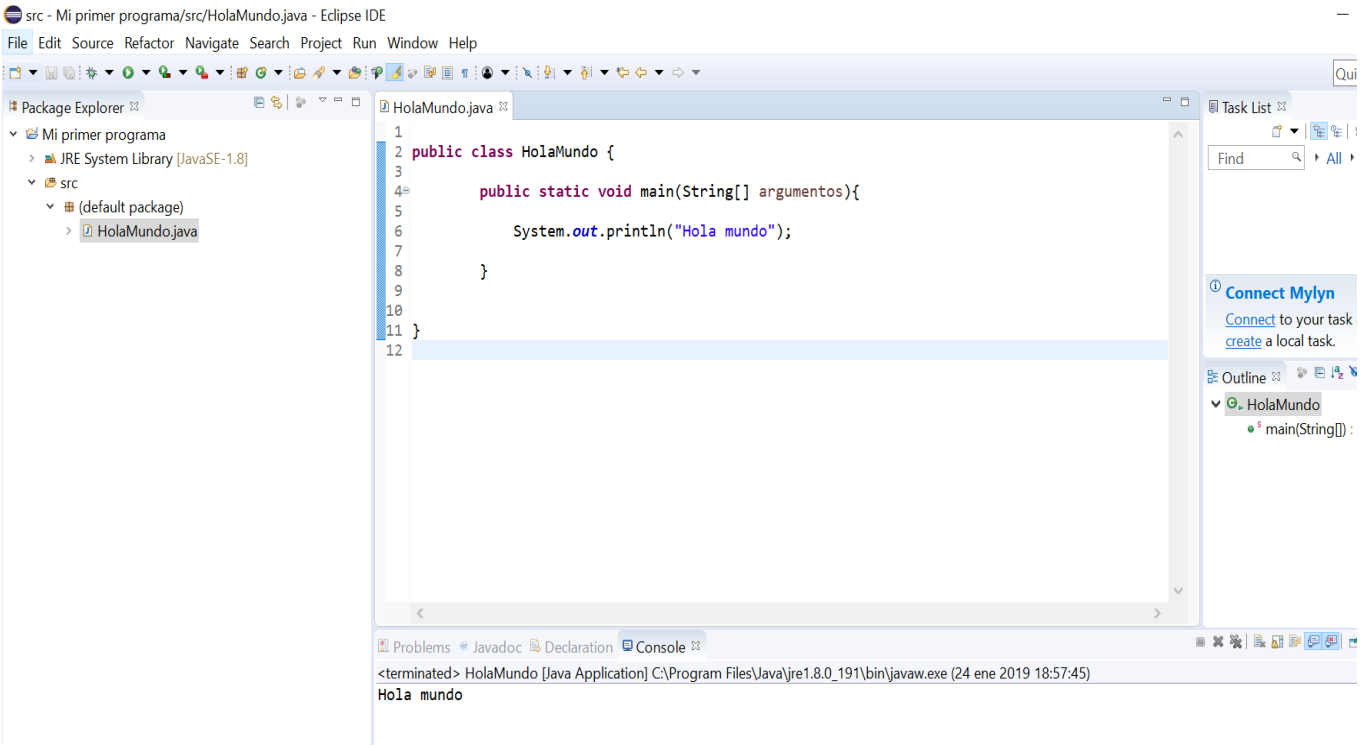
El siguiente programa imprime en pantalla el texto ‘Hola mundo’, su diagrama de flujo se muestra del lado derecho.

```
public class HolaMundo {

	public static void main(String[] argumentos){

		System.out.println("Hola mundo");

	}

}
```



Ejemplo 3.1

Escribe este programa en la parte del editor de texto del IDE, como se muestra en la siguiente imagen. Para ver el resultado se da clic en el botón de ejecución:



Si no hay ningún error, en la parte de abajo, aparecerá la consola, donde se desplegará el mensaje 'Hola mundo'. Prueba agregando otro mensaje.

```
public class HolaMundo {

public static void main(String[] argumentos){

    System.out.println("Hola mundo");
  System.out.println("Adiós mundo cruel");

                    }

                    }
```

Actividad 3.1

Escribe una carátula con los siguientes datos, y una definición de cibernética. También escribe en tu cuaderno el diagrama de flujo.

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

COLEGIO DE CIENCIAS Y HUMANIDADES – PLANTEL NAUCALPAN

CIBERNÉTICA Y COMPUTACIÓN

GRUPO

NOMBRE

FECHA

Definición de cibernética

Tipos de datos en Java

Un tipo de dato es el conjunto de valores que puede tomar una variable. Los siguientes son los tipos de datos simples.

Enteros

Son números completos, positivos y negativos. La palabra reservada es ***int***.

Ejemplo:

```
int a = 4558;
```

Reales o de punto flotante

Son números decimales. Su palabra reservada es ***float***.

Ejemplo:

```
float = 99.9f;
```

Caracteres

Corresponde a los símbolos Unicode o ASCII. (Busca en internet alguna tabla con estos símbolos). Su palabra reservada es ***char***.

Ejemplo:

char letra = 'A';

Lógicos

También conocidos como booleanos, toman valores true o false. Su palabra reservada es ***boolean***.

Ejemplo:

boolean indicador = true;

Ejemplo 3.2

El siguiente programa utiliza los tipos de datos simples para mostrar en pantalla los datos de una persona. Utiliza tus propios datos para escribir este ejemplo en Eclipse.

```
public class Presentacion {

	public static void main(String[] args) {

		int numero = 7;
		char simbolo = '@';
		float altura = 1.78f;
		boolean respuesta = false;

		System.out.println("Mi nombre es salvador y mi número
favorito es el "+numero+", mi símbolo es "+simbolo+" y mi altura es
"+altura);
		System.out.println("¿Me gustan los gatos? "+respuesta);

	}

}
```

Ejemplo 3.3

El siguiente programa usa un tipo de dato simple para calcular el área de un círculo.

```
public class Area {

	public static void main(String[] args) {
		float pi = 3.1416f;
		float area;
		float radio;

		radio = 2;

		area = pi*radio*radio;

		System.out.println("Para un radio "+radio+" el área es "+area);


	}

}
```

Actividad 3.2

Utiliza los tipos de datos de Java para escribir un programa que presente los datos de tres alumnos como se muestra a continuación.

Juan Álvarez tiene 17 años, un promedio de 9.53 y su identificador es el *. Inscrito: true

María Martínez tiene 16 años, un promedio de 8.12 y su identificador es el #. Inscrito: false

Esteban González tiene 19 años, un promedio de 6.7 y su identificador es el °. Inscrito: true

Clase Scanner: entrada de datos

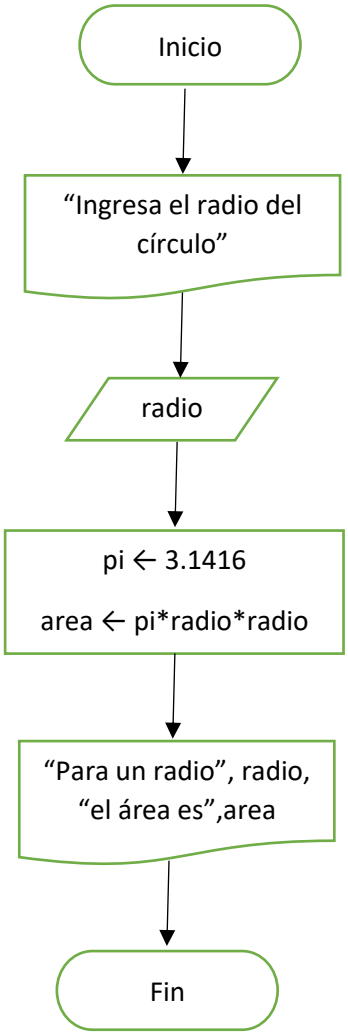
Para poder recibir datos desde el teclado es necesario declarar una variable del tipo Scanner. Esta es una clase que se usa para la entrada de datos de varios tipos (enteros, reales, caracteres, reales, etc.), pero ahora sólo usaremos datos de tipo real.

Ejemplo 3.4

El siguiente código ejemplifica cómo usar la clase Scanner. Observa que para recibir datos de tipo real (flotante) se usa el método *nextFloat()*.

```
import java.util.Scanner;

public class Area {

	public static void main(String[] args) {

		Scanner entrada = new Scanner(System.in);

		float pi = 3.1416f;
		float area;
		float radio;

		System.out.println("Ingresa el radio del círculo");
		radio = entrada.nextFloat();

		area = pi*radio*radio;

		System.out.println("Para un radio "+radio+" el área es "+area);


	}
```

El diagrama de flujo del programa anterior se muestra a continuación.



Clase String y otros tipos de datos

Para leer o declarar una palabra o frase, se utiliza la clase String. Un tipo de dato String está compuesto por una cadena de caracteres. Una variable de tipo cadena se declara de la siguiente forma.

```
String   nombre_variable;
```

Para leer una variable de tipo cadena desde el teclado se usa la función readLine() de la siguiente forma.

nombre_variable = entrada.readLine();

Ya se vio cómo declarar variables de tipo entero, real y booleano, y cómo recibir datos del tipo real. A continuación, se muestra cómo recibir un tipo de dato entero.

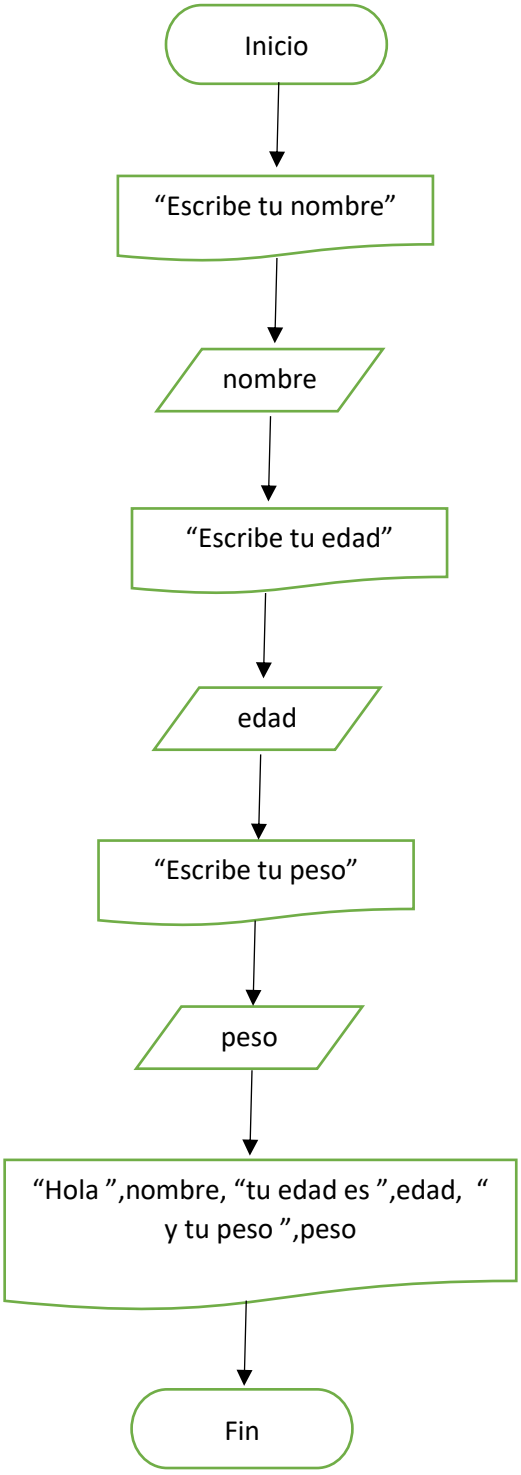
nombre_variable = entrada.nextInt();

Ejemplo 3.5

El siguiente programa pide algunos datos de diferentes tipos desde el teclado y los muestra en pantalla.

```
import java.util.Scanner;

public class Presentacion2 {

	public static void main(String[] args) {

		Scanner entrada = new Scanner(System.in);

		String nombre;

		int edad;

		double peso;

		System.out.println("Escribe tu nombre");
		nombre = entrada.nextLine();

		System.out.println("Escribe tu edad");
		edad = entrada.nextInt();

		System.out.println("Escribe tu peso");
		peso = entrada.nextFloat();

		System.out.println("Hola "+nombre+" tu edad es "+edad+" y tu peso "+peso);
	}

}
```

Su respectivo diagrama de flujo es el siguiente.



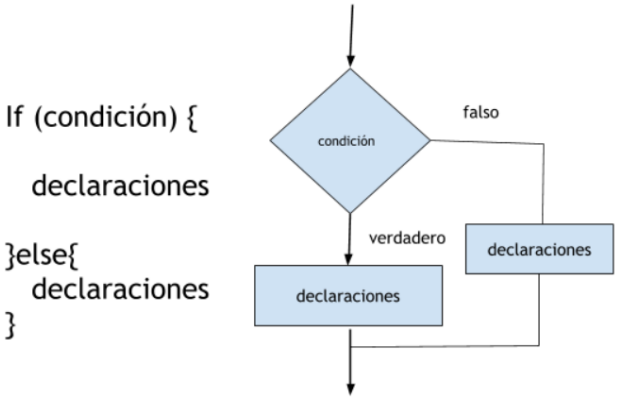
Actividad 3.3

Escribe un programa en Java (con diagrama de flujo) que:

1. Pida el nombre de un empleado, su sueldo base, número de empleado, año de nacimiento, piso en el que trabaja, cuántas horas extras trabajó.
2. A partir del año de nacimiento calculará su edad; al sueldo base se le sumará las horas extras (cada hora extra se paga $200) y se le restará el 10% del sueldo base, esto es el sueldo neto.
3. Muestre el nombre, número de empleado, edad, piso en el que trabaja y sueldo neto.

Sentencia if-else

La sentencia *if* funciona a partir de una condición o pregunta, si la condición (o pregunta) es verdadera, entonces se ejecutará el bloque, en caso contrario la respuesta será falsa y el programa tomará el otro camino. A continuación se muestra su estructura.



Ejemplo 3.6

El siguiente programa pide el nombre, el sueldo y la antigüedad de un empleado; por medio de una estructura if-else, se le sumará 2500 al sueldo si tiene más de 8 de antigüedad, en caso contrario no se le suma nada.

```
import java.util.Scanner;

public class Condicion {

	public static void main(String[] args) {
		// TODO Auto-generated method stub
		Scanner entrada = new Scanner(System.in);

		float sueldo;
		int antiguedad;
		String nombre;

		System.out.println("Dame el nombre del empleado");
		nombre = entrada.nextLine();

		System.out.println("Dame el sueldo del empleado");
		sueldo = entrada.nextFloat();

		System.out.println("Dame la antigüedad del empleado");
		antiguedad = entrada.nextInt();

		if (antiguedad>8) {
			sueldo=sueldo+2500;
				}
		else {
			System.out.println("El empleado no recibió ningún aumento");
		}

		System.out.println("El empleado "+nombre+" con antigüeda de "+antiguedad+"
tiene un sueldo de "+sueldo);


	}

}
```

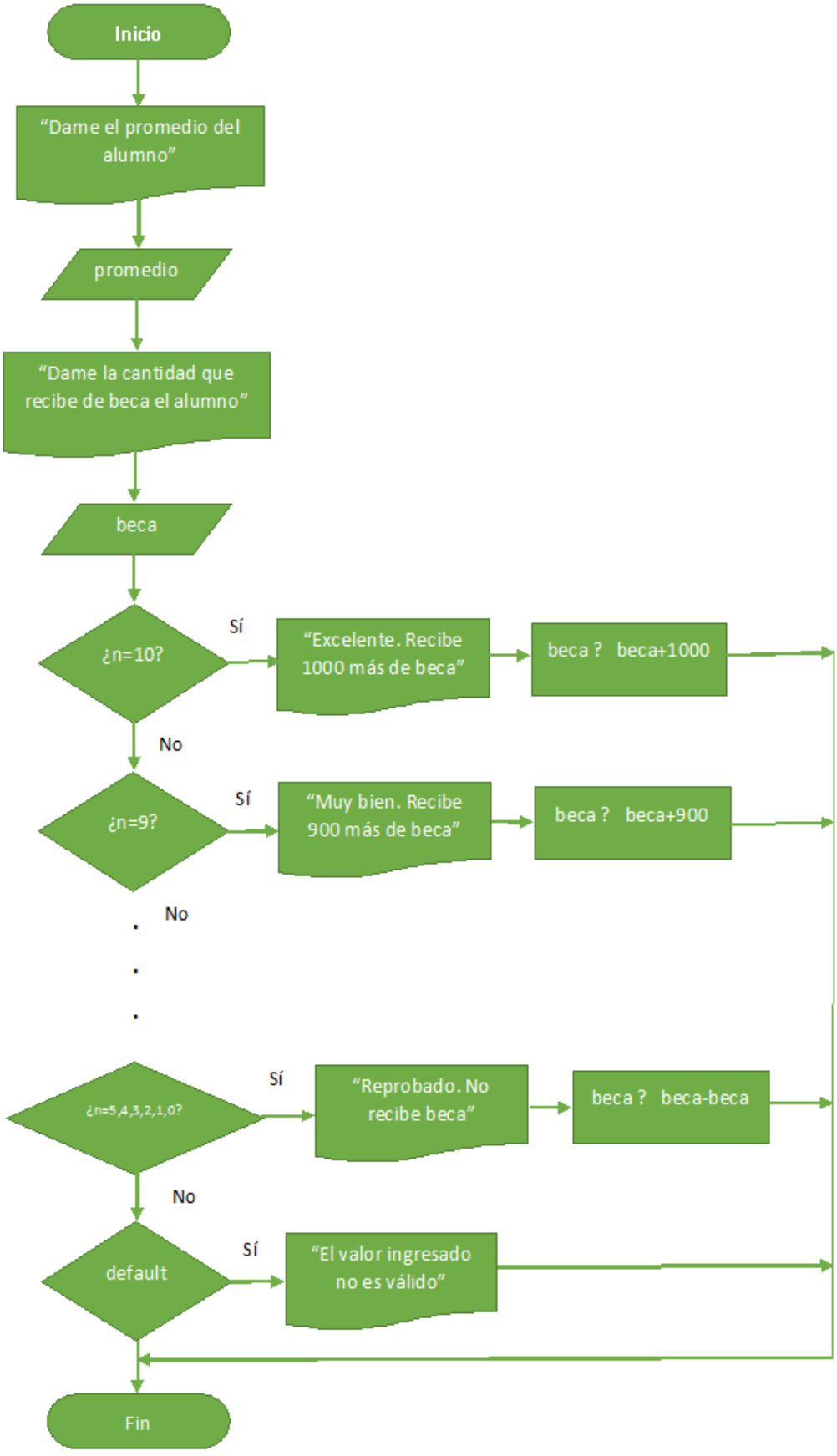
Sentencia SWITCH

La sentencia switch funciona como un menú. A partir del valor que recibe una variable llamada selector, que puede ser solo *int* o *char*, se elige una de múltiples opciones. La siguiente es la forma general para su utilización.

```
switch(selector){
        case valor1:    sentencias;
        break;
        case valor2:    sentencias;
        break;
        .
        .
        .
        case valorn:    sentencias;
        break;
        default: sentencias;
}
```

Ejemplo 3.7

El siguiente ejemplo pide el promedio y la cantidad que recibe de beca un alumno. El programa ejecutará un mensaje y una operación dependiendo del valor ingresado en la variable *promedio*, que se usa como selector. Primero se muestra su diagrama de flujo.


Inicio
"Dame el promedio del alumno"
promedio
"Dame la cantidad que recibe de beca el alumno"
beca
¿n=10?
Sí
"Excelente. Recibe 1000 más de beca"
beca ? beca+1000
No
¿n=9?
Sí
"Muy bien. Recibe 900 más de beca"
beca ? beca+900
No
¿n=5,4,3,2,1,0?
Sí
"Reprobado. No recibe beca"
beca ? beca-beca
No
default
Sí
"El valor ingresado no es válido"
Fin

```
import java.util.Scanner;

public class Menu {

	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);

		int promedio;
		double beca;

		System.out.println("Dame el promedio del alumno");
		promedio = entrada.nextInt();

		System.out.println("Dame la cantidad que recibe de beca el alumno");
		beca = entrada.nextDouble();

		switch (promedio) {

		case 10: System.out.println("Excelente. Recibe 1000 más de beca");
		    beca = beca+1000;
		    break;
		case 9: System.out.println("Muy bien. Recibe 900 más de beca");
		      beca = beca+900;
		      break;
		case 8: System.out.println("Bien. Recibe 800 más de beca");
		             beca = beca+800;
		             break;
		case 7: System.out.println("Regular. Recibe 700 más de beca");
		             beca = beca+700;
		             break;
		case 6: System.out.println("Suficiente. Recibe 600 más de beca");
		             beca = beca+600;
		             break;
		case 5:case 4:case 3:case 2:case 1:case 0:
		             System.out.println("Reprobado. No recibe beca");
		             beca = beca-beca;
		             break;

		default: System.out.println("El valor ingresado no es válido");

		}
		System.out.println("El alumno tiene un promedio de "+promedio+" por lo que
su beca es de "+beca);

	}

}
```

Ciclo For

La sentencia for es un ciclo o bucle que ejecuta ciertas sentencias un número fijo de veces. Este ciclo utiliza una variable como contador para indicar al ciclo dónde inicia, cómo se incrementa y, a partir de una condición, dónde acaba. La sintaxis se muestra a continuación.

```
for (inicio; condición; incremento){
  sentencias;
}
```

Ejemplo 3.8

El siguiente ejemplo muestra los números del 1 al 10 usando un ciclo for.

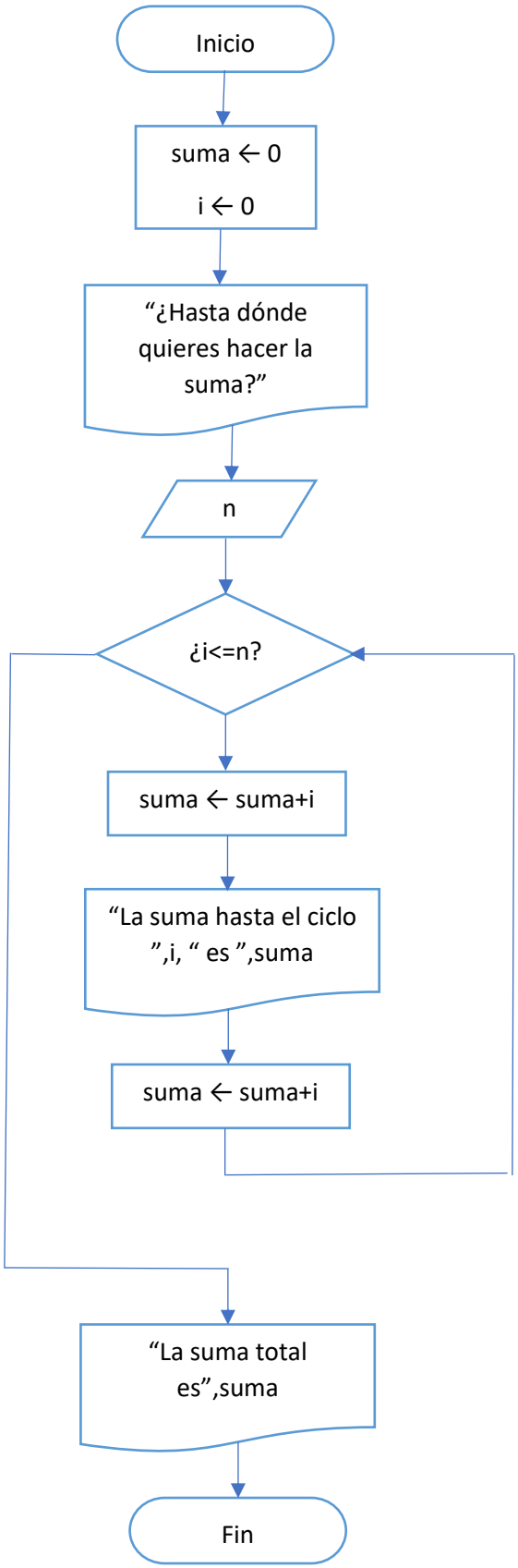
```
public class CicloFor {

      public static void main(String[] args) {
            int i;

            for(i=1;i<11;i=i+1) {
             System.out.println(i);
          }

    }

}
```

Ejemplo 3.9

El siguiente programa calcula la suma de los números naturales hasta n.

```
import java.util.Scanner;
public class Suma {
	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);

		int i;
		int suma=0;
		int n;

		System.out.println("¿Hasta dónde quieres hacer la suma?");
		n = entrada.nextInt();

		for(i=0;i<=n;i=i+1) {
		suma=suma+i;
		System.out.println("La suma hasta el ciclo "+i+" es "+suma);

	}
		System.out.println("La suma total es "+suma);
    }
}
```

Su diagrama de flujo se muestra a continuación.


Inicio
suma ← 0
i ← 0
“¿Hasta dónde quieres hacer la suma?”
n
¿i<=n?
suma ← suma+i
“La suma hasta el ciclo ”,i, “ es ”,suma
suma ← suma+i
“La suma total es”,suma
Fin

Ciclo While

La sentencia while, al igual que la sentencia for, es un ciclo o bucle. La diferencia, además de la sintaxis, es que el ciclo while se ejecuta uno número indeterminado de veces, en realidad podría no ejecutarse; en cambio, el ciclo for desde un principio queda explícito cuántas veces se va a repetir. Su sintaxis es la siguiente.

```
while(condición){


}
```

Ejemplo 3.10

El siguiente programa usa un ciclo while (izq.) y un ciclo for (der.) para sumar los n primeros números enteros. Compara la forma de escribir cada uno de ellos.

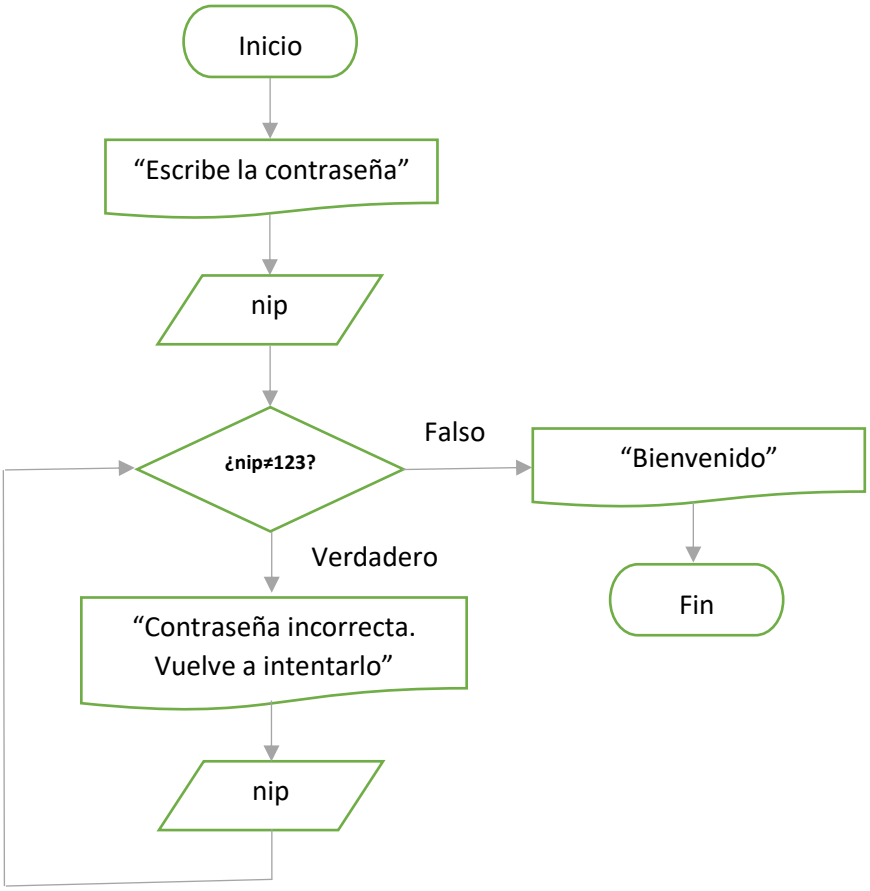
```
import java.util.Scanner;
public class Suma {

        public static void main(String[] args) {
                Scanner entrada = new
Scanner(System.in);
                int i=1;
                int suma=0;
                int n;

                System.out.println("Dime hasta dónde
quieres hacer la suma");

                n = entrada.nextInt();

                while(i<=n) {

                        suma=suma+i;

                        i=i+1;

                }

        System.out.println("La suma total es "+suma);

        }

}
```

```
import java.util.Scanner;
public class Suma {
        public static void main(String[] args) {
                Scanner entrada = new Scanner(System.in);

                int i;
                int suma=0;
                int n;

                System.out.println("¿Hasta dónde quieres
hacer la suma?");

                n = entrada.nextInt();

                for(i=0; i<=n; i=i+1) {

                suma=suma+i;




            }
        System.out.println("La suma total es "+suma);
    }
}
```

Ejemplo 3.11

El siguiente programa pide una contraseña (sólo de número enteros). Mientras la contraseña sea incorrecta, continúa solicitándola. Si se ingresa la contraseña correcta (123) aparece un mensaje de bienvenida.

```
import java.util.Scanner;
public class Contrasena {

	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);

		int nip;
		System.out.println("Escribe la contraseña");
		nip = entrada.nextInt();

		while(nip!=123) {
			System.out.println("Contraseña incorrecta. Vuelve a intenrarlo:");
			nip = entrada.nextInt();
		}

		System.out.println("Bienvenido");
	}

}
```



Ciclo FOR con IF anidado

El siguiente ejemplo muestra cómo usar un ciclo FOR con un IF anidado. El uso compuesto de estas dos estructuras de control es muy útil en la resolución de problemas con computadoras.

Ejemplo 3.12

El siguiente programa registra el nombre y n materias de un alumno. El programa contará cuántas materias aprobó y cuántas reprobó. Al final mostrará todos los datos anteriores y el promedio.

```java
import java.util.Scanner;

public class Promedios {

	public static void main(String[] args) {
		Scanner entrada = new Scanner(System.in);
		//Se declaran variables
		int i,n,apro = 0,rep = 0;
		double cal,promedio=0,acu=0;
		String nombre;

		//Se piden los datos
		System.out.println("Dame el nombre del alumno");
		nombre = entrada.nextLine();
		System.out.println("¿Cuántas materias vas a registrar?");
		n = entrada.nextInt();


		for(i=1;i<=n;i=i+1) {
			System.out.println("Dame la calificación "+i);
			cal = entrada.nextInt();
			acu=acu+cal; //En este acumulador se van sumando las calificaciones

			if(cal<=5) {
				rep=rep+1;//Se suma uno por cada materia reprobada
			}else {
				apro=apro+1; //Se suma uno por cada materia aprobada
			}

			 promedio = acu/n; //Se calcula el promedio


		}

		System.out.println("El alumno "+nombre+" tiene un promedio de "+promedio+" con
"+apro+" aprobadas y "+rep+" reprobadas");

	}

}
```

Significado de las variables

nombre : nombre del alumno

n : número de calificaciones

i : contador

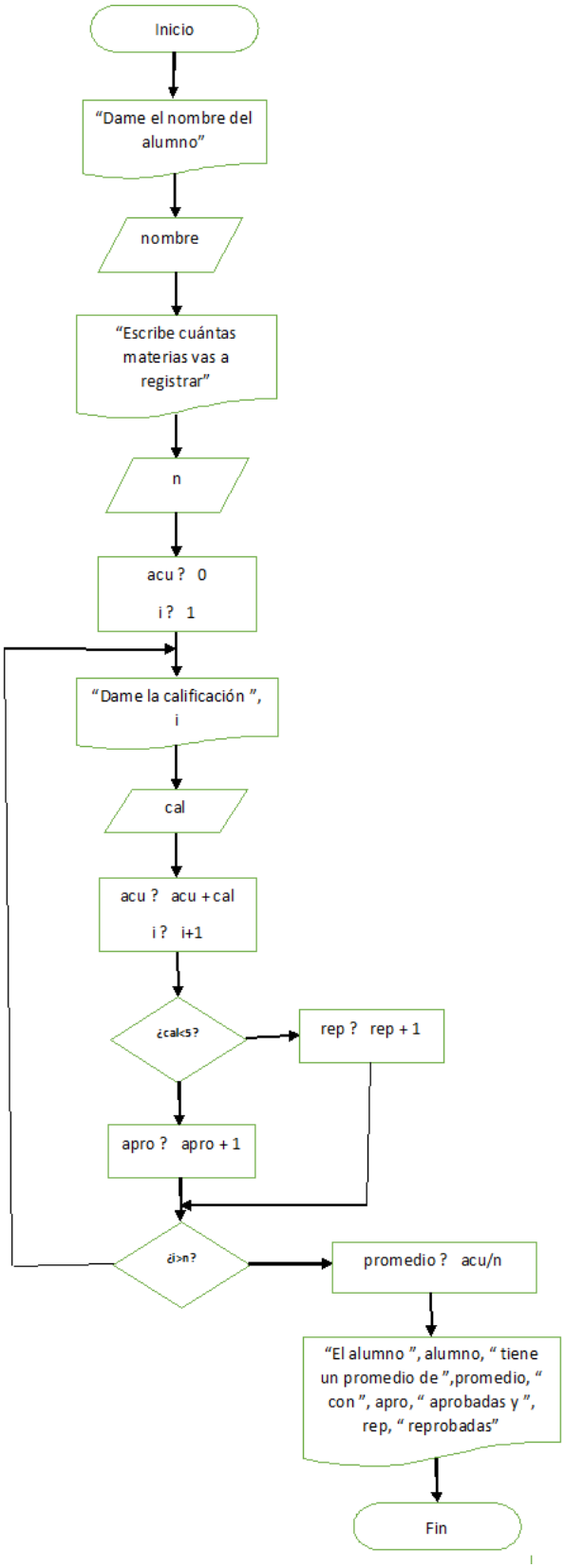
acu : acumulador donde se van sumando las materias

cal : calificación de alguna materia

apro : materias aprobadas

rep : materias reprobadas

promedio : promedio de las materias registradas



Ejercicios

1. Escribe un programa (y su diagrama de flujo) que calcule el punto medio $p_m(x_m, y_m)$ entre los puntos $p_1(x_1, y_1)$ y $p_2(x_2, y_2)$.
2. Escribe un programa que pida los siguientes datos de un alumno:
    a. Nombre
    b. Edad
    c. Genero
    d. Calificación en Cibernética
    e. Calificación en Física
    f. Calificación en Cálculo
    g. Calificación de Química
    h. Calificación de Biología
    i. Cuánto recibe actualmente de beca

    Si el promedio de las cinco materias es mayor a nueve, se le sumará 300 a su beca actual. Al final aparecerá el siguiente mensaje.

    "El alumno (Nombre) de edad (Edad) y género (Género) tiene actualmente una beca de (Beca) por su promedio de (Promedio)."
3. Un aeropuerto pide los siguientes datos de un viajero:
    a. Nombre
    b. Edad
    c. Nacionalidad
    d. Dinero en efectivo en dólares con el que viaja

    Hacer un programa que primero convierta de dólares a pesos. Si el dinero es mayor a 10 mil pesos se le descontará el 10% de impuestos, en caso contrario no hará ningún descuento. Al final aparecerá un mensaje que diga:
    "El viajero (Nombre) de edad (Edad) y nacionalidad (Nacionalidad) lleva consigo la cantidad de (Dinero), descuento incluido."

4. Una tienda de ropa necesita un programa para calcular el descuento en alguna prenda. Se debe pedir el nombre de la prenda y su precio, luego se mostrará el siguiente menú.

1: Descuento del 10%
2: Descuento del 20%
3: Descuento del 30%
4: Descuento del 40%
5: Descuento del 50%

Dependiendo de la opción de la prenda aparecerá un mensaje del tipo "La prenda (nombre) con descuenta del (porcentaje) ahora cuesta (precio con descuento)". Recuerda hacer el diagrama de flujo.

5. Realiza los siguientes programas en Java usando el ciclo for.

5.1 Que calcule el factorial de un número *n*. (No hacer diagrama de flujo.)

5.2 Que pida un número n y calcule las tablas de multiplicar. Por ejemplo, si se ingresa el número 5, deberá aparecer en pantalla lo siguiente. Hacer diagrama de flujo.

5x1=5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50

6. Escribe los siguientes programas en Java usando únicamente el ciclo while.
    a. Que al ingresar un número entero se desplieguen sus tablas de multiplicar.
    b. Que pida un número entero. Mientras sea positivo aparecerá un mensaje de "Número positivo" y seguirá pidiendo números; si se ingresa uno negativo dirá "Número negativo" y dejará de pedir números.
7. Una empresa necesita un programa para calcular el sueldo de sus empleados. Escribe un programa en Java que pida el nombre del empleado, su antigüedad, cuánto se paga por hora y cuántas horas trabajó en el día 1 hasta n. Si el empleado tiene más de 8 años de antigüedad se le incrementará el 10% a su sueldo, en caso contrario no habrá ningún incremento. Es necesario utilizar un ciclo para pedir las horas trabajadas por día.

    El siguiente es un ejemplo de lo que debe aparecer en pantalla si se ingresan los siguientes datos.

Escribe el nombre del empleado

Juan Martinez

Escribe su antigüedad

10

Cuanto se paga por hora

100

Cuantos dias trabajo

4

Cuantas horas trabajo en el día 1

5

Cuantas horas trabajo en el día 2

4

Cuantas horas trabajo en el día 3

5

Cuantas horas trabajo en el día 4

4

Juan Martinez trabajó en 4 días un total de 18 horas, debido a su antiguedad (10 años), su sueldo fue de $1980 pesos

Bibliografía

Wiener, N. (1988). Cibernética y sociedad. Buenos Aires: Editorial Sudamericana.

Coello Coello, Carlos. (Enero-marzo 2016). Norbert Wiener: de la gloria al olvido. Ciencia, Volumen 67 número 1, 12-25.

Morris, M. (2003). Diseño digital. México: Pearson Educación.

Floyd, T. (2006). FUNDAMENTOS DE SISTEMAS DIGITALES. Madrid: Pearson Educación.

Pinales Delgado, F. & Velázquez Amador C. (2014). ALGORITMOS RESUELTOS CON DIAGRAMAS DE FLUJO Y PSEUDOCÓDIGO. México: Universidad Autónoma de Aguascalientes.

Joyanes Aguilar, Luis. (2011). Programación en Java 6. México D.F.: McGraw Hill.

Eckel, B. (2007). Piensa en Java. Madrid: Pearson Educación.

Cairo, Osvaldo. (2006). Metodología de la programación. México: Alfaomega.

Deitel, P. (2012). Cómo programar en Java. México: Pearson Educación.