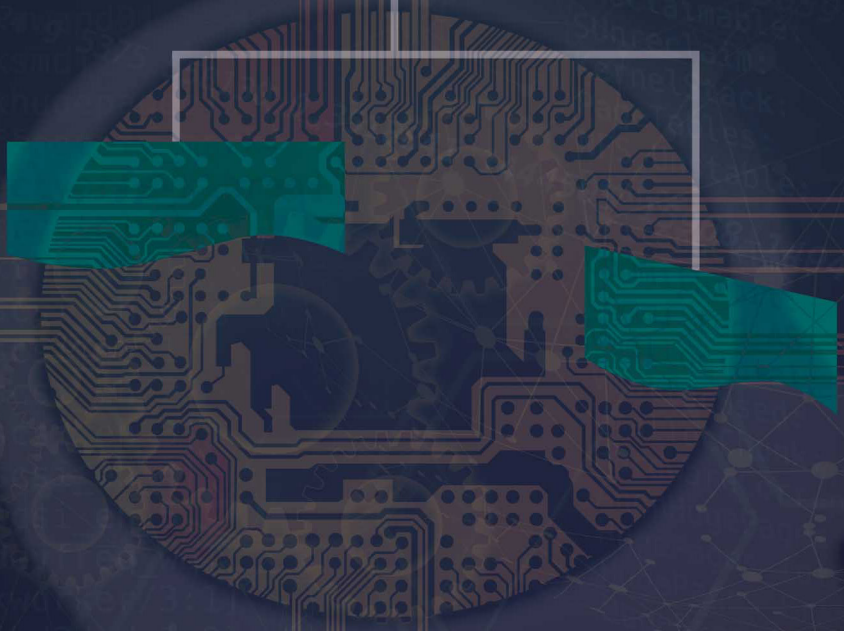
ACADÉMICA

Jairo Hernando Ramírez Marín

Fundamentos iniciales de lógica de programación I.

Algoritmos en PSeInt y Python



INSTITUCIÓN UNIVERSITARIA DE ENVIGADO
Ciencia, educación y desarrollo
Vigilada Mineducación

Fondo Editorial
Institución Universitaria de Envigado

Fundamentos iniciales de lógica de programación I.

Algoritmos en PSeInt y Python

Fundamentos iniciales de lógica de programación I.

Algoritmos en PSeInt y Python

Jairo Hernando Ramírez Marín

Fondo Editorial
Institución Universitaria de Envigado

Ramírez Marín, Jairo Hernando

Fundamentos iniciales de lógica de programación I. Algoritmos en PseInt y Python / Jairo Hernando Ramírez Marín – Envigado: Institución Universitaria de Envigado, 2019.

278 páginas.

ISBN - pdf: 978-958-52600-4-7

Programación lógica – 2. Algoritmos – 3. Ingeniería de software

005.1 (SCDD 22 ed.)

Fundamentos iniciales de lógica de programación I. Algoritmos en PseInt y Python

Edición: diciembre de 2019
ISBN – pdf: 978-958-52600-4-7

Rectora
Blanca Libia Echeverri Londoño

Director de Publicaciones
Jorge Hernando Restrepo Quirós

Coordinadora de Publicaciones
Lina Marcela Patiño Olarte

Asistente Publicaciones
Diana Yaned Marroquín Carmona

Diseño y diagramación
Leonardo Sánchez Perea
Corrección de texto
Erika Tatiana Agudelo

Editado en Envigado – Antioquia
Sello Editorial Fondo Editorial IUE
Institución Universitaria de Envigado
Carrera 27B #39ª Sur 57
Tel: (+4) 339 10 10 ext. 1524
www.iue.edu.co
publicaciones@iue.edu.co

Contenido

Agradecimientos

El primer agradecimiento siempre será para Dios, por inundarme de bendiciones en todos los aspectos de mi vida.

A mi esposa Ledy Yovana Cañaveral Ocampo, por su paciencia y por ser una excelente compañera en este viaje de vida. A mi hijo Miguel Ángel Ramírez Cañaveral, ese ser angelical que es el motor de todo cuanto hago, quien es mi mayor bendición junto con mi esposa, familia y amigos.

A mis padres Jairo Antonio Ramírez Quiros y Orfelina María Marín, por enseñarme valores a través del ejemplo. Gracias a ellos pude realizar mis estudios. Mi padre, quien me enseñó a ser responsable y luchador; mi madre, que me heredó su lógica y creatividad. Toda mi admiración y agradecimiento para ellos. No pude haber tenido mejores padres.

A mis hermanos John Diego y Lina María, que, a pesar de la distancia, siempre estamos unidos como familia para afrontar los problemas y superarlos; y a mis sobrinos John Fernando, Zorelly, Yoselin y Samuel; que trajeron alegría a nuestra familia con sus llegadas y que esperamos se levanten unidos y con lazos de hermandad.

A Oscar Alberto Saavedra Vásquez, Liliana María Loaiza y Samuel Saavedra Loaiza, a Alexander García Berrio, Paula Andrea Cañaveral Ocampo y Samuel García Cañaveral, a Carlos Andrés Cañaveral Ocampo y Marisella Pérez Martínez: unas familias maravillosas que nos han permitido compartir momentos inolvidables.

A mis amigos, especialmente a Felipe Bedoya, Wilmar Ocampo, Guelmer Hincapié, Alexander Ospina, Conrado y Enrique Zapata, quienes forman parte del círculo familiar.

A mis padrinos Fabiola Montoya y Bernardo Marín, quienes acogieron a unos desconocidos y les brindaron apoyo cuando más lo necesitaban. A mi familia,

a mis tías: Omaira, Hortencia, Olga y Olivia y a mis primos: Yesender y Yonier García; Bayron, Johana y Nilton Vasquez; Yuri Restrepo; Henry (q. e. p. d.) y Jeison (q. e. p. d.) Marín, que son los únicos familiares con los que hemos contado.

A Beatriz Hernández Obando y Sergio Alberto Estrada, quienes me dieron la oportunidad de iniciarme como docente de media técnica en el mejor colegio de todos (Colegio Monseñor Gustavo Calle Giraldo, en Bello). Así mismo, al Mg. Gustavo Adolfo Moreno López, quien me apoyó en el aspecto laboral, me motivó a estudiar la Maestría en software libre y fue un motivador constante para escribir este libro.

Al Decano de la Escuela de Ingeniería de la Institución Universitaria Salazar y Herrera (IUSH): Jorge Alonso Monsalve Jaramillo y a la Coordinadora del programa de Ingeniería Industrial: Yenny Alejandra Aguirre, quienes me motivaron a retomar la publicación de este libro y con la oportunidad brindada por Lina Marcela Patiño del Fondo Editorial IUE de la Institución Universitaria de Envigado, tomé un nuevo aire que me permitió culminar este anhelado sueño.

A Rafael Quintana, Carlos Monsalve y Alberto Sánchez mis profesores de Lógica, Algoritmos y Programación cuando realicé mi Tecnología en Sistemas. Sin olvidar a los profesores que me aportaron para culminar mis estudios de Ingeniería en Sistemas. A Roberto Flórez Rueda, profesor de la Universidad de Antioquia y experto en algoritmos, y a Hernán Rendón Valencia, Secretario General de la IUSH, quienes hicieron la revisión en el aspecto temático y de estilo, respectivamente.

A todos los estudiantes que he tenido en las diferentes instituciones: Colegio Monseñor Gustavo Calle Giraldo, Politécnico Marco Fidel Suárez, Corporación Tecnológica Católica de Occidente y la Institución Universitaria Salazar y Herrera, especialmente a Luisa Fernanda Molina Betancur, Juan Guillermo Martínez Álvarez, Joseph Mateus Raigoza Mesa, quienes me motivaron a terminar este libro y me ayudaron a revisar aspectos del mismo y realizar el testing de los ejercicios.

Presentación

En la mayoría de los hogares, empresas e instituciones educativas hay computadoras para diferentes usos, lo que propició que su manejo pasara de ser una necesidad a ser una obligación. La mejor forma de actualizarse y ofrecer posibles soluciones a los múltiples problemas que se presentan es utilizando óptimamente estas herramientas. Pero luego de operarlas, se espera subir a un nivel más avanzado que es el mismo desarrollo del *software*; pasamos del nivel de ser operarios de los programas a ser los creadores de estos, adecuándolos a las necesidades y exigencias de nuestro entorno sin importar el nivel de complejidad y su área de desempeño.

Con este libro se busca brindar la posibilidad de empezar a explorar todo este fantástico mundo de la programación, ampliando la posibilidad laboral y personal del lector: estudiante, docente de sistemas y de otras áreas o cualquier persona interesada en el tema; puesto que al adquirir los conceptos fundamentales de lógica de programación y algoritmos se pueden pasar fácilmente de algoritmos, con una sintaxis similar al software PSeInt, a un lenguaje de programación que le permita desarrollar aplicaciones para celulares y páginas web, configurar periféricos e incluso realizar videojuegos con mayor facilidad.

La importancia de este libro radica en que está enfocado a personas que apenas inician en este mundo de la programación: estudiantes de media técnica que tienen énfasis en sistemas, estudiantes de los primeros semestres de tecnologías e ingenierías de sistemas y programas afines.

Es de resaltar que el 'arte' de programar no es una tarea fácil, pero se puede llegar a entender y manejar, siempre y cuando se tenga disciplina y dedicación, y al final, esta actividad se puede convertir en un juego lógico y entretenido con resultados muy satisfactorios. Debido a lo anterior se ha tomado la iniciativa de compartir el conocimiento adquirido durante el proceso de más de dos décadas de docencia en colegios con énfasis en sistemas, instituciones tecnológicas e instituciones universitarias; y se busca entregar

estrategias que permitan entender el funcionamiento de los programas y la forma más fácil y acertada de desarrollar algoritmos, como primer paso, antes de ir a un lenguaje de programación.

Este libro lleva una secuencia lógica, coherente e incremental, donde se exponen conceptos teóricos, variedad de ejemplos para entender la temática y se termina con una serie de ejercicios prácticos que sirven para evidenciar el manejo del tema. Siempre se busca un avance empezando por el ejercicio más fácil hasta terminar con el más complejo. Se agregan varias soluciones a un mismo ejercicio para facilitarle al estudiante la identificación de la forma personal y propia de fortalecer su lógica, y se le motiva a crear nuevas soluciones. Se proponen algoritmos actuales con entorno reales, lo que facilita más el entendimiento del ejercicio planteado (primer paso del desarrollo de algoritmos).

En *Fundamentos iniciales de lógica de programación I. Algoritmos en PSeInt y Python* se encuentra la solución de ejercicios de algoritmos a través de pseudocódigos (similar al lenguaje de programación) y se habla superficialmente de la técnica de diagramación libre (usada por profesionales de electrónica o industrial). Está distribuido en tres unidades, la solución de algunos ejercicios propuestos y un apartado de anexos, en las que se abordan temas fundamentales de lógica de programación: conceptos y estructuras básicas de un algoritmo, estructuras de decisión y de selección.

Espero que este material sea de su agrado y le sirva para motivarlo a incursionar en el mundo de la programación, donde su éxito depende del manejo óptimo de la lógica de programación aplicada en el diseño de algoritmos y la programación.

Unidad 1
Conceptos básicos de lógica de programación

Introducción

En esta unidad se explican los siguientes temas (véase figura 1):

- Operadores usados en lógica.
- Jerarquía de los operadores.
- Expresiones algorítmicas.
- Lógica de programación y algoritmos.
- Conceptos básicos de algoritmos.
- Representación de los algoritmos.
- Datos usados en los algoritmos.
- Ejercicios resueltos y propuestos.

Estos temas facilitan la adquisición de las competencias que se describen a continuación:

- Conocer conceptos fundamentales de algoritmos y lógica de programación, como los tipos de algoritmos y sus características primordiales.
- Entender las diferentes formas de representar un algoritmo e identificar la mejor al momento de dar solución a un problema.
- Convertir cualquier fórmula y operación matemática en una expresión algorítmica válida y entendible para la computadora.
- Desarrollar los pasos necesarios para la realización de un algoritmo en la búsqueda de soluciones a problemas planteados.
- Reconocer el concepto de dato y su interacción con los diversos procesos de entrada y de salida.

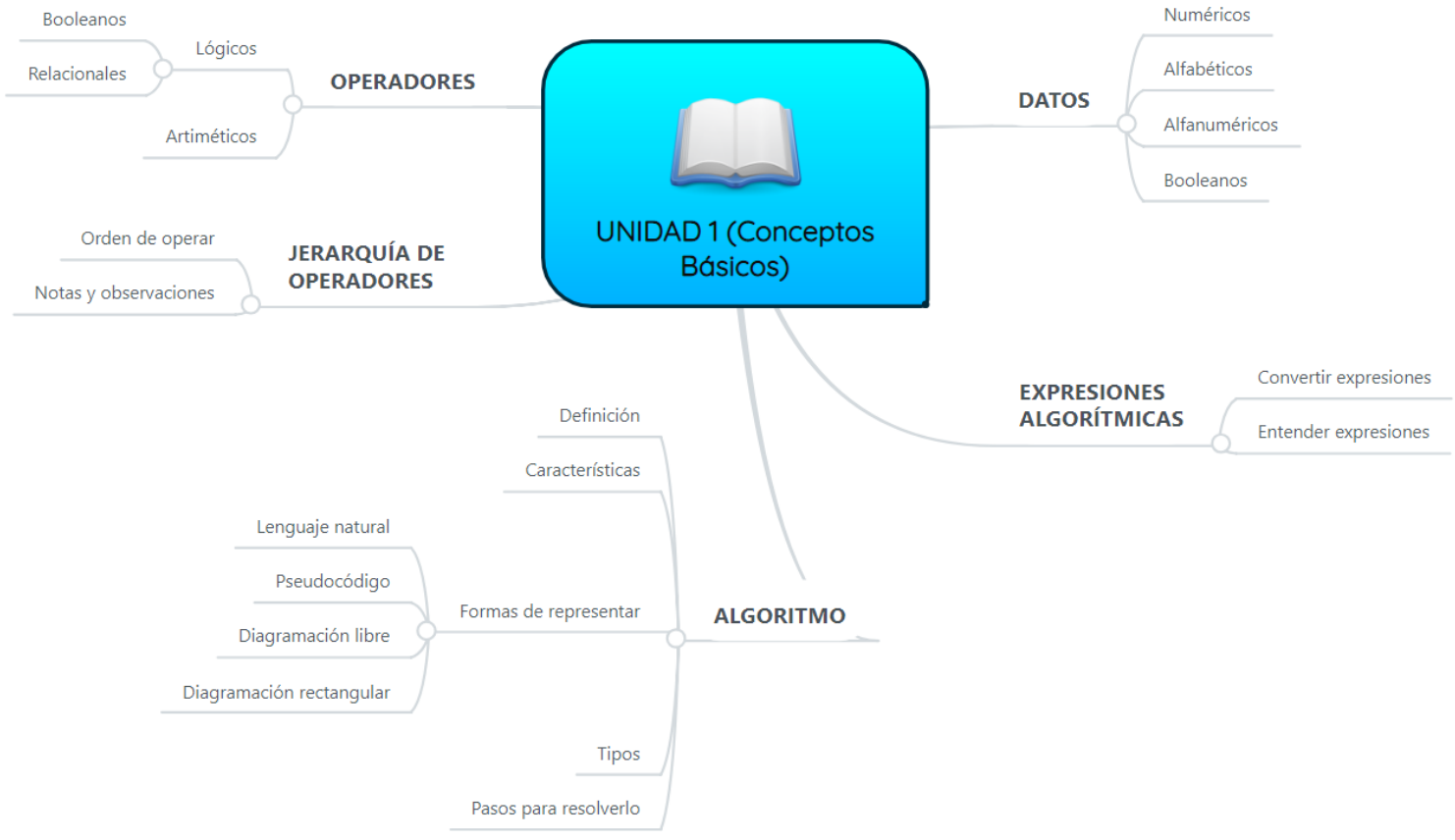


Figura 1. Unidad 1. Conceptos básicos.

1. Conceptos básicos de lógica de programación

1.1 Operadores usados en lógica

Corona y Ancona (2011) enuncian los operadores usados en lógica como símbolos que relacionan valores, variables o constantes dando un resultado determinado según el tipo de dato procesado.

Toda expresión manipula unos datos que arrojan resultados dependiendo de su tipo; por ejemplo, al realizar la operación 5 / 2, podría arrojar como resultado 2.5 si se realiza en Python o 2 si es en otro lenguaje como C++ o Java con dos número enteros; así mismo, en la comparación de 2 < 5 arroja un resultado *Verdadero* (resultado booleano a pesar de hacer operación con dos números enteros).[1]

1.1.1 Operadores aritméticos

Los operadores aritméticos permiten la realización de operaciones matemáticas con variables y constantes. Estos operadores pueden ser utilizados con tipos de datos enteros, reales o cadenas, dependiendo del lenguaje de programación se pueden concatenar con el operador +. A continuación, en la tabla 1, se muestran los operadores aritméticos usados en los lenguajes de programación de computadoras:

Tabla 1.
Operadores aritméticos

Operación	Símbolo	Nombre	Alternativo
Suma o adición	+	Más	
Resta o diferencia	–	Menos	
Multiplicación o producto	*	Asterisco	
División o cociente decimal	/	*Slash*	
División o cociente entero	\	*Backslash*	Div
Operación módulo o de residuo	Mod	Módulo	%
Exponenciación o potencia	^	Circunflejo	**

[1] En este libro se utilizará el punto (.) como separador de los números decimales puesto que en los lenguajes de programación si se utilizan números separados por coma (,) presentarán error.

La división entera solo es usada en algunos lenguajes de programación. Por tal motivo, en este libro que es enfocado a lógica de programación, no se usará con mucha frecuencia. En algunos lenguajes se usan funciones de truncamiento, de redondeo o se soluciona este problema en la declaración de variables de tipo entera, la cual omite los decimales.

1.1.2 Operadores lógicos relacionales y booleanos

Los operadores lógicos permiten hacer comparaciones entre variables y/o valores. Estos operadores se dividen en: operadores lógicos relacionales y operadores lógicos booleanos.

Se utilizan para establecer una relación entre dos valores por medio de una comparación y produciendo un resultado de *Verdadero* o *Falso*. Los operadores relacionales comparan valores del mismo tipo, tienen el mismo nivel de prioridad al evaluarse y tienen menor prioridad que los operadores aritméticos (véase tabla 2).

Tabla 2.
Operadores relacionales.

Comparación	**Operador**	**Ejemplo**	Python
Mayor que	>	50 > 10	>
Menor que	<	5 < 15	<
Mayor o igual que	> =	10 >= 10	>=
Menor o igual que	<=	–5 <= 5	<=
Igual	=	25 = 25	==
Diferente	< >	8 <> 4	!=

Los símbolos de los operadores relacionales: mayor igual que (≥), menor o igual que (≤) y diferente (≠), son usados en lógica matemática, pero no en lógica de programación. En el lenguaje de programación Python se usa el operador != (no igual) que es equivalente al operador <> (diferente).

Los operadores lógicos booleanos se utilizan para establecer relaciones entre valores lógicos, y estos valores pueden ser resultado de una expresión relacional. Al comparar valores de *Falso* o *Verdadero* usando los operadores lógicos booleanos de la tabla 3, se produce un resultado del mismo tipo: *Verdadero* o *Falso*.

Tabla 3.
Operadores booleanos.

Operación	Símbolo (Nombre)	Python
Conjunción	*Λ (Y)*	And
Disyunción	*V (O)*	Or
Negación	*! (NO)*	Not

Como lo dice (EcuRed, 2018), al momento de hacer comparaciones se debe tener en cuenta la *tabla de verdad* o *tabla de valores de verdad* vista en matemática o lógica matemática. La tabla 3 muestra el valor de verdad de una proposición compuesta para cada combinación de valores que se puedan asignar a sus componentes. Las tablas de verdad fueron desarrolladas por Charles Sanders Peirce en los años de 1880, pero el formato más popular es el que introdujo Ludwig Wittgenstein en 1921. La tabla de los valores de verdad es usada en el ámbito de la lógica, para obtener *Verdadero (V)* o *Falso (F)* de una expresión o de una proposición.

Estas tablas pueden construirse haciendo una interpretación de los operadores lógicos booleanos. En consecuencia, las tablas de verdad constituyen un método de decisión para chequear si una proposición es o no un teorema. Normalmente, empieza con la letra *p* con valores *Verdadero, Verdadero, Falso, Falso* y una variable *q* con valores *Verdadero, Falso, Verdadero y Falso*, como se muestra en la tabla 4.

Tabla 4.
Tabla de verdad.

P	*Q*
Verdadero	*Verdadero*
Verdadero	*Falso*
Falso	*Verdadero*
Falso	Falso

Al construir tablas de verdad se pueden encontrar alguna negación, la cual es el valor contrario de la proposición, esta se representa con el símbolo ~. Por ejemplo, si *p = Verdadero*, entonces *~p = Falso* y si *p = Falso*, entonces *~p = Verdadero*. La tabla de verdad de la negación de *p* y *q* se muestra en la tabla 5.

Tabla 5.
Tabla de verdad de la negación.

P	***Q***	*~p*	*~q*
Verdadero	*Verdadero*	*Falso*	*Falso*
Verdadero	*Falso*	*Falso*	*Verdadero*
Falso	*Verdadero*	*Verdadero*	*Falso*
Falso	*Falso*	*Verdadero*	Verdadero

Por su parte, la conjunción se representa con el símbolo Λ. Se tienen que cumplir las dos condiciones en *Verdadero* para que el resultado sea *Verdadero*, en los demás casos el resultado es *Falso*. Por ejemplo, si ponemos la siguiente condición: compro ese libro si tengo dinero (*p*) y si la tienda está abierta (*q*). Se pueden dar estos casos:

- Si tengo dinero (*p* = *Verdadero*) y la tienda está abierta (*q* = *Verdadero*) → Me compro el libro (*Verdadero*). Se tienen que cumplir las dos condiciones para poder comprar el libro.
- Si tengo dinero (*p* = *Verdadero*) y la tienda no está abierta (*~q* = *Falso*) → No compro el libro (*Falso*). Al no estar abierta la tienda no puedo comprar el libro.
- Si no tengo dinero (*~p* = *Falso*) y la tienda está abierta (*q* = *Verdadero*) → No compro el libro (*Falso*). Al no tener dinero, tampoco puedo comprar el libro.
- Si no tengo dinero (*~p* = *Falso*) y la tienda no está abierta (*~q* = *Falso*) → No compro el libro (*Falso*). Al no tener dinero y la tienda no estar abierta no puedo comprar el libro.

En resumen, *p* = *Verdadero* y *q* = *Verdadero*, entonces *p* **Λ** *q* = *Verdadero,* en los demás casos es *Falso* (véase tabla 6).

Tabla 6.
Tabla de verdad de la conjunción.

P	***Q***	*p* Λ *q*
Verdadero	*Verdadero*	*Verdadero*
Verdadero	*Falso*	*Falso*
Falso	*Verdadero*	*Falso*
Falso	*Falso*	*Falso*

La disyunción se representa con el símbolo *V*. Da como resultado *Falso* cuando las dos condiciones son falsas, en los demás casos es *Verdadero*. Por ejemplo, si ponemos la siguiente condición, me compro un vehículo si me aprueban un préstamo en el banco (*p*) o si me gano la lotería (*q*). Se pueden dar estos casos:

- Si me aprueban un préstamo en el banco (*p = Verdadero*) o me gano la lotería (*q = Verdadero*) → Me compro el vehículo (*Verdadero*).
- Si me aprueban un préstamo en el banco (*p = Verdadero*) o no me gano la lotería (*~q = Falso*) → Me compro el vehículo (*Verdadero*).
- Si no me aprueban un préstamo en el banco (*~p = Falso*) o me gano la lotería (*q = Verdadero*) → Me compro el vehículo (*Verdadero*).
- Si no me aprueban un préstamo en el banco (*~p = Falso*) o no me gano la lotería (*~q = Falso*) → No puedo comprar el vehículo (*Falso*).

En resumen, *p = Falso* y *q = Falso*, entonces *p V q = Falso,* en los demás casos es *Verdadero* (véase tabla 7).

Tabla 7.
Tabla de verdad de la disyunción.

P	***q***	*p* V *q*
Verdadero	*Verdadero*	*Verdadero*
Verdadero	*Falso*	*Verdadero*
Falso	*Verdadero*	*Verdadero*
Falso	*Falso*	*Falso*

1.2 Tipos de división en lógica de programación

Antes de empezar a hacer operaciones con jerarquía de los operadores es fundamental entender los tres tipos de división que existen en lógica y en lenguajes de programación que, a pesar de su similitud, tienen algunas diferencias marcadas.

1.2.1 División decimal

Es la división que se realiza por defecto, es la más usada. Se representa con signo / (*slash*). Es aquella que da resultados con decimales, en caso de tenerlos. Una calculadora siempre arroja los resultados de esta forma. En este libro se usan dos decimales para dar resultados de forma más precisa y así unificar respuestas en la solución de los ejercicios (tener en cuenta conceptos de redondeo para estas respuestas).

Por ejemplo, 51 / 2 = 25.5 no tiene decimales para hacer aproximación y 100 / 3 = 33.3333333, quedaría 33.33 (aproximando o truncando a dos decimales). El resultado se muestra completo con sus decimales. En caso de ser una división cuyo resultado no tenga decimales, este se puede presentar de dos formas, dependiendo del formato de salida: 20 / 5 se puede visualizar como 4 o 4.0 y ambos resultados son válidos.

1.2.2 División entera

Se representa con signo \ (*backslash*). Esta división no tiene en cuenta los decimales del resultado, no importa su valor decimal. Aquí no se puede aproximar o redondear, sino que se aplica el concepto de truncamiento al resultado.

Según Ruiz, Llorente, González, Aparicio y Arribas (2019, p.19) "para hacer truncamientos de orden *n* se eliminan todas las cifras a partir de ese orden", en nuestros ejemplos se aplica truncamiento de orden 0 quitando los dígitos a la derecha del separador decimal y dejando los dígitos que se encuentra a la izquierda del punto, por ejemplo, al hacer la división entera 59 999 \ 10 000 se obtiene como respuesta 5.9999 y aplicarle el concepto de truncamiento, el resultado sería 5.

Existe truncamiento a un número de dígitos, por ejemplo, al truncar el número 3.14159265 a cuatro dígitos significativos a la derecha, el resultado es 3.1415; pero en este caso, usaremos el truncamiento con cero cantidades de dígitos significativos a la derecha, para lo cual se quita los valores ubicados después del punto (.) sin importar cual sea ese valor. Ahora aplicando el concepto de división entera a 51 \ 2, el resultado es 25.

En la práctica normal, la división entera se aplica declarando variables de este tipo o usando funciones de truncamiento y redondeo. Por ejemplo, si se calcula una operación como *Division* = 8 / 3, el resultado es 2.666666 si la variable *Division* es de tipo real; pero si es de tipo entero el resultado es 2. Por lo tanto, se realiza el truncamiento por la declaración de la variable.

1.2.3 Operación módulo

Es conocida como división de residuo; debido a que usa el residuo de la división de los dos valores enteros para arrojar un resultado. Por ejemplo, 51 mod 2 es igual a 1. Normalmente se usa el operador % o la palabra mod. Para explicar el resultado véase la figura 2.

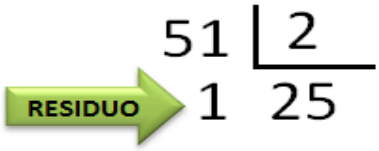


Figura 2. Ejemplo de operación módulo.

Para entender mejor los tipos de división, se realizan las siguientes aclaraciones:

- La operación decimal 46 / 7 es igual a 6.57142857, aproximando la respuesta a dos decimales sería 6.57. Este resultado es el que sacamos frecuentemente en las calculadoras, celulares o computadoras.
- La operación entera 46 \ 7 es igual a 6, dado que se aplica el truncamiento al resultado quitando los decimales de la respuesta anterior: 6.57La operación módulo 46 mod 7 es igual a 4; el cual corresponde al residuo luego de ejecutar la división de 46 entre 7 (el 7 en el 46 está 6 veces, al multiplicar 7 por 6 da 42, quedando como residuo 4, que es lo que falta para llegar a 46). Para entender mejor los conceptos, observar los resultados de cada una de las operaciones en la tabla 8.

Tabla 8.
Resultados de los tipos de división.

Operación	Decimal	Entera	Módulo
11 dividido 6	1.83333333 ≈ 1.83	1	5
29 dividido 9	3.22222222 ≈ 3.22	3	2
38 dividido 8	4.75	4	6
45 dividido 2	22.50	22	1
52 dividido 9	5.77777778 ≈ 5.78	5	7
64 dividido 7	9.14285714 ≈ 9.14	9	1
80 dividido 8	10.00	10	0
98 dividido 9	10.8888888889 ≈ 10.89	10	8
100 dividido 3	33.3333333 ≈ 33.33	33	1
148 dividido 84	1.7619047619047 ≈ 1.76	1	64
1583 divido 78	20.29487179487 ≈ 20.29	20	23
4562 divido 145	31.46206896551 ≈ 31.46	31	67
7821 divido 23	340.04347826 ≈ 340.04	340	1

Se recomienda tener en cuenta lo siguiente al momento de resolver los diferentes tipos de división:

- Cuando una división es exacta en la operación módulo, el resultado siempre es igual a cero, porque no queda ningún residuo. Por ejemplo: al realizar la operación 42 mod 7 (el 7 en el 42 está 6 veces, 7 multiplicado por 6) da como resultado 42 y lo que le faltaría para llegar a 42 es 0.
- Cuando la división es exacta, la división decimal y la división entera son iguales. Por ejemplo: 42 / 7 = 6.00 es igual a 42 \ 7 = 6.
- Cuando el denominador es mayor que el numerador, el resultado de la división entera es igual 0 y de la operación módulo es igual al numerador. Por ejemplo: 7 / 42 = 0.1666 y al realizar el truncamiento de los decimales a la respuesta anterior, el resultado es 0. Y al realizar la operación 7 mod 42 es resultado es igual a 7.

Para entender mejor las últimas observaciones, ver los resultados de cada una de las divisiones en la tabla 9.

Tabla 9.
Resultados de los casos especiales en los tipos de división.

Operación	Decimal	Entera	Módulo
8 dividido 16	0.50	0	8
15 dividido 5	3.00	3	0
48 dividido 8	6.00	6	0
14 dividido 24	0.58333333 ≈ 0.58	0	14
78 dividido 13	6.00	6	0
97 dividido 101	0.96039604 ≈ 0.96	0	97
645 dividido 4521	0.142667551	0	645
4335 dividido 255	17.00	17	0

1.3 Jerarquía de los operadores

Una computadora, para dar el resultado de una operación matemática, realiza cálculos teniendo siempre en cuenta el orden de los operadores que la conforman. Este orden es el que se conoce como jerarquía de los operadores, la cual efectúa las operaciones de forma estricta según los niveles mostrados en la tabla 10.

Tabla 10.
Jerarquía de los operadores de cuatro niveles.

Nivel	Operación por desarrollar	Símbolo
1	Potenciación o exponenciación	^
2	Multiplicación y división decimal	* /
3	División entera y operación módulo	\ mod
4	Suma y resta	+ –

Al resolver cualquier expresión matemática, se desarrollan las operaciones que están entre paréntesis, y en el interior se van aplicando cada uno de los niveles que se muestran. Es importante tener en cuenta que un paréntesis desaparece cuando no tenga operaciones para realizar en su interior. Luego, se repasan los niveles de arriba hacia abajo hasta reducir a un solo término.

1.3.1 Ejemplos de jerarquía de los operadores

Para realizar cualquier operación matemática es necesario utilizar la jerarquía de los operadores antes mencionada, debido a que esta indica el orden a seguir en el desarrollo de cada una de las operaciones, dependiendo de los operadores aritméticos que la componen.

Ejemplo 1. La expresión 2 + 5 * 3 podrá generar duda al momento de evaluar cuál operación se ejecuta primero: ¿multiplicación o suma?, si se suma primero 2 + 5 da como resultado 7 y al multiplicarlo por 3, da como respuesta 21; pero si primero se multiplica 5 * 3 da como resultado 15 y luego de sumarle 2, da como respuesta 17. Entonces, ¿cuál es la respuesta correcta: 21 o 17?

Para no tener dudas al momento de resolver una operación matemática, se deben seguir las instrucciones de la jerarquía de los operadores, donde se observa que primero se ejecuta la multiplicación y luego la suma, por lo tanto, la respuesta correcta será 17, quedando así:

2 + 5 * 3

2 + 15

17

Ejemplo 2. Para dar el resultado a la expresión *40 / 5 + 8 ^ 2 * 3*, se debe hacer lo siguiente:

- *40 / 5 + 8 ^ 2 * 3* → Primero se resuelve la potenciación, que es el operador de mayor jerarquía (debido a que no tiene paréntesis), dando como resultado: *40 / 5 + 64 * 3*.
- *40 / 5 + 64 * 3* → En este momento se pasa a dar solución a la división, que está al mismo nivel que la multiplicación, puesto que operaciones del mismo nivel se desarrollan de izquierda a derecha, dando como resultado: *8 + 64 * 3*.
- *8 + 64 * 3* → Luego se resuelve la multiplicación, que está por encima de la suma en la jerarquía, dando como resultado: 8 + *192*.
- *8 + 192* → Por último, se realiza la suma dando como resultado *200*.

1.3.2 Ejercicios resueltos

a. 9 + 2 * 12 / 2 ^ 2 + (5 ^ 3 / (10 + 2.5))
   9 + 2 * 12 / 2 ^ 2 + (5 ^ 3 / 12.5)
   9 + 2 * 12 / 2 ^ 2 + (125 / 12.5)
   9 + 2 * 12 / 2 ^ 2 + 10
   9 + 2 * 12 / 4 + 10
   9 + 24 / 4 + 10
   9 + 6 + 10
   15 + 10
   25

Al encontrarse con dos paréntesis se empieza a resolver por el más interno, y solo desaparece cuando queda un solo término en medio (en la segunda línea desaparece un paréntesis, debido a que al sumar *10 + 2.5* queda un solo termino que es 12.5; mientras que el segundo no desaparece porque aún faltan operaciones por realizar en su interior: potenciación y división. Este desaparece en la cuarta línea).

b. 20 / 2 / 4 / 5 + ((41 / 2 / 5 + 2.9) mod 3)
   20 / 2 / 4 / 5 + ((20.5 / 5 + 2.9) mod 3)
   20 / 2 / 4 / 5 + ((4.1 + 2.9) mod 3)
   20 / 2 / 4 / 5 + (7 mod 3)
   20 / 2 / 4 / 5 + 1
   10 / 4 / 5 + 1
   2.5 / 5 + 1
   0.5 + 1
   1.5

En este ejemplo se presenta una situación similar relacionada con el manejo de paréntesis. En la línea 4 y en la línea 6 desaparecen los paréntesis.

c. (10 ^ 3 + 5 * 100 / 5) + 54 – (605 mod 2) * 1150
   (1000 + 5 * 100 / 5) + 54 – (605 mod 2) * 1150
   (1000 + 500 / 5) + 54 – (605 mod 2) * 1150

(1000 + 100) + 54 – (605 mod 2) * 1150
1100 + 54 – (605 mod 2) * 1150
1100 + 54 – 1 * 1150
1100 + 54 – 1150
1154 – 1150
4

Cuando un paréntesis desaparece y queda solo un número, al momento de destruirlo, se aplica la ley de signos. En el caso del 605 módulo 2 el resultado es positivo: +1, pero al estar precedido de un signo negativo, la respuesta será con signo negativo: –1.

d. (12 + 3) * 8 – (9 – 4 ^ 2 – 10)
15 * 8 – (9 – 4 ^ 2 – 10)
15 * 8 – (9 – 16 – 10)
15 * 8 – (–7 – 10)
15 * 8 – (–17)
15 * 8 + 17
120 + 17
137

En este ejercicio se aplican las aclaraciones referentes a cuando dos términos tienen signos iguales, estos se suman y conservan el signo: *–7 – 10* es igual a –17; y al desaparecer el paréntesis precedido con signo negativo se cambiaría a positivo: – (–17) es igual a +17.

e. –7 * 10 – 50 mod 3 * 4 – 12 * 8 + (–3) ^ 3 – 5
–7 * 10 – 50 mod 3 * 4 – 12 * 8 + (–27) – 5
–7 * 10 – 50 mod 3 * 4 – 12 * 8 – 27 – 5
–70 – 50 mod 3 * 4 – 12 * 8 –27 – 5
–70 – 50 mod 12 – 12 * 8 – 27 – 5
–70 – 50 mod 12 – 96 – 27 – 5
–70 – 2 – 96 – 27 – 5
–72 – 96 – 27 – 5

–168 – 27 – 5

–195 – 5

–200

En este ejercicio se aplica la aclaración de cuando un número negativo es elevado a una potencia impar, el resultado es negativo. Por ejemplo, *(–3) ^ 3* da como resultado un número con signo negativo –27. Y en la séptima línea, cuando dos términos tienen signos iguales, estos se suman y conservan el signo: –70 – 2 es igual a –72.

Al ejecutar la expresión del literal *e* en Python, el resultado es –206 (véase la figura 3)*;* pero este resultado es diferente al 200 obtenido al resolver el ejemplo anterior.

Figura 3. Ejecución de operaciones en Python.

La explicación a la diferencia entre las dos respuestas radica en que Python tiene una estructura de jerarquía de operadores de tres niveles, donde se combinan en uno solo las multiplicaciones y las divisiones (véase tabla 11).

Tabla 11.
Jerarquía de los operadores de tres niveles.

Nivel	Operación para desarrollar
1	Potenciación o exponenciación.
2	Multiplicación, división (decimal y entera) y operación módulo.
3	Suma y resta.

En la tabla anterior no se agrega la potenciación, porque en algoritmos y en programación no existe esta operación, sino que se usa la potenciación para obtener las raíces.

f. –7 * 10 – 50 mod 3 * 4 – 12 * 8 + (–3) ^ 3 – 5
–7 * 10 – 50 mod 3 * 4 – 12 * 8 + (–27) – 5
–7 * 10 – 50 mod 3 * 4 – 12 * 8 – 27 – 5
–70 – 50 mod 3 * 4 – 12 * 8 –27 – 5
–70 – 2 * 4 – 12 * 8 – 27 – 5
–70 – 8 – 12 * 8 – 27 – 5
–70 – 8 – 96 – 27 – 5
–78 – 96 – 27 – 5
–174 – 27 – 5
–201 – 5
–206

En la cuarta línea pasa algo sumamente significativo; se realiza primero la operación módulo que las multiplicaciones; y esto pasa en computadoras que manejen un nivel de jerarquía de tan solo tres niveles, donde se combinan en uno solo las multiplicaciones y las divisiones (véase tabla 11).

g. (17 / 7 * 2) + (4 * 3 / 5 – 2) – (40 / 3 / 2 * 4 – 5)
(2.43 * 2) + (4 * 3 / 5 – 2) – (40 / 3 / 2 * 4 – 5)
4.86 + (4 * 3 / 5 – 2) – (40 / 3 / 2 * 4 – 5)
4.86 + (12 / 5 – 2) – (40 / 3 / 2 * 4 – 5)
4.86 + (2.4 – 2) – (40 / 3 / 2 * 4 – 5)
4.86 + 0.4 – (40 / 3 / 2 * 4 – 5)
4.86 + 0.4 – (13.33 / 2 * 4 – 5)
4.86 + 0.4 – (6.67 * 4 – 5)
4.86 + 0.4 – (26.68 – 5)
4.86 + 0.4 – 21.68
5.26 – 21.68
–16.42

Al encontrarse con dos paréntesis se resuelven de izquierda a derecha; por eso se empieza por el primer paréntesis (*17 / 7 * 2*) y luego con otro (*4 * 3 / 5 – 2*).

h. 5 ^ 2 ^ 2 + 8 * 2 ^ 3 ^ 2 / 2 ^ 3 mod 23

5 ^ 4 + 8 * 2 ^ 3 ^ 2 / 2 ^ 3 mod 23

625 + 8 * 2 ^ 3 ^ 2 / 2 ^ 3 mod 23

625 + 8 * 2 ^ 9 / 2 ^ 3 mod 23

625 + 8 * 512 / 2 ^ 3 mod 23

625 + 8 * 512 / 8 mod 23

625 + 4096 / 8 mod 23

625 + 512 mod 23

625 + 6

631

En otros lenguajes, las potencias sucesivas, se resuelven de derecha a izquierda, como es el caso del lenguaje de programación Python. Recuerde que en la parte de operadores se explicó que la potenciación se puede representar con el doble asterisco (**). El resultado sería como se puede observar en la figura 4.

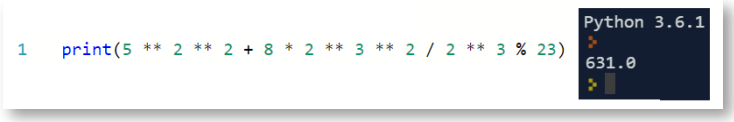


Figura 4. Ejecución de potencias sucesivas en Python.

1.3.3 Ejercicios propuestos

Dar el resultado de las siguientes expresiones.

a. 2 + 3 * 4

b. 10 ^ 2 ^ 2 / 7

c. 42 / 2 / 3 / 7 ^ 3

d. 8 + 7 * 3 + 4 * 6

e. 5 ^ 3 + 120 – 3 ^ 5

f. 12 + 3 * 7 + 5 * 4

g. 4 + 5 ^ 2 – 500 + 20 ^ 2 * 10

h. (4 ^ 3 + 2 * 40 – 140) ^ 2

i. 5 + (25 * 2 + 5 * 8 / 2 – 10) * 2
j. 2 * 25 / 5 + 5 ^ 2 – 5
k. 15 – ((4 *5 + (8 / 2 –5) * 2) + 8 * 10 / 2)
l. 2 – 5 * 12 / 2 + 8
m. (8 ^ 2 + 3 * 10 + 5) – (6 ^ 2 – 2 * 5 – 6)
n. (33 + 5 * 3 ^ 5 + 11 * 3 + 14) / (3 + 2)
o. (2 * 5 * 3 * 4 ^ 2) / (48 / 2 ^ 2 – 20 + 2 ^ 2) / 10
p. 4 ^ 2 + 3 ^ 2 * 2 ^ 2 + 5 ^ 3 – 4 ^ ((7 * 3 mod 9) mod 25)
q. (4 – (9 * 8 * 5 mod 3) * 3) / (100 mod 8 mod 3)
r. 208 / 4 / (455 * 2 / 5 mod 4 + 50)
s. (17 / 7 * 2) + (4 * 3 / 5 – 2) – (40 / 3 / 2 – 5)
t. 4 ^ 2 * 5 * 3 – 8 – 152 – (5 ^ 3 * 10 \ 4) mod 2 * 4
u. 8 * (4 – 3 ^ 2) ^ 2 – 9 * (3 ^ 4 – 4 ^ 3) ^ 3
v. (4 + 2 * 9 / 5) – (2 ^ 2 * 8) / 2 ^ 2
w. 8 ^ 2 / 4 ^ 3 mod 10 ^ 2 / 5 + 100
x. (80 / 8) ^ 2 – 4 / 2 + (20 / 4) ^ 3
y. 89 + (66 ^ 2 mod 2011 + 400) ^ (2 mod 8) / 3 ^ (3 + 2)
z. 14 – (7 + 4 * 3 – (–2) ^ 2 * 2 – 6) + (2 ^ 2 + 6 – 5 * 3)

1.4 Expresiones algorítmicas

Las expresiones algorítmicas son expresiones manejadas dentro de lógica de programación que usan una sintaxis que permite ser procesada por la computadora. Estas expresiones son diferentes a las expresiones matemáticas.

Para crear expresiones algorítmicas se requiere la aplicación de los operadores aritméticos, haciendo reemplazo de los operadores matemáticos por operadores aritméticos. Una expresión matemática como $a^2 + bx + c$, se convierte a expresión algorítmica, quedando de esta forma $a \wedge 2 + b * x + c$.

1.4.1 Ejemplos de conversión de expresiones algorítmicas

Ejemplo 1. En la expresión matemática: $a^2 + 2ab + b^2$, se puede observar que, tanto la letra *a* como la letra *b*, están elevadas al cuadrado, por tal motivo se debe utilizar el operador circunflejo (^). De igual forma, los términos 2, *a* y *b*, se están multiplicando entre sí, por lo que deben ser separados por el operador de la multiplicación (*). Por último, la suma queda tal cual está ubicada en la expresión. El resultado al realizar la conversión sería: *a ^ 2 + 2 * a * b + b ^ 2.*

Ejemplo 2. La expresión matemática *X + 2Y*, para ser convertida a expresión algorítmica, simplemente se agregar el asterisco (*) al termino *2Y*, considerando que el número 2 se está multiplicando con la letra *Y* dando como resultado *X + 2 * Y.*

Ejemplo 3. La expresión $5a^3 + 2b^4 – 4c$ tiene multiplicación, potencia, suma y resta. Dando como resultado: *5 * a ^ 3 + 2 * b ^ 4 – 4 * c.*

Pero no todas las expresiones son tan fáciles de convertir; aquí se darán cuatro ejemplos donde se empezará a generar un poco más de complejidad, donde varias expresiones matemáticas diferentes, generan la misma expresión algorítmica.

a. $\frac{x+y}{m} n^3$ → x + y / m * n ^ 3

b. $\frac{x+y}{m\, n^3}$ → x + y / m * n ^ 3

c. $x + \frac{y}{m\, n^3}$ → x + y / m * n ^ 3

d. $x + \frac{y}{m}\, n^3$ → x + y / m * n ^ 3

Para poder solucionar la dificultad anterior, se deben usar paréntesis para empezar a romper la jerarquía y que se hagan unas operaciones antes que otras.

Los resultados finales son los siguientes:

a. $\frac{x+y}{m} n^3$ → (x + y) / m * n ^ 3

b. $\frac{x+y}{m\, n^3}$ → (x + y) / (m * n ^ 3)

c. $x + \frac{y}{m\, n^3}$ → x + y / (m * n ^ 3)

d. $x + \frac{y}{m} n^3$ → x + y / m * n ^ 3

Observe que cada presencia o ausencia de paréntesis diferencia cada expresión algorítmica, de ahí lo fundamental de identificar el lugar donde se van a agregar.

1.4.2 Convertir expresiones algorítmicas en matemáticas

Los ejercicios también se pueden convertir en sentido contrario, de una expresión algorítmica sacar una expresión matemática, como se muestra en estos ejemplos:

a. x * y * z → xyz

b. a * (x – y) ^ 3 → $a\,(x - y)^3$

c. 40 / 5 + 8 ^ 2 – 3 → $\frac{40}{5} + 8^2 - 3$

d. x ^ 2 / a ^ 2 + y ^ 2 → $\frac{x^2}{a^2} + y^2$

e. x ^ 2 / (a ^ 2 + y ^ 2) → $\frac{x^2}{a^2 + y^2}$

f. x ^ (a – 2) / (x ^ 2 – 2 + y ^ (2 * n)) → $\frac{x^{a-2}}{x^2 - 2 + y^{2n}}$

1.4.3 Ejercicios resueltos

a. $\frac{x + 2y}{2^3}$ → (x + 2 * y) / 2 ^ 3

b. $x + \frac{2y}{2^3}$ → x + 2 * y / 2 ^ 3

c. $\frac{m^2 + n^2}{4xy^3}$ → (m ^ 2 + n ^ 2) / (4 * x * y ^ 3)

d. $\frac{(m - n)^2}{(4x)(3y^3)}$ → (m – n) ^ 2 / (4 * x * (3 * y ^ 3))

→ (m – n) ^ 2 / (4 * x * 3 * y ^ 3) // Se puede ignorar el paréntesis sin afectar el resultado.

El doble *slash* (//) se usa en el libro para realizar comentarios, así como dentro de los pseudocódigos.

e. $x + y + \frac{a^2 + b^2}{2}$ → x + y + (a ^ 2 + b ^ 2) / 2

f. $\frac{x + y + a^2 + b^2}{2}$ → (x + y + a ^ 2 + b ^ 2) / 2

g. $x + y + a^2 + \frac{b^2}{2}$ → x + y + a ^ 2 + b ^ 2 / 2

h. $\frac{\frac{m + n}{p}}{\frac{q - r}{s}}$ → (m + n) / p / ((q – r) / s)

i. $\frac{m + \frac{n}{p}}{q - \frac{r}{s}}$ → (m + n / p) / (q – r / s)

j. $\frac{m + \frac{n}{p}}{\frac{q - r}{s}}$ → (m + n / p) / ((q – r) / s)

1.4.4 Ejercicios propuestos

Convertir las siguientes expresiones matemáticas en expresiones algorítmicas:

a. $\frac{2}{3} + \frac{2}{4} + \frac{3}{5}$

b. $y + \frac{x}{z+5}$

c. $\frac{x^2}{a^2} + \frac{y^2}{b^2}$

d. $\frac{\frac{x^2}{a^2} + y^2}{b^2}$

e. $\frac{x^2 + y^2}{a^2 + b^2}$

f. $\frac{a^2 (c^2 - 2m + y^2)}{2^2}$

g. $\frac{\frac{x^2 + 3xy}{a^2 + y^2}}{b^2}$

h. $\frac{a^2 (a^2 - c^2)}{z}$

i. $\left(\frac{a^2+bx+c^3}{x}\right)^2$

j. $\frac{b.h}{2} + \frac{y^2}{2} - x^2 . \frac{8x.4}{2}$

k. $\left(\frac{a^2-2mn^2-n^3}{pqr}\right)^2$

l. $\frac{a}{b^2} + \left(\frac{a}{b}\right)^2 - \left(\frac{m^2n^3}{4x}\right)^3$

Convertir las siguientes expresiones algorítmicas a expresiones matemáticas:

m. 7 * (1 + y)

n. a ^ 3 + b ^ 3

o. (x + y) / (u + w / a)

p. a ^ 3 + 3 * a ^ 2 * b + 3 * a * b ^ 2 + b ^ 3

q. (a + b) ^ 2 – (a – b ^ 2)

r. x – y ^ 2 * (x – y) ^ (2 – n ^ 3)

s. (a / b) + (c / a) + c

t. a / (b + c) / (a / b + c)

1.5 Lógica de programación y algoritmos

La base fundamental al momento de empezar a programar son la lógica y los algritmos. A continuación unas definiciones cortas con unas explicaciones de las caracteristicas, partes, tipos y pasos para resolver un algoritmo.

1.5.1 Definiciones

El término "lógica (del griego *logos*) significa estudio o tratado racional y así, el objeto de estudio de esta disciplina es el razonamiento y éste se expresa mediante el lenguaje" (García, Ordoñez y Ruiz citados en Guzmán y López, 2019, p. 76). Por lo tanto, *lógica* es todo aquello relacionado con la razón o el pensamiento.

El *algoritmo* es el conjunto finito de pasos, operaciones o procedimientos secuenciales que sirven para obtener una solución a un problema determinado a partir de ciertas reglas definidas. El término *algoritmo* viene, según la traducción de Fibonacci de las palabras *Algoritmi dicit*, de la obra *Quitab Al Jabr Al Mugabala* del autor árabe Muhammed ibn Musa al–Khwarizmi (también llamado Al–Jwarizmi o Al–Juarismi), matemático árabe del siglo IX que aparece con un turbante en el famoso libro del *Álgebra de Baldor* (Niño, Cobos y Mendoza, 2010).

La lógica que se aprende en el desarrollo de los temas se aplica en lenguajes de programación, de ahí el concepto de lógica de programación.

1.5.2 Características de un algoritmo

Todo algoritmo debe cumplir con las siguientes características:

- *Ser finito:* todo algoritmo debe tener un inicio y en un momento determinado debe llegar a su fin. Tiene un número finito de instrucciones.
- *Ser definido:* debe tener un desarrollo claro y coherente. No debe permitir dobles interpretaciones.
- *Ser preciso:* un algoritmo solo debe hacer lo solicitado en un orden lógico y estricto. Se debe generar un orden estructurado en el desarrollo de la solución. Se puede presentar errores si se hace más o menos de lo pedido, de igual forma cuando se realizan instrucciones en un orden equivocado.
- *Ser legible*: debe estar bien estructurado y organizado para entenderlo fácilmente.[2]
- *Ser eficiente*: debe hacer lo solicitado con el mínimo de instrucciones posibles.
- *Ser general:* debe soportar todas las variaciones que sean posible para dar solución a problemas con las mismas especificaciones.
- *Ser neutral de material:* puede ser realizado en papel, pizarras, microcontroladores, simuladores o computadoras.

---

[2] Se recomienda tener en cuenta esta y las siguientes características, aunque no son de tanta rigurosidad como las anteriores que sí son casi de estricto cumplimiento.

A pesar de la complejidad que se maneja en los conceptos de algoritmos, estos son muy comunes, se encuentran en todo lugar y se aplican constantemente. El ejemplo más claro está en una receta de cocina, que no es otra cosa que un algoritmo, por poco parecido que tenga con un cálculo matemático o un programa informático, cumple con cada una de las características de un algoritmo. A continuación, una breve explicación, las recetas de cocina son:

- Instrucciones paso a paso que sirven para dar solución a un problema, en este caso si se busca una receta es porque se requiere hacer un postre, ensalada o cualquier plato, y se encontrarán unas instrucciones secuenciales para su realización.
- Finitas porque tienen un proceso que empieza con una serie de ingredientes y unas instrucciones que van a finalizar con la realización de un producto esperado.
- Definidas porque tienen un mismo procedimiento y un resultado correcto, si se aplica correctamente.
- Precisas porque tienen instrucciones claras que se deben realizar de forma ordenada.
- Legibles porque tienen un desarrollo claro y estructurado. Normalmente pueden ser realizadas, sin ningún problema, por cualquier persona.
- Eficientes porque se hace el producto con el mínimo de recursos y en el menor tiempo posible.
- Generales porque permiten variantes de acuerdo con la experiencia de quien la realiza; pero a pesar de estas el resultado siempre va a ser el mismo.
- Son neutrales respecto al material, puesto que pueden estar escritas en un libro de recetas, pueden ser vista en un programa de televisión o simplemente existir en grabaciones.

1.5.3 Partes de un algoritmo

a. *Entrada:* es el lugar donde se relaciona la información que va a ser leída o ingresada. Aquí van todos los datos que se desconocen y se requieren

para realizar cálculos, operaciones o procesos. En el pseudocódigo la entrada empieza con palabras como *ingresar, obtener, iniciar* o *leer* y se describen las variables necesarias para crear una solución, separadas por coma (,). Estas variables deben cumplir con ciertas condiciones (condiciones para nombrar variables o constantes), tema ubicado al finalizar esta primera unidad. También se recomienda acompañar estas entradas de mensajes que permitan identificar con mayor facilidad lo que se requiere, entregando instrucciones claras y precisas al usuario de la forma como debe entregar los datos.

b. *Proceso:* es el lugar donde se realizan las operaciones y cálculos necesarios para alcanzar el resultado esperado. Cada cálculo se recomienda almacenarlo en una variable y se le asigna información por medio del signo igual (=) o a con el símbolo (←). Se recomienda realizar un cálculo en cada renglón, aunque cada cálculo puede incluir varias operaciones matemáticas. En el pseudocódigo el proceso puede calcularse asignando los cálculos o describiendo las acciones a realizar, por ejemplo, calcular, restar o sumar.

c. *Salidala cual dará respuesta a los requerimientos o peticiones que tenía el algoritmo al empezar." :* es el lugar donde se describe la información que se va a imprimir, la cual dará respuesta a los requerimientos o peticiones que tenía el algoritmo al empezar. En el pseudocódigo la salida empieza con las palabras visualizar, imprimir, escribir o mostrar y se escriben las variables a mostrar separadas por coma (,). También se recomienda acompañar las salidas de mensajes que permitan identificar con mayor facilidad los datos que se quieren mostrar.

1.5.4 Tipos de algoritmos

a. *Algoritmo cualitativo:* "son algoritmos usados generalmente para describir procesos de la vida cotidiana que no incluyen cálculos numéricos" (Guerrero, 2016, p.3). También conocido como *algoritmo no computacional*. Ejemplos claros y cotidianos de algoritmos cualitativos pueden ser: el modo de operar una lavadora, tocar música por medio de partituras, construir un aeroplano, búsqueda de un número en el direc-

torio telefónico, el montaje de una llanta pinchada, la búsqueda de una dirección, la adquisición de algún producto o incluso hacer trucos de magia, entre otros.

b. *Algoritmo cuantitativo:* "son algoritmos muy objetivos, pues al tener siempre correspondencia con problemas matemáticos, demanda la inclusión de cálculos numéricos para la obtención de los resultados" (Guerrero, 2016, p.4). También conocido como *algoritmo computacional*. Ejemplos de algoritmos cuantitativos pueden ser: la división para calcular el cociente de dos números, la liquidación de una nómina, la solución a cualquier ecuación o fórmula, el registro de transacciones en un banco o la realización de un determinado estudio estadístico, entre otros.

1.5.5 Pasos para solucionar un problema computacional

a. *Definición y delimitación del problema:* para resolver un problema es fundamental conocerlo por completo, y que esté definido y delimitado. Esta fase está dada por el enunciado del problema, que luego de ser leído cuantas veces sea necesario, se pasará a determinar si está claro o no y si es posible darle una solución. Si no se entiende el enunciado no se podrá continuar con los demás pasos. Se debe tener cuidado con no confundir el enunciado dado que se puede terminar dando solución a otro problema inexistente.

b. *Análisis del problema*: una vez que se ha comprendido completamente qué es lo que se va a desarrollar, es necesario definir los datos de entrada, los cálculos necesarios para procesar los datos, los datos de salida y las restricciones que pueda tener la solución planteada.

c. *Diseño del algoritmo:* en esta fase se determinan los pasos o instrucciones que se llevarán a cabo, el orden lógico de ejecución y la manera como se desarrollará todo el programa para un eficiente funcionamiento. La forma de representarlo puede ser pseudocódigo, diagramación libre o diagramación rectangular.

d. *Codificación:* consiste en pasar el diseño del algoritmo, en cualquiera de las formas de representación, en un lenguaje comprendido por la

computadora llamado lenguaje de programación. Existen varios lenguajes con diferentes características y fabricantes, por ejemplo*:* Lenguaje C, C++, Java*,* Python*,* PHP y todas las herramientas ofrecidas por Microsoft (Visual Basic, Visual Studio C#, Visual Studio C++, J#, ASP .NET*,* entre otros*)*.

e. *Compilación:* en esta etapa la computadora hace una revisión de las instrucciones digitadas por el programador y muestra los errores que se hayan producido al no cumplir con la sintaxis del lenguaje. Algo importante para resaltar es que ningún compilador detecta los errores lógicos en la realización de una solución, solo los de sintaxis.

f. *Depuración:* consiste en identificar y corregir los errores reportados en el proceso de compilación. Este proceso se repite cuantas veces necesario, hasta dejar el programa sin ningún error.

g. *Construcción del programa*: al no tener ningún tipo de error, el compilador construye un programa ejecutable que permitirá verificar el funcionamiento de la solución propuesta. Hay que tener en cuenta que, un solo error evitará la creación de este programa ejecutable.

h. *Ejecución:* luego de haber corregido todos los errores y haber compilado el programa, se pasa a ejecutarlo para ver los resultados obtenidos. En esta etapa se utilizan los dispositivos de entrada y de salida, siendo el teclado y el monitor, respectivamente, los más utilizados. Los resultados tienen que ser los esperados, de lo contrario, se tienen que repetir los pasos anteriores para detectar los errores; normalmente, aquí se detectan los errores de lógica que no detectan los compiladores.

i. *Documentación:* se divide en *documentación interna*, que son aquellos comentarios o mensajes que se añaden al código fuente para hacer más claro su entendimiento y el de algunos procesos; y *documentación externa* la cual es un documento escrito donde se hace descripción paso a paso de la manera cómo se debe usar el programa y se describe todo lo relacionado con este. Las documentaciones externas más comunes son los manuales de usuario y los manuales técnicos.

j. *Mantenimiento:* se lleva a cabo después de terminado el programa cuando se detecta que es necesario hacer algún cambio, ajuste o com-

plementación para que siga trabajando de manera correcta. Para poder realizar este trabajo se requiere que el programa este correctamente documentado. Luego de todo mantenimiento se recomienda hacer ajustes a los manuales realizados en la parte de documentación.

1.5.6 Pasos de elaboración de un algoritmo

a. Leer el enunciado del problema, cuantas veces sea necesario, hasta entenderlo completamente.
b. Determinar claramente los datos de entrada requeridos para poder dar solución al problema.
c. Definir cálculos y comparaciones necesarias para llegar al resultado esperado.
d. Aclarar y determinar los datos de salida, los cuales contienen la información o resultados que se soliciten.
e. Tener en cuenta condiciones y restricciones para la solución del problema.

Los cinco pasos anteriores gozan de la misma importancia. La falta de análisis a alguno de estos causará problemas en el transcurso del desarrollo del algoritmo. No se debe continuar con alguno de los pasos hasta tener claridad de cada uno de los anteriores.

1.6 Formas de representación de un algoritmo

Al momento de plantear una solución a través de un algoritmo es importante saber que existen varias técnicas para representarlos: lenguaje natural, pseudocódigo, diagramación libre y diagramación rectangular. En este libro se hace un enfoque mayor en el pseudocódigo por ser la representación más cercana al lenguaje de programación.

1.6.1 Algoritmo en lenguaje natural

Un *algoritmo en lenguaje natural* es definido por (Gómez y Moraleda, 2015) como una solución basada en el lenguaje que utilizamos para comunicarnos, mediante explicaciones más o menos precisas se puede describir una

solución a un problema usando palabras del lenguaje común que expresan operaciones, cálculos y procesos sin tener ninguna sintaxis de programación. Un algoritmo que permita calcular la ganancia que puede obtener un comerciante luego de realizar la venta de un producto puede representarse como se hace en la figura 5.

- *Entrada:* precio de compra y precio de venta.
- *Proceso:* la ganancia, restando al precio de venta el precio de compra.
- *Salida:* la ganancia obtenida por dicha venta.

Figura 5. Primer ejemplo de un algoritmo en lenguaje natural.

1.6.2 Pseudocódigo

Gómez y Moraleda (2015) describen el *pseudocódigo* como una notación basada en un lenguaje de programación estructurado del que se excluyen todos los aspectos de declaración de constantes, tipos, variables y subprogramas; no tiene sintaxis estricta, como sucede con los lenguajes de programación.

Por otro lado, Juganaru (2014) lo define como un "lenguaje falso" formado por una serie de palabras con un formalismo muy sencillo que describe el funcionamiento de un programa; combina frases de lenguaje común e instrucciones de programación.

Un pseudocódigo ocupa menos espacio en una hoja de papel, permite representar en forma fácil operaciones repetitivas complejas, es más fácil de realizar su transformación a un lenguaje de programación y se pueden observar claramente los niveles de las operaciones. En ocasiones los pseudocódigos se enfocan a determinado lenguaje de programación. Por lo anterior es que la mayoría de los desarrolladores profesionales prefieren utilizarlo antes que el diagrama de flujo que es más enfocado a programadores principiantes o profesionales de otras carreras afines.

En este libro usamos el pseudocódigo como principal forma de representación de un algoritmo, aunque al inicio de cada unidad se brindan soluciones combinadas con los diagramas de flujos.

Se pone como ejemplo el mismo ejercicio de la ganancia obtenida por un comerciante utilizado en el lenguaje natural: un algoritmo que permita calcular las ganancias que puede obtener un comerciante luego de realizar la venta de un producto determinado.

Para hacer un pseudocódigo hay que aplicar conceptos que se trataron en las partes que conforman un algoritmo (véase figura 6).

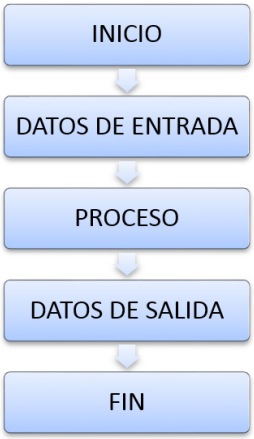


Figura 6. Partes de un algoritmo.

En la figura 7 se puede visualizar un algoritmo en pseudocódigo, tomando el ejemplo de la ganancia obtenida sobre la venta de un producto.

```
INICIO
        Leer Precio_Compra, Precio_Venta
        Ganancia = Precio_Venta - Precio_Compra
        Imprimir Ganancia
FIN
```

Figura 7. Primer ejemplo de un algoritmo en pseudocódigo.

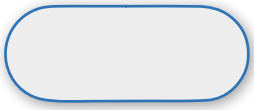
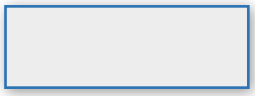
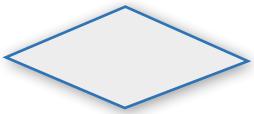
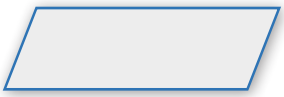
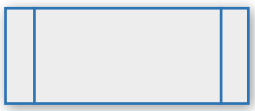
La solución es casi la misma del lenguaje natural, solo que ya con un poco más estructura y una sintaxis diferente. A medida que se avanza en el libro se observarán las diferencias, similitudes y sobre todo las ventajas de cada una de estas formas de representación.

1.6.3 Diagramación libre

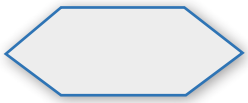
La diagramación libre es también conocida como diagramas de flujo, permite ilustrar la secuencia de pasos de un algoritmo por medio de símbolos especializados y líneas de flujo. La combinación de símbolos especializados y líneas de flujo describe la lógica para la solución del problema (algoritmo). Entonces se puede afirmar que "El diagrama de flujo es la representación gráfica de un algoritmo; para ello se utiliza unconjunto de símbolos estándares mundialmente utilizados" (Sauceda, 2019, p. 4).

Los símbolos usados en la diagramación libre y sus significados se presentan en la tabla 12.[3]

Tabla 12.
Símbolos de diagramación libre (Microsoft Visio).

Símbolo	Significado
	*Símbolo de inicio y final:* se usa para indicar el *Inicio* y el *Fin* del diagrama de flujo.
	*Símbolo de proceso y asignación:* se usa para realizar instrucciones, cálculos y operaciones de asignación.
	*Símbolo de decisión:* se usa para realizar operaciones de comparación de valores con estructuras de decisión.
	*Símbolo de entrada de datos:* se usa para indicar las variables que van a servir de datos de entrada al diagrama.
	*Símbolo de subprogramas:* se usa para representar todos los subprogramas o subrutinas (funciones y procedimientos).
	*Símbolo de impresión:* se usa para mostrar los datos que serán visualizados en la impresora o en la pantalla de la computadora (siendo este último, el dispositivo de salida más usado).

[3] Los símbolos utilizados han sido normalizados por el Instituto Norteamericano de Normalización (ANSI).

Símbolo	Significado
	*Símbolo de estructura repetitiva:* se usa para representar ciclos o estructuras repetitivas dentro del diagrama de flujo.
	*Símbolo de línea o conectores:* se usa para conectar los diferentes símbolos dentro del diagrama de flujo.

En la figura 8 se puede visualizar un algoritmo en diagramación libre a partir del ejemplo de la ganancia obtenida sobre la venta de un producto.

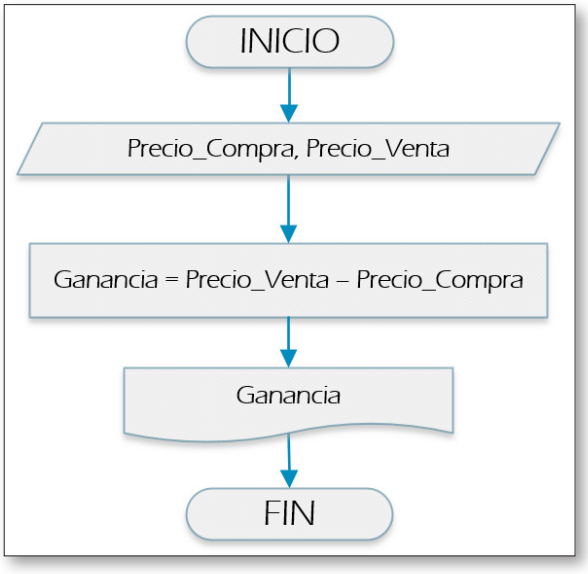


Figura 8. Primer ejemplo de un algoritmo en diagramación libre.

Para realizar un diagrama de flujo se debe tener en cuenta lo siguiente:

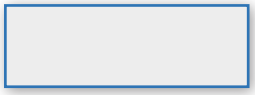
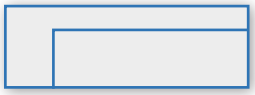
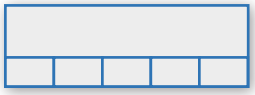
- Usar solo flechas horizontales y/o verticales.
- Evitar el cruce de líneas.
- No dejar líneas de flujo (flechas) sin conectar.
- Trazar los símbolos de manera que puedan leerse de arriba a abajo y de izquierda a derecha.
- Todo texto dentro de un símbolo tiene que ser escrito claramente.

Se puede observar con claridad que a pesar de ser casi idénticos, el diagrama ocupa más espacio que el pseudocódigo. En el diagrama desaparecen las palabras *Leer* e *Imprimir*, porque las figuras tienen su propio significado y es redundante poner el símbolo y la palabra.[4]

1.6.4 Diagramación rectangular

También conocida con el nombre de *diagrama estructurado N–S* (*Nassi–Shneiderman*). En esta diagramación se utilizan una serie de rectángulos consecutivos que son de diferentes tipos y representan diferentes partes de un algoritmo. Los símbolos usados y sus significados se presentan en la tabla 13.

Tabla 13.
Símbolos de diagramación rectangular.

Símbolo	Significado
	*Símbolo general:* se usa para el *Inicio* y el *Fin* del diagrama, para leer e imprimir datos y para la realización de cálculos.
	*Símbolo de decisión:* se usa para realizar operaciones de comparación de valores con estructuras de decisión.
	*Símbolo de estructura repetitiva:* se usa para representar ciclos o estructuras repetitivas dentro del diagrama de flujo.
	*Símbolo de estructuras caso:* se usa para hacer operaciones con estructuras de selección múltiple o caso.

En la figura 9 se puede visualizar un algoritmo en diagramación rectangular a partir del ejemplo de la ganancia obtenida sobre la venta de un producto.

[4] Para profundizar todos los conceptos de diagramación libre y diagramas de flujo (Sauceda, 2019, p. 4).

INICIO
Leer Precio_Compra, Precio_Venta
Ganancia = Precio_Venta - Precio_Compra
Imprimir Ganancia
FIN

Figura 9. Primer ejemplo de algoritmo en diagramación rectangular.

1.6.5 Ejemplo de algoritmo cualitativo

a. *Problema:* diseñar un algoritmo para preparar una limonada para un grupo de varias personas. Tener en cuenta las diferentes formas de representación de un algoritmo y los conceptos vistos hasta ahora.

b. *Algoritmo en lenguaje natural: e*n la figura 10 se puede visualizar un algoritmo en lenguaje natural, dando solución al problema de preparar una limonada para un grupo de personas.

c. *Algoritmo en pseudocódigo:* en la figura 11 se puede visualizar un algoritmo en pseudocódigo, dando solución al problema de preparar una limonada para un grupo de personas.

- *Entrada:* son los ingredientes y utensilios necesarios para hacer la limonada:
  - 1 jarra.
  - 1 litro de agua.
  - 2 limones.
  - 6 cucharadas de azúcar.
- *Proceso:*
  - Se toma la jarra y se echa el agua.
  - Se echa el jugo de los limones.
  - Se le echa el azúcar.
  - Se revuelve y queda lista la limonada.
- *Salida:*
  - La limonada.

Figura 10. Segundo ejemplo de un algoritmo en lenguaje natural.

```
INICIO
        Llenar una jarra con un litro de agua
        Echar el jugo de los 2 limones
        Echar las 6 cucharadas de azúcar
        Revolver hasta disolver completamente el azúcar
        Queda lista la limonada para ser consumida
FIN
```

Figura 11. Segundo ejemplo de un algoritmo en pseudocódigo.

a. *Algoritmo en diagramación libre:* en la figura 12 se puede visualizar un algoritmo en diagramación libre, dando solución al problema de preparar una limonada para un grupo de personas.

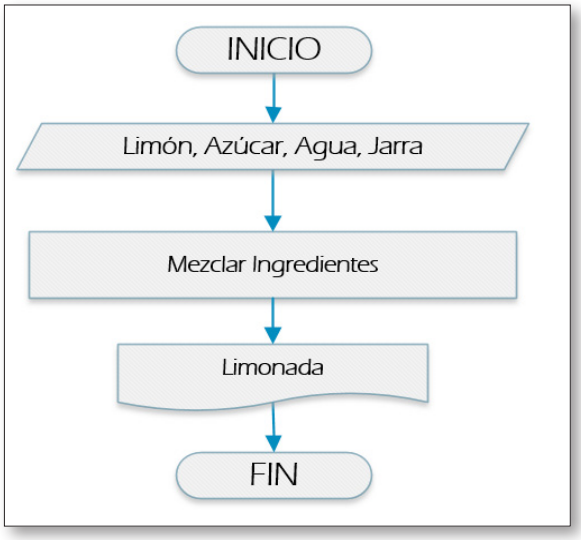


Figura 12. Segundo ejemplo de un algoritmo en diagramación libre.

b. *Algoritmo en diagramación rectangular:* en la figura 13 se puede visualizar un algoritmo en diagramación rectangular, dando solución al problema de preparar una limonada para un grupo de personas:

INICIO
Obtener Limón, Azúcar, Agua, Jarra
Mezclar ingredientes
Limonada terminada
FIN

Figura 13. Segundo ejemplo de un algoritmo en diagramación rectangular.

En el lenguaje natural se pueden omitir algunas consideraciones que se tienen que cumplir en el algoritmo cuantitativo enfocado a pasarse a un lenguaje de programación, por ejemplo, las tildes en las variables al momento de explicar los pasos a realizar.

El leguaje natural es muy flexible y se enfoca principalmente en desglosar y mostrar la solución propuesta de una forma más clara y entendible.

Una operación de leer es similar a la obtención de la información en el desarrollo de un algoritmo, mientras que la operación de imprimir es similar a la entrega de resultados requeridos.

1.6.6 Ejemplo de algoritmo cuantitativo

a. *Problema:* diseñar un algoritmo que permita hallar la suma de tres números enteros.
b. *Algoritmo en lenguaje natural:* en la figura 14 se puede visualizar un algoritmo en lenguaje natural, dando solución al problema de calcular e imprimir la suma de tres números.

- Entrada: tres números que se almacenan en tres variables con nombres nemotécnicos: Num1, Num2 y Num3.
- Proceso: sumar las tres variables: Num1 + Num2 + Num3.
- Salida: resultado de la suma.

Figura 14. Tercer ejemplo de un algoritmo en lenguaje natural.

c. *Algoritmo en pseudocódigo:* en la figura 15 se puede visualizar en algoritmo en pseudocódigo, dando solución al problema de calcular e imprimir la suma de tres números.

```
INICIO
        Leer Num1, Num2, Num3
        Suma = Num1 + Num2 + Num3
        Imprimir Suma
FIN
```

Figura 15. Tercer ejemplo de un algoritmo en pseudocódigo.

d. *Algoritmo en diagramación libre:* en la figura 16 se puede visualizar en algoritmo en diagramación libre, dando solución al problema de calcular e imprimir la suma de tres números.

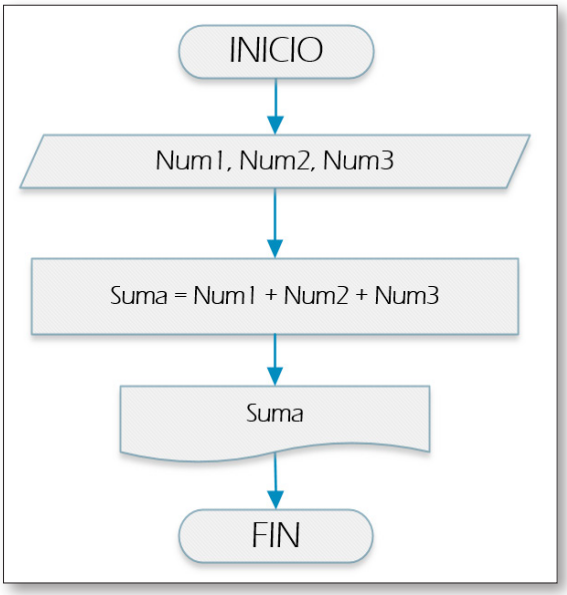


Figura 16. Tercer ejemplo de un algoritmo en diagramación libre.

e. *Algoritmo en diagramación rectangular:* en la figura 17 se puede visualizar en algoritmo en diagramación rectangular, dando solución al problema de calcular e imprimir la suma de tres números.

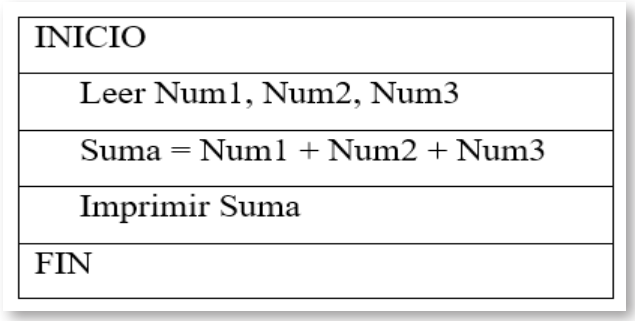


Figura 17. Tercer ejemplo de un algoritmo en diagramación rectangular.

Para practicar este tema explique el proceso que se debe realizar paso a paso, en cada una de estas situaciones: cambiar una bombilla fundida; cambiar un neumático chuzado de un vehículo; realizar una llamada telefónica; funcionamiento de un cajero electrónico al realizar un retiro de dinero; y el funcionamiento de una máquina expendedora de gaseosas.

1.7 Datos manejados en un algoritmo

Todos los datos que se manejan dentro de un algoritmo tienen un tipo asociado. Un dato puede ser un simple carácter tal como *b*, un valor entero tal como 35 o un valor alfanumérico como Calle 42 # 83-60. El tipo de dato determina la naturaleza del conjunto de valores que puede tomar una variable.

Los tipos de datos más frecuentes son alfabéticos, numéricos, alfanuméricos y lógicos.

Dentro de los lenguajes de programación se encuentran con una gran variedad de tipos de datos para el almacenamiento de variables. A continuación, una explicación breve de estos tipos.

1.7.1 Datos numéricos

Normalmente se reconocen dos tipos de variables numéricas:

- *Entera:* permite almacenar datos numéricos enteros (sin decimales); por ejemplo, una edad de 20, una cantidad de 5, un número positivo como 60, un número negativos como –58.
- *Real:* permite almacenar datos numéricos que tengan decimales, por ejemplo, una estatura de 1.75, un peso de 70.2 o el valor de una retención de 0.08.

Cualquier operación entre números reales da como resultado un número real. Cualquier operación entre números enteros da un resultado entero, excepto en la división que pueda dar un número real.

1.7.2 Datos alfabéticos

Normalmente se reconocen dos tipos de variables alfabéticas:

- *Carácter:* permite almacenar una letra, un número o un símbolo en medio de comillas sencillas rectas (‘); por ejemplo, una respuesta ‘S’ o ‘N’, dos valores ‘1’ o ‘5’, o un símbolo como ‘@’.
- *Cadena:* permite almacenar varios caracteres en medio de comillas dobles rectas (“); por ejemplo, un nombre como “Miguel Ángel”, un apellido como “Cañaveral”, una ciudad como “Medellín” o una ocupación como “Docente”.

Además, dentro de lógica de programación y los lenguajes de programación, también son clasificadas como alfabéticas aquellas variables que almacenen números con los cuales no se van a realizar cálculos o no se hacen operaciones matemáticas, por ejemplo, un número de cédula “1.023.245”, un NIT “1023245-3” o un código postal “259”. Lo anterior es fundamental para optimizar el manejo de memoria en una computadora; una vez que una variable numérica ocupa más espacio que una variable de tipo alfabética.

1.7.3 Datos alfanuméricos

Los datos alfanuméricos permiten almacenar combinaciones de datos alfabéticos, datos numéricos que serán tratados como alfabéticos en forma de cadena de caracteres, símbolos o caracteres especiales. Los ejemplos más claros son una dirección de una residencia (Carrera 50 # 10-00), una

dirección de correo electrónico (micorreo_123@servidor.com), una placa de un vehículo (MNU664) o el código de un producto (REF–4560).

1.7.4 Datos lógicos o booleanos

Los datos lógicos o booleanos permiten almacenar aquellos datos que tengan solo valores de *Verdadero* o *Falso*, 1 o 0, *F* o *V*, entre otros. Estas variables son resultado de una comparación entre otros datos. Por ejemplo, si se tiene: A=1, B=10, C=–5. El resultado de hacer algunas comparaciones se muestra en la tabla 14.

Tabla 14.
Comparación entre diferentes tipos de datos.

Al comparar	El resultado sería	Explicación
A > B	*Falso*	A (1) no es mayor que B (10)
A != B	*Verdadero*	A (1) no es igual B (10)
C > A	*Falso*	C (–5) no es mayor que A (1)
C < A	*Verdadero*	C (–5) es menor que A (1)
C > 0	*Falso*	C (–5) no es mayor que 0
B > 0	*Verdadero*	B (10) es mayor que 0
B < 0	*Falso*	B (10) no es menor que 0
B = 10	*Verdadero*	B (10) es igual a 10
A = 3	*Falso*	A (1) no es igual a 3
8 <> C	*Verdadero*	8 es diferente de C (–5)

Al momento de hacer comparaciones se debe tener en cuenta la tabla de verdad vista en matemáticas básicas. La tabla 15 muestra tablas de verdad para los operadores de negación (~), conjunción (Λ) y disyunción (V).[5]

Tabla 15.
Tabla de verdad de la negación, conjunción y disyunción.

p	Q	NO (~p)	NO (~q)	Y (p Λ q)	O (p V q)
Verdadero	*Verdadero*	*Falso*	*Falso*	*Verdadero*	*Verdadero*
Verdadero	*Falso*	*Falso*	*Verdadero*	*Falso*	*Verdadero*
Falso	*Verdadero*	*Verdadero*	*Falso*	*Falso*	*Verdadero*
Falso	*Falso*	*Verdadero*	*Verdadero*	*Falso*	*Falso*

[5] Para profundizar en el tema, se recomienda el texto *Lógica Matemática para Ingeniería de Sistemas y Computación* de Cardona Torres, Hernández Rodríguez y Jaramillo Valbuena (2010).

Por ejemplo, 1 < 7 Λ 5 < 10 = *Verdadero* y 5 > 20 Λ 'A' < 'B' = *Falso* (mirar la explicación en la figura 18).

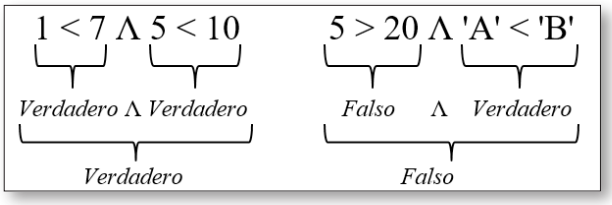


Figura 18. Ejemplo de comparaciones con la conjunción.

Cuando se evalúa una expresión teniendo el operador de la conjunción (Λ), se tienen que cumplir las dos condiciones para que el resultado sea *Verdadero*; mientras que con el operador de la disyunción (V), basta con que se cumpla una sola para que el resultado sea *Verdadero*. Mirar, en la figura 19, el mismo ejemplo anterior, pero cambiando el operador de conjunción por el de disyunción.

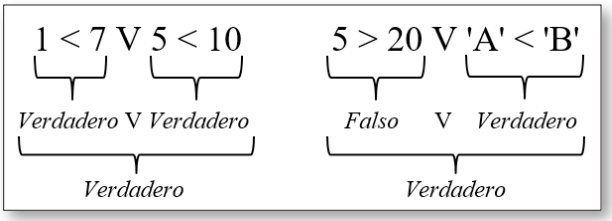


Figura 19. Ejemplo de comparaciones con la disyunción.

Cuando se comparan variables alfabéticas teniendo en cuenta que la 'A' es menor que la 'B', la letra 'Z' es mayor que todas las letras. Cuando una computadora procesa estas letras tiene en cuenta su código ASCII, la letra 'A' tiene un código de 65, la letra 'B' tiene un código de 66 y la 'Z' de 90 (véase anexo 9).

1.7.5 Ejercicios propuestos de tipos de datos

Indicar el tipo de dato (si es Numérico-Entero, Numérico-Real, Alfabético-Cadena, Alfabético-Carácter o Lógico) de los siguientes valores:

a. ‘A’
b. –89
c. “125”
d. “–9”
e. –5
f. 0
g. 125.00
h. 4.000
i. ‘+’
j. “9.12”
k. 325
l. “fin del algoritmo”
m. –5698.23
n. V
o. 2.333
p. Falso
q. 0.0005
r. 30 000
s. “Sí”
t. –500.00

1.7.6 Ejercicios propuestos de expresiones booleanas

Indique el resultado de las siguientes expresiones (*Falso* o *Verdadero*):

a. 3 < 6
b. 5 >= 10
c. ‘A’ > ‘Z’
d. ‘8’ > ‘700’
e. 8 <> 8
f. 64 >= 64

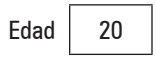
g. ‘8’ > ‘7’
h. 45 mod 2 = 0
i. 2 ^ 4 <> 84
j. 85.36 = 85.34 V 235 = 114
k. ~ (‘A’ >= ‘Z’ V ‘8’ <= ‘0’)
l. 12 <> 15 Λ 111<= 114
m. ~ (17 < 698 Λ ‘J’ <> ‘K’)
n. 158 >= 158 Λ 256 <= 256
o. 25 > 12 V 0 > 14
p. 25 > 12 Λ 0 > 14
q. 3 ^ 2 = 36 / 6 + 3
r. 5 * 4 + 2 > 45 mod 13

1.7.7 Datos variables

Son espacios de memoria que almacenan valores que pueden cambiar en el transcurso de un algoritmo. Usualmente son introducidas por el usuario.

Toda variable consta de un *nombre* y un *tipo*. El nombre es un conjunto de letras o caracteres y números, en el cual se almacena la información ingresada por el usuario o asignada por el programador. El tipo indica la clase de información que almacena la variable, la cual puede ser entera, real, cadena o lógica.

Por ejemplo, al declarar una variable llamada *Edad* de tipo *Entero,* en memoria quedaría representado así:



1.7.8 Datos constantes

Son espacios de memoria que no cambian en el transcurso de un algoritmo y cuyos valores son asignados al momento de su declaración. En determinados casos se puede presentar que el valor de una variable no cambie en

el transcurso de un algoritmo; pero por este hecho no puede ser catalogada como constante

Una constante es un dato alfabético, numérico, alfanumérico o lógico que no cambia durante la ejecución del programa. Un ejemplo claro de un valor constante es el valor de Π (*Pi* siempre va a ser igual a 3.14159265358979, y este valor nunca cambia). En memoria quedaría representado así:

Pi	3.14159265358979

Cuando en otro lugar del algoritmo se intente cambiar el valor de Pi por 0; el algoritmo lanzaría una alerta indicando que las constantes no pueden ser modificadas o se suspendería su ejecución.

1.7.9 Condiciones para nombrar variables y constantes

Para definir el nombre de una variable se deben tener en cuenta las siguientes condiciones:

- Debe comenzar con una letra, mayúscula o minúscula comprendida entre la 'A' y la 'Z' o la 'a' y la 'z' (omitiendo la letra ñ que no existe en el idioma inglés y los lenguajes de programación están en ese idioma; ya que se reconoce como un carácter especial).
- No debe contener espacios en blanco.
- No se pueden utilizar caracteres especiales (´ ¨ ¿ ¡), símbolos (% & # @), operadores aritméticos (+ – * /) o signos ortográficos (, ; . :).
- Los dígitos y el carácter de subraya o guion bajo (_) están permitidos después del primer carácter; aunque algunos lenguajes como Java o C++ permiten usarlo al inicio.

1.7.10 Ejercicios propuestos

Coloque al frente de cada nombre de variable, si est**á** *Correcta* o *Incorrecta* y la explicación de su respuesta, solo cuando esta sea *Incorrecta*:

a. $Sueldo

b. A

c. cElUlAr

d. Dir_casa
e. Edad
f. Notas
g. N-Tel
h. SB512
i. Tel
j. c@rreo
k. Apellido paterno
l. Nombre
m. 2Salarios
n. Pensión
o. N1
p. Primera/Nota
q. 1Nota
r. Año

Unidad 2
Estructura fundamental de un algoritmo

Introducción

En esta unidad se explican los siguientes temas (ver el resumen en la figura 20).

- Estructura secuencial.
- Datos de entrada, proceso y datos de salida.
- Asignación interna, asignación externa y actualización.
- Operaciones primordiales y básicas.
- Porcentajes.
- Conversión de medidas.
- Prueba de escritorio.
- Ejercicios resueltos y propuestos.

Estos temas facilitan la obtención de las competencias que se describen a continuación:

- Utilizar de forma correcta los diferentes operadores aritméticos y lógicos, teniendo en cuenta la jerarquía que indica el orden en que deben ejecutarse las operaciones.
- Conocer las principales partes de un algoritmo secuencial, identificando datos de entrada, proceso y datos de salida.
- Comprender el funcionamiento de las computadoras y la integración que tiene con la lógica de programación al momento de dar solución a problemas planteados.
- Manejar de forma correcta la asignación de información a una variable por medio de la asignación interna, la asignación externa o la actualización.
- Resolver ejercicios que requieran del manejo de operaciones primordiales y la conversión de fórmulas a expresiones algorítmicas.

Figura 20. Unidad 2. Estructura fundamental de un algoritmo.

- Calcular porcentajes dentro de algoritmos utilizando las tres fórmulas de acuerdo con el tipo de problema planteado.
- Solucionar cualquier ejercicio planteado relacionado con las estructuras secuenciales que componen un algoritmo.

2. Estructura secuencial de un algoritmo

Un algoritmo está conformado por varias estructuras fundamentales, cada una con un propósito y uso determinado. Estas estructuras son llamadas *estructuras de control.* La primera estructura es la secuencial, la cual se trabajará en esta segunda unidad; y la otras son la estructura de decisión y la estructura de selección múltiple (caso), que se trabajarán en la unidad 3.

2.1 Estructura secuencial

Es la estructura más simple dentro de los algoritmos, donde una acción o instrucción sigue a otra en secuencia hasta llegar al fin del algoritmo. Esta estructura consta de tres partes primordiales: los datos de entrada, el proceso y los datos de salida, temas tratados en el numeral 1.5.3 de la primera unidad.

Tomando como base la representación de una computadora en su unidad central de proceso (Joyanes, 2008), se plantea la siguiente relación (figura 21).

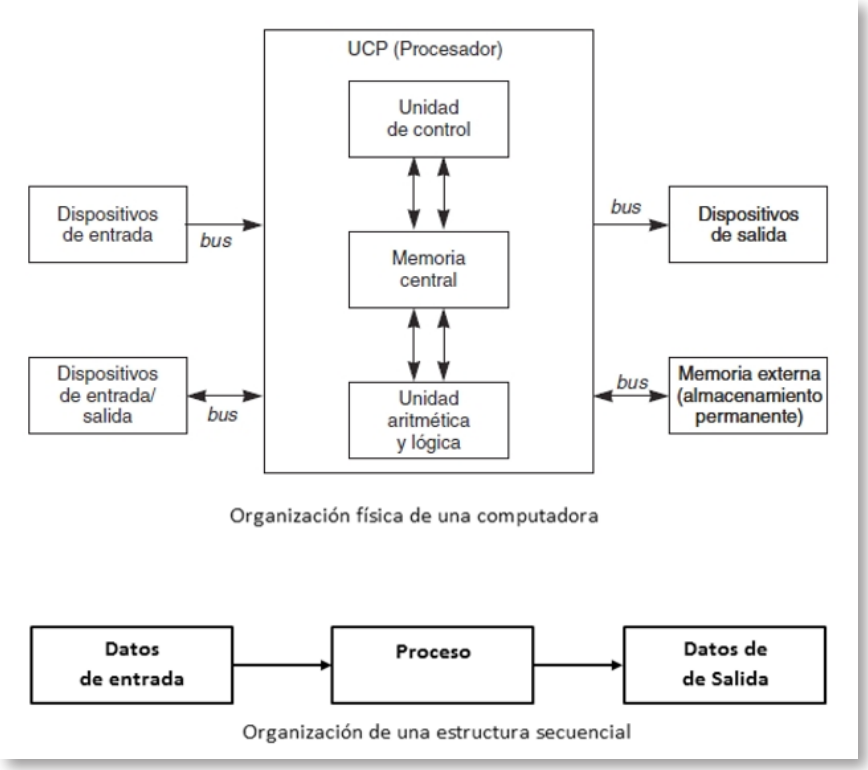


Figura 21. Organización física de una computadora.
Fuente: Edukativos (2013).

Todo programador debería tener presente la relación entre la computadora y estructuras secuenciales (Edukativos, 2013). Los datos de entrada son los que se ingresan a la computadora por medio de los dispositivos de entrada (teclado, *mouse*, cámara, entre otros); en nuestro caso se usará únicamente el teclado para ingresar la información. El proceso son los cálculos y operaciones de un algoritmo que se realizan en la unidad central de proceso, estas se almacenan en variables (creadas anteriormente) y después se convierten en los datos de salida, los cuales se muestran a través de los diferentes dispositivos de salida (pantalla, impresora, parlantes, entre otros); en nuestro caso se usará únicamente la pantalla para visualizar los resultados.

De esta forma, todo lo que se haga en un algoritmo se enfoca a lo ingresado por el usuario, el proceso que se realiza y su respectiva salida. En

muchas ocasiones es bueno pensar en situaciones de la vida cotidiana en la cual se refleja esta relación, por ejemplo: al acercarse a un cajero el cliente ingresa datos; la máquina realiza operaciones internas (normalmente no son visibles ni entendibles para los usuarios); y finalmente, se produce una salida determinada.

Para entender mejor el tema, mirar el siguiente ejemplo de un algoritmo.

Ejemplo. Desarrolle un algoritmo que calcule e imprima la multiplicación de dos valores.

Normalmente antes de realizar un algoritmo se debe hacer un preanálisis de cómo y con qué se va a resolver. Aquí solo se hace este proceso para ver su uso; pero luego se requiere que esa etapa se haga mentalmente.

Datos de entrada (datos necesarios para realizar los cálculos): para multiplicar los dos valores, se necesitan estos dos datos. Por lo tanto, manejamos dos variables *Num1* y *Num2*, cuyos nombres de variables representan esos dos valores. Además, en la parte del proceso, se necesita una variable *Mult* para almacenar el resultado. También se dice que la línea de datos de entrada empieza con la palabra *Leer* y las variables separadas por coma (,); quedando así: *Leer Num1, Num2*.

Proceso (cálculos solicitados): para multiplicar dos valores, *Num1* y *Num2,* se puede poner *Num1x Num2, Num1. Num2 o (Num1) (Num2)*; pero la multiplicación en los algoritmos se representa por medio del operador aritmético asterisco (*). Por lo tanto, se pone *Num1 * Num2*. Además, la teoría dice que el cálculo se guarda en una variable (*Mult*), quedando así: *Mult = Num1 * Num2*.

Datos de salida (datos que tienen los cálculos solicitados para dar solución al problema): en este caso es la multiplicación que se almacenó en la variable *Mult*. Además, se dice que la línea de datos de salida empieza con la palabra *Imprimir.* En este libro se va a manejar este comando poniendo las variables separadas por coma (,), en caso de que existe más de una; quedando así: *Imprimir Mult*.

Con lo anterior hemos hecho el análisis al primer algoritmo enfocado en las partes de un algoritmo: entrada, proceso y salida. Ahora se agregan mensajes para los datos de entrada y los datos de salida.

En la figura 22 se visualiza el pseudocódigo en PSeInt y su respectiva ejecución en caso de digitar los números 4 y 6.

```
Algoritmo Inicio
    Imprimir "Ingrese número 1: "
    Leer Num1
    Imprimir "Ingrese número 2: "
    Leer Num2
    Mult = Num1 * Num2
    Imprimir Mult
    Imprimir Num1, " multiplicado por ", Num2, " es igual a: ", Mult
    Imprimir Num1, " X ", Num2, " = ", Mult
FinAlgoritmo
```

Ejecución

```
Ingrese número 1:
> 4
Ingrese número 2:
> 6
24
4 multiplicado por 6 es igual a: 24
4 X 6 = 24
```

Figura 22. Ejemplo de salida en pantalla usando PSeInt.

En PSeInt se usará la opción del perfil: *Flexible* ('Menú Configurar' – 'Opciones del Lenguaje' ('Perfiles'), la cual permite usar una estructura similar al del pseudocódigo, pero sin la sintaxis estricta que tiene un lenguaje de programación, por ejemplo, declarar y definir variables o terminar instrucciones en punto y coma (;).

A partir de esta sección del libro se empezarán a trabajar los algoritmos en pseudocódigo y su equivalente en el lenguaje de programación Python, por lo tanto, es de suma importancia que se aborden primero los textos de Python *Software Foundation* (2017), Gilpérez (2015) y Sientes (2017) relacionados con dicho lenguaje, allí encontrará explicación de la sintaxis y lo básico para tener en cuenta al momento de programar.

Véase la figura 23 para observar el algoritmo codificado en Python con su respectiva ejecución y comparar ambas soluciones.

main.py

```
print("Ingrese número 1: ")
num1 = float(input())
print("Ingrese número 2: ")
num2 = float(input())
mult = num1 * num2
print(mult)
print(num1," multiplicado por ",num2," es igual a: ", mult)
print(num1," X ",num2," = ",mult)
```

```
Ingrese número 1:
6
Ingrese número 2:
4
24.0
6.0 multiplicado por 4.0 es igual a: 24.0
6.0 X 4.0 = 24.0
```

Figura 23. Ejemplo de salida en Python.

2.2 Asignación de información

Asignar información a una variable consiste en darle un valor a esta. En algoritmos se pueden asignar valores y/o variables de tres formas: asignación interna, asignación externa y actualización, las cuales se explican a continuación.

2.2.1 Asignación interna

La asignación interna se da dentro de un algoritmo cuando se le lleva un valor a una variable por medio del signo igual (=). También se puede asignar variables a otras variables.

Por ejemplo, cuando se coloca en un algoritmo una instrucción como *X = 15*, se le está asignando el valor de 15 a la variable *X*. También se pudo colocar *X = A*, y estamos haciendo una asignación interna de variable a variable. El valor de la derecha es llevado a la variable de la izquierda.

Existen muchas formas de representar este tipo de asignación, las cuales dependen del año de publicación del libro o del lenguaje de programación que se tome como base en la implementación de los algoritmos. Algunos de los ejemplos más reconocidos son: *X←15* se usa en pseudocódigo tradicional (PSeInt) y *X=15* se usa en pseudocódigo moderno y lenguajes de programación como

En este libro usaremos el signo igual (=) para referirnos a la asignación interna. Por ejemplo, para asignar 15 a la variable *X* se usa la instrucción *X* = 15.

2.2.2 Asignación externa

La asignación externa se da cuando se establece un valor a una variable por medio de la instrucción *Leer*; pero este valor solo lo toma cuando se hace una ejecución o una prueba de escritorio al ejercicio (tema que se explica más adelante). Para asignar el valor a una variable de forma externa se pueden usar la sentencia *Leer*, *Lea*, *Ingresar* o *Ingrese*. En el lenguaje de programación Python se usa el comando *input*.

En este libro usaremos la instrucción *Leer* para realizar asignación externa a una variable en un algoritmo. Por ejemplo, *Leer variable*.

2.2.3 Actualización

La actualización se presenta cuando una variable cambia de valor en el transcurso del algoritmo. Por ejemplo, cuando se coloca EDAD = 20 a través de una asignación interna; pero luego se coloca otra instrucción como EDAD = EDAD + 10, en ese momento la variable se *actualiza* al valor de 30.

Cuando se está programando, es muy común realizar actualización a diferentes variables en el desarrollo y codificación de este.

2.3 Operaciones básicas en algoritmos

Las operaciones básicas son todas aquellas que incluyen los operadores aritméticos, combinados con algunas variables y constantes. El ejemplo más sencillo se presenta es cuando en un algoritmo se realizan sumas, restas, multiplicaciones, divisiones, potenciaciones, fraccionarios o radicales. En la tabla 16 se presentan algunos ejemplos de operaciones.

Tabla 16.
Ejemplos de operaciones primordiales.

Operación	Instrucción en algoritmos
Suma de dos valores	Valor1 + Valor2
Resta de dos valores	Valor1 – Valor2
División de dos valores	Valor1 / Valor2
Multiplicación de dos valores	Valor1 * Valor2
Cuadrado de un valor	Valor ^ 2 o Valor * Valor
Cubo de un valor	Valor ^ 3 o Valor * Valor * Valor
Potencia N de un valor	Valor ^ N
Doble de un valor	Valor * 2 o Valor + Valor
Triple de valor	Valor * 3 o Valor + Valor + Valor
Raíz cuadrada de un valor	Valor ^ (1 / 2) o Valor ^ 0.5

Operación	Instrucción en algoritmos
Raíz cúbica de un valor	Valor ^ (1 / 3)
Raíz enésima de un valor	Valor ^ (1 / N)
Mitad de un valor	Valor / 2 o Valor * 0.5
Tercera parte de un valor	Valor / 3
Enésima parte de un valor	Valor / N
10% de un valor	Valor * 0.1 o Valor * 10/100
4% de un valor	Valor * 0.04 o Valor * 4/100

Los operadores aritméticos cambian dependiendo el lenguaje de programación; pero lo que debe hacer todo programador es consultar la documentación del lenguaje y adaptar sus conceptos de lógica a los del lenguaje, lo más importante es identificarlos y conocer su funcionamiento. Por ejemplo, en Python se representaría *N ** 3*.

Para identificar los diferentes operadores en el lenguaje de programación Python pueden revisar la información documentada por Python *Software Foundation* (2017), Gilpérez (2015) y Sientes (2017) correspondiente a ese lenguaje.

Ejemplo 1. Realizar un algoritmo que lea tres números, calcule e imprima la suma y el producto de esos números.

En la figura 24 se muestra el pseudocódigo en PSeInt y en la figura 25 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Ingrese número 1: "
3      Leer Num1
4      Imprimir "Ingrese número 2: "
5      Leer Num2
6      Imprimir "Ingrese número 3: "
7      Leer Num3
8      Sum = Num1 + Num2 + Num3
9      Pdto = Num1 * Num2 * Num3
10     Imprimir "Suma de los 3 números: ", Sum
11     Imprimir "Producto de los 3 números: ", Pdto
12 FinAlgoritmo
```

Ejecución

```
Ingrese número 1:
> 2
Ingrese número 2:
> 4
Ingrese número 3:
> 5
Suma de los 3 números: 11
Producto de los 3 números: 40
```

Figura 24. Primer ejemplo de operaciones básicas en PSeInt.

main.py

```
print("Ingrese número 1: ")
num1 = float(input())
print("Ingrese número 2: ")
num2 = float(input())
print("Ingrese número 3: ")
num3 = float(input())
sum = num1 + num2 + num3
pdto = num1 * num2 * num3
print("Suma de los 3 números: ", sum)
print("Producto de los 3 números: ", pdto)
```

```
Ingrese número 1:
2
Ingrese número 2:
4
Ingrese número 3:
5
Suma de los 3 números:  11.0
Producto de los 3 números:  40.0
```

Figura 25. Primer ejemplo de operaciones básicas en Python.

En la ejecución en Python aparecen las respuestas con decimal debido a que las variables se leyeron de tipo *float* (flotantes o reales).

Ejemplo 2. Hacer un algoritmo al que se le ingrese un número e imprima el cuadrado y el cubo del número. Así mismo, la raíz cuadrada y la raíz cúbica del mismo número.

En la figura 26 se muestra el pseudocódigo en PSeInt y en la figura 27 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese número: "
    Leer Num
    Cuad = Num ^ 2
    Cubo = Num ^ 3
    Raiz_Cuad = Num ^ 0.5
    Raiz_Cub = Num ^ (1/3)
    Imprimir "Cuadrado del número: ", Cuad
    Imprimir "Cubo del número: ", Cubo
    Imprimir "Raíz cuadrada del número: ", Raiz_Cuad
    Imprimir "Raíz cúbica del número: ", Raiz_Cub
FinAlgoritmo
```

Ejecución

```
Ingrese número:
> 10
Cuadrado del número: 100
Cubo del número: 1000
Raíz cuadrada del número: 3.1622776602
Raíz cúbica del número: 2.15443469
```

Figura 26. Segundo ejemplo de operaciones básicas en PSeInt.

main.py

```
print("Ingrese número: ")
num = float(input())
cuad = num ** 2
cubo = num ** 3
raiz_cuad = num ** 0.5
raiz_cub = num ** (1/3)
print("Cuadrado del número: ", cuad)
print("Cubo del número: ", cubo)
print("Raíz cuadrada del número: ", raiz_cuad)
print("Raíz cúbica del número: ", raiz_cub)
```

```
Ingrese número:
10
Cuadrado del número:  100.0
Cubo del número:  1000.0
Raíz cuadrada del número:  3.16227
Raíz cúbica del número:  2.1544346
```

Figura 27. Segundo ejemplo de operaciones básicas en Python.

Ejemplo 3. Desarrolle un algoritmo que lea un número y permita calcular e imprimir el doble de este número.

Todo algoritmo empieza con *Inicio*, luego van los datos de entrada con la palabra *Leer* seguida de las variables necesarias (en este caso solo es una variable llamada *Numero)*, a la cual se le va a calcular el doble. Luego en el proceso, se calcula el doble al valor (se multiplica la variable *Numero* * *2*). Finalmente se ponen los datos de salida que empiezan con la palabra *Imprimir*. Se debe recordar que todo algoritmo termina con la palabra *Fin*. Ver solución.

En la figura 28 se muestra el pseudocódigo en PSeInt y en la figura 29 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1 Algoritmo Inicio
2     Imprimir "Ingrese un número: "
3     Leer Numero
4     Doble = Numero * 2
5     Imprimir "Doble del número: ", Doble
6 FinAlgoritmo
```

Ejecución

```
Ingrese un número:
> 16
Doble del número: 32
```

Figura 28. Tercer ejemplo de operaciones básicas en PSeInt.

main.py

```
1  print("Ingrese un número: ")
2  numero = float(input())
3  doble = numero * 2
4  print("Doble del número: ",doble)
```

```
Ingrese un número:
16
Doble del número:  32.0
```

Figura 29. Tercer ejemplo de operaciones básicas en Python.

Ejemplo 4. Hacer un pseudocódigo que permita calcular e imprimir el doble de la suma de dos números y el triple de la resta de esos mismos números.

En la figura 30 se muestra el pseudocódigo en PSeInt y en la figura 31 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese un número: "
    Leer Num1
    Imprimir "Ingrese otro número: "
    Leer Num2
    Doble = (Num1 + Num2) * 2
    Triple = (Num1 - Num2) * 3
    Imprimir "El doble de la suma es: ", Doble
    Imprimir "El triple de la resta es: ", Triple
FinAlgoritmo
```

Ejecución

```
Ingrese un número:
> 2
Ingrese otro número:
> 4
El doble de la suma es: 12
El triple de la resta es: -6
```

Figura 30. Cuarto ejemplo de operaciones básicas en PSeInt.

main.py

```
print("Ingrese un número: ")
num1 = float(input())
print("Ingrese otro número: ")
num2 = float(input())
doble = (num1 + num2) * 2
tripl = (num1 - num2) * 3
print("El doble de la suma es: ", doble)
print("El triple de la resta es: ", tripl)
```

```
Ingrese un número:
2
Ingrese otro número:
4
El doble de la suma es:  12.0
El triple de la resta es:  -6.0
```

Figura 31. Cuarto ejemplo de operaciones básicas en Python.

Ejemplo 5. Realizar un algoritmo que permita calcular la raíz cuadrada de la mitad de la multiplicación de dos números. Además, calcular la raíz cúbica del triple de la cuarta parte de la suma del cuadrado del primer número más el cubo del segundo número.

En la figura 32 se muestra el pseudocódigo en PSeInt y en la figura 33 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite Número 1: "
    Leer Num1
    Imprimir "Digite Número 2: "
    Leer Num2
    Raiz_Cuad = (Num1 * Num2 / 2) ^ (1 / 2)
    Raiz_Cub = ((Num1 ^ 2 + Num2 ^ 3) / 4 * 3) ^ (1 / 3)
    Imprimir "La raíz cuadrada es: ", Raiz_Cuad
    Imprimir "La raíz cúbica es: ", Raiz_Cub
FinAlgoritmo
```

Ejecución

```
Digite Número 1:
> 5
Digite Número 2:
> 6
La raíz cuadrada es: 3.8729833462
La raíz cúbica es: 5.6540472737
```

Figura 32. Quinto ejemplo de operaciones básicas en PSeInt.

main.py

```
1   import math
2
3   print("Digite Número 1: ")
4   num1 = float(input())
5   print("Digite Número 2: ")
6   num2 = float(input())
7   raiz_cuad = math.sqrt(num1 * num2 / 2)
8   raiz_cub = math.pow(((num1 ** 2 + num2 ** 3) / 4 * 3),  (1 / 3))
9   print("La raíz cuadrada es: ", raiz_cuad)
10  print("La raíz cúbica es: ", raiz_cub)
```

```
Digite Número 1:
 5
Digite Número 2:
 6
La raíz cuadrada es:  3.872983346207417
La raíz cúbica es:  5.654047273715072
```

Figura 33. Quinto ejemplo de operaciones básicas en Python.

En Python se debe importar una librería llamada *math* para un mejor funcionamiento de un ejercicio que requiera de funciones matemáticas. Estos conceptos son fundamentales en los lenguajes de programación actuales. Si desea compilar en varios lenguajes de programación puede dirigirse al sitio web https://repl.it, el cual ofrece compiladores en diferentes lenguajes en línea (Neoreason, 2018).

2.4 Fórmulas y ecuaciones en algoritmos

Luego de trabajar la conversión de expresiones matemáticas en expresiones algorítmicas, será muy fácil trabajar con fórmulas y ecuaciones, debido a que debe escribirse de tal forma que sea entendible para una computadora. Esa es la finalidad de los algoritmos: realizar soluciones a problemas usando los conceptos lógicos aplicados a programación. La recopilación de fórmulas generales de figuras geométricas se encuentra en el Anexo 1.

Para resolver ejercicios de conversión de unidades se recomienda remitirse a los anexos 3, 4, 5, 6 y 7 donde hago una recopilación de algunos apartados del Manual de fórmulas y tablas matemáticas (Spiegel, 1991). Para ejercicios que requieran fórmulas de conversión de temperaturas y equivalencias de computación remitirse a los anexos 2 y 8 respectivamente.

Ejemplo 1. Se tiene como referencia un problema que permita calcular y mostrar el volumen de una caja. Analizar e identificar las variables a usar teniendo en cuenta la figura 34 (Disfruta Las Matemáticas, 2011).

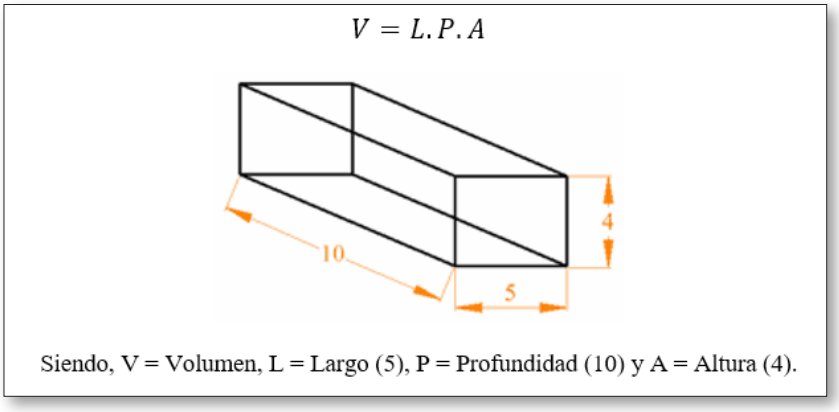


Figura 34. Fórmula para calcular el volumen de una caja.
Fuente: Disfruta Las Matemáticas (2011).

Para resolver problemas de este tipo se efectúan tres etapas:

Etapa 1: en esta etapa, se leen los datos de entrada que corresponden a las variables necesarias para calcular la fórmula (*L*, *P* y *A*). Estas variables se identifican fácilmente porque están ubicadas al lado derecho del signo igual de la fórmula. La instrucción quedaría así: *Leer L, P, A*; pero al momento de realizar el pseudocódigo en PSeInt se agregan los mensajes correspondientes y se lee una variable por cada línea.

Etapa 2: en esta etapa, se convierte la fórmula a expresión algorítmica, teniendo en cuenta almacenar el cálculo en una variable con nombre nemotécnico. La instrucción quedaría así: $V = L * P * A$.

Etapa 3: en esta etapa, se imprime(n) la(s) variable(s) que guarda(n) el (los) cálculo(s). Como ayuda, estas variables se encuentran al lado izquierdo del signo igual de la fórmula. La instrucción quedaría así: *Imprimir "Volumen de la caja = ", V*.

En la figura 35 se muestra el pseudocódigo en PSeInt y en la figura 36 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite largo: "
    Leer L
    Imprimir "Digite profundidad: "
    Leer P
    Imprimir "Digite altura: "
    Leer A
    V = L * P * A
    Imprimir "Volumen de la caja = ", V
FinAlgoritmo
```

Ejecución

```
Digite largo:
> 2.5
Digite profundidad:
> 3.3
Digite altura:
> 4.5
Volumen de la caja = 37.125
```

Figura 35. Primer ejemplo de fórmulas y ecuaciones en PSeInt.

main.py

```
print("Digite largo: ")
L = float(input())
print("Digite profundidad: ")
P = float(input())
print("Digite altura: ")
A = float(input())
V = L * P * A
print("Volumen de la caja = ", V)
```

```
Digite largo:
2.5
Digite profundidad:
3.3
Digite altura:
4.5
Volumen de la caja =  37.125
```

Figura 36. Primer ejemplo de fórmulas y ecuaciones en Python.

Aquí se refleja lo sencillo que es resolver ejercicios que requieran de fórmulas y ecuaciones en algoritmos secuenciales.

Ejemplo 2. Se pide hacer un algoritmo que determine el área y el perímetro de un cuadrado cuyas medidas están en centímetros.

Las fórmulas necesarias para resolver el ejemplo son:

- Área = L*ado* × *Lado*. También se puede usar $Lado^2$ que es exactamente lo mismo.
- *Perímetro* = *Lado* + *Lado* + *Lado* + *Lado*. La suma de los lados o también se puede usar *4* × *Lado*, que es exactamente lo mismo.

Las fórmulas y ecuaciones manejan formato matemático, pero al momento de realizar el cálculo debe tenerse en cuenta convertirlo a expresión algorítmica y tener en cuenta las condiciones para nombrar variables, descritas en el numeral 1.7.9.

Aplicar las tres etapas para la conversión de fórmulas y ecuaciones a expresiones algorítmicas a este ejemplo: Leer datos (solo sería el *Lado*, en

este caso la variable *L*), convertir la fórmula en una expresión algorítmica guardando el resultado en variables como *Area* y *Perim*; y finalmente, imprimir las variables que tienen los cálculos.

En la figura 37 se muestra el pseudocódigo en PSeInt y en la figura 38 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1 Algoritmo Inicio
2     Imprimir "Digite lado (cm): "
3     Leer L
4     Area = L * L
5     Perim = 4 * L
6     Imprimir "Área cuadrado = ", Area, " cm2"
7     Imprimir "Perímetro cuadrado = ", Perim, " cm"
8 FinAlgoritmo
```

Ejecución

```
Digite lado (cm):
> 5.5
Área cuadrado = 30.25 cm2
Perímetro cuadrado = 22 cm
```

Figura 37. Segundo ejemplo de fórmulas y ecuaciones en PSeInt.

main.py

```
1  print("Digite lado (cm): ")
2  L = float(input())
3  Area = L * L
4  Perim = 4 * L
5  print("Área cuadrado = ", Area," cm2")
6  print("Perímetro cuadrado = ", Perim," cm")
```

```
Digite lado (cm):
5.5
Area cuadrado =  30.25  cm2
Perímetro cuadrado =  22.0  cm
```

Figura 38. Segundo ejemplo de fórmulas y ecuaciones en Python.

Ejemplo 3. Hacer un algoritmo que permita resolver la siguiente ecuación de segundo grado (trinomio cuadrado):

$$ax^2 + bx + c$$

Las variables necesarias para resolver esta ecuación son *a*, *x*, *b* y *c*. Por lo tanto, se leen y la fórmula se convierte en expresión algorítmica.

En la figura 39 se muestra el pseudocódigo en PSeInt y en la figura 40 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese valor de a: "
    Leer a
    Imprimir "Ingrese valor de x: "
    Leer x
    Imprimir "Ingrese valor de b: "
    Leer b
    Imprimir "Ingrese valor de c: "
    Leer c
    Ecuac = a * x ^ 2 + b * x + c
    Imprimir "Resultado ecuación = ", Ecuac
FinAlgoritmo
```

Ejecución

```
Ingrese valor de a:
> 2
Ingrese valor de x:
> 1
Ingrese valor de b:
> 3
Ingrese valor de c:
> 5
Resultado ecuación = 10
```

Figura 39. Tercer ejemplo de fórmulas y ecuaciones en PSeInt.

main.py

```
print("Ingrese valor de a: ")
a = float(input())
print("Ingrese valor de x: ")
x = float(input())
print("Ingrese valor de b: ")
b = float(input())
print("Ingrese valor de c: ")
c = float(input())
ecuac = a * x ** 2 + b * x + c
print("Resultado ecuación = ", ecuac)
```

```
Ingrese valor de a:
2
Ingrese valor de x:
1
Ingrese valor de b:
3
Ingrese valor de c:
5
Resultado ecuación =  10.0
```

Figura 40. Tercer ejemplo de fórmulas y ecuaciones en Python.

Ejemplo 4. Un vehículo recorre una distancia en un tiempo determinado. Se pide dar una solución que calcule la velocidad a la que viaja este vehículo teniendo en cuenta que se debe leer la distancia en kilómetros y el tiempo en horas. La fórmula es la siguiente:

$$V = \frac{d}{t}$$

La fórmula de la velocidad *(V)* requiere de la distancia *(d)* y el tiempo *(t)* para realizar su división. La solución sería la siguiente:

En la figura 41 se muestra el pseudocódigo en PSeInt y en la figura 42 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese distancia (km): "
    Leer d
    Imprimir "Ingrese tiempo (horas): "
    Leer t
    V = d / t
    Imprimir "La velocidad es: ", V, " km/h"
FinAlgoritmo
```

Ejecución

```
Ingrese distancia (km):
> 200
Ingrese tiempo (horas):
> 2.5
La velocidad es: 80 km/h
```

Figura 41. Cuarto ejemplo de fórmulas y ecuaciones en PSeInt.

main.py

```
print("Ingrese distancia (km): ")
d = float(input())
print("Ingrese tiempo (horas): ")
t = float(input())
v = d / t
print("La velocidad es: ", v," km/h")
```

```
Ingrese distancia (km):
200
Ingrese tiempo (horas):
2.5
La velocidad es:  80.0  km/h
```

Figura 42. Cuarto ejemplo de fórmulas y ecuaciones en Python.

Ejemplo 5. Desarrolle un algoritmo que lea el valor de una hora y el número de horas laboradas por un empleado e imprima su salario devengado.

Las variables necesarias para resolver este ejemplo son el valor de hora *(Vlr_Hora)* y el número de horas *(Num_Hora)*. En la figura 43 se muestra el pseudocódigo en PSeInt y en la figura 44 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite valor hora: "
    Leer Vlr_Hora
    Imprimir "Digite horas laboradas:"
    Leer Num_Hora
    Sal_Dev = Num_Hora * Vlr_Hora
    Imprimir "Salario devengado: $", Sal_Dev
FinAlgoritmo
```

Ejecución

```
Digite valor hora:
> 5500
Digite horas laboradas:
> 190
Salario devengado: $1045000
```

Figura 43. Quinto ejemplo de fórmulas y ecuaciones en PSeInt.

main.py

```
print("Digite valor hora: ")
Vlr_Hora = float(input())
print("Digite horas laboradas: ")
Num_Hora = float(input())
Sal_dev = Num_Hora * Vlr_Hora
print("Salario devengado: $", Sal_dev)
```

```
Digite valor hora:
5500
Digite horas laboradas:
190
Salario devengado: $ 1045000.0
```

Figura 44. Quinto ejemplo de fórmulas y ecuaciones en Python.

Ejemplo 6. Calcular e imprimir, a través de un algoritmo, el resultado de la ecuación *(R)*:

$$R = \sqrt[5]{\frac{8\%\ x}{0.4\%\ w} + 150\%\ z}$$

En la figura 45 se muestra el pseudocódigo en PSeInt y en la figura 46 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite valor de x: "
    Leer x
    Imprimir "Digite valor de w: "
    Leer w
    Imprimir "Digite valor de z: "
    Leer z
    R = (0.08 * x / (0.04 * w) + 1.5 * z) ^ 0.2
    Imprimir "Resultado de R = ", R
FinAlgoritmo
```

Ejecución

```
Digite valor de x:
> 5
Digite valor de w:
> 6
Digite valor de z:
> 7
Resultado de R = 1.6482926496
```

Figura 45. Sexto ejemplo de fórmulas y ecuaciones en PSeInt.

main.py

```
import math
print("Ingrese valor de X: ")
X = float(input())
print("Ingrese valor de W: ")
W = float(input())
print("Ingrese valor de Z: ")
Z = float(input())
R = math.pow(0.08 * X / (0.004 * W) + 1.5 * Z, 0.2)
print("Resultado de R = ", R)
```

```
Ingrese valor de X:
5
Ingrese valor de W:
6
Ingrese valor de Z:
7
Resultado de R =  1.935562818214166
```

Figura 46. Sexto ejemplo de fórmulas y ecuaciones en Python.

2.5 Porcentajes en algoritmos

Los porcentajes se encuentran en la cotidianidad de muchas personas, por ejemplo, un descuento en algún producto que se desea comprar, requerir el precio de un producto al que se le debe incluir el IVA, al liquidar una nómina

con múltiples porcentajes, determinar un porcentaje para representar los índices de inflación, entre otros.

Para calcular porcentajes dentro de un algoritmo existen varias formas, de las cuales se explicarán algunas, buscando que se entiendan sus diferencias y se logre identificar la eficiencia y las ventajas y desventajas. Lo más importante de este tema es recordar que para determinar el equivalente de un porcentaje se debe multiplicar un valor por respectivo el porcentaje. Por ejemplo, para determinar el valor un incremento del 25% de un valor de un artículo, se multiplica dicho valor por 0.25.

Ejemplo 1. A un empleado le aplican una retención del 18% sobre su salario básico y le entregan una bonificación del 1.3% del mismo salario. Desarrolle un algoritmo que permita calcular e imprimir el salario neto y los valores de sus respectivos porcentajes.

En la figura 47 se muestra el pseudocódigo en PSeInt y en la figura 48 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Digite salario: $"
3      Leer Sal_Bas
4      Reten = Sal_Bas * 0.18
5      Bonif = Sal_Bas * 0.013
6      Sal_Neto = Sal_Bas - Reten + Bonif
7      Imprimir "Retención: $", Reten
8      Imprimir "Bonificación: $", Bonif
9      Imprimir "Salario neto: $", Sal_Neto
10 FinAlgoritmo
```

Ejecución

```
Digite salario: $
> 800000
Retención: $144000
Bonificación: $10400
Salario neto: $666400
```

Figura 47. Primer ejemplo de porcentajes en PSeInt.

main.py

```
print("Digite salario: $")
Sal_Bas = float(input())
Reten = Sal_Bas * 0.18
Bonif = Sal_Bas * 0.013
Sal_Neto = Sal_Bas - Reten + Bonif
print("Retención: $", Reten)
print("Bonificación: $", Bonif)
print("Salario neto: $", Sal_Neto)
```

```
Digite salario: $
800000
Retención: $ 144000.0
Bonificación: $ 10400.0
Salario neto: $ 666400.0
```

Figura 48. Primer ejemplo de porcentajes en Python.

Ejemplo 2. La medida del ancho de una cancha de fútbol fue ampliada en un 20%. Determinar el área total en metros de esa cancha.

En la figura 49 se muestra el pseudocódigo en PSeInt y en la figura 50 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Ancho de la cancha(mt)?: "
    Leer Med_Can
    Ampl = Med_Can * 0.2
    Area_Tot = Med_Can + Ampl
    Imprimir "El área actual : ", Area_Tot, " mts"
FinAlgoritmo
```

Ejecución

```
¿Ancho de la cancha(mt)?:
> 80.5
El área actual : 96.6 mts
```

Figura 49. Segundo ejemplo de porcentajes en PSeInt.

main.py

```
print("¿Ancho de la cancha(mt)?: ")
Med_Can = float(input())
Ampl = Med_Can * 0.2
Area_Tot = Med_Can + Ampl
print("El área actual : ", Area_Tot," mts")
```

Ejecución

```
¿Ancho de la cancha(mt)?:
80.5
El área actual :  96.6  mts
```

Figura 50. Segundo ejemplo de porcentajes en Python.

Observar con cuidado el nombre de las variables que deben ser nemotécnicas y tener mucho cuidado con los signos de la retención y la bonificación. Es fundamental deducir el signo que debe ir antes de la variable, ya

que, si se coloca el equivocado, afectaría totalmente el resultado. Hay que recordar que 20% es igual a 0.2 (resultado de dividir 20 entre 100).

En estos dos ejemplos se calculan los valores de los porcentajes en cada línea diferente, luego se suman o restan para poder determinar el valor final. Se recomienda este uso en algoritmos que pidan mostrar los valores equivalentes de los porcentajes.

Ejemplo 3. Calcular el salario neto de un empleado, sabiendo que se le hace una retención del 30% de su salario básico.

En la figura 51 se muestra el pseudocódigo en PSeInt y en la figura 52 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1 Algoritmo Inicio
2     Imprimir "Digite salario: $"
3     Leer Sal_Bas
4     Sal_Neto = Sal_Bas - Sal_Bas * 0.3
5     Imprimir "Salario neto: $", Sal_Neto
6 FinAlgoritmo
```

Ejecución

```
Digite salario: $
> 800000
Salario neto: $560000
```

Figura 51. Tercer ejemplo de porcentajes en PSeInt.

main.py

```
1  print("Digite salario: $")
2  Sal_Bas = float(input())
3  Sal_Net = Sal_Bas-Sal_Bas * 0.3
4  print("Salario neto: $", Sal_Net)
```

```
Digite salario: $
800000
Salario neto: $ 560000.0
```

Figura 52. Tercer ejemplo de porcentajes en Python.

En estos dos últimos ejemplos se calculan los valores finales en una sola línea. Se recomienda su uso en algoritmos que solo pidan el valor final y no el valor de los porcentajes de forma separada.

Ejemplo 4. Calcular el salario neto de empleado, sabiendo que se le hace una retención del 45% de su salario básico.

En la figura 53 se muestra el pseudocódigo en PSeInt y en la figura 54 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite su salario: $"
    Leer Sal_Bas
    Sal_Neto = Sal_Bas * 0.55
    Imprimir "Salario Neto: $", Sal_Neto
FinAlgoritmo
```

Ejecución

```
Digite su salario: $
> 800000
Salario Neto: $440000
```

Figura 53. Cuarto ejemplo de porcentajes en PSeInt.

main.py

```
print("Digite su salario: $")
Sal_Bas = float(input())
Sal_Neto = Sal_Bas*0.55
print("Salario Neto: $",Sal_Neto)
```

```
Digite su salario: $
800000
Salario Neto: $ 440000
```

Figura 54. Cuarto ejemplo de porcentajes en Python.

Esta fórmula se enfoca en sumar y restar los porcentajes partiendo de un 100%. Lo que está en medio de los paréntesis se resuelve y solo se debe colocar el último resultado. Por ejemplo, por lógica, si al salario básico de un trabajador se hace una retención de un 45%, este recibe un equivalente al 55%, por lo tanto, se multiplica la variable por 0.55 (100%–45% = 55%).

Ejemplo 5. El valor del desempleo aumentó en el primer trimestre un 9.5% y en el segundo disminuyó en 1.5%. Calcular el valor del desempleo actual.

En la figura 55 se muestra el pseudocódigo en PSeInt y en la figura 56 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones. Al realizar las operaciones con los porcentajes el resultado sería: 100% + 9.5% – 1.5% = 108% = 1.08.

```
Algoritmo Inicio
    Imprimir "Digite valor desempleo: "
    Leer Val_Des
    Val_Des_Act = Val_Des * 1.08
    Imprimir "Valor desempleo actual: ", Val_Des_Act
FinAlgoritmo
```

Ejecución

```
Digite valor desempleo:
> 13.8
Valor desempleo actual: 14.904
```

Figura 55. Quinto ejemplo de porcentajes en PSeInt.

main.py

```
print("Digite valor desempleo: ")
Val_Des = float(input())
Val_Des_Act = Val_Des * 1.08
print("Valor desempleo actual: ", Val_Des_Act)
```

```
Digite valor desempleo:
13.8
Valor desempleo actual:  14.904
```

Figura 56. Quinto ejemplo de porcentajes en Python.

En estos dos últimos ejemplos se calculan los valores finales en una sola línea; pero realizando la operación con los porcentajes. Se recomienda su uso en algoritmos que solo pidan el valor final y no el valor de los porcentajes de forma separada. Es la forma más recursiva para calcular valores finales luego de aplicados los porcentajes.

Ejemplo 6. Calcular el área total de un terreno en metros sabiendo que esta fue reducida en un 10%, y posteriormente, le fue adicionado un 50% con relación al área después de la reducción.

En la figura 57 se muestra el pseudocódigo en PSeInt y en la figura 58 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite área terreno: "
    Leer Area_Ter
    Area_Red = Area_Ter - Area_Ter * 0.1
    Area_Act = Area_Red + Area_Red * 0.5
    Imprimir "Área actual: ", Area_Act, " mts"
FinAlgoritmo
```

Ejecución

```
Digite área terreno:
> 20
Área actual: 27 mts
```

Figura 57. Sexto ejemplo de porcentajes en PSeInt.

main.py

```
print("Digite área terreno: ")
Area_Ter = float(input())
Area_Red = Area_Ter - Area_Ter * 0.1
Area_Act = Area_Red + Area_Red * 0.5
print("Área actual: ", Area_Act," mts")
```

```
Digite área terreno:
20
Area actual:  27.0  mts
```

Figura 58. Sexto ejemplo de porcentajes en Python.

Este último ejemplo muestra un caso especial y muy común con los porcentajes, y es aquel donde el segundo porcentaje no se determina con

el valor inicial, sino del valor que queda después de aplicado el porcentaje. En este caso, el 50% que se adicionó no es de los 20 metros, sino de los 18 metros que quedan después de quitarle el 10%. 20 metros – 2 metros (equivalentes al 10%) = 18 metros. Ahora a esos 18 se les agrega el 50% que equivalente a 9 metros, quedando un total de 27 (18 + 9 = 27).

2.6 Conversión de unidades

Para la conversión de unidades solo se requieren manejar operaciones básicas de multiplicación y división, conocer los valores equivalentes entre las diferentes unidades e implementar el concepto de una regla de tres.

Por ejemplo, para hacer una conversión de un tiempo en horas a minutos, hay que saber la cantidad de minutos que tiene una hora, para responder la pregunta: ¿A cuántos minutos equivale 1 hora? y se implementa una regla de tres con esa información.

Ejemplo 1. Hacer un algoritmo que permita determinar las horas a las que equivale una cantidad de minutos ingresado por el usuario.

En la figura 59 se muestra el pseudocódigo en PSeInt y en la figura 60 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1 Algoritmo Inicio
2     Imprimir "¿Cantidad de minutos?: "
3     Leer Cant_Min
4     Cant_Hor = Cant_Min / 60
5     Imprimir "Cantidad horas = ", Cant_Hor
6 FinAlgoritmo
```

Ejecución

```
¿Cantidad de minutos?:
> 210
Cantidad horas = 3.5
```

Figura 59. Primer ejemplo de conversión de unidades en PSeInt.

main.py

```
1  print("¿Cantidad de minutos?: ")
2  Cant_Min = float(input())
3  Cant_Hor = Cant_Min / 60
4  print("Cantidad horas = ", Cant_Hor)
```

```
¿Cantidad de minutos?:
210
Cantidad horas =  3.5
```

Figura 60. Primer ejemplo de conversión de unidades en Python.

Ejemplo 2. Hacer un algoritmo que permita determinar la cantidad de minutos a los que equivale una cantidad de horas ingresadas por el usuario.

En la figura 61 se muestra el pseudocódigo en PSeInt y en la figura 62 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Cantidad de horas?: "
    Leer Cant_Horas
    Cant_Min = Cant_Horas * 60
    Imprimir "Cantidad minutos = ", Cant_Min
FinAlgoritmo
```

Ejecución

```
¿Cantidad de horas?:
> 3.5
Cantidad minutos = 210
```

Figura 61. Segundo ejemplo de conversión de unidades en PSeInt.

main.py

```
print("¿Cantidad de horas?: ")
Cant_Horas = float(input())
Cant_Min = Cant_Horas * 60
print("Cantidad minutos = ", Cant_Min)
```

```
¿Cantidad de horas?:
3.5
Cantidad minutos =  210.0
```

Figura 62. Segundo ejemplo de conversión de unidades en Python.

Ejemplo 3. Desarrollar un algoritmo que permita convertir de quintal a kilogramos, tenga en cuenta que un 1 quintal equivale a 100 kilogramos.

En la figura 63 se muestra el pseudocódigo en PSeInt y en la figura 64 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Cantidad de quintales?: "
    Leer Cant_Quintal
    Cant_Kg = Cant_Quintal * 100
    Imprimir "Cantidad kilogramos = ", Cant_Kg
FinAlgoritmo
```

Ejecución

```
¿Cantidad de quintales?:
> 6.5
Cantidad kilogramos = 650
```

Figura 63. Tercer ejemplo de conversión de unidades en PSeInt.

main.py

```
print("¿Cantidad de quintales?: ")
Cant_Quintal = float(input())
Cant_Kg = Cant_Quintal * 100
print("Cantidad kilogramos = ", Cant_Kg)
```

```
¿Cantidad de quintales?:
6.5
Cantidad kilogramos =  650.0
```

Figura 64. Tercer ejemplo de conversión de unidades en Python.

Ejemplo 4. Se pide realizar un algoritmo que lea una cantidad en metros y determine su equivalente en centímetros y en kilómetros. Tenga en cuenta que 1 metro tiene 100 centímetros y 1 kilómetro tiene 1000 metros.

En la figura 65 se muestra el pseudocódigo en PSeInt y en la figura 66 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Cantidad de metros?: "
    Leer Cant_M
    Cant_Cm = Cant_M * 100
    Cant_Km = Cant_M / 1000
    Imprimir Cant_M, " m equivalen a ", Cant_Cm, " cm"
    Imprimir Cant_M, " m equivalen a ", Cant_Km, " km"
FinAlgoritmo
```

Ejecución

```
¿Cantidad de metros?:
> 8500
8500 m equivalen a 850000 cm
8500 m equivalen a 8.5 km
```

Figura 65. Cuarto ejemplo de conversión de unidades en PSeInt.

main.py

```
print("¿Cantidad de metros?: ")
Cant_M = float(input())
Cant_Cm = Cant_M * 100
Cant_Km = Cant_M / 1000
print(Cant_M,"m equivalen a ", Cant_Cm,"cm")
print(Cant_M,"m equivalen a ", Cant_Km,"km")
```

```
¿Cantidad de metros?:
8500
8500.0 m equivalen a  850000.0 cm
8500.0 m equivalen a  8.5 km
```

Figura 66. Cuarto ejemplo de conversión de unidades en Python.

Observe que mientras en la primera conversión se multiplicó en la segunda se dividió. Todo es cuestión de lógica y hacer operaciones con valores asignadas a las variables. Observar esta forma de *imprimir* tan completa y diferente a otros *imprimir* usados anteriormente.

Ejemplo 5. Diseñar un algoritmo que lea una distancia en millas y la convierta a kilómetros y metros. Tenga en cuenta que una milla tiene 1609.344 metros y 1.609344 kilómetros.

En la figura 67 se muestra el pseudocódigo en PSeInt y en la figura 68 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Cantidad de millas?: "
    Leer Cant_Millas
    Cant_Km = Cant_Millas * 1.609344
    Cant_M = Cant_Millas * 1609.344
    Imprimir "Cantidad kilometros = ", Cant_Km
    Imprimir "Cantidad metros = ", Cant_M
FinAlgoritmo
```

Ejecución

```
¿Cantidad de millas?:
> 3.8
Cantidad kilometros = 6.1155072
Cantidad metros = 6115.5072
```

Figura 67. Quinto ejemplo de conversión de unidades en PSeInt.

main.py

```
print("¿Cantidad de millas?: ")
Cant_Millas = float(input())
Cant_Km = Cant_Millas * 1.609344
Cant_M = Cant_Millas * 1609.344
print("Cantidad kilometros = ", Cant_Km)
print("Cantidad metros = ", Cant_M)
```

```
¿Cantidad de millas?:
3.8
Cantidad kilometros =  6.1155072
Cantidad metros =  6115.5072
```

Figura 68. Quinto ejemplo de conversión de unidades en Python.

Ejemplo 6. Se requiere de una solución en la cual se lea una distancia en metros y la convierta a millas. Tenga en cuenta que una milla tiene 1609.344 metros.

En la figura 69 se muestra el pseudocódigo en PSeInt y en la figura 70 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Cantidad de metros?: "
    Leer Cant_M
    Cant_Millas = Cant_M / 1609.344
    Imprimir "Cantidad millas = ", Cant_Millas
FinAlgoritmo
```

Ejecución

```
¿Cantidad de metros?:
> 6115.5
Cantidad millas = 3.799995
```

Figura 69. Sexto ejemplo de conversión de unidades en PSeInt.

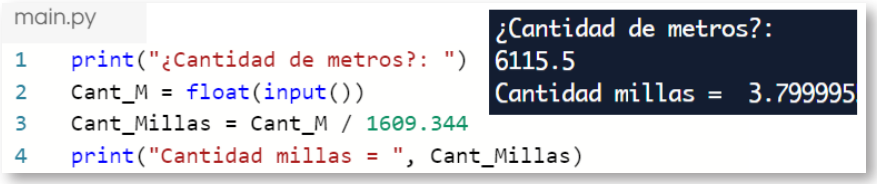


Figura 70. Sexto ejemplo de conversión de unidades en Python.

Luego de trabajar cada uno de los ejemplos, puede notar que Python es un lenguaje simple, sencillo y eficiente. En la actualidad, es uno de los lenguajes de mayor demanda; debido a que es potente y una excelente herramienta para el análisis de datos (*big data*) al combinarse con R y Hadoop.

2.7 Prueba de escritorio

Una prueba de escritorio consiste en seguir paso a paso lo que hace el algoritmo. No existe una representación formal de la prueba de escritorio, se pueden usar casillas para las variables y sus valores o se puede utilizar una columna (López y Gutiérrez, 2014).

Para hacer una prueba de escritorio se debe tener en cuenta el ingreso de datos, los cálculos y la respectiva salida. Mirar los siguientes pasos:

1. Colocar de forma horizontal todas las variables usadas en el algoritmo y agregar una columna para la instrucción *Imprimir*. Estas se ponen en forma de encabezado.
2. Hacer un seguimiento al algoritmo paso por paso, desde la primera línea hasta la última.
3. Asignar cualquier valor cuando se solicite un dato por medio de las palabras *Leer*, *Ingresar* o *Digitar.*
4. Asignar los respectivos valores cuando se encuentre con un operador igual.
5. Imprimir los valores de las variables que tienen los resultados solicitados.

Si los resultados arrojados por la prueba de escritorio son los esperados, el algoritmo se considera que está correcto, de lo contrario se deben hacer las respectivas correcciones.

Ejemplo 1. Para explicar esta prueba de escritorio se toma el *Ejemplo 2* de manejo de porcentajes en algoritmos. La medida del ancho de una cancha de fútbol fue ampliada en un 20%. Determinar el área total en metros de esa cancha.

1. Algoritmo Inicio
2. Imprimir "¿Ancho de la cancha (mt)?: "
3. Leer Med_Can
4. Ampl = Med_Can * 0.2
5. Area_Tot = Med_Can + Ampl
6. Imprimir "El área actual: ", Area_Tot, " mts"
7. FinAlgoritmo

Paso 1: colocar de forma horizontal (en forma de encabezado) todas las variables usadas en el algoritmo y agregar una columna para la instrucción de *Imprimir*. En este caso, se agrega una columna al lado izquierdo para indicar el número de la línea en la cual se ejecuta la instrucción y cuatro columnas más: una para la variable *Med_Can* (medidas del ancho de la cancha), otra para la variable *Ampl* (ampliación de la cancha), otra para la variable *Area_Tot* (área total de la cancha) y una última para la instrucción *Imprimir*. El resultado se presenta en la tabla 17.

Tabla 17.
Primer ejemplo de prueba de escritorio. Paso 1.

# Línea	Med_Can	Ampl	Area_Tot	Imprimir
1				

Paso 2: hacer un seguimiento al algoritmo paso por paso, desde la primera línea hasta la última. En este caso, en la *Línea 1*: el encabezado *Algoritmo Inicio*, no realiza ninguna instrucción. La *Línea 2* muestra un mensaje que le indica al usuario que ingrese el ancho de la cancha que se requiere para dar solución al ejercicio planteado. El resultado se presenta en la tabla 18.

Tabla 18.
Primer ejemplo de prueba de escritorio. Paso 2.

# Línea	Med_Can	Ampl	Area_Tot	Imprimir
1				
2				**¿Ancho de la cancha (m)?:**

Paso 3: asignar cualquier valor cuando se solicite un dato por medio de las palabras *Leer*, *Ingresar* o *Digitar*. En este caso, en *Línea 3* se lee el valor de 80.5 metros, el cual se almacena en la variable *Med_Can.* El resultado se presenta en la tabla 19.

Tabla 19.
Primer ejemplo de prueba de escritorio. Paso 3.

# Línea	Med_Can	Ampl	Area_Tot	Imprimir
1				
2				¿Ancho de la cancha (m)?:
3	**80.5**			

Paso 4: asignar los respectivos valores cuando se encuentre con un operador igual. En este caso, en la *Línea 4*, se asigna 16.1 a la variable *Ampl*, el cual es el resultado equivalente al 20% de 80.5 que es el ancho de la cancha; mientras que en la línea 5, se asigna 96.6 a la variable *Area_Tot*, el cual corresponde al área actual de la cancha. El resultado se presenta en la tabla 20.

Tabla 20.
Primer ejemplo de prueba de escritorio. Paso 4.

# Línea	Med_Can	Ampl	Area_Tot	Imprimir
1				
2				¿Ancho de la cancha (m)?:
3	80.5			
4		**16.1**		
5			**96.6**	

Paso 5: imprimir los valores de las variables que tienen los resultados solicitados. En este caso, en la *Línea 6* se imprime un mensaje con el área actual de la cancha acompañado del resultado del área total en metros: *96.6*. El resultado se presenta en la tabla 21.

Tabla 21.
Primer ejemplo de prueba de escritorio. Paso 4.

# Línea	Med_Can	Ampl	Area_Tot	Imprimir
1				
2				¿Ancho de la cancha (m)?:
3	80.5			
4		16.1		
5			96.6	
6				**El área actual: 96.6 m**

La *Línea 7* no realiza ninguna instrucción, solo indica que se finaliza la ejecución del algoritmo. El resultado final de la prueba de escritorio se muestra en la figura 71.

```
1. Algoritmo Inicio
2.     Imprimir "¿Ancho de la cancha (m)?: "
3.     Leer Med_Can
4.     Ampl = Med  Can * 0.2
5.     Area_Tot = Med_Can + Ampl
6.     Imprimir "El área actual: ", Area_Tot, " m"
7. FinAlgoritmo
```

# Línea	Med_Can	Ampl	Area_Tot	Imprimir
1				
2				¿Ancho de la cancha (m)?:
3	80.5			
4		16.1		
5			96.6	
6				El área actual: 96.6 m
7				

Figura 71. Primer ejemplo de prueba de escritorio. Resultado final.

Comparando con la solución realizada en el lenguaje de programación Python *(*figura 50), y al ver que los resultados son los esperados de acuerdo con los valores ingresados, se puede decir que el algoritmo está correcto, aunque se recomienda hacer la prueba de escritorio por lo menos tres veces para mayor seguridad en la respuesta.

Ejemplo 2. Para explicar esta prueba de escritorio se toma el *Ejemplo 6* del tema manejo de porcentajes en algoritmos. Calcular el área total de un terreno en metros sabiendo que esta fue reducida en un 10%, y posteriormente, le fue adicionado un 50% con relación al área después de la reducción.

```
1. Algoritmo Inicio
2.     Imprimir "Digite área terreno:"
3.     Leer Area_Ter
4.     Area_Red = Area_Ter – Area_Ter * 0.1
5.     Area_Act = Area_Red + Area_Red * 0.5
6.     Imprimir "Área actual: ", Area_Act, " mts"
7. FinAlgoritmo
```

Paso 1: colocar de forma horizontal (en forma de encabezado) todas las variables usadas en el algoritmo y agregar una columna para la instrucción de *Imprimir*. En este caso, se agrega una columna al lado izquierdo para indicar el número de la línea en la cual se ejecuta la instrucción y cuatro columnas más: una para la variable *Area_Ter* **(área del terreno), otra para la variable** *Area_Red* **(área reducida), otra para la variable** *Area_Act* **(área actual del** terreno) y una última para la instrucción *Imprimir*. El resultado se presenta en la tabla 22.

Tabla 22.
Segundo ejemplo de prueba de escritorio. Paso 1

# Línea	Area_Ter	Area_Red	Area_Act	Imprimir
1				

Paso 2: hacer un seguimiento al algoritmo paso por paso, desde la primera línea hasta la última. En este caso, en la *Línea 1*: el encabezado *Algoritmo Inicio*, no realiza ninguna instrucción. La *Línea 2* muestra un mensaje que le indica al usuario que ingrese el área del terreno que se requiere para dar solución al ejercicio planteado. El resultado se presenta en la tabla 23.

Tabla 23.
Segundo ejemplo de prueba de escritorio. Paso 2.

# Línea	Area_Ter	Area_Red	Area_Act	Imprimir
1				
2				**Digite área del terreno:**

Paso 3: asignar cualquier valor cuando se solicite un dato por medio de las palabras *Leer*, *Ingresar* o *Digitar*. En este caso, en *Línea 3* se lee el valor de 20 metros que es el área del terreno inicial, el cual se almacena en la variable *Area_Ter.* El resultado se presenta en la tabla 24.

Tabla 24.
Segundo ejemplo de prueba de escritorio. Paso 3.

# Línea	Area_Ter	Area_Red	Area_Act	Imprimir
1				
2				Digite área del terreno:
3	**20**			

Paso 4: asignar los respectivos valores cuando se encuentre con un operador igual. En este caso, en la *Línea 4*, se asigna 2 a la variable *Area_Red*, el cual es el resultado equivalente al 10% de 20 que es el área del terreno; mientras que en la línea 5, se asigna 27 a la variable *Area_Act*, el cual corresponde al área actual del terreno, luego del incremento del 50% sobre los 18 metros que habían quedado (18 + 9 = 27). El resultado se presenta en la tabla 25.

Tabla 25.
Segundo ejemplo de prueba de escritorio. Paso 4.

# Línea	Area_Ter	Area_Red	Area_Act	Imprimir
1				
2				Digite área del terreno:
3	20			
4		**2**		
5			**27**	

Paso 5: imprimir los valores de las variables que tienen los resultados solicitados. En este caso, en la *Línea 6* se imprime un mensaje con el Área actual del terreno acompañado de su resultado en metros: *27*. El resultado se presenta en la tabla 26.

Tabla 26.
Segundo ejemplo de prueba de escritorio. Paso 5.

# Línea	Area_Ter	Area_Red	Area_Act	Imprimir
1				
2				Digite área del terreno:
3	20			
4		2		
5			27	
6				El área actual: 27 m
7				

La *Línea 7* no realiza ninguna instrucción, solo indica que se finaliza la ejecución del algoritmo. El resultado final de la prueba de escritorio se muestra en la figura 72.

```
1. Algoritmo Inicio
2.     Imprimir "Digite área terreno: "
3.     Leer Area_Ter
4.     Area_Red = Area_Ter - Area_Ter * 0.1
5.     Area_Act = Area_Red + Area_Red * 0.5
6.     Imprimir "Área actual: ", Area_Act, " m"
7. FinAlgoritmo
```

# Línea	Area_Ter	Area_Red	Area_Act	Imprimir
1				
2				Digite área del terreno:
3	20			
4		2		
5			27	
6				**El área actual: 27 m**
7				

Figura 72. Segundo ejemplo de prueba de escritorio. Resultado final.

Comparando con la solución realizada en el lenguaje de programación Python (figura 58), y al ver que los resultados son los esperados de acuerdo con los valores ingresados, se puede decir que el algoritmo está correcto.

Ejemplo 3. Para explicar esta prueba de escritorio se toma el *Ejemplo 3* del tema de conversión de unidades. Desarrollar un algoritmo que permita convertir de quintal a kilogramos, tenga en cuenta que un 1 quintal equivale a 100 kilogramos.

1. Algoritmo Inicio
2. Imprimir "¿Cantidad de quintales?: "
3. Leer Cant_Quintal
4. Cant_Kg = Cant_Quintal * 100
5. Imprimir "Cantidad kilogramos = ", Cant_Kg
6. FinAlgoritmo

Paso 1: colocar de forma horizontal (en forma de encabezado) todas las variables usadas en el algoritmo y agregar una columna para la instrucción de *Imprimir*. En este caso, se agrega una columna al lado izquierdo para indicar el número de la línea en la cual se ejecuta la instrucción y tres columnas más: una para la variable *Cant_Quintal* (cantidad de quintales), otra para la variable *Cant_Kg* (cantidad de kilogramos) y una última para la instrucción *Imprimir*. El resultado se presenta en la tabla 27.

Tabla 27.
Tercer ejemplo de prueba de escritorio. Paso 1.

# Línea	Cant_Quintal	Cant_Kg	Imprimir
1			

Paso 2: hacer un seguimiento al algoritmo paso por paso, desde la primera línea hasta la última. En este caso, en la *Línea 1*: el encabezado *Algoritmo Inicio*, no realiza ninguna instrucción. La *Línea 2* muestra un mensaje que le indica al usuario que ingrese la cantidad de quintales que se requieren para dar solución al ejercicio planteado. El resultado se presenta en la tabla 28.

Tabla 28.
Tercer ejemplo de prueba de escritorio. Paso 2.

# Línea	Cant_Quintal	Cant_Kg	Imprimir
1			
2			**¿Cantidad de quintales?:**

Paso 3: asignar cualquier valor cuando se solicite un dato por medio de las palabras *Leer, Ingresar* o *Digitar*. En este caso, en *Línea 3* se lee el valor de 6.5 quintales, el cual se almacena en la variable *Cant_Quintal.* El resultado se presenta en la tabla 29.

Tabla 29.
Tercer ejemplo de prueba de escritorio. Paso 3.

# Línea	Cant_Quintal	Cant_Kg	Imprimir
1			
2			¿Cantidad de quintales?:
3	**6.5**		

Paso 4: asignar los respectivos valores cuando se encuentre con un operador igual. En este caso, en la *Línea 4*, se asigna 650 a la variable *Cant_Kg*, el cual es el resultado de multiplicar 6.5 quintales por 100 que es su equivalente. El resultado se presenta en la tabla 30.

Tabla 30.
Tercer ejemplo de prueba de escritorio. Paso 4.

# Línea	Cant_Quintal	Cant_Kg	Imprimir
1			
2			¿Cantidad de quintales?:
3	6.5		
4		**650**	

Paso 5: imprimir los valores de las variables que tienen los resultados solicitados. En este caso, en la *Línea 5* se imprime un mensaje con la *Cantidad de kilogramos* acompañado del resultado de la multiplicación: *650*. El resultado se presenta en la tabla 31.

Tabla 31.
Tercer ejemplo de prueba de escritorio. Paso 5.

# Línea	Cant_Quintal	Cant_Kg	Imprimir
1			
2			¿Cantidad de quintales?:
3	6.5		
4		650	
5			Cantidad kilogramos = 650

La *Línea 7* no realiza ninguna instrucción, solo indica que se finaliza la ejecución del algoritmo. El resultado final de la prueba de escritorio se muestra en la figura 73.

```
1. Algoritmo Inicio
2.     Imprimir "¿Cantidad de quintales?: "
3.     Leer Cant_Quintal
4.     Cant_Kg = Cant_Quintal * 100
5.     Imprimir "Cantidad kilogramos = ", Cant_Kg
6. FinAlgoritmo
```

# Línea	Cant_Quintal	Cant_Kg	Imprimir
1			
2			¿Cantidad de quintales?:
3	6.5		
4		650	
5			Cantidad kilogramos = 650
6			

Figura 73. Tercer ejemplo de prueba de escritorio. Resultado final.

Comparando con la solución realizada en el lenguaje de programación Python (figura 64), y al ver que los resultados son los esperados de acuerdo con los valores ingresados, se puede decir que el algoritmo está correcto.

Ejemplo 4. Para explicar esta prueba de escritorio se toma el *Ejemplo 6* del tema de conversión de unidades. Se requiere de una solución en la cual se lea una distancia en metros y la convierta a millas. Tenga en cuenta que una milla tiene 1609.344 metros.

```
1. Algoritmo Inicio
2.     Imprimir "¿Cantidad de metros?: "
3.     Leer Cant_M
4.     Cant_Millas = Cant_M / 1609.344
5.     Imprimir "Cantidad millas = ", Cant_Millas
6. FinAlgoritmo
```

Paso 1: colocar de forma horizontal (en forma de encabezado) todas las variables usadas en el algoritmo y agregar una columna para la instrucción de *Imprimir*. En este caso, se agrega una columna al lado izquierdo para indicar el número de la línea en la cual se ejecuta la instrucción y tres columnas más: una para la variable *Cant_M* (cantidad de metros), otra para la variable *Cant_Millas* (cantidad de millas) y una última para la instrucción *Imprimir*. El resultado se presenta en la tabla 32.

Tabla 32.
Cuarto ejemplo de prueba de escritorio. Paso 1.

# Línea	Cant_M	Cant_Millas	Imprimir
1			

Paso 2: hacer un seguimiento al algoritmo paso por paso, desde la primera línea hasta la última. En este caso, en la *Línea 1*: el encabezado *Algoritmo Inicio*, no realiza ninguna instrucción. La *Línea 2* muestra un mensaje que le indica al usuario que ingrese la cantidad de metros que se requieren para dar solución al ejercicio planteado. El resultado se presenta en la tabla 33.

Tabla 33.
Cuarto ejemplo de prueba de escritorio. Paso 2.

Línea	Cant_M	Cant_Millas	Imprimir
1			
2			¿Cantidad de metros?:

Paso 3: asignar cualquier valor cuando se solicite un dato por medio de las palabras *Leer, Ingresar* o *Digitar*. En este caso, en *Línea 3* se lee el valor

de 6115.5 metros, el cual se almacena en la variable *Cant_M.* El resultado se presenta en la tabla 34.

Tabla 34.
Cuarto ejemplo de prueba de escritorio. Paso 3.

# Línea	Cant_M	Cant_Millas	Imprimir
1			
2			¿Cantidad de metros?:
3	**6115.5**		

Paso 4: asignar los respectivos valores cuando se encuentre con un operador igual. En este caso, en la *Línea 4*, se asigna 3.799995 a la variable *Cant_Millas*, el cual es el resultado de dividir 6115.5 entre 1609.344 que es su equivalente. El resultado se presenta en la tabla 35.

Tabla 35.
Cuarto ejemplo de prueba de escritorio. Paso 4.

# Línea	Cant_M	Cant_Millas	Imprimir
1			
2			"¿Cantidad de metros?: "
3	6115.5		
4		3.799995	

Paso 5: imprimir los valores de las variables que tienen los resultados solicitados. En este caso, en la *Línea 5* se imprime un mensaje con la *Cantidad de millas* acompañado del resultado de la división: *3.799995*. El resultado se presenta en la tabla 36.

Tabla 36.
Cuarto ejemplo de prueba de escritorio. Paso 5.

# Línea	Cant_M	Cant_Millas	Imprimir
1			
2			¿Cantidad de metros?:
3	6115.5		
4		3.799995	
5			**Cantidad millas = 3.799995**

La *Línea 7* no realiza ninguna instrucción, solo indica que se finaliza la ejecución del algoritmo. El resultado final de la prueba de escritorio se muestra en la figura 74.

Comparando con la solución realizada en el lenguaje de programación Python (figura 70), y al ver que los resultados son los esperados de acuerdo con los valores ingresados, se puede decir que el algoritmo está correcto.

```
1. Algoritmo Inicio
2.     Imprimir "¿Cantidad de metros?: "
3.     Leer Cant_M
4.     Cant_Millas = Cant_M / 1609.344
5.     Imprimir "Cantidad millas = ", Cant_Millas
6. FinAlgoritmo
```

# Línea	Cant_M	Cant_Millas	Imprimir
1			
2			¿Cantidad de metros?:
3	6115.5		
4		3.799995	
5			Cantidad millas = 3.799995
6			

Figura 74. Cuarto ejemplo de prueba de escritorio. Resultado final.

2.8 Ejercicios resueltos

Cuando se desarrolla un algoritmo es fundamental hacer un preanálisis antes de empezar. Este proceso consiste en hacer un listado detallado de las variables de entrada, el proceso a realizar, las variables de salida y el nombre de cada una de las variables. En el preanálisis no hay mucha exigencia en cuanto al nombramiento de las variables; teniendo en cuenta que no se hace mucho énfasis en la sintaxis, sino en la lógica para solucionar el problema. Ver este preanálisis en los dos primeros ejercicios.

2.8.1 Primer ejercicio

Un cliente de telefonía celular realiza cuatro llamadas: dos a un primer operador y dos a un segundo. El cliente desea conocer el costo de cada llamada, el costo total de las llamadas a cada operador y el total de las cuatro llamadas. Tenga en cuenta que se debe leer el número de minutos de cada una de las llamadas (ingresar un número entero) y el valor por minuto a cada operador.

Explicación de las variables a usar:

- Minutos primera llamada primer operador: *M1_Op1.*
- Minutos segunda llamada primer operador: *M2_Op1.*
- Minutos tercera llamada segundo operador: *M1_Op2.*
- Minutos cuarta llamada segundo operador: *M2_Op2.*
- Valor minutos primer operador: *Val_Op1.*
- Valor minutos segundo operador: *Val_Op2.*
- Costo de la primera llamada: *Cto_Llama1*.
- Costo de la segunda llamada: *Cto_Llama2.*
- Costo de la tercera llamada: *Cto_Llama3*.
- Costo de la cuarta llamada: *Cto_Llama4.*
- Costo total de llamadas al primer operador: *Cto_Tot_Op1.*
- Costo total de llamadas al segundo operador: *Cto_Tot_Op2.*
- Costo total de las cuatro llamadas: *Tot_Llamadas.*

Datos de entrada:

- M1_Op1, M2_Op1, M1_Op2, M2_Op2.
- Val_Op1, Val_Op2.

Datos de salida:

- Cto_Llama1, Cto_Llama2, Cto_Llama3, Cto_Llama4.
- Cto_Tot_Op1, Cto_Tot_Op2, Tot_Llamadas.

Proceso:

- Para calcular el costo de cada llamada hay que tomar el número de minutos al operador y multiplicarlo por el valor del minuto a dicho operador: Valor_Llamada = Minutos × Valor_Minuto; pero discriminado cada cálculo: Cto_Llama1 = M1_Op1 * Val_Op1, Cto_Llama2 = M2_Op1 * Val_Op1, Cto_Llama3 = M1_Op2 * Val_Op2 y Cto_Llama4 = M2_Op2 * Val_Op2.
- Para calcular el costo a cada operador, se suma el costo de la primera llamada con el costo de la segunda llamada para el primer operador, y luego los costos de la tercera y cuarta llamada se suman para el segundo operador. Cto_Tot_Op1 = Cto_Llama1 + Cto_Llama2 y Cto_Tot_Op2 = Cto_Llama3 + Cto_Llama4.
- El valor de todas las llamadas se calcula sumando el costo de llamadas de los dos operadores. Tot_Llamadas = Cto_Tot_Op1 + Cto_Tot_Op2.

La verificación de un algoritmo puede ser sencilla si se aplica a casos de la vida cotidiana. Siempre se debe verificar que se suministre una respuesta a cada uno de los requerimientos con valores reales y correctos.

Por ejemplo, en este caso, una persona hace dos llamadas de 8 y 10 minutos a un primer operador con un costo de $ 100 por minuto, luego realiza otras dos llamadas de 2 y 4 minutos a un segundo operador con un costo de $ 150 por minuto. El costo de cada llamada, por separado es: $ 800 (8 minutos × $ 100), $ 1000 (10 minutos × $ 100), $ 300 (2 minutos × $ 150) y $ 600 (4 minutos × $ 150). El costo del primer operador es de $ 1800 (suma de las dos primeras llamadas) y el costo del segundo operador es de $ 900 (suma de las dos últimas llamadas). Finalmente, se tiene que pagar $ 2700 (la suma de los dos operadores $1.800 + $ 900 o la suma de todas las cuatro llamadas que es lo mismo $ 800 + $ 1000 + $ 300 + $ 600).

En la figura 75 se muestra el pseudocódigo en PSeInt y en la figura 76 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite duración en minutos primera llamada operador 1: "
    Leer M1_Op1
    Imprimir "Digite duración en minutos segunda llamada operador 1: "
    Leer M2_Op1
    Imprimir "Digite duración en minutos primera llamada operador 2: "
    Leer M1_Op2
    Imprimir "Digite duración en minutos segunda llamada operador 2: "
    Leer M2_Op2
    Imprimir "Digite valor minutos al operador 1: $"
    Leer Val_Op1
    Imprimir "Digite valor minutos al operador 2: $"
    Leer Val_Op2
    Cto_Llama1 = M1_Op1 * Val_Op1
    Cto_Llama2 = M2_Op1 * Val_Op1
    Cto_Llama3 = M1_Op2 * Val_Op2
    Cto_Llama4 = M2_Op2 * Val_Op2
    Cto_Tot_Op1 = Cto_Llama1 + Cto_Llama2
    Cto_Tot_Op2 = Cto_Llama3 + Cto_Llama4
    Tot_Llamadas = Cto_Tot_Op1 + Cto_Tot_Op2
    Imprimir "Costo llamada 1: $", Cto_Llama1
    Imprimir "Costo llamada 2: $", Cto_Llama2
    Imprimir "Costo llamada 3: $", Cto_Llama3
    Imprimir "Costo llamada 4: $", Cto_Llama4
    Imprimir "Costo llamada operador1: $", Cto_Tot_Op1
    Imprimir "Costo llamada operador2: $", Cto_Tot_Op2
    Imprimir "Costo de todas las llamadas: $", Tot_Llamadas
FinAlgoritmo
```

Ejecución

```
Digite duración en minutos primera llamada operador 1:> 8
Digite duración en minutos segunda llamada operador 1:> 10
Digite duración en minutos primera llamada operador 2:> 2
Digite duración en minutos segunda llamada operador 2:> 4
Digite valor minutos al operador 1: $ > 100
Digite valor minutos al operador 2: $ > 150
Costo llamada 1: $800
Costo llamada 2: $1000
Costo llamada 3: $300
Costo llamada 4: $600
Costo llamada operador1: $1800
Costo llamada operador2: $900
Costo de todas las llamadas: $2700
```

Figura 75. Primer ejercicio resuelto en PSeInt.

main.py

```
print("Digite duración en minutos primera llamada operador 1: ")
m1_op1 = float(input())
print("Digite duración en minutos segunda llamada operador 1: ")
m2_op1 = float(input())
print("Digite duración en minutos primera llamada operador 2: ")
m1_op2 = float(input())
print("Digite duración en minutos segunda llamada operador 2: ")
m2_op2 = float(input())
print("Digite valor minutos al operador 1: $")
val_op1 = float(input())
print("Digite valor minutos al operador 2: $")
val_op2 = float(input())
cto_llama1 = m1_op1*val_op1
cto_llama2 = m2_op1*val_op1
cto_llama3 = m1_op2*val_op2
cto_llama4 = m2_op2*val_op2
cto_tot_op1 = cto_llama1+cto_llama2
cto_tot_op2 = cto_llama3+cto_llama4
tot_llamadas = cto_tot_op1+cto_tot_op2
print("Costo llamada 1: $",cto_llama1)
print("Costo llamada 2: $",cto_llama2)
print("Costo llamada 3: $",cto_llama3)
print("Costo llamada 4: $",cto_llama4)
print("Costo llamada operador1: $",cto_tot_op1)
print("Costo llamada operador2: $",cto_tot_op2)
print("Costo de todas las llamadas: $",tot_llamadas)
```

```
Digite duración en minutos primera llamada operador 1:
8
Digite duración en minutos segunda llamada operador 1:
10
Digite duración en minutos primera llamada operador 2:
2
Digite duración en minutos segunda llamada operador 2:
4
Digite valor minutos al operador 1: $
100
Digite valor minutos al operador 2: $
150
Costo llamada 1: $ 800.0
Costo llamada 2: $ 1000.0
Costo llamada 3: $ 300.0
Costo llamada 4: $ 600.0
Costo llamada operador1: $ 1800.0
Costo llamada operador2: $ 900.0
Costo de todas las llamadas: $ 2700.0
```

Figura 76. Ejecución del primer ejercicio resuelto en Python.

2.8.2 Segundo ejercicio

Una computadora realiza las siguientes acciones: solicita tres números al usuario y muestra la suma de estos números, luego muestra la división de la suma de los dos primeros entre la resta de los dos últimos y, finalmente, muestre la multiplicación de los resultados de la suma y la división. Diseñar un algoritmo que permita realizar estas acciones y muestre los resultados.

Explicación de las variables a usar:

- Primer número: *Num1.*
- Segundo número: *Num2.*
- Tercer número: *Num3.*
- Suma de los tres números: *Suma.*
- División de la suma y la resta: *Div.*
- El producto de la suma y la división: *Prod.*

Datos de entrada:

- Num1, Num2, Num3.

Datos de Salida

- Suma, Div, Prod.

Proceso:

- Para sumar los tres números se realiza la adición: Suma = Num1 + Num2 + Num3.
- La división de la suma entre la resta se calcula: Div = $\frac{Num1 + Num2}{Num2 - Num3}$
- El producto de la suma y la división se calcula: Prod = Suma × Div.
- Cada cálculo debe almacenarse en una variable.

En la figura 77 se muestra el pseudocódigo en PSeInt y en la figura 78 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite # 1: "
    Leer Num1
    Imprimir "Digite # 2: "
    Leer Num2
    Imprimir "Digite # 3: "
    Leer Num3
    Suma = Num1 + Num2 + Num3
    Div = (Num1 + Num2) / (Num2 - Num3)
    Prod = Suma * Div
    Imprimir "La suma es: ", Suma
    Imprimir "La división es: ", Div
    Imprimir "El producto es: ", Prod
FinAlgoritmo
```

Ejecución

```
Digite # 1:
> 18
Digite # 2:
> 14
Digite # 3:
> 100
La suma es: 132
La división es: -0.3720930233
El producto es: -49.1162790698
```

Figura 77. Segundo ejercicio resuelto en PSeInt.

main.py

```
print("Digite # 1: ")
num1 = float(input())
print("Digite # 2: ")
num2 = float(input())
print("Digite # 3: ")
num3 = float(input())
suma = num1 + num2 + num3
div = (num1+num2)/(num2-num3)
prod = suma * div
print("La suma es: ", suma)
print("La división es: ", div)
print("El producto es: ", prod)
```

```
Digite # 1:
18
Digite # 2:
14
Digite # 3:
100
La suma es:  132.0
La división es:  -0.37209302325581395
El producto es:  -49.116279069767444
```

Figura 78. Ejecución del segundo ejercicio resuelto en Python.

2.8.3 Tercer ejercicio

El propietario de una vivienda necesita renovar parte de esta, para lo cual tiene planeado enchapar los muros de su baño. La persona responsable de hacer este trabajo mide el alto y el largo de los muros. Se pide crear una solución algorítmica para calcular e imprimir el área total del baño y el número de cajas de baldosas necesarias para cubrir esta área del baño, sabiendo que cada caja trae 3.5 metros cuadrados.

En la figura 79 se muestra el pseudocódigo en PSeInt y en la figura 80 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite largo muro: "
    Leer Largo
    Imprimir "Digite alto del muro: "
    Leer Alto
    Area = Largo * Alto
    N_Cajas = Area / 3.5
    Imprimir "Área total: ", Area, " mt2"
    Imprimir "Se requieren ", N_Cajas, " cajas"
FinAlgoritmo
```

Ejecución

```
Digite largo muro:
> 7
Digite alto del muro:
> 6
Área total: 42 mt2
Se requieren 12 cajas
```

Figura 79. Tercer ejercicio resuelto en PSeInt.

main.py

```
print("Digite largo muro: ")
largo = float(input())
print("Digite alto del muro: ")
alto = float(input())
area = largo * alto
n_cajas = area / 3.5
print("Área total: ", area," mt2")
print("Se requieren ", n_cajas," cajas")
```

```
Digite largo muro:
7
Digite alto del muro:
6
Área total:  42.0  mt2
Se requieren  12.0  cajas
```

Figura 80. Ejecución del tercer ejercicio resuelto en Python.

Si se hace una gráfica del ejercicio anterior para verificar sus resultados usando los últimos valores, el resultado del área total del baño junto con su distribución, sería un *área de 42 metros cuadrados y se requieren 12 cajas exactamente como se muestra* en la figura 81.

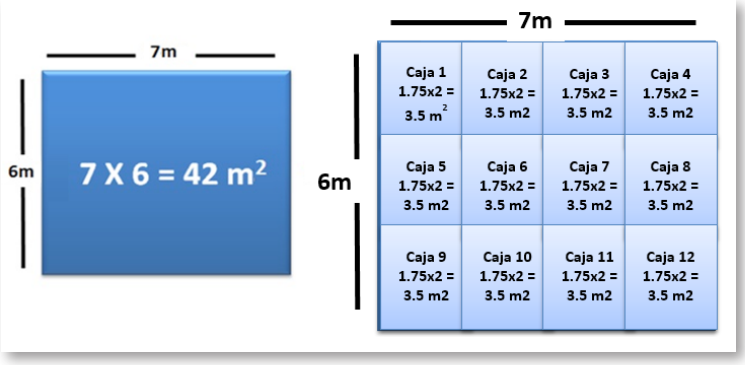


Figura 81. Representación gráfica del área y distribución de un terreno.

2.8.4 Cuarto ejercicio

Un atleta tiene la costumbre de medir el tiempo (en minutos) y la distancia (en metros) en sus tres días de entrenamiento. Al final de la semana quiere saber el total de tiempo que duró el entrenamiento, la distancia total recorrida, el tiempo promedio y la distancia promedio. Plantear una solución que muestre los totales y los promedios esperados.

En la figura 82 se muestra el pseudocódigo en PSeInt y en la figura 83 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese 1er. tiempo: "
    Leer T1
    Imprimir "Ingrese 2do. tiempo: "
    Leer T2
    Imprimir "Ingrese 3er. tiempo: "
    Leer T3
    Imprimir "Ingrese 1er. distancia: "
    Leer D1
    Imprimir "Ingrese 2do. distancia: "
    Leer D2
    Imprimir "Ingrese 3er. distancia: "
    Leer D3
    Total_T = T1 + T2 + T3
    Total_D = D1 + D2 + D3
    Prom_T = Total_T / 3
    Prom_D = Total_D / 3
    Imprimir "Tiempo que duró el entrenamiento: ", Total_T, " minutos"
    Imprimir "Distancia total recorrida: ", Total_D, " metros"
    Imprimir "Tiempo promedio: ", Prom_T, " minutos"
    Imprimir "Distancia promedio: ", Prom_D, " metros"
FinAlgoritmo
```

Ejecución

```
Ingrese 1er. tiempo:
> 80
Ingrese 2do. tiempo:
> 120
Ingrese 3er. tiempo:
> 70
Ingrese 1er. distancia:
> 4.6
Ingrese 2do. distancia:
> 1.8
Ingrese 3er. distancia:
> 7
Tiempo que duró el entrenamiento: 270 minutos
Distancia total recorrida: 13.4 metros
Tiempo promedio: 90 minutos
Distancia promedio: 4.4666666667 metros
```

Figura 82. Cuarto ejercicio resuelto en PSeInt.

main.py

```
print("Ingrese 1er. tiempo: ")
t1 = float(input())
print("Ingrese 2do. tiempo: ")
t2 = float(input())
print("Ingrese 3er. tiempo: ")
t3 = float(input())
print("Ingrese 1er. distancia: ")
d1 = float(input())
print("Ingrese 2do. distancia: ")
d2 = float(input())
print("Ingrese 3er. distancia: ")
d3 = float(input())
total_t = t1 + t2 + t3
total_d = d1 + d2 + d3
prom_t = total_t / 3
prom_d = total_d / 3
print("Tiempo que duró el entrenamiento: ", total_t," minutos")
print("Distancia total recorrida: ", total_d," metros")
print("Tiempo promedio: ", prom_t," minutos")
print("Distancia promedio: ", prom_d," metros")
```

```
Ingrese 1er. tiempo:
80
Ingrese 2do. tiempo:
120
Ingrese 3er. tiempo:
70
Ingrese 1er. distancia:
4.6
Ingrese 2do. distancia:
1.8
Ingrese 3er. distancia:
7
Tiempo que duró el entrenamiento:  270.0  minutos
Distancia total recorrida:  13.399 metros
Tiempo promedio:  90.0  minutos
Distancia promedio:  4.466666666666666  metros
```

Figura 83. Ejecución del cuarto ejercicio resuelto en Python.

2.8.5 Quinto ejercicio

Una madre y sus cuatro hijos se acercan a recibir información por parte de un abogado referente al dinero que les corresponde en una herencia dejada por su esposo y padre respectivamente. El testamento tiene estas condiciones: a la esposa le deja el 40% del total heredado y el resto se distribuye entre los cuatro hijos teniendo en cuenta que les corresponde los siguientes porcentajes: 30%, 20%, 40% y 10% respectivamente. Se necesita de un algoritmo que lea los datos necesarios y muestre lo que le corresponde a la madre, a los hijos en general y a cada uno de ellos.

En la figura 84 se muestra el pseudocódigo en PSeInt y en la figura 85 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Cuál es el monto de la herencia?: $"
    Leer Monto_Her
    Esposa = Monto_Her * 40/100
    Resto = Monto_Her - Esposa
    Hijo1 = Resto * 0.3
    Hijo2 = Resto * 0.2
    Hijo3 = Resto * 0.4
    Hijo4 = Resto * 0.1
    Imprimir "Total recibido esposa: $", Esposa
    Imprimir "Total recibido hijos: $", Resto
    Imprimir "Total recibido 1er. hijo: $", Hijo1
    Imprimir "Total recibido 2do. hijo: $", Hijo2
    Imprimir "Total recibido 3er. hijo: $", Hijo3
    Imprimir "Total recibido 4to. hijo: $", Hijo4
FinAlgoritmo
```

Ejecución

```
¿Cuál es el monto de la herencia?: $
> 250000000
Total recibido esposa: $100000000
Total recibido hijos: $150000000
Total recibido 1er. hijo: $45000000
Total recibido 2do. hijo: $30000000
Total recibido 3er. hijo: $60000000
Total recibido 4to. hijo: $15000000
```

Figura 84. Quinto ejercicio resuelto en PSeInt.

main.py

```
print("¿Cuál es el monto de la herencia?: $")
monto_her = float(input())
esposa = monto_her*40/100
resto = monto_her-esposa
hijo1 = resto * 0.3
hijo2 = resto * 0.2
hijo3 = resto * 0.4
hijo4 = resto * 0.1
print("Total recibido esposa: $", esposa)
print("Total recibido hijos: $", resto)
print("Total recibido 1er. hijo: $", hijo1)
print("Total recibido 2do. hijo: $", hijo2)
print("Total recibido 3er. hijo: $", hijo3)
print("Total recibido 4to. hijo: $", hijo4)
```

```
¿Cuál es el monto de la herencia?: $
250000000
Total recibido esposa: $ 100000000.0
Total recibido hijos: $ 150000000.0
Total recibido 1er. hijo: $ 45000000.0
Total recibido 2do. hijo: $ 30000000.0
Total recibido 3er. hijo: $ 60000000.0
Total recibido 4to. hijo: $ 15000000.0
```

Figura 85. Ejecución del quinto ejercicio resuelto en Python.

2.8.6 Sexto ejercicio

El terreno comprado por un político tuvo la siguiente destinación: 40% para cultivos, 25% para construir viviendas y 15% para zonas verdes. Leer el área total del terreno en metros cuadrados e imprimir el área de cada destinación y el área que queda disponible para otros proyectos.

En la figura 86 se muestra el pseudocódigo en PSeInt y en la figura 87 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese el área del terreno: "
    Leer Area_Terr
    Cultivo = Area_Terr * 0.4
    Vivienda = Area_Terr * 0.25
    Zonas_V = Area_Terr * 0.15
    Area_Disp = Area_Terr - Cultivo - Vivienda - Zonas_V
    Imprimir "Área cultivos: ", Cultivo, " metros cuadrados"
    Imprimir "Área viviendas: ", Vivienda, " metros cuadrados"
    Imprimir "Área zonas verdes: ", Zonas_V, " metros cuadrados"
    Imprimir "Área disponible: ", Area_Disp, " metros cuadrados"
FinAlgoritmo
```

Ejecución

```
Ingrese el área del terreno:
> 2850
Área cultivos: 1140 metros cuadrados
Área viviendas: 712.5 metros cuadrados
Área zonas verdes: 427.5 metros cuadrados
Área disponible: 570 metros cuadrados
```

Figura 86. Sexto ejercicio resuelto en PSeInt.

main.py

```
print("Ingrese el área del terreno: ")
area_terr = float(input())
cultivo = area_terr * 0.4
vivienda = area_terr * 0.25
zonas_v = area_terr * 0.15
area_disp = area_terr - cultivo - vivienda - zonas_v
print("Área cultivos: ", cultivo," metros cuadrados")
print("Área viviendas: ", vivienda," metros cuadrados")
print("Área zonas verdes: ", zonas_v," metros cuadrados")
print("Área disponible: ", area_disp," metros cuadrados")
```

```
Ingrese el área del terreno:
2850
Area cultivos:  1140.0  metros cuadrados
Area viviendas:  712.5  metros cuadrados
Area zonas verdes:  427.5  metros cuadrados
Area disponible:  570.0  metros cuadrados
```

Figura 87. Ejecución del sexto ejercicio resuelto en Python.

Si se requiere de un análisis a los resultados de la última prueba de escritorio donde el área del terreno es igual a 2000 metros, la figura 88 ayuda a este análisis.

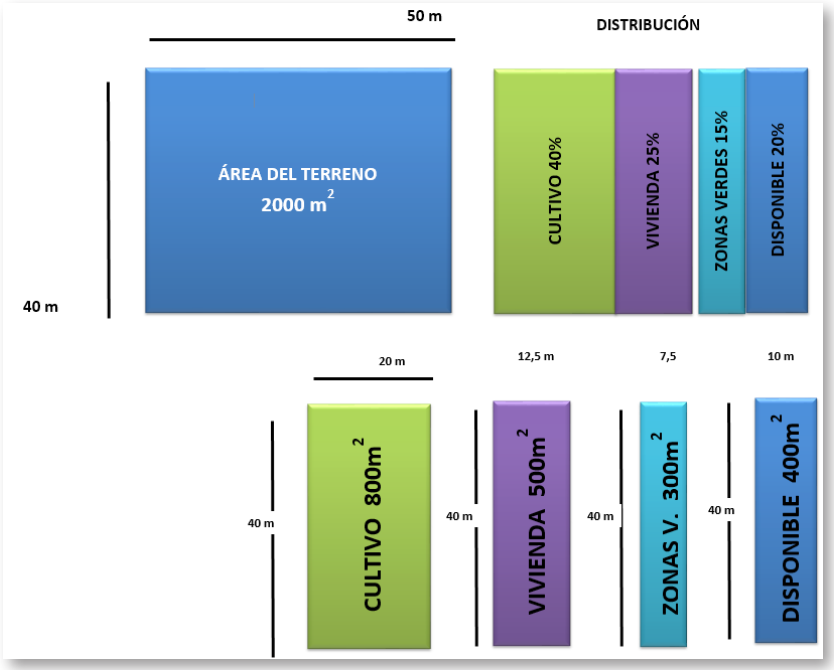


Figura 88. Representación gráfica de los porcentajes.

La sumatoria de cada una de las áreas es: 800 m^2 (cultivo) + 500 m^2 (vivienda) + 300 m^2 (zonas verdes) + 400 m^2 (disponible) = 2000 m^2 (área terreno). Aunque la distribución puede ser de múltiples tipos, por ejemplo, no aplicar el porcentaje sobre el ancho sino al largo del terreno.

2.8.7 Séptimo ejercicio

La temperatura (en grados centígrados) de un líquido ubicado en un recipiente aumentó un 43% luego de ser puesto sobre un mechero en un laboratorio. Luego de 10 minutos se mezcló con otro líquido congelado, lo que ocasionó que la temperatura descendiera 35% con respecto a la última temperatura (la temperatura después del aumento). Finalmente, luego de otros 10 minutos la temperatura volvió a subir un 150% con respecto a la última temperatura (la temperatura después de la disminución).

Hacer un algoritmo que lea la temperatura inicial del líquido en grados centígrados e imprima la temperatura de aumento o disminución, las temperaturas parciales, cambio de temperatura y la temperatura final.

En la figura 89 se muestra el pseudocódigo en PSeInt y en la figura 90 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite temperatura inicial (°C): "
    Leer T_Inic
    Aum1 = T_Inic * 0.43
    T_Parc1 = T_Inic + Aum1
    Dism = T_Parc1 * 0.35
    T_Parc2 = T_Parc1 - Dism
    Aum2 = T_Parc2 * 1.5
    T_Actual = T_Parc2 + Aum2
    Imprimir "Primer aumento de temperatura: ", Aum1, " °C"
    Imprimir "Temperatura luego del mechero: ", T_Parc1, " °C"
    Imprimir "Segundo aumento de temperatura: ", Aum2, " °C"
    Imprimir "Temperatura luego del líquido congelado: ", T_Parc2, " °C"
    Imprimir "Disminución de temperatura: ", Dism, " °C"
    Imprimir "Temperatura final: ", T_Actual, " °C"
FinAlgoritmo
```

Ejecución

```
Digite temperatura inicial (°C):
> 32
Primer aumento de temperatura: 13.76 °C
Temperatura luego del mechero: 45.76 °C
Segundo aumento de temperatura: 44.616 °C
Temperatura luego del líquido congelado: 29.744 °C
Disminución de temperatura: 16.016 °C
Temperatura final: 74.36 °C
```

Figura 89. Séptimo ejercicio resuelto en PSeInt.

main.py

```
print("¿Temperatura inicial?:")
t_inic = float(input())
aum1 = t_inic * 0.43
t_parc1 = t_inic+aum1
dism = t_parc1 * 0.35
t_parc2 = t_parc1 - dism
aum2 = t_parc2 * 1.5
t_actual = t_parc2 + aum2
print("Primer aumento de temperatura: ", aum1," °C")
print("Temperatura luego del mechero: ", t_parc1," °C")
print("Segundo aumento de temperatura: ", aum2," °C")
print("Temperatura luego del líquido congelado: ", t_parc2," °C")
print("Disminución de temperatura: ", dism," °C")
print("Temperatura final: ", t_actual," °C")
```

```
¿Temperatura inicial?:
32
Primer aumento de temperatura:  13.76  °C
Temperatura luego del mechero:  45.76  °C
Segundo aumento de temperatura:  44.616  °C
Temperatura luego del líquido congelado:  29.744 °C
Disminución de temperatura:  16.016  °C
Temperatura final:  74.36  °C
```

Figura 90. Ejecución del séptimo ejercicio resuelto en Python.

2.8.8 Octavo ejercicio

En la plaza de mercado se venden mangos por unidad; pero de igual forma se empacan en cajas de 6, 10, 20, 50 y 100 unidades. Hacer un algoritmo

que lea el valor unitario del mango y muestre el valor de cada una de las cajas, teniendo en cuenta que las cajas de 20 tienen un descuento del 10%, las de 50 del 25% y las de 100 se venden a mitad de precio.

En la figura 91 se muestra el pseudocódigo en PSeInt y en la figura 92 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Valor unitario del mango?: "
    Leer Val_Unit
    V_Caja6 = Val_Unit * 6
    V_Caja10 = Val_Unit * 10
    V_Caja20 = Val_Unit * 20 * 0.9
    V_Caja50 = Val_Unit * 50 * 0.75
    V_Caja100 = Val_Unit * 100 / 2
    Imprimir "Valor 6 unidades: $", V_Caja6
    Imprimir "Valor 10 unidades: $", V_Caja10
    Imprimir "Valor 20 unidades (descuento): $", V_Caja20
    Imprimir "Valor 50 unidades (descuento): $", V_Caja50
    Imprimir "Valor 100 unidades (descuento): $", V_Caja100
FinAlgoritmo
```

Ejecución

```
¿Valor unitario del mango?:
> 2500
Valor 6 unidades: $15000
Valor 10 unidades: $25000
Valor 20 unidades (descuento): $45000
Valor 50 unidades (descuento): $93750
Valor 100 unidades (descuento): $125000
```

Figura 91. Octavo ejercicio resuelto en PSeInt.

main.py

```
print("¿Valor unitario del mango?: ")
val_unit = float(input())
v_caja6 = val_unit * 6
v_caja10 = val_unit * 10
v_caja20 = val_unit * 20 * 0.9
v_caja50 = val_unit * 50 * 0.75
v_caja100 = val_unit * 100 / 2
print("Valor 6 unidades: $", v_caja6)
print("Valor 10 unidades: $", v_caja10)
print("Valor 20 unidades (descuento): $", v_caja20)
print("Valor 50 unidades (descuento): $", v_caja50)
print("Valor 100 unidades (descuento): $", v_caja100)
```

```
¿Valor unitario del mango?:
2500
Valor 6 unidades: $ 15000.0
Valor 10 unidades: $ 25000.0
Valor 20 unidades (descuento): $ 45000.0
Valor 50 unidades (descuento): $ 93750.0
Valor 100 unidades (descuento): $ 125000.0
```

Figura 92. Ejecución del octavo ejercicio resuelto en Python.

2.8.9 Noveno ejercicio

En clase de algoritmos y programación se manejan cuatro notas del 15%, 30%, 30% y 25% respectivamente. Se pide diseñar un algoritmo que permita mostrar la nota definitiva de un estudiante, teniendo en cuenta que la primera nota consta del promedio de dos talleres, la segunda nota sale del promedio de tres evaluaciones, la tercera nota es de un trabajo final y la última es el promedio de cuatro notas de seguimiento.

En la figura 93 se muestra el pseudocódigo en PSeInt y en la figura 94 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Nota taller #1?: "
    Leer Tal1
    Imprimir "¿Nota taller #2?: "
    Leer Tal2
    Imprimir "¿Nota evaluación #1?: "
    Leer Ev1
    Imprimir "¿Nota evaluación #2?: "
    Leer Ev2
    Imprimir "¿Nota evaluación #3?: "
    Leer Ev3
    Imprimir "¿Nota trabajo final?: "
    Leer Final
    Imprimir "¿Nota seguimiento #1?: "
    Leer Seg1
    Imprimir "¿Nota seguimiento #2?: "
    Leer Seg2
    Imprimir "¿Nota seguimiento #3?: "
    Leer Seg3
    Imprimir "¿Nota seguimiento #4?: "
    Leer Seg4
    Not1 = (Tal1 + Tal2) / 2
    Not2 = (Ev1 + Ev2 + Ev3) / 3
    Not4 = (Seg1 + Seg2 + Seg3 + Seg4) / 4
    Defi = Not1*0.15+Not2*0.3+Final*0.3+Not4*0.25
    Imprimir "Nota definitiva: ", Defi
FinAlgoritmo
```

Ejecución

```
¿Nota taller #1?:
> 2.5
¿Nota taller #2?:
> 3.5
¿Nota evaluación #1?:
> 4.5
¿Nota evaluación #2?:
> 5.0
¿Nota evaluación #3?:
> 5.0
¿Nota trabajo final?:
> 1.0
¿Nota seguimiento #1?:
> 4.8
¿Nota seguimiento #2?:
> 2.2
¿Nota seguimiento #3?:
> 5.0
¿Nota seguimiento #4?:
> 1.0
Nota definitiva: 3.0125
```

Figura 93. Noveno ejercicio resuelto en PSeInt.

main.py

```
print("¿Nota taller #1?: ")
tal1 = float(input())
print("¿Nota taller #2?: ")
tal2 = float(input())
print("¿Nota evaluación #1?: ")
ev1 = float(input())
print("¿Nota evaluación #2?: ")
ev2 = float(input())
print("¿Nota evaluación #3?: ")
ev3 = float(input())
print("¿Nota trabajo final?: ")
final = float(input())
print("¿Nota seguimiento #1?: ")
seg1 = float(input())
print("¿Nota seguimiento #2?: ")
seg2 = float(input())
print("¿Nota seguimiento #3?: ")
seg3 = float(input())
print("¿Nota seguimiento #4?: ")
seg4 = float(input())
not1 = (tal1+tal2)/2
not2 = (ev1+ev2+ev3)/3
not4 = (seg1+seg2+seg3+seg4)/4
defi = not1*0.15+not2*0.3+final*0.3+not4*0.25
print("Nota definitiva: ", defi)
```

```
¿Nota taller #1?:
2.5
¿Nota taller #2?:
3.5
¿Nota evaluación #1?:
4.5
¿Nota evaluación #2?:
5.0
¿Nota evaluación #3?:
5.0
¿Nota trabajo final?:
1.0
¿Nota seguimiento #1?:
4.8
¿Nota seguimiento #2?:
2.2
¿Nota seguimiento #3?:
5.0
¿Nota seguimiento #4?:
1.0
Nota definitiva:  3.01249
```

Figura 94. Ejecución del noveno ejercicio resuelto en Python.

2.8.10 Décimo ejercicio

Una persona va a un gimnasio pesando un valor inicial (en kilogramos). Luego de varios días de ejercicios su peso sufre una disminución del 28.5% y al cabo de un mes tiene un incremento de peso del 5% (tome como referencia el peso después de la disminución). Hacer un algoritmo que permita determinar su peso actual en kilogramos, miligramos, libras, onzas y quintales.

Tener en cuenta que: 1 kilogramo equivale a 1000 gramos; 1 kilogramo equivale a 1 000 000 miligramos; 1 kilogramo tiene 2.2205 libras; 1 libra equivale a 16 onzas y 1 quintal equivale a 100 kilogramos.

En la figura 95 se muestra el pseudocódigo en PSeInt y en la figura 96 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite su peso inicial: "
    Leer Peso_X
    Reduc = Peso_X * 0.285
    Pes_Desp_Red = Peso_X - Reduc
    Aumen = Pes_Desp_Red * 0.05
    Pes_Act_K = Pes_Desp_Red + Aumen
    Pes_Mil = Pes_Act_K * 1000000
    Pes_Lib = Pes_Act_K * 2.205
    Pes_Onz = Pes_Lib * 16
    Pes_Quin = Pes_Act_K / 100
    Imprimir "Peso actual kilogramos: ", Pes_Act_K
    Imprimir "Peso actual miligamos: ", Pes_Mil
    Imprimir "Peso actual libras: ", Pes_Lib
    Imprimir "Peso actual onzas: ", Pes_Onz
    Imprimir "Peso actual quintal: ", Pes_Quin
FinAlgoritmo
```

Ejecución

```
Digite su peso inicial:
> 80
Peso actual kilogramos: 60.06
Peso actual miligamos: 60060000
Peso actual libras: 132.4323
Peso actual onzas: 2118.9168
Peso actual quintal: 0.6006
```

Figura 95. Décimo ejercicio resuelto en PSeInt.

main.py

```
print("Digite su peso inicial: ")
peso_x = float(input())
reduc = peso_x * 0.285
pes_desp_red = peso_x - reduc
aumen = pes_desp_red * 0.05
pes_act_k = pes_desp_red + aumen
pes_mil = pes_act_k * 1000000
pes_lib = pes_act_k * 2.205
pes_onz = pes_lib * 16
pes_quin = pes_act_k / 100
print("Peso actual kilogramos: ", pes_act_k)
print("Peso actual miligamos: ", pes_mil)
print("Peso actual libras: ", pes_lib)
print("Peso actual onzas: ", pes_onz)
print("Peso actual quintal: ", pes_quin)
```

```
Digite su peso inicial:
80
Peso actual kilogramos:  60.06
Peso actual miligamos:  60060000.0
Peso actual libras:  132.4323
Peso actual onzas:  2118.9168
Peso actual quintal:  0.6006
```

Figura 96. Ejecución del décimo ejercicio resuelto en Python.

En la solución anterior se pudo calcular el peso actual en una instrucción más corta y optimizada: *Pes_Act_K = Peso_X * 0.725 * 1.05*; pero es un poco más compleja, porque si se multiplica el peso por 0.725 (72.5%) significa que se hizo una reducción de 28.5 y al multiplicar por 1.05 (105%), significa que aumentó un 5% tomando como base el peso resultante, después de la reducción.

2.9 Ejercicios propuestos

2.9.1 Primer ejercicio

El sistema de liquidación que hacen los conductores de los buses y los colectivos a las diferentes empresas consiste en tomar un número de la registradora al iniciar el día y otro al terminarlo. La diferencia de estos dos números determina el número de pasajeros transportados en el díaRealizar un algoritmo que permita leer estos dos números y el valor de un pasaje. Calcular e imprimir el total de pasajeros, el valor liquidado al conductor y el total liquidado a la empresa. Tenga en cuenta que la empresa recibe tres cuartas partes del dinero mientras que el conductor recibe el resto.

2.9.2 Segundo ejercicio

Una persona compra una computadora portátil y una impresora en Estados Unidos, con precios en dólares. Calcular por medio de un algoritmo el precio en pesos colombianos de cada artículo y el precio total de la compra. Tenga en cuenta que se debe leer el valor al que equivale un dólar en pesos (este valor es conocido como TRM —tasa representativa del mercado—, la cual es la cantidad de pesos colombianos por un dólar de los Estados Unidos).

2.9.3 Tercer ejercicio

Un almacén tiene cuatro sucursales en la ciudad. El dueño de este almacén requiere del diseño de un algoritmo que lea la venta semanal por cada una de las sucursales y muestre el total vendido y el promedio de ventas realizadas en una semana.

2.9.4 Cuarto ejercicio

Un investigador desea conocer todos los datos equivalentes a la edad de un fósil. Para esto se requiere de un algoritmo al que se le ingrese la cantidad de años e imprima su equivalente en: milenios, siglos, décadas, meses, días y horas.

Tener en cuenta que: 1 día equivale a 24 horas, 1 mes equivale a 30 días, 1 año equivale a 12 meses, 1 década equivale a 10 años, 1 siglo equivale a 100 años y 1 milenio equivale a 1000 años.

2.9.5 Quinto ejercicio

En un encuentro mundial de deportistas que participan en varias maratones se tiene el problema que, al dar una distancia de una de esas pruebas, no se alcanza a dimensionar sus medidas porque no están acostumbrados a dichas escalas.

Desarrollar un algoritmo que ingrese una distancia en metros y muestre los siguientes equivalentes: kilómetros, pies, yardas, brazas inglesas, leguas y millas.

Tener en cuenta que: 1 kilómetro equivale a 1000 metros, 1 yarda equivale a 0.9144 metros, 1 pulgada equivale a 2.54 centímetros, 1 metro equivale a 3.28084 pies, 1 yarda equivale a 3 pies, 1 legua equivale a 5.5727 kilómetros, 1 milla equivale a 1.609 kilómetros y 1 braza equivale a 1.8288 metros (1 braza también equivale a 2 yardas o 6 pies).

2.9.6 Sexto ejercicio

Un antioqueño necesita viajar a Inglaterra en el mes de diciembre, para lo cual debe cambiar sus pesos colombianos a libras esterlinas. Se pide hacer un algoritmo que lea el número de pesos y le permita ayudar a esta persona a saber la equivalencia exacta de su dinero en libras, sabiendo que cada libra esterlina cuesta el equivalente de 129% de un dólar y el valor del dólar hay que leerlo.

2.9.7 Séptimo ejercicio

Un campesino compró un terreno en forma triangular. La primera tarea que debe realizar es poner alambre a todos los linderos, pero no sabe la forma de hallar la cantidad de alambre que debe comprar. La única información con la que cuenta es con la dimensión de cada uno de los tres linderos (todos son diferentes). Hacer un algoritmo que permita imprimir la cantidad de

metros de alambre que requiere para hacer esta actividad, tenga en cuenta la siguiente fórmula:

Perímetro triángulo escaleno = Lado1 + Lado2 + Lado3

2.9.8 Octavo ejercicio

La parte central de la institución universitaria tiene forma circular. Para iniciar el nuevo año se tomó la decisión de cambiar todo el piso y comprar una nueva baldosa. La persona encargada del trabajo requiere conocer el área total de la misma (véase figura 97).

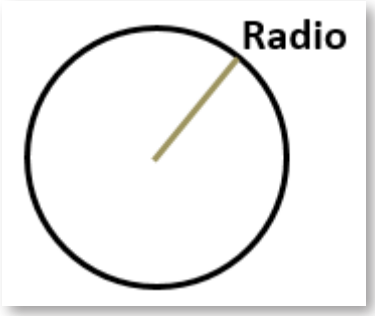


Figura 97. Representación de la circunferencia.

Hacer un algoritmo que permita ingresar la distancia que hay desde el centro a cualquiera de los extremos (radio del círculo) e imprima el área total usando la siguiente fórmula:

Área de Circunferencia = π .Radio^2

2.9.9 Noveno ejercicio

Desarrollar un algoritmo que permita dar solución a la siguiente ecuación:

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

2.9.10 *Décimo ejercicio*

Un profesor de matemáticas requiere de un algoritmo para hallar el área de las siguientes figuras geométricas: cubo, rombo, romboide y cilindro. Tenga en cuenta: no repetir variables de entrada debido a que las figuras son diferentes. Estas son las fórmulas correspondientes:

$$\text{Área del cubo} = 6 \cdot \text{Lado}^2$$

$$\text{Área del rombo} = \frac{\text{Diagonal1} \cdot \text{Diagonal2}}{2}$$

$$\text{Área del romboide} = \text{Base} \cdot \text{Altura}$$

$$\text{Área del cilindro} = 2 \cdot \pi \cdot \text{Radio} \cdot (\text{Altura} + \text{Base})$$

2.9.11 Décimo primer ejercicio

Luego de la caída del puente que comunicaba a los municipios de Medellín y San Jerónimo, se debe tomar una ruta alterna la cual incrementa el tiempo en un 70% y la distancia en un 55%, se pide leer el tiempo y la distancia anterior y determinar el tiempo y la distancia actual.

2.9.12 Décimo segundo ejercicio

En la parte alta de la esquina de una habitación se instala una cámara que detecta el movimiento como se muestra en la figura 98.

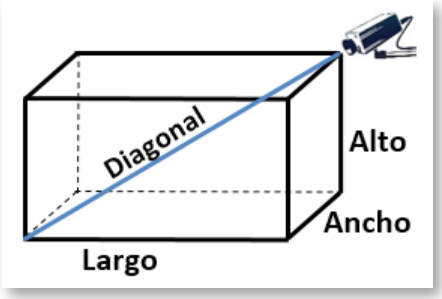


Figura 98. Representación de la diagonal de un rectángulo.

Para su configuración se requiere conocer la distancia más lejana que hay entre esta y la parte inferior de la esquina opuesta a la cámara. Se pide realizar

un algoritmo que solicite el largo, el ancho y el alto de la habilitación y use la fórmula de la diagonal de un rectángulo para dar una adecuada solución. Ver la siguiente fórmula:

2.9.13 Décimo tercer ejercicio

Se requiere hacer un algoritmo que calcule e imprima el volumen de estas cuatro figuras: cubo, ortoedro, cono y esfera. Tenga en cuenta, no repetir variables de entrada debido a que las figuras son diferentes. Las fórmulas son las siguientes:

$$\text{Diagonal} = \sqrt{\text{Alto}^2 + \text{Ancho}^2 + \text{Largo}^2}$$

$$\text{Volumen del cubo} = \text{Lado}^3$$

$$\text{Volumen del ortoedro} = \text{Largo} + \text{Ancho} + \text{Alto}$$

$$\text{Volumen del cono} = \frac{\pi \,.\, \text{Radio}^2 . \text{Altura}}{3}$$

$$\text{Volumen del esfera} = \frac{4}{3} \,.\, \pi \,.\, \text{Radio}^3$$

2.9.14 Décimo cuarto ejercicio

Existe una fórmula para hallar el número de diagonales de un polígono:

$$\text{Numero\_Diagonales} = \frac{\text{Numero\_Lados}\ (\text{Numero\_Lados} - 3)}{2}$$

Se pide hacer un algoritmo al que se le ingrese el número de lados del polígono y muestre el número de diagonales posibles (véase figura 99).

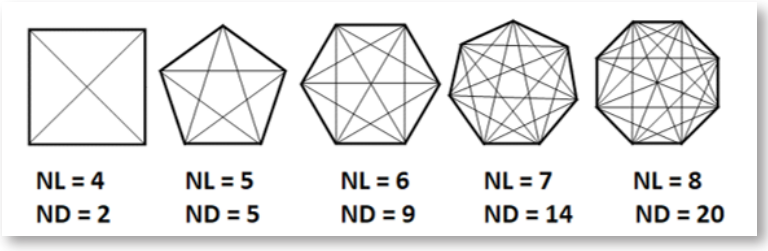


Figura 99. Representación gráfica del número de diagonales.
Fuente: Fernández (2017).

2.9.15 Décimo quinto ejercicio

En un edificio se demolió el 45% de sus pisos. Hacer un algoritmo que lea el número inicial de pisos e imprimir el número de pisos demolidos y el número de pisos después de la demolición.

2.9.16 Décimo sexto ejercicio

Una piscina en forma circular fue ampliada en varios metros respetando la circunferencia. Se pide hacer un algoritmo que lea el radio inicial de la piscina y el radio final de la piscina ampliada. Calcular el área que aumentó la piscina teniendo en cuenta la fórmula de la corona circular:

$$\text{Área} = \pi\ (\text{Radio1}^2 - \text{Radio2}^2)$$

2.9.17 Décimo séptimo ejercicio

Una persona que no conoce la escala de temperatura Kelvin ni la de Fahrenheit recibe una medida en cada escala. Para poder entender estas lecturas se pide hacer un algoritmo que halle el equivalente en grados centígrados. Tenga en cuenta las siguientes fórmulas:

$$^\circ \text{ Centígrados} = \text{Kelvin} - 273.15$$

$$^\circ \text{ Centígrados} = (^\circ \text{ Farenheit} - 32) / 1.8$$

2.9.18 Décimo octavo ejercicio

Para hallar un valor esperado en un experimento se usa la ecuación V*x.* Se pide hacer un algoritmo que permita leer el valor de la variable *x*, el valor de la variable *y* e imprima el resultado de esta ecuación:

$$\text{Vx} = \frac{1}{\text{x} + 2\text{y}}$$

2.9.19 Décimo noveno ejercicio

Realice un algoritmo que permita efectuar las conversiones de unidades que se encuentran en la tabla 37. La unidad de origen es la ubicada al lado

izquierdo y la unidad de destino es ubicada al lado derecho. Por ejemplo, en la primera línea se pide convertir una cantidad de gramos cúbicos en libras teniendo en cuenta ese equivalente. Realizar lo mismo con cada una de las líneas.

Tabla 37.
Resumen de equivalencias. Ejercicio 2.9.19.

Escala	Conversión
Densidad	1 gramo cúbico = 62.43 libras
Información	1 kilobyte = 1024 bytes
Longitud	1 pulgada = 2.54 centímetros
Masa	1 libra = 453.6 gramos
Potencia	1 kilovatio = 3412.142 BTU/Hora (Unidad t**érmica** británica)
Presión	1 atm**ó**sfera = 14.6959 PSI (libra por pulgada cuadrada)
Temperatura	1 ° Fahrenheit = 1.8 (° Cent**í**grados) + 32
Volumen	1 pulgada cúbica = 16.39 centímetros cúbicos

2.9.20 Vigésimo ejercicio

La ecuación contable o ecuación de contabilidad es la que hace relación al activo, al pasivo y al patrimonio (o capital). Una empresa requiere por medio de esta fórmula imprimir el total del activo con el que cuenta en este momento, mientras que otra empresa requiere imprimir el total de su patrimonio. La ecuación contable es:

$$\text{Activo} = \text{Pasivo} + \text{Patrimonio}.$$

2.9.21 Vigésimo primer ejercicio

En la asociación de cafeteros de un municipio requieren desarrollar un algoritmo que tome una medición en gramos y muestre su equivalente en kilos, arrobas, libras, onzas y quintales.

Tener en cuenta que: 1 kilogramo equivale a 1000 gramos, 1 arroba equivale a 15 kilos o 400 onzas, 1 libra equivale a 16 onzas, 1 kilogramo equivale a 2.2046 libras y 1 quintal equivale a 100 kilogramos.

2.9.22 Vigésimo segundo ejercicio

Hacer un algoritmo que permita calcular el perímetro de dos triángulos diferentes (uno equilátero y otro isósceles). Para hacer este proceso tenga en cuenta las fórmulas respectivas:

Perímetro triángulo equilátero = 3 .Lado

Perímetro triángulo isósceles= 2 .Altura + Base

2.9.23 Vigésimo tercer ejercicio

Realizar un algoritmo que permita convertir una cantidad de hectáreas en acres, áreas, metros y millas cuadrados.

Tener en cuenta que: 1 acre equivale a 0.4047 hectáreas, 1 área equivale a 100 metros cuadrados, 1 hectárea equivale a 10 000 metros cuadrados y 1 milla cuadrada equivale a 2.589 kilómetros cuadrados.

2.9.24 Vigésimo cuarto ejercicio

Un vendaval derribó el 40% de las matas de plátano de una finca. Posteriormente el dueño de esta pudo levantar el 7% de estas. Hacer un algoritmo que permita calcular el número de matas derribadas, el número de matas levantadas luego del vendaval y, finalmente, el total de matas que quedaron en pie. Tenga en cuenta que se debe leer el total de matas de plátano que tenía la finca antes del vendaval.

2.9.25 Vigésimo quinto ejercicio

Un zancudo saca el 0.0000005% de la sangre de una persona, mientras que su propia sangre aumenta un 200%. Se requiere calcular el aumento de sangre del zancudo, la disminución de sangre de la persona y el nivel actual de sangre de los dos.

Como información, la Asociación Americana de Control de Mosquitos (AMCA) dice que la picadura media de un mosquito quita cinco millonésimas partes de un litro. Cada insecto ingiere unos 5 miligramos, lo que supone el doble de su propia masa corporal que es de 2.5 miligramos

aproximadamente (Rey, 2018) y el cuerpo humano de un adulto tiene entre 4.25 y 5.67 litros.

2.9.26 Vigésimo sexto ejercicio

Convertir una cantidad de litros en pies cúbicos, pintas y galones ingleses, pintas y galones americanos. Realizar un algoritmo para hacer esas conversiones.

Tener en cuenta que: 1 pie cúbico equivale a 28.3168 litros, 1 pinta inglesa equivale a 0.5683 litros, 1 pinta americana equivale a 0.473 litros, 1 galón inglés equivale a 4.5461 litros y 1 galón americano equivale a 3.7854 litros.

2.9.27 Vigésimo séptimo ejercicio

Un terremoto destruyó el 60% de una vivienda. Luego de evaluar los daños, se tomó la decisión de derrumbar el 75% de lo que había quedado luego del terremoto. Se requiere de un algoritmo que tome el área total de la vivienda antes del terremoto y muestre el área destruida, el área derrumbada y el área actual luego de estas dos situaciones.

2.9.28 Vigésimo octavo ejercicio

Elabore un algoritmo que determine el espacio recorrido por un móvil junto con su velocidad final; teniendo en cuenta el movimiento rectilíneo uniformemente acelerado (MRUA). Leer las variables necesarias para dar una solución. Según EducaMadrid (2018), las fórmulas de la velocidad final (v_f) y el espacio (s) hacen relación a la aceleración (a), la velocidad inicial (v_i) y el tiempo (t):

$$vf = vi + a \,.t$$
$$s = vi.t + \frac{1}{2} a \,.t^2$$

2.9.29 Vigésimo noveno ejercicio

Un turista tiene dinero en pesos colombianos, en dólares y en euros. Como desea realizar un viaje a Estados Unidos y Europa, requiere cambiar la

cantidad de pesos colombianos en dólares y euros. Diseñar un algoritmo que le permita ingresar la cantidad de pesos colombianos que tiene, la cantidad de dólares y la cantidad de euros y visualice el total de dólares y el total de euros con el que termina después de hacer el respectivo cambio. Tenga en cuenta que la mitad del dinero en pesos se va a cambiar por dólares y el resto en euros; y para hacer este proceso también se debe leer el valor de un dólar en pesos colombianos (TRM) y el valor de un euro en pesos colombianos.

2.9.30 Trigésimo ejercicio

Una persona está interesada en cercar su finca para hacer sembrado de algunos productos. Esta persona hará una distribución en forma de polígono regular (véase la zona azul de la figura 100).

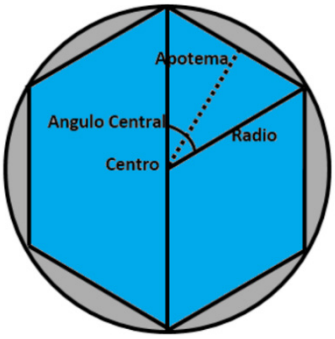


Figura 100. Representación del polígono regular.

Se requiere un algoritmo que lea la dimensión de uno de los lados y la menor distancia desde el centro hacia cualquier lado (apotema). Muestre el área total de la finca y el alambre total requerido para realizar el cercado. Las fórmulas relacionadas al polígono regular son:

$$P = N.L$$

$$A = \frac{P \,.\, Ap}{2}$$

Siendo, *P* el perímetro, *N* el número de lados, *L* la dimensión del lado, *A* el área del polígono regular y *Ap* la apotema.

2.9.31 Trigésimo primer ejercicio

Un coleccionista de billetes guarda en una caja solo pesos colombianos. Luego de varios años desea abrirla y conocer el monto total al que asciende su dinero. En la caja hay billetes de $ 1000, $ 2000, $ 5000, $ 10 000, $ 20 000 y $ 50 000. Hacer un algoritmo que lea la cantidad de billetes de cada denominación y permita mostrar ese monto total.

2.9.32 Trigésimo segundo ejercicio

Un estudiante de matemáticas recibe la tarea de determinar el área y el volumen de cuatro extrañas figuras: un tetraedro, un octaedro, un icosaedro y un dodecaedro. Se pide realizar un algoritmo que le permita a este estudiante imprimir los cálculos solicitados. Para ayudarle un poco se darán las fórmulas respectivas:

$$\text{Área tetraedro} = \sqrt{3}.\ \text{Lado}^2$$

$$\text{Volumen tetraedro} = \frac{\sqrt{2}}{12}.\ \text{Lado}^3$$

$$\text{Área octaedro} = 2\sqrt{3}.\ \text{Lado}^2$$

$$\text{Volumen octaedro} = \frac{\sqrt{2}}{3}.\ \text{Lado}^3$$

$$\text{Área icosaedro} = 5\sqrt{3}.\ \text{Lado}^2$$

$$\text{Volumen icosaedro} = \frac{5}{12}\left(3+\sqrt{5}\right).\text{Lado}^3$$

$$\text{Área dodecaedro} = 30.\ \text{Lado}\ .\text{Apotema}$$

$$\text{Volumen dodecaedro} = \frac{1}{4}(15+7\sqrt{5})\ .\ \text{Lado}^3$$

2.9.33 Trigésimo tercer ejercicio

El grado de alcohol de un licor depende de la cantidad de agua destilada (CAD) que se mezcle con este. La fórmula para determinar la cantidad exacta de agua destilada que permita bajar el grado de alcohol es

$$\text{CAD} = \text{CLA} - \frac{\text{CLA}\ .\ \text{GLA}}{\text{GLE}}$$

Siendo, CLA la cantidad de licor actual, GLA son los grados de licor actual y GLE son los grados de licor esperado.

Hacer un algoritmo que permita resolver la fórmula.

2.9.34 Trigésimo cuarto ejercicio

Hacer un algoritmo que permita leer los datos: *X*, *Y* y *Z* y resuelva la siguiente ecuación matemática:

$$W = X^{Y+Z} - Y^{X+Z} + Z^{Y+X}$$

2.9.35 Trigésimo quinto ejercicio

Una persona emprendedora compra una máquina plana, una fileteadora y una recubridora, cada una con los siguientes descuentos: 5%, 15% y 18% respectivamente. Hacer un algoritmo que lea el valor de cada máquina e imprima el valor del descuento de cada una de estas, el valor neto de cada una, el total descontado y total pagado.

2.10 Práctica final de algoritmos secuenciales

Con este tipo de ejercicio se pretende consolidar todos los conceptos aprendidos en esta unidad; a pesar de lo extenso, recuerde que los algoritmos y la programación siempre buscarán dividir un problema en fracciones más pequeñas que faciliten llegar a una solución clara, coherente y eficiente.

Una persona que inicia en ingeniería en la IUSH quiere adquirir una computadora portátil y encuentra en internet que las provenientes de la India son las más económicas del mercado. Luego de varios minutos de búsqueda, encuentra varias computadoras similares a las que desea comprar, pero el precio aparece en rupias y en las formas de pagos aparecen dólares, euros y *bitcoin*.

Esta persona no recuerda haber visto conceptos de conversión de unidades en la universidad, por lo tanto, solicita a un amigo programador que desarrolle un *software* para realizar una conversión de rupias a estas unidades agregándole la conversión a pesos colombianos.

Al momento de tener los valores, esta persona paga con su tarjeta de crédito, observando que el proveedor le realizó un descuento del 18.87% sobre el valor de la computadora y eligió diferirlo a 20 cuotas. Finalmente, le sale un mensaje indicándole que debe recordar que las compras realizadas con tarjeta de crédito tienen un recargo del 4%.

Luego de algunos días llega la computadora y empieza a realizar descargas de juegos, películas, *software*, cursos y vídeo-tutoriales; pero al poco tiempo se presenta un problema con el disco duro, el cual le produce la pérdida del 12.7% de su tamaño (capacidad en *terabyte*). Posteriormente, somete el disco duro a un programa de recuperación de información (*Data Recovery*), y puede recuperar el 40.2% de lo que se había perdido (no es el 40.2% de todo el disco sino de lo perdido).

Hacer un algoritmo que simule el *software* realizado por el programador; leer el valor de la computadora en rupias y mostrar los equivalentes respectivos en dólares, euros, *bitcoin* y pesos colombianos.

Tener en cuenta que: 1 rupia equivale a 42.25 pesos colombianos; 1 bitcoin equivale a 6408.12 dólares; 1 euro equivale a 1.16 dólares y si lo requiere puede leer el TRM actual. Así mismo, mostrar en pesos colombianos el valor del descuento, el valor del recargo, el valor de la compra con el descuento y el valor de cada cuota mensual (incluyendo el recargo) y el valor neto de la compra (incluyendo el descuento y el recargo).

Además, imprimir en *gigabyte*: la capacidad del disco duro inicial, la capacidad perdida, la capacidad recuperada, la capacidad actual y, finalmente, el equivalente en *megabyte, kilobyte*, CD y DVD de la capacidad actual (en Gb).

Tener en cuenta que: 1 *terabyte* equivale a 1024 *gigabyte*; 1 *gigabyte* equivale a 1024 *megabyte*; 1 *megabyte* equivale a 1024 *kilobyte*; 1 CD tiene capacidad de 640 *megabyte* y 1 DVD tiene 4.7 *gigabyte*.

Unidad 3
Estructuras de decisión y selección múltiple

Introducción

En esta unidad se explican los siguientes temas (véase el resumen en la figura 101):

- Definición de una estructura de decisión.
- Tipos de estructuras de decisión: simples, compuestas, múltiples y anidadas.
- Estructuras de selección múltiple o estructuras caso.
- Definición y aplicación de las diferentes estructuras.
- Ejercicios resueltos y propuestos.

Estos temas facilitan la obtención de las competencias que se describen a continuación:

- Conocer las partes que conforman una estructura de decisión.
- Construir condiciones válidas para una estructura de decisión.
- Identificar la importancia de las estructuras de decisión dentro del desarrollo de algoritmos.
- Diferenciar los tipos de estructuras de decisión y aplicarlos de manera correcta en la solución de problemas determinados.
- Optimizar instrucciones dentro de un algoritmo por medio de la estructura de decisión anidada y la estructura de selección múltiple.
- Reconocer los componentes de las estructuras de selección múltiple y las ventajas que tienen frente a las estructuras de decisión.
- Identificar las similitudes y diferencias entre la estructura de decisión y selección.

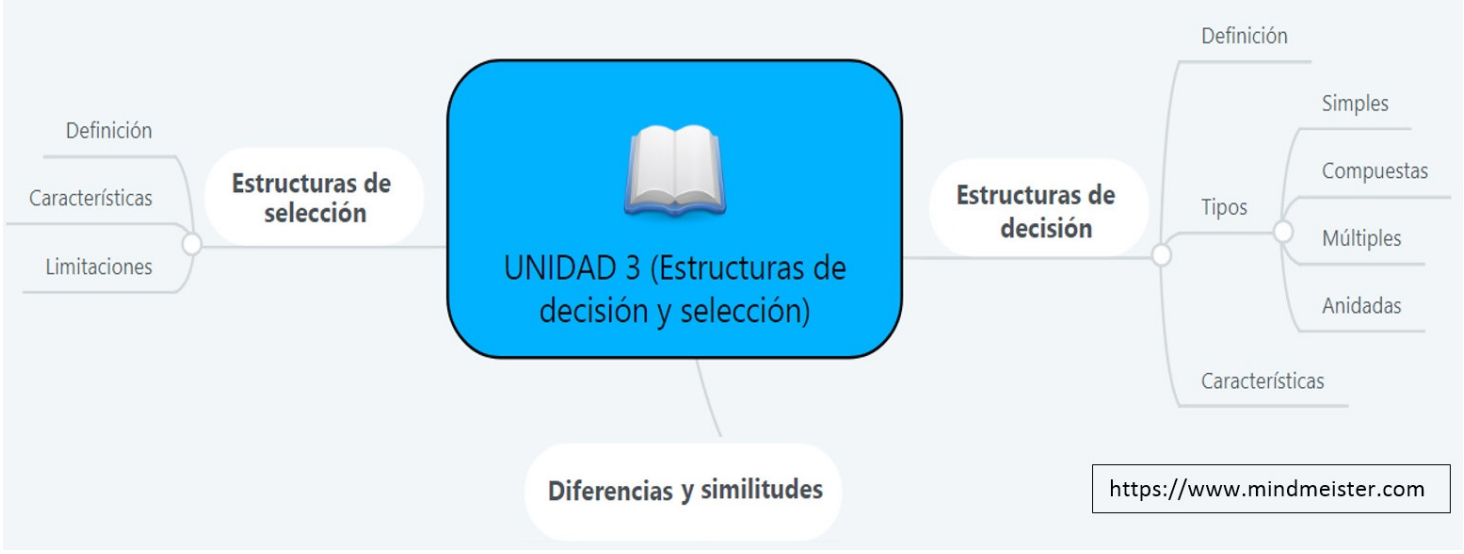


Figura 101. Unidad 3. Estructuras de decisión y selección.

3. Estructuras de decisión y selección

Las estructuras de decisión y selección se usan en algoritmos que requieren evaluar algún tipo de condición o que tenga que elegir entre varias alternativas para llegar a una solución más precisa. También se conocen con el nombre de *estructuras de control selectivas* o *alternativas.* Estos dos tipos de estructuras, decisión y selección, se complementan. A continuación se hace una explicación breve, una representación gráfica y se explican varios ejemplos con problemas resueltos.

3.1 Estructuras de decisión

La estructura de decisión es uno de los pilares de la lógica, para realizarla se deben utilizar los operadores relacionales, los cuales son operadores usados para establecer una comparación entre dos o más valores o variables.

Los operadores lógicos relacionales utilizados en lógica de programación para estructuras de decisión se resumen en la tabla 38.

Tabla 38.
Operadores lógicos relacionales.

Operador	Comparación
>	Mayor que
<	Menor que
>=	Mayor o igual que
<=	Menor o igual que
=	Igual
<>	Diferente
!=	Diferente

En el lenguaje de programación Python, el operador <> no se usa, sino que se usa *!=*, pero su funcionalidad en la misma. Este operador es muy usado en otros lenguajes populares como Java, C++, C o C#.

Las condiciones se pueden realizar con variables, constantes, cálculos o combinación de estos. Algunos ejemplos a continuación:

- Si (Salario = Salario_Neto) Entonces → Variable vs. Variable
- Si (Salario > 5000000) Entonces → Variable vs. Constante
- Si (500000 < Salario_Neto) Entonces → Constante vs. Variable
- Si (Salario_Neto >= Salario + Aumento) Entonces → Variable vs. Cálculo
- Si (Salario – Aumento <= Salario_Neto) Entonces → Cálculo vs. Variable
- Si (450000 <> Salario – Disminucion) Entonces → Constante vs. Cálculo
- Si (Salario_Neto + Aumento = 550000) Entonces → Cálculo vs. Constante
- Si (Basico – Retencion > Basico + Aumento) Entonces → Cálculo vs. Cálculo.

En la tabla 39 se muestra un listado de preguntas frecuentes en ejercicios de estructuras de decisión. En esta unidad se usan los operadores *Y O*, en lugar de los símbolos Λ y *V* usados en lógica matemática que se utilizaron en la primera unidad.

Tabla 39.
Ejemplos de condiciones en un algoritmo.

¿Cómo se pregunta en lenguaje natural?	¿Cómo se pregunta en un **algoritmo?**
Si una persona es mayor de edad	*Si (Edad >= 18) Entonces*
Si una persona es menor de edad	*Si (Edad < 18) Entonces*
Si una persona es soltera	*Si (EstadoCivil = "Soltero") Entonces*
Si una persona es casada	*Si (EstadoCivil = "Casado") Entonces*
Si una persona no es casada	*Si (EstadoCivil <> "Casado") Entonces*
Si una persona es hombre	*Si (Sexo = "Masculino") Entonces*
Si una persona es mujer	*Si (Sexo = "Femenino") Entonces*
Si un número es positivo	*Si (Numero > 0) Entonces*
Si un número es negativo	*Si (Numero < 0) Entonces*
Si un número es cero o neutro	*Si (Numero = 0) Entonces*
Si un número es par	*Si (Numero Mod 2 = 0) Entonces*
Si un número es par (otra forma)	*Si (Numero Mod 2 <> 1) Entonces*

¿Cómo se pregunta en lenguaje natural?	¿Cómo se pregunta en un **algoritmo?**
Si un número es impar	*Si (Numero Mod 2 = 1) Entonces*
Si un número es impar (otra forma)	*Si (Numero Mod 2 <> 0) Entonces*
Si un número es múltiplo de 5	*Si (Numero Mod 5 = 0) Entonces*
Si un número no es par	*Si (Numero Mod 2 <> 0) Entonces*
Si un número no es impar	*Si (Numero Mod 2 <> 1) Entonces*
Si un número es impar y múltiplo de 3	*Si (Num Mod 2=1 Y Num Mod 3=0) Entonces*
Si un número no es múltiplo de 3	*Si (Numero Mod 3 <> 0) Entonces*
Si un número es diferente de otro	*Si (Numero1 <> Numero2) Entonces*
Si un número está entre 10 y 20	*Si (Numero >= 10 Y Numero <= 20) Entonces*
Si una edad no alcanza los 20 años	*Si (Edad < 20) Entonces*
Si un triángulo es equilátero	*Si (Lad1 = Lad2 Y Lad1 = Lad3) Entonces*
Si un triángulo es escaleno	*Si (L1 <> L2 Y L1 <> L3 Y L2 <> L3) Entonces*
Si una nota está ganada	*Si (Nota >= 3.0) Entonces*
Si una nota está perdida	*Si (Nota < 3.0) Entonces*
Si una nota está entre 2.0 y 4.0	Si (Nota >= 2.0 Y Nota <= 4.0) Entonces

3.2 Tipos de estructuras de decisión

Al momento de construir una estructura de decisión, debe identificarse el número de condiciones o posibles preguntas, para determinar el tipo de estructura a utilizar. Los tipos de estructuras son: simples, compuestas, múltiples y anidadas. A continuación se da una explicación de cada uno y se muestran varios ejemplos de problemas resueltos.

3.2.1 Estructura de decisión simple

Es una estructura que se usa cuando un algoritmo requiere de una sola pregunta. Su sintaxis comienza con la palabra *Si*, seguida de una condición encerrada entre paréntesis y termina esta línea con la palabra *Entonces*; luego van las instrucciones que se realizan dependiendo la condición realizada. Finalmente, toda pregunta (*Si*), de una estructura de decisión, cierra con la instrucción *FinSi*. Así como todo algoritmo tiene su instrucción de *Fin*, todas las estructuras de decisión obligatoriamente tienen su instrucción de cierre.

Si (Condición) Entonces

Instrucciones (si la condición es *Verdadera*)

FinSi

A partir de este párrafo, todo lo relacionado a las instrucciones, cálculos, operaciones y demás conceptos similares y relacionados se llamarán simplemente instrucciones.

En la mayoría de herramientas para crear pseudocódigo la palabra *Entonces* se omite; por tal motivo queda a consideración del programador, si la escribe o no al momento de realizar el pseudocódigo. En este libro, dicha palabra, siempre se coloca al final de esta línea de la estructura, donde va una pregunta compuesta por una condición lógica. En medio de la estructura de decisión, solo se colocan las instrucciones que se van a desarrollar cuando la pregunta sea *Verdadera*, las demás operaciones van fuera de la estructura.

Las instrucciones se deben colocar un poco desplazadas hacia la derecha para que se pueda identificar perfectamente donde empiezan y donde terminan. Esto es conocido con el nombre de *indentación del* código.

La sintaxis de la estructura de decisión simple en diagramación libre se puede ver en la figura 102.

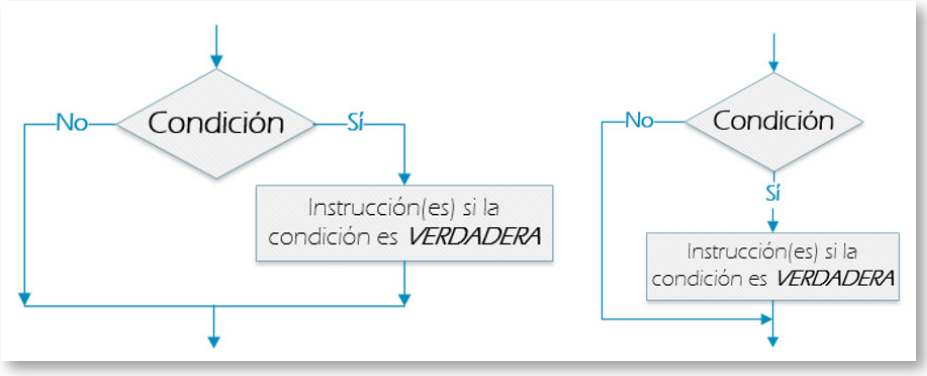


Figura 102. Sintaxis de estructura de decisión simple en diagramación libre.

Ejemplo 1. Realizar un algoritmo que lea dos números. Calcular la división del primero entre el segundo. Tenga en cuenta que solo se puede dividir si el segundo número es diferente de cero.

En la figura 103 se muestra el pseudocódigo en PSeInt y en la figura 104 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Digite número 1: "
3      Leer Num1
4      Imprimir "Digite número 2: "
5      Leer Num2
6      Si (Num2 <> 0) Entonces
7          Div = Num1 / Num2
8          Imprimir "La división es: ", Div
9      FinSi
10 FinAlgoritmo
```

Ejecución Condición Verdadera

```
Digite número 1:
> 28
Digite número 2:
> 5
La división es: 5.6
```

Ejecución Condición Falsa

```
Digite número 1:
> 28
Digite número 2:
> 0
```

Figura 103. Primer ejemplo de estructura de decisión simple en PSeInt.

main.py

```
1  print("Digite número 1: ")
2  num1 = float(input())
3  print("Digite número 2: ")
4  num2 = float(input())
5  if (num2 != 0):
6      div = num1 / num2
7      print("La división es: ", div)
```

```
Digite número 1:
28
Digite número 2:
5
La división es:  5.6
```

```
Digite número 1:
28
Digite número 2:
0
```

Figura 104. Primer ejemplo de estructura de decisión simple en Python.

Observe que PSeInt y Python agregan líneas verticales que permitan identificar con precisión el inicio y el final de las estructuras. Siempre se recomienda que estén alineados: el *Si*, el *Sino* y el *FinSi*, aunque en este ejercicio no se tiene una instrucción de *Sino*. Las instrucciones dentro de

Si o del *Sino* se ponen tabuladas hacia la derecha, este concepto se conoce como indentación o sangrado y es fundamental para programar.

En uno de los ejemplos de ejecución se ingresa el segundo número diferente a cero siendo esta una condición *Verdadera* y en otro ejemplo de ejecución se ingresa el segundo número igual a cero siendo esta una condición *Falsa*. Observe que en uno se calcula e imprime el resultado en el otro no se realiza ninguna instrucción.

En la figura 105 se muestra la solución de este ejercicio en diagramación libre.

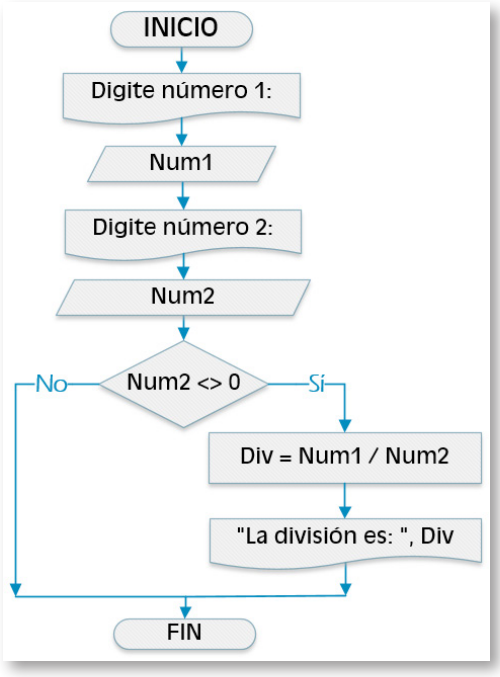


Figura 105. Estructura de decisión simple en diagramación libre.

Ejemplo 2. Realizar un algoritmo que pida a una persona el nombre de un día de la semana e imprima un mensaje "Es el mejor día de la semana", solo en caso de que se digite el viernes.

En la figura 106 se muestra el pseudocódigo en PSeInt y en la figura 107 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite nombre de un día de la semana: "
    Leer Dia
    Si (Dia = "VIERNES" O Dia = "Viernes" O Dia = "viernes") Entonces
        Imprimir "Es el mejor día de la semana."
    FinSi
FinAlgoritmo
```

Ejecución Condición Verdadera

```
Digite nombre de un día de la semana:
> Viernes
Es el mejor día de la semana.
```

Ejecución Condición Falsa

```
Digite nombre de un día de la semana:
> Lunes
```

Figura 106. Segundo ejemplo de estructura de decisión simple en PSeInt.

main.py

```
print("Digite nombre de un día de la semana: ")
dia = input()
if (dia=="VIERNES" or dia=="Viernes" or dia=="viernes"):
    print("Es el mejor día de la semana.")
```

```
Digite nombre de un día de la semana:
Viernes
Es el mejor día de la semana.
```

```
Digite nombre de un día de la semana:
Lunes
```

Figura 107. Segundo ejemplo de estructura de decisión simple en Python.

Observe que en la condición se preguntó por varias opciones con las que el usuario podría contestar: mayúsculas, minúsculas o alternándolas. Para corregir esta situación, se pueden usar funciones propias del lenguaje, como las funciones *mayusculas/minusculas* de PSeInt, que convierten la respuesta y así será más fácil realizar la condición. Tenga en cuenta que, si se coloca la función de *mayusculas*, la palabra dentro de la condición debe estar en mayúsculas también. Observe el siguiente ejemplo:

Dia = mayusculas(Dia).
Si (Dia = "VIERNES") Entonces
Imprimir "Es el mejor día de la semana"
FinSi

3.2.2 Estructura de decisión compuesta

Es una estructura que se usa cuando un algoritmo requiere ejecutar dos acciones diferentes con base en un resultado arrojado después de evaluar una condición. También es conocida como estructura de decisión doble. Su sintaxis es idéntica a la estructura simple, solo que, al requerirse dos posibilidades, se agrega en medio de la estructura la palabra *Sino* para la otra acción.

Si (Condición) Entonces
    Instrucciones (si la condición es Verdadera)
Sino
    Instrucciones (si la condición es Falsa)
FinSi

En este caso, las primeras instrucciones se desarrollan cuando la pregunta sea *Verdadera*, y las otras cuando sea *Falsa*. La sintaxis de la estructura de decisión compuesta en diagramación libre se puede ver en la figura 108.

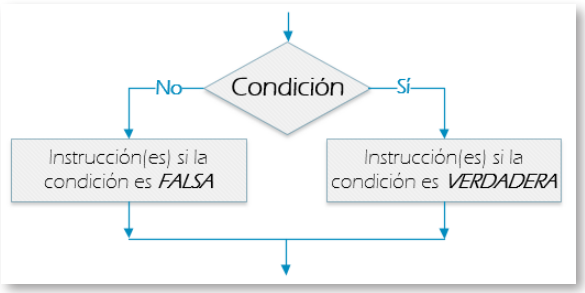


Figura 108. Sintaxis de estructura de decisión compuesta en diagramación libre.

Ejemplo 1. Desarrollar un algoritmo que permita mostrar un mensaje de alerta que indique si es suficiente o no el número de sillas que se llevarán a un evento. Para hacer esta validación se tiene que ingresar el número de sillas y el número de personas que van a asistir.

En la figura 109 se muestra el pseudocódigo en PSeInt y en la figura 110 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Ingrese número de sillas: "
3      Leer Num_Sillas
4      Imprimir "Ingrese número de asistentes: "
5      Leer Num_Asist
6      Si (Num_Sillas >= Num_Asist) Entonces
7          Imprimir "Número de sillas es suficiente. "
8      Sino
9          Imprimir "Número de sillas no es suficiente."
10     FinSi
11 FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingrese número de sillas:
> 120
Ingrese número de asistentes:
> 100
Número de sillas es suficiente.
```

Ejecución Condición Falsa

```
Ingrese número de sillas:
> 100
Ingrese número de asistentes:
> 200
Número de sillas no es suficiente.
```

Figura 109. Primer ejemplo de estructura de decisión compuesta en PSeInt.

main.py

```
1  print("Ingrese número de sillas: ")
2  num_sillas = input()
3  print("Ingrese número de asistentes: ")
4  num_asist = input()
5  if (num_sillas >= num_asist):
6      print("Número de sillas es suficiente.")
7  else:
8      print("Número de sillas no es suficiente.")
```

```
Ingrese número de sillas:
120
Ingrese número de asistentes:
100
Número de sillas es suficiente.
```

```
Ingrese número de sillas:
100
Ingrese número de asistentes:
120
Número de sillas no es suficiente.
```

Figura 110. Primer ejemplo de estructura de decisión compuesta en Python.

En la ejecución del lado izquierdo se ingresa 120 en el número de sillas y 100 asistentes, siendo 120 una cantidad suficiente para el evento y dando como resultado *Verdadero* al momento de evaluar la condición; mientras que en la ejecución del lado derecho izquierdo se ingresa 100 en el número de sillas y 120 asistentes, siendo 100 una cantidad insuficiente para el evento y dando como resultado *Falso*. Observe que en cada caso imprime una instrucción diferente dependiendo la evaluación realizada por la condición.

En la figura 111, se muestra la solución de este ejercicio en diagramación libre.

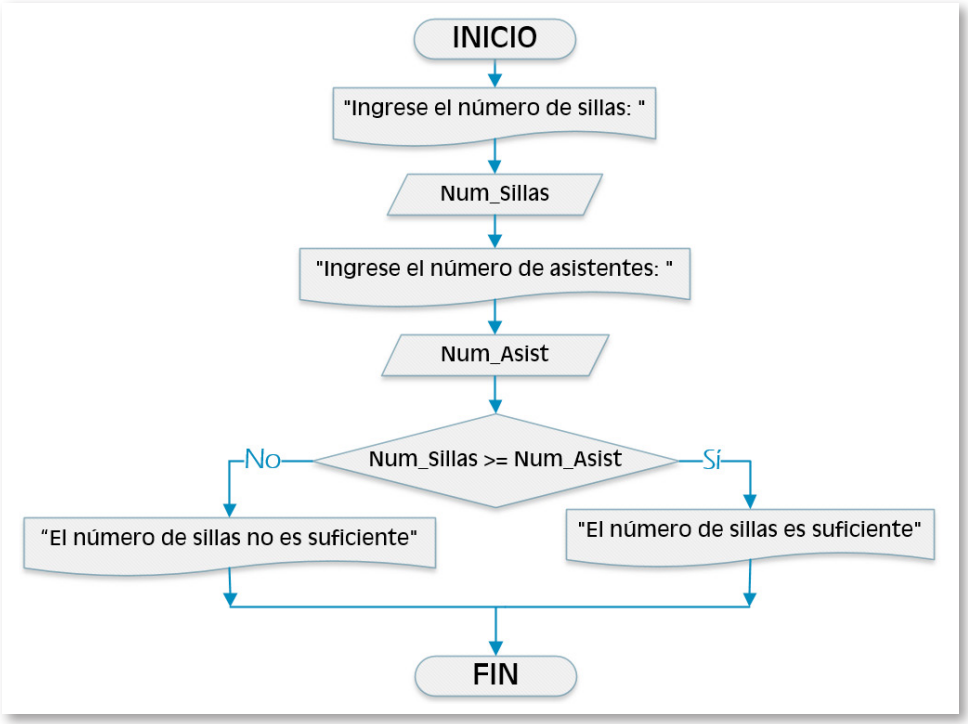


Figura 111. Estructura de decisión compuesta en diagramación libre.

Ejemplo 2. El vigilante de un casino de la ciudad le pregunta a la persona que va ingresar su edad, si la persona es mayor de edad le da la bienvenida y le avisa que puede ingresar, mientras que a los menores de edad les avisa que tienen prohibido el ingreso al casino.

Realizar un algoritmo que lea la edad de la persona que desea ingresar y simule el proceso realizado por este vigilante en el casino, mostrando mensajes de "Bienvenido, puede ingresar" o "Disculpe, tiene prohibido el ingreso".

En la figura 112 se muestra el pseudocódigo en PSeInt y en la figura 113 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese su edad: "
    Leer Edad
    Si (Edad >= 18) Entonces
        Imprimir "Bienvenido, puede ingresar"
    Sino
        Imprimir "Disculpe, tiene prohibido el ingreso"
    FinSi
FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingrese su edad:
> 25
Bienvenido, puede ingresar
```

Ejecución Condición Falsa

```
Ingrese su edad:
> 17
Disculpe, tiene prohibido el ingreso
```

Figura 112. Segundo ejemplo de estructura de decisión compuesta en PSeInt.

main.py

```
print("Ingrese su edad: ")
edad = int(input())
if (edad >= 18):
  print("Bienvenido, puede ingresar")
else:
  print("Disculpe, tiene prohibido el ingreso")
```

```
Ingrese su edad:
25
Bienvenido, puede ingresar
```

```
Ingrese su edad:
17
Disculpe, tiene prohibido el ingreso
```

Figura 113. Segundo ejemplo de estructura de decisión compuesta en Python.

3.2.3 Estructura de decisión múltiple

Es una estructura que se usa cuando un algoritmo requiere ejecutar más de dos acciones con base en un resultado arrojado después de evaluar una condición. Su sintaxis consiste en hacer varias copias de la estructura anterior. Se inicia igual que la anterior con un *Si* y luego se deben agregar instrucciones de *Sino / Si* para cada una de las siguientes acciones, excepto la última, que algunas veces se puede ignorar porque, en muchos casos, resulta obvia dicha pregunta.

Tenga en cuenta que el número de *Si* de una estructura de decisión, tiene que ser igual al número de *FinSi*. Cada estructura debe tener su indentación correspondiente. Quedando de forma paralela, el *Si,* el *Sino* y el *FinSi*.

La sintaxis de la estructura de decisión múltiple es la siguiente:

Si (Condición_1) Entonces
  Instrucciones (si Condición_1 es *Verdadera*).
Sino
  Si (Condición_2) Entonces
    Instrucciones (si Condición_2 es *Verdadera*).
  Sino
    Si (Condición_N) Entonces
      Instrucciones (si Condición_N es *Verdadera*).
    Sino
      Instrucciones (si ninguna condición es *Verdadera*).
    FinSi
  FinSi
FinSi

La sintaxis de la estructura de decisión múltiple en diagramación libre se puede ver en la figura 114.

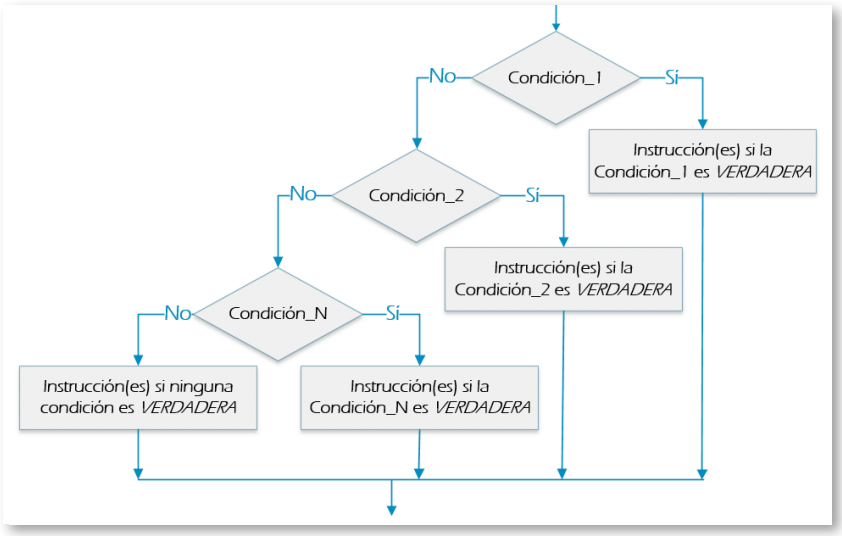


Figura 114. Sintaxis de estructura de decisión múltiple en diagramación libre.

Si la primera condición se cumple, realiza las instrucciones que hay dentro de esa condición y se va directo al *FinSi* sin pasar por las demás. Si no cumple la condición pasará por cada una de las siguientes condiciones que se encuentran hacia abajo, repitiendo el mismo proceso de validación.

Ejemplo 1. El costo de un cuaderno depende del número de hojas que tiene. Estos costos están consolidados en la tabla 40.

Tabla 40.
Costo cuadernos. Ejemplo1. Estructura de decisión múltiple

Número de hojas	Costo ($)
250	16 000
100	11 000
80	8 000
50	4 500

Realizar un algoritmo que permita leer el número de hojas de un cuaderno e imprima su costo respectivo. Si se produce un error al ingresar el número de hojas, se debe imprimir un mensaje de error indicando que el número de hojas es incorrecto.

En la figura 115 se muestra el pseudocódigo en PSeInt y en la figura 116 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Número de hojas del cuaderno?: "
    Leer Num_Hojas
    Si (Num_Hojas = 250) Entonces
        Imprimir "El cuaderno cuesta $16.000."
    Sino
        Si (Num_Hojas = 100) Entonces
            Imprimir "El cuaderno cuesta $11.000."
        Sino
            Si (Num_Hojas = 80) Entonces
                Imprimir "El cuaderno cuesta $8.000."
            Sino
                Si (Num_Hojas = 50) Entonces
                    Imprimir "El cuaderno cuesta $4.500."
                Sino
                    Imprimir "Número de hojas incorrecto."
                FinSi
            FinSi
        FinSi
    FinSi
FinAlgoritmo
```

Ejecución Condición 3 Verdadera

```
¿Número de hojas del cuaderno?:
> 80
El cuaderno cuesta $8.000.
```

Ejecución Todas Condiciones Falsas

```
¿Número de hojas del cuaderno?:
> 75
Número de hojas incorrecto.
```

Figura 115. Primer ejemplo de estructura de decisión múltiple en PSeInt.

En la ejecución del lado izquierdo se ingresa un número de hojas igual a 80, siendo la tercera condición *Verdadera* mostrando el costo de un cuaderno de 80 hojas; mientras que en la ejecución de la derecha se ingresa un número de hojas igual a 75, mostrando el último mensaje, ya que todas las condiciones son *Falsas*.

main.py

```
print("¿Número de hojas del cuaderno?: ")
num_hojas = int(input())
if (num_hojas == 250):
  print("El cuaderno cuesta $16.000.")
else:
  if (num_hojas == 100):
    print("El cuaderno cuesta $11.000.")
  else:
    if (num_hojas == 80):
      print("El cuaderno cuesta $8.000.")
    else:
      if (num_hojas == 50):
        print("El cuaderno cuesta $4.500.")
      else:
        print("Número de hojas incorrecto.")
```

```
¿Número de hojas del cuaderno?:
80
El cuaderno cuesta $8.000.
```

```
¿Número de hojas del cuaderno?:
75
Número de hojas incorrecto.
```

Figura 116. Primer ejemplo de estructura de decisión múltiple en Python.

En la figura 117 se muestra la solución del ejercicio en diagramación libre.

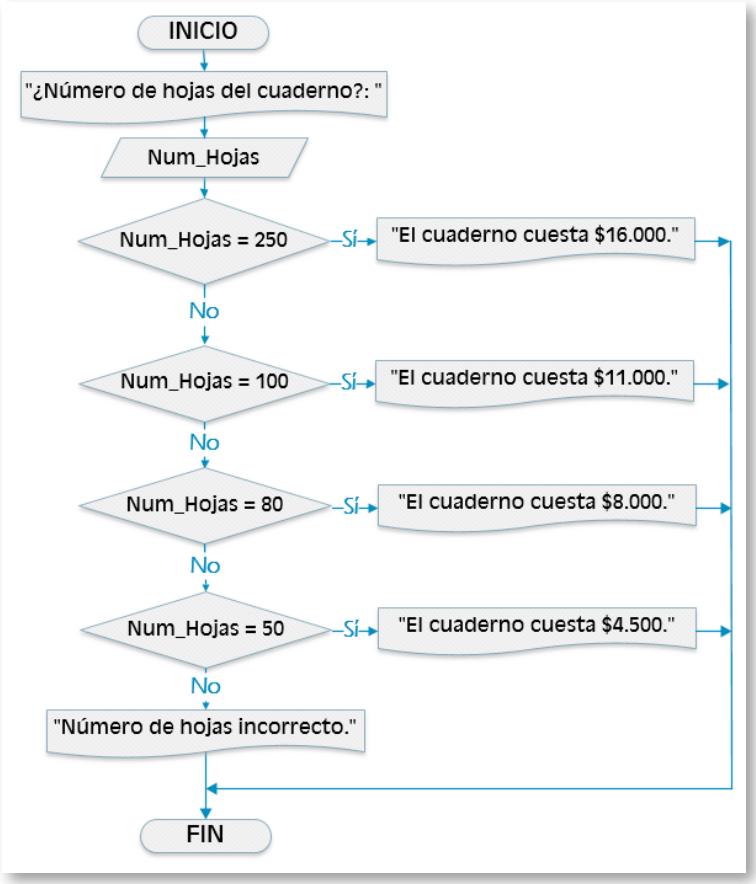


Figura 117. Estructura de decisión múltiple en diagramación libre.

Ejemplo 2. Hacer un algoritmo que lea 4 números e imprima el mayor de estos. Tenga en cuenta que todos los números leídos van a ser diferentes.

En la figura 118 se muestra el pseudocódigo en PSeInt y en la figura 119 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Escribir "Digite número 1:"
    Leer Num1
    Escribir "Digite número 2:"
    Leer Num2
    Escribir "Digite número 3:"
    Leer Num3
    Escribir "Digite número 4:"
    Leer Num4
    Si (Num1> Num2 Y Num1 > Num3 Y Num1 > Num4) Entonces
        May = Num1
    Sino
        Si (Num2 > Num1 Y Num2 > Num3 Y Num2 > Num4) Entonces
            May = Num2
        Sino
            Si (Num3 > Num1 Y Num3 > Num2 Y Num3 > Num4) Entonces
                May = Num3
            Sino
                May = Num4
            FinSi
        FinSi
    FinSi
    Escribir "El número mayor es el: ", May
FinAlgoritmo
```

Ejecución Condición 3 Verdadera

```
Digite número 1:> 2
Digite número 2:> 5
Digite número 3:> 8
Digite número 4:> 3
El número mayor es el: 8
```

Ejecución Todas Condiciones Falsas

```
Digite número 1:> 2
Digite número 2:> 3
Digite número 3:> 5
Digite número 4:> 8
El número mayor es el: 8
```

Figura 118. Segundo ejemplo de estructura de decisión múltiple en PSeInt.

main.py

```
1  print("Digite número 1:")
2  num1 = int(input())
3  print("Digite número 2:")
4  num2 = int(input())
5  print("Digite número 3:")
6  num3 = int(input())
7  print("Digite número 4:")
8  num4 = int(input())
9  if (num1>num2 and num1>num3 and num1>num4):
10   may = num1
11 else:
12   if (num2>num1 and num2>num3 and num2>num4):
13     may = num2
14   else:
15     if (num3>num1 and num3>num2 and num3>num4):
16       may = num3
17     else:
18       may = num4
19 print("El número mayor es el: ",may)
```

```
Digite número 1: 2
Digite número 2: 5
Digite número 3: 8
Digite número 4: 3
El número mayor es el:  8
```

```
Digite número 1: 2
Digite número 2: 3
Digite número 3: 5
Digite número 4: 8
El número mayor es el:  8
```

Figura 119. Segundo ejemplo de estructura de decisión múltiple en Python.

En las ejecuciones anteriores, a pesar de mostrar el mismo resultado, es importante entender que al ingresar los números 2, 5, 8 y 3, se cumple la tercera condición (el tercer número: *num3,* es mayor que los otros tres); mientras que la del lado derecho no se cumple ninguna condición y se iría a ejecutar la línea 18 donde a la variable *may* se le asigna el cuarto número: *num4*.

Existen métodos para hallar el mayor o el menor valor de un grupo de valores. Por ejemplo, comparando cada variable con las demás o haciendo condiciones anidadas (tema siguiente). Pero en la figura 120 se muestra otra solución para hallar el elemento mayor de un grupo de valores.

```
1  Algoritmo Inicio
2      Imprimir "Digite número 1:"
3      Leer Num1
4      Imprimir "Digite número 2:"
5      Leer Num2
6      Imprimir "Digite número 3:"
7      Leer Num3
8      Imprimir "Digite número 4:"
9      Leer Num4
10     May = Num1
11     Si (Num2 > May)
12         May = Num2
13     FinSi
14     Si (Num3 > May)
15         May = Num3
16     FinSi
17     Si (Num4 > May)
18         May = Num4
19     FinSi
20     Imprimir "El número mayor es: ", May
21 FinAlgoritmo
```

Ejecución

```
Digite número 1:
> 2
Digite número 2:
> 5
Digite número 3:
> 8
Digite número 4:
> 3
El número mayor es: 8
```

Figura 120. Resumen de una estructura de decisión múltiple.

3.2.4 Estructura de decisión anidada

La estructura de decisión anidada se usa cuando una pregunta va dentro de otra(s), sin estar separadas por la instrucción *Sino*. Este tipo de estructura se usa cuando se necesita que se cumpla una condición para seguir con la otra.

Estas estructuras pueden reemplazar las estructuras de decisión que usan el operador *Y,* así como en algunos casos donde se requiera agrupar o resumir estructuras de decisión múltiple.

Su sintaxis es la siguiente:

Si (Condición_1) Entonces
  Si (Condición_2) Entonces
    Si (Condición_N) Entonces
      Instrucciones (si las condiciones son *Verdaderas*).
    Sino
      Instrucciones (si Condición_N es *Falsa*).
    FinSi
  Sino
    Instrucciones (si Condición_2 es *Falsa*).
  FinSi
Sino
    Instrucciones (si Condición_1 es *Falsa*).
FinSi

La sintaxis de la estructura anidada en diagramación libre se ve en la figura 121.

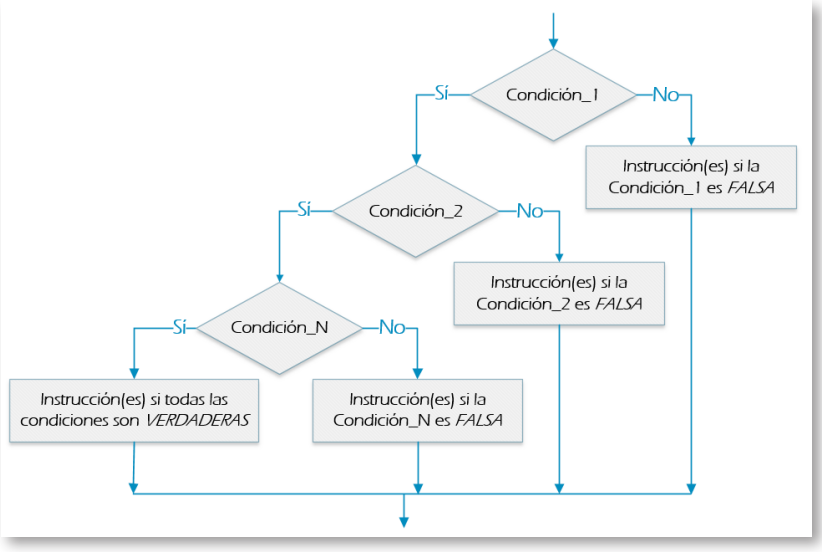


Figura 121. Sintaxis de estructura de decisión anidada en diagramación libre.

La única forma de ejecutar las primeras instrucciones es en el caso en el cual se cumplan todas las condiciones. Esta estructura es muy similar a la anterior; pero en esta estructura anidada se abren todas las preguntas con la palabra *Si* y, en la mayoría de los casos, se agregan cada uno de los *Sino* y *FinSi*. Comparar la sintaxis con la estructura de decisión múltiple para entender mejor la explicación.

Ejemplo 1. Calcular la suma de dos números solo si los dos números son pares y positivos. En caso de no poder realizar la operación mostrar mensajes indicando los motivos por los cuales no pudo realizarla. Por ejemplo que, el primer número no sea par o el segundo no sea par o el primero no sea positivo o que el segundo número no sea positivo, evitaría la ejecución de la suma.

Un número es par si el resultado de dividir ese número entre 2 da residuo igual 0 (Numero *MOD* 2 = 0). Por lo tanto, se recomienda usar la operación módulo y un número es positivo si es mayor que cero.

En la figura 122 se muestra el pseudocódigo en PSeInt y en la figura 123 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Ingrese el primer número: "
3      Leer Num1
4      Imprimir "Ingrese el segundo número: "
5      Leer Num2
6      Si (Num1 Mod 2=0 Y Num2 Mod 2=0 Y Num1>0 Y Num2>0) Entonces
7          Suma = Num1 + Num2
8          Imprimir "La suma es: ", Suma
9      Sino
10         Imprimir "No cumple con las condiciones establecidas."
11     FinSi
12 FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingrese el primer número:
> 2
Ingrese el segundo número:
> 4
La suma es: 6
```

Ejecución Condición Falsa

```
Ingrese el primer número:
> 2
Ingrese el segundo número:
> 3
No cumple con las condiciones establecidas.
```

Figura 122. Primer ejemplo de estructura de decisión con conjunción en PSeInt.

main.py

```
1  print("Ingrese el primer número: ")
2  num1 = int(input())
3  print("Ingrese el segundo número: ")
4  num2 = int(input())
5  if (num1%2==0 and num2%2==0 and num1>0 and num2>0):
6      suma = num1+num2
7      print("La suma es: ",suma)
8  else:
9      print("No cumple con las condiciones establecidas.")
```

```
Ingrese el primer número:
2
Ingrese el segundo número:
4
La suma es:  6
```

```
Ingrese el primer número:
2
Ingrese el segundo número:
3
No cumple con las condiciones establecidas.
```

Figura 123. Primer ejemplo de estructura de decisión con conjunción en Python

Para que este ejercicio pueda calcular la suma, tienen que cumplirse las cuatro condiciones antes mencionadas. Pero el mensaje no es específico al momento de indicar el motivo por el cual no se pudo realizar el cálculo; por lo tanto, se recomienda plantear una solución con estructuras de decisión anidadas como la siguiente.

En esta propuesta se va a leer el primer número, cuando cumpla las dos condiciones de ser par y positivo, se procede a leer el segundo y si cumple esas mismas condiciones, se pasa a calcular e imprimir la suma.

En la figura 124 se muestra el pseudocódigo en PSeInt y en la figura 125 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese el primer número: "
    Leer Num1
    Si (Num1 Mod 2 = 0) Entonces
        Si (Num1 > 0) Entonces
            Imprimir "Ingrese el segundo número: "
            Leer Num2
            Si (Num2 Mod 2 = 0) Entonces
                Si (Num2 > 0) Entonces
                    Suma = Num1 + Num2
                    Imprimir "La suma es: ", Suma
                Sino
                    Imprimir "El segundo número no es positivo."
                FinSi
            Sino
                Imprimir "El segundo número no es par."
            FinSi
        Sino
            Imprimir "El primer número no es positivo."
        FinSi
    Sino
        Imprimir "El primer número no es par."
    FinSi
FinAlgoritmo
```

Ejecución Todas Condiciones Verdaderas

```
Ingrese el primer número:
> 2
Ingrese el segundo número:
> 4
La suma es: 6
```

Ejecución Condición 3 Falsa

```
Ingrese el primer número:
> 2
Ingrese el segundo número:
> 3
El segundo número no es par.
```

Figura 124. Primer ejemplo de estructura de anidada en PSeInt.

main.py

```
print("Ingrese el primer número: ")
num1 = int(input())
if (num1%2==0):
  if (num1>0):
    print("Ingrese el segundo número: ")
    num2 = int(input())
    if (num2%2==0):
      if (num2>0):
        suma = num1+num2
        print("La suma es: ",suma)
      else:
        print("El segundo número no es positivo.")
    else:
      print("El segundo número no es par.")
  else:
    print("El primer número no es positivo.")
else:
  print("El primer número no es par.")
```

```
Ingrese el primer número:
2
Ingrese el segundo número:
4
La suma es:  6
```

```
Ingrese el primer número:
2
Ingrese el segundo número:
3
El segundo número no es par.
```

Figura 125. Primer ejemplo de estructura de anidada en Python.

Las estructuras de decisión pueden ser anidadas de forma horizontal como en el ejemplo de estructuras de decisión con la conjunción, donde se colocan varias condiciones dentro de una pregunta, separadas con los operadores: Λ (Y) y V (O), o pueden ser vertical, colocando una pregunta dentro de otra como el primer ejemplo de estructuras de decisión anidadas.

A pesar de ser similares las estructuras de decisión compuestas y anidadas; en la segunda se ahorran instrucciones en cada pregunta y los algoritmos con estas estructuras son mucho más eficientes. Para entender mejor el funcionamiento de la estructura de decisión, especialmente, la estructura de decisión anidada se hace un paso a paso dependiendo de los valores ingresados.

En el caso donde se digita ambos números pares positivos, donde todas las condiciones son *Verdaderas*, ver secuencia de pasos en figura 126.

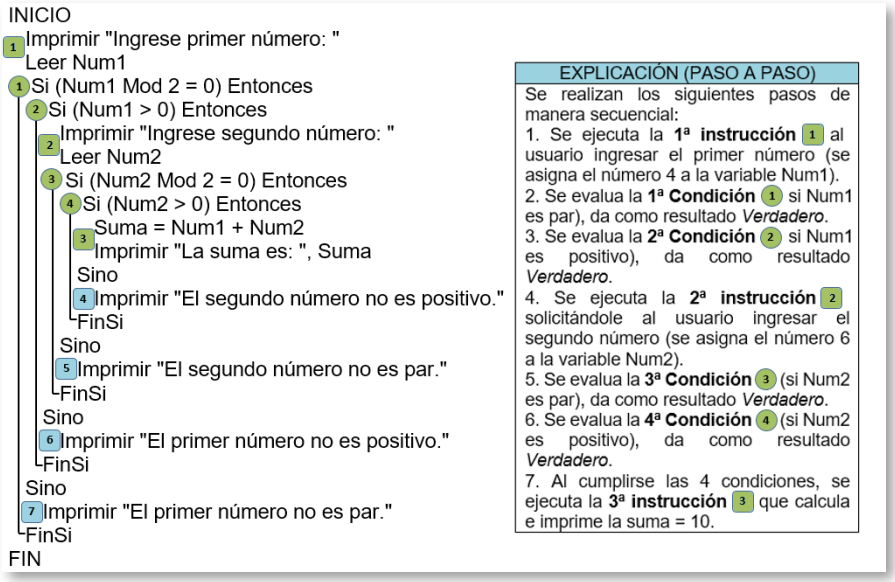


Figura 126. Primer seguimiento a una estructura de decisión anidada.

En el caso donde se digita, el primer número par positivo (1ª y 2ª Condición *Verdadera*) y el segundo número par negativo (3ª Condición *Verdadera* y 4ª Condición *Falsa*), ver secuencia de pasos en la figura 127.

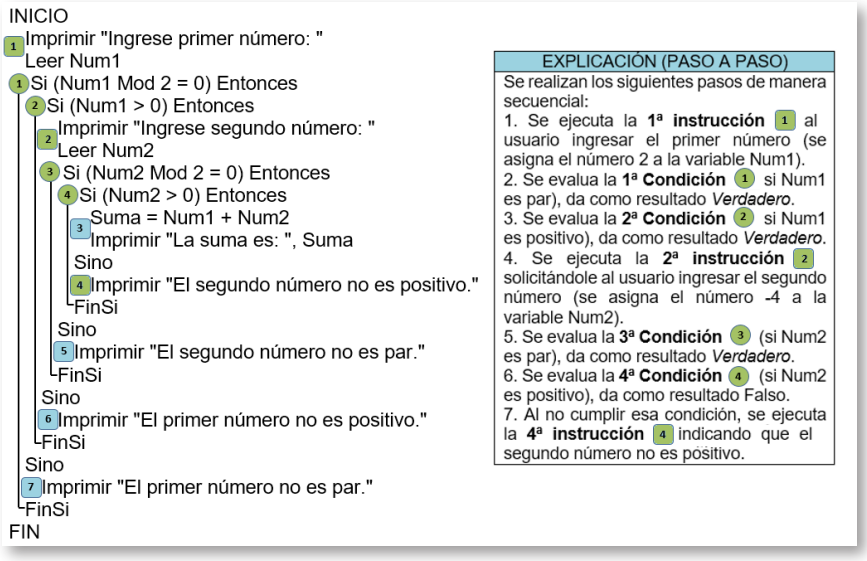


Figura 127. Segundo seguimiento a una estructura de decisión anidada.

En el caso donde se digita, el primer número par positivo (1ª y 2ª Condición *Verdadera*) y el segundo impar positivo (3ª Condición *Falsa*), ver secuencia de pasos en la figura 128.

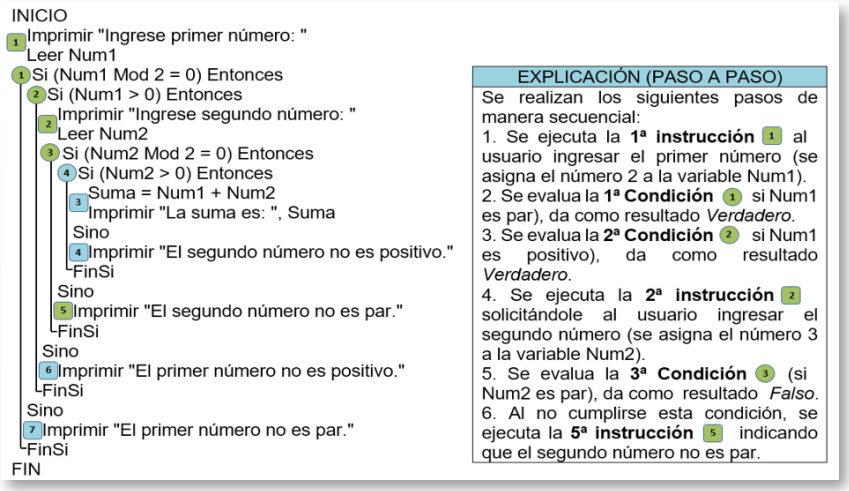


Figura 128. Tercer seguimiento a una estructura de decisión anidada.

En el caso donde se digita, el primer número par negativo (1ª Condición *Verdadera* y 2ª Condición *Falsa*), ver secuencia de pasos en la figura 129.

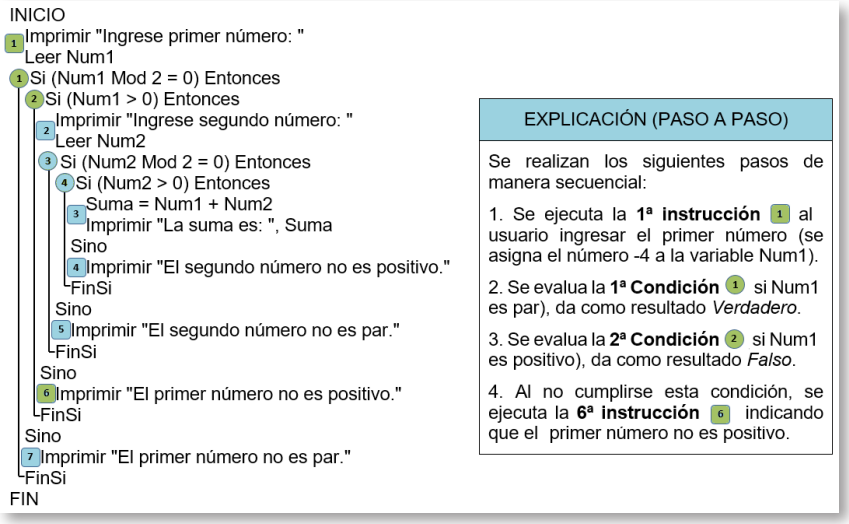


Figura 129. Cuarto seguimiento a una estructura de decisión anidada.

En el caso donde se digita, el primer número impar positivo (1ª *Falsa*), ver secuencia de pasos en la figura 130.

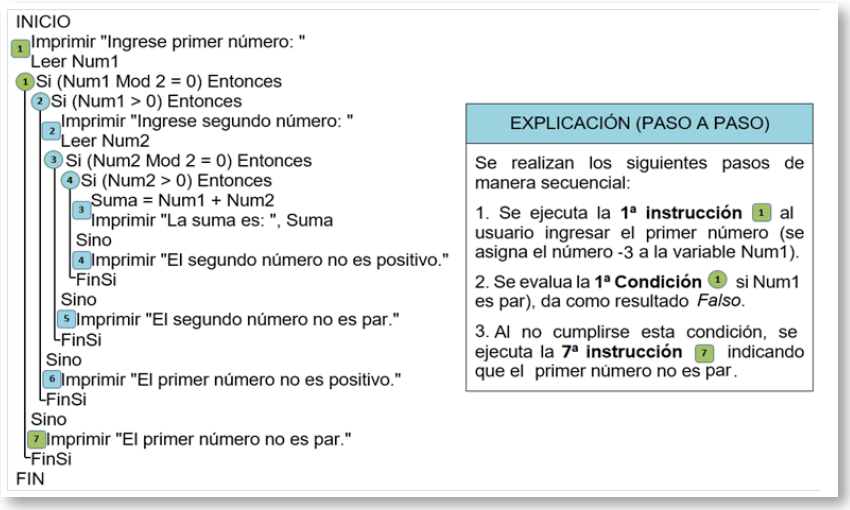


Figura 130. Quinto seguimiento a una estructura de decisión anidada.

En la figura 131 se muestra la solución de este ejercicio en diagramación libre.

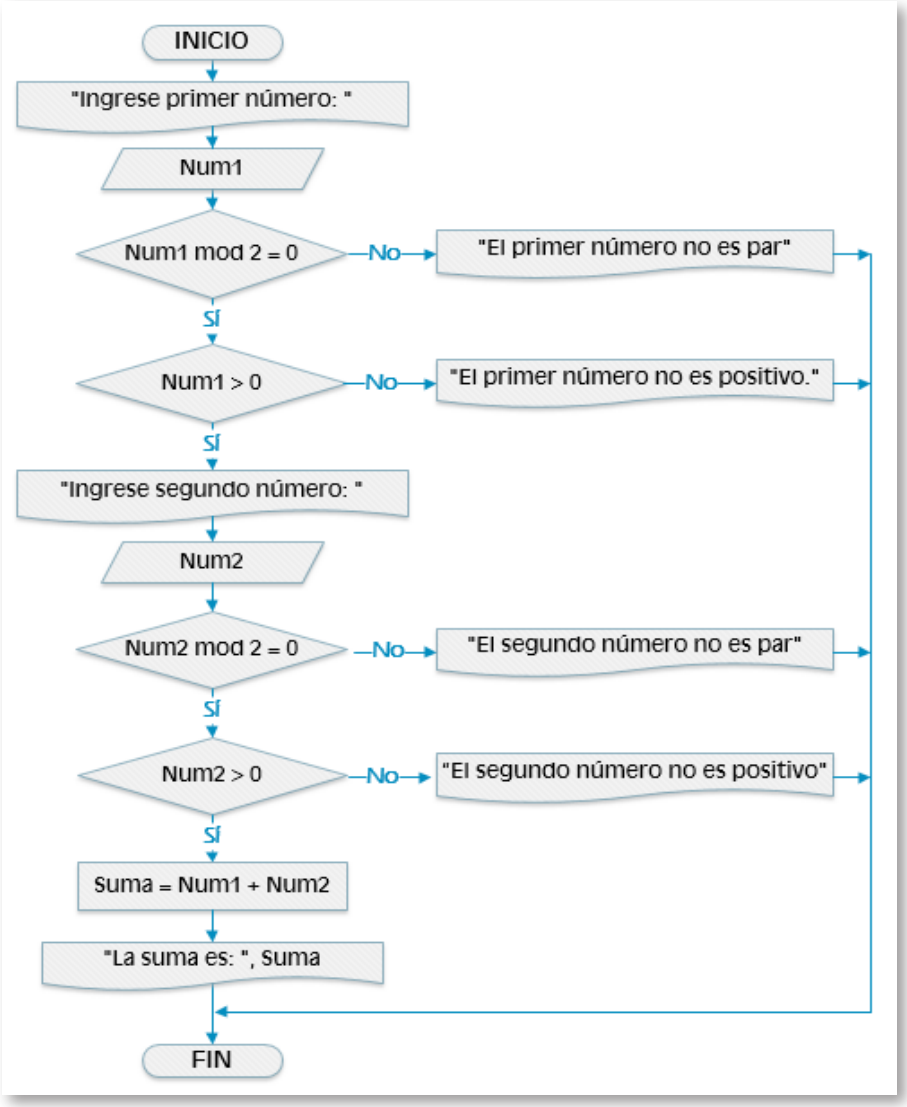


Figura 131. Estructura de decisión anidada en diagramación libre.

Ejemplo 2. Desarrollar un algoritmo que permita leer el sexo y el programa al que pertenece un estudiante de la institución universitaria. Imprimir un mensaje que indique alguna de las siguientes situaciones: si el estudiante es hombre y estudia Sistemas, o es hombre y estudia Electrónica, o es hombre y estudia Maquinaria, o es hombre y estudia Industrial, o es una mujer y estudia algunos de estos mismos programas.

En la figura 132 se muestra el pseudocódigo en PSeInt y en la figura 133 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "¿Sexo (M:Masculino - F:Femenino)?: "
    Leer Sex
    Sex = mayusculas(Sex)
    Imprimir "¿Programa (S:Sistemas - E:Electrónica - M:Maquinaria - I:Industrial)?: "
    Leer Pgma
    Pgma = mayusculas(Pgma)
    Si (Sex = "M") Entonces
        Si (Pgma = "S") Entonces
            Imprimir "Hombre que estudia Sistemas."
        Sino
            Si (Pgma = "E") Entonces
                Imprimir "Hombre que estudia Electrónica."
            Sino
                Si (Pgma = "M") Entonces
                    Imprimir "Hombre que estudia Maquinaria."
                Sino
                    Si (Pgma = "I") Entonces
                        Imprimir "Hombre que estudia Industrial."
                    FinSi
                FinSi
            FinSi
        FinSi
    Sino
        Si (Pgma = "S") Entonces
            Imprimir "Mujer que estudia Sistemas."
        Sino
            Si (Pgma = "E") Entonces
                Imprimir "Mujer que estudia Electrónica."
            Sino
                Si (Pgma = "M") Entonces
                    Imprimir "Mujer que estudia Maquinaria."
                Sino
                    Si (Pgma = "I") Entonces
                        Imprimir "Mujer que estudia Industrial. "
                    FinSi
                FinSi
            FinSi
        FinSi
    FinSi
FinAlgoritmo
```

Ejecución 4a. Condición Verdadera

```
¿Sexo (M:Masculino - F:Femenino)?:
> M
¿Programa (S:Sistemas - E:Electrónica - M:Maquinaria - I:Industrial)?:
> M
Hombre que estudia Maquinaria.
```

Ejecución 6a. Condición Verdadera

```
¿Sexo (M:Masculino - F:Femenino)?:
> F
¿Programa (S:Sistemas - E:Electrónica - M:Maquinaria - I:Industrial)?:
> S
Mujer que estudia Sistemas.
```

Figura 132. Segundo ejemplo de estructura de decisión anidada en PSeInt.

main.py

```
print("¿Sexo (M:Masculino - F:Femenino)?: ")
sex = input()
sex = str.upper(sex)
print("¿Programa (S:Sistemas - E:Electrónica - M:Maquinaria - I:Industrial)?: ")
pgma = input()
pgma = str.upper(pgma)
if (sex=="M"):
  if (pgma=="S"):
    print("Hombre que estudia Sistemas.")
  else:
    if (pgma=="E"):
      print("Hombre que estudia Electrónica.")
    else:
      if (pgma=="M"):
        print("Hombre que estudia Maquinaria.")
      else:
        if (pgma=="I"):
          print("Hombre que estudia Industrial.")
else:
  if (pgma=="S"):
    print("Mujer que estudia Sistemas.")
  else:
    if (pgma=="E"):
      print("Mujer que estudia Electrónica.")
    else:
      if (pgma=="M"):
        print("Mujer que estudia Maquinaria.")
      else:
        if (pgma=="I"):
          print("Mujer que estudia Industrial. ")
```

```
¿Sexo (M:Masculino - F:Femenino)?:
M
¿Programa (S:Sistemas - E:Electrónica - M:Maquinaria - I:Industrial)?:
M
Hombre que estudia Maquinaria.
```

```
¿Sexo (M:Masculino - F:Femenino)?:
F
¿Programa (S:Sistemas - E:Electrónica - M:Maquinaria - I:Industrial)?:
S
Mujer que estudia Sistemas.
```

Figura 133. Segundo ejemplo de estructura de decisión anidada en Python.

3.3 Ejercicios resueltos

Para entender mejor el tema de estructuras de decisión y empezar a diferenciar sus tipos, se muestra una serie de soluciones a problemas planteados en PSeInt y Python.

3.3.1 Primer ejercicio

Desarrollar un algoritmo que permita calcular el cuadrado de un número, teniendo en cuenta que solo se debe hacer el proceso si se ingresa 10, 20, 30 o 40.; en los demás casos hacer el cuadrado igual a 0.

En la figura 134 se muestra el pseudocódigo en PSeInt y en la figura 135 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Ingrese un número:"
3      Leer Num
4      Si (Num=10 O Num=20 O Num=30 O Num=40) Entonces
5          Cuad = Num ^ 2
6      Sino
7          Cuad = 0
8      FinSi
9      Imprimir "El cuadrado es: ", Cuad
10 FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingrese un número:
> 20
El cuadrado es: 400
```

Ejecución Condición Falsa

```
Ingrese un número:
> 80
El cuadrado es: 0
```

Figura 134. Primer ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
1  print("Ingrese un número:")
2  num = float(input())
3  if (num == 10 or num == 20 or num == 30 or num == 40):
4      cuad = num ** 2
5  else:
6      cuad = 0
7  print("El cuadrado es: ", cuad)
```

```
Ingrese un número:
20
El cuadrado es:  400.0
```

```
Ingrese un número:
80
El cuadrado es:  0
```

Figura 135. Primer ejercicio resuelto de estructuras de decisión en Python.

3.3.2 Segundo ejercicio

Hacer un programa para un laboratorio de química que lea un símbolo químico e imprimir el elemento al cual corresponde.

> Tenga presente que solo se cuenta con los elementos hidrógeno, oxígeno y nitrógeno. En los demás casos, imprima un mensaje que diga "Elemento no contemplado".

En la figura 136 se muestra el pseudocódigo en PSeInt y en la figura 137 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Ingrese el símbolo químico: "
3      Leer Simboloq
4      Si (Simboloq = "H") Entonces
5          Imprimir "El elemento es el hidrógeno."
6      Sino
7          Si (Simboloq = "O") Entonces
8              Imprimir "El elemento es el oxígeno."
9          Sino
10             Si (Simboloq = "N") Entonces
11                 Imprimir "El elemento es el nitrógeno."
12             SiNo
13                 Imprimir "Elemento no contemplado"
14             FinSi
15         FinSi
16     FinSi
17 FinAlgoritmo
```

Ejecución 1a. Condición Verdadera

```
Ingrese el símbolo químico:
> O
El elemento es el oxígeno.
```

Ejecución Ninguna Condición Verdadera

```
Ingrese el símbolo químico:
> K
Elemento no contemplado
```

Figura 136. Segundo ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese el símbolo químico: ")
simboloq = input()
if (simboloq == "H"):
    print("El elemento es el hidrógeno.")
elif (simboloq == "O"):
    print("El elemento es el oxígeno.")
elif (simboloq == "N"):
    print("El elemento es el nitrógeno.")
else:
    print("Elemento no contemplado")
```

```
Ingrese el símbolo químico:
O
El elemento es el oxígeno.
```

```
Ingrese el símbolo químico:
K
Elemento no contemplado.
```

Figura 137. Segundo ejercicio resuelto de estructuras de decisión en Python.

3.3.3 Tercer ejercicio

Resolver la siguiente ecuación por medio de un algoritmo, teniendo en cuenta que solo se puede realizar si la variable "r" es diferente de 2; en caso contrario hacer P = 1.

$$P = (r - 2)^3$$

En la figura 138 se muestra el pseudocódigo en PSeInt y en la figura 139 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese el valor de r: "
    Leer r
    Si (r <> 2) Entonces
        P = (r - 2) ^ 3
    SiNo
        P = 0
    FinSi
    Imprimir "El valor de P es: ", P
FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingrese el valor de r:
> 8.5
El valor de P es: 274.625
```

Ejecución Condición Falsa

```
Ingrese el valor de r:
> 2
El valor de P es: 1
```

Figura 138. Tercer ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese el valor de r: ")
r = float(input())
if (r != 2):
    p = (r - 2) ** 3
else:
    p = 0
print("El valor de P es: ", p)
```

```
Ingrese el valor de r:
8.5
El valor de P es:  274.625
```

```
Ingrese el valor de r:
2
El valor de P es:  1
```

Figura 139. Tercer ejercicio resuelto de estructuras de decisión en Python.

3.3.4 Cuarto ejercicio

Calcule el promedio de goles anotados por un jugador en cuatro encuentros, solo si la suma de estos es mayor a 10; de lo contrario imprima "No se pide determinar el promedio".

En la figura 140 se muestra el pseudocódigo en PSeInt y en la figura 141 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite cantidad de goles primer encuentro: "
    Leer Cg1
    Imprimir "Digite cantidad de goles segundo encuentro: "
    Leer Cg2
    Imprimir "Digite cantidad de goles tercer encuentro: "
    Leer Cg3
    Imprimir "Digite cantidad de goles cuarto encuentro: "
    Leer Cg4
    TotalGoles = Cg1 + Cg2 + Cg3 + Cg4
    Si (TotalGoles > 10) Entonces
        Prom = TotalGoles / 4
        Imprimir "Promedio de goles: ", Prom
    SiNo
        Imprimir "No se puede determinar el promedio."
    FinSi
FinAlgoritmo
```

Ejecución Condición Verdadera

```
Digite cantidad de goles primer encuentro:
> 8
Digite cantidad de goles segundo encuentro:
> 15
Digite cantidad de goles tercer encuentro:
> 35
Digite cantidad de goles cuarto encuentro:
> 20
Promedio de goles: 19.5
```

Ejecución Condición Falsa

```
Digite cantidad de goles primer encuentro:
> 2
Digite cantidad de goles segundo encuentro:
> 0
Digite cantidad de goles tercer encuentro:
> 3
Digite cantidad de goles cuarto encuentro:
> 1
No se puede determinar el promedio.
```

Figura 140. Cuarto ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Digite cantidad de goles primer encuentro: ")
cg1 = float(input())
print("Digite cantidad de goles segundo encuentro: ")
cg2 = float(input())
print("Digite cantidad de goles tercer encuentro: ")
cg3 = float(input())
print("Digite cantidad de goles cuarto encuentro: ")
cg4 = float(input())
Total_Goles = cg1 + cg2 + cg3 + cg4
if (Total_Goles > 10):
    prom = Total_Goles / 4
    print("Promedio de goles: ", prom)
else:
    print("No se pide determinar el promedio.")
```

```
Digite cantidad de goles primer encuentro:
8
Digite cantidad de goles segundo encuentro:
15
Digite cantidad de goles tercer encuentro:
35
Digite cantidad de goles cuarto encuentro:
20
Promedio de goles:  19.5
```

```
Digite cantidad de goles primer encuentro:
2
Digite cantidad de goles segundo encuentro:
0
Digite cantidad de goles tercer encuentro:
3
Digite cantidad de goles cuarto encuentro:
1
No se puede determinar el promedio.
```

Figura 141. Cuarto ejercicio resuelto de estructuras de decisión en Python.

3.3.5 Quinto ejercicio

Desarrolle un algoritmo que ingrese dos números y luego los imprima de forma ascendente (primero se imprime el menor y luego el mayor).

En la figura 142 se muestra el pseudocódigo en PSeInt y en la figura 143 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese dos números: "
    Leer N1, N2
    Si (N1 < N2) Entonces
        Imprimir N1, ", ", N2
    Sino
        Imprimir N2, ", ", N1
    FinSi
FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingrese dos números:
> 20
> 5
5, 20
```

Ejecución Condición Falsa

```
Ingrese dos números:
> 3
> 7
3, 7
```

Figura 142. Quinto ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese dos números: ")
n1 = int(input())
n2 = int(input())
if (n1 < n2):
  print(n1,", ",n2)
else:
  print(n2,", ",n1)
```

```
Ingrese dos números:
20
5
5 ,  20
```

```
Ingrese dos números:
3
7
3 ,  7
```

Figura 143. Quinto ejercicio resuelto de estructuras de decisión en Python.

3.3.6 Sexto ejercicio

Desarrollar un algoritmo que resuelva la siguiente ecuación (las restricciones son: las variables *b* y *c* no pueden ser iguales a cero y *a* y *b* deben ser diferentes).

$$Ec = \frac{-b + a}{2bc}$$

n la figura 144 se muestra el pseudocódigo en PSeInt y en la figura 145 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Ingresar valor de variable a: "
3      Leer a
4      Imprimir "Ingresar valor de variable b: "
5      Leer b
6      Imprimir "Ingresar valor de variable c: "
7      Leer c
8      Si (b <> 0 y C <> 0) Entonces
9          Ec = (b * (-1) + a) / (2 * b * c)
10         Imprimir "Resultado de la ecuación = ", Ec
11     Sino
12         Imprimir "Error. División por cero."
13     FinSi
14 FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingresar valor de variable a:
> 24
Ingresar valor de variable b:
> 15
Ingresar valor de variable c:
> 25
Resultado de la ecuación = 0.012
```

Ejecución Condición Falsa

```
Ingresar valor de variable a:
> 5
Ingresar valor de variable b:
> 8
Ingresar valor de variable c:
> 0
Error. División por cero.
```

Figura 144. Sexto ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingresar valor de variable a: ")
a = float(input())
print("Ingresar valor de variable b: ")
b = float(input())
print("Ingresar valor de variable c: ")
c = float(input())
if (b != 0 and c != 0):
    ec = (b * (-1) + a) / (2 * b * c)
    print("Resultado de la ecuación = ", ec)
else:
    print("Error. División por cero.")
```

```
Ingresar valor de variable a:
24
Ingresar valor de variable b:
15
Ingresar valor de variable c:
25
Resultado de la ecuación =  0.012
```

```
Ingresar valor de variable a:
5
Ingresar valor de variable b:
8
Ingresar valor de variable c:
0
Error. División por cero.
```

Figura 145. Sexto ejercicio resuelto de estructuras de decisión en PSeInt.

3.3.7 Séptimo ejercicio

Realizar un algoritmo que permita leer la temperatura de un día en grados centígrados. Imprimir un mensaje “Hace frío” si la temperatura es menor a 23°C; “Buen día” si la temperatura es mayor o igual que 23°C e inferior a 30°C; mientras que si la temperatura sea igual o superior a 30°C imprima “Hace calor”.

En la figura 146 se muestra el pseudocódigo en PSeInt y en la figura 147 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese temperatura en °C: "
    Leer Temp_Gc
    Si (Temp_Gc < 23) Entonces
        Imprimir "Hace frío."
    Sino
        Si (Temp_Gc < 30) Entonces
            Imprimir "Buen día."
        Sino
            Imprimir "Hace calor."
        FinSi
    FinSi
FinAlgoritmo
```

Ejecución 1a. Condición Verdadera

```
Ingrese temperatura en °C:
> -5
Hace frío.
```

Ejecución 2a. Condición Verdadera

```
Ingrese temperatura en °C:
> 28
Buen día.
```

Ejecución Ninguna Condición Verdadera

```
Ingrese temperatura en °C:
> 35
Hace calor.
```

Figura 146. Séptimo ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese la temperatura en °C: ")
temp_gc = float(input())
if (temp_gc < 23):
  print("Hace frío.")
elif (temp_gc < 30):
  print("Buen día.")
else:
  print("Hace calor.")
```

```
Ingrese la temperatura en °C:
-5
Hace frío.
```

```
Ingrese la temperatura en °C:
28
Buen día.
```

```
Ingrese la temperatura en °C:
35
Hace calor.
```

Figura 147. Séptimo ejercicio resuelto de estructuras de decisión en Python.

Esta es la forma más organizada para realizar pseudocódigos. Visualmente muestran orden, jerarquía y claridad. En algoritmos que son muy extensos y tienen muchas condiciones es muy útil, aunque todo es gusto del programador (por ejemplo,algunos programadores prefieren diagramación libre al pseudocódigo).

3.3.8 Octavo ejercicio

El profesor de matemáticas requiere de una solución a través de un algoritmo que encuentre el valor absoluto de un número determinado para aplicarlo a una de sus fórmulas. El valor absoluto de cualquier número es su equivalente positivo. Por ejemplo, 5 es el valor absoluto de 5 y de –5.

En la figura 148 se muestra el pseudocódigo en PSeInt y en la figura 149 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Introduzca un número: "
3      Leer Num
4      Si (Num > 0) Entonces
5          Vabs = Num
6      Sino
7          Vabs = Num * (-1)
8      FinSi
9      Imprimir "El valor absoluto es: ", Vabs
10 FinAlgoritmo
```

Ejecución Condición Verdadera

```
Introduzca un número:
> -8
El valor absoluto es: 8
```

Ejecución Condición Falsa

```
Introduzca un número:
> 8
El valor absoluto es: 8
```

Figura 148. Octavo ejercicio resuelto de estructuras de decisión en PSeInt.

```
main.py
1   print("Introduzca un número: ")
2   num = int(input())
3   if (num > 0):
4       vabs = num
5   else:
6       vabs = num * (-1)
7   print("El valor absoluto es: ", vabs)
```

```
Introduzca un número:
-8
El valor absoluto es:  8
```

```
Introduzca un número:
8
El valor absoluto es:  8
```

Figura 149. Octavo ejercicio resuelto de estructuras de decisión en Python.

3.3.9 Noveno ejercicio

Desarrollar un algoritmo que permita leer una nota entre 0.0 y 5.0. Imprimir su nota cualitativa equivalente, teniendo en cuenta la tabla 41.

Tabla 41.
Equivalencias de notas. Ejercicio 3.3.9.

Nota numérica	Nota cualitativa
Nota mayor o igual que 4.6 y menor o igual que 5.0	Excelente
Nota mayor o igual que 3.6 y menor que 4.6	Buena
Nota mayor o igual que 3.0 y menor que 3.6	Aceptable
Nota mayor o igual que 2.0 y menor que 3.0	Insuficiente
Nota menor que 2.0 y mayor o igual que 0.0	Deficiente

En la figura 150 se muestra el pseudocódigo en PSeInt y en la figura 151 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese una nota: "
    Leer Nota
    Si (Nota >= 4.6 Y Nota <= 5.0) Entonces
        Imprimir "Excelente."
    Sino
        Si (Nota < 4.6 Y Nota >= 3.6) Entonces
            Imprimir "Buena."
        Sino
            Si (Nota < 3.6 Y Nota >= 3.0) Entonces
                Imprimir "Aceptable."
            Sino
                Si (Nota < 3.0 Y Nota >= 2.0) Entonces
                    Imprimir "Insuficiente."
                Sino
                    Si (Nota < 2.0 Y Nota >=0) Entonces
                        Imprimir "Deficiente."
                    Sino
                        Imprimir "Error en la nota."
                    FinSi
                FinSi
            FinSi
        FinSi
    FinSi
FinAlgoritmo
```

Ejecución 3a. Condición Verdadera

```
Ingrese una nota:
> 3.5
Aceptable.
```

Ejecución Ninguna Condición Verdadera

```
Ingrese una nota:
> -4.2
Error en la nota.
```

Figura 150. Noveno ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese una nota: ")
nota = float(input())
if (nota >= 4.6 and nota <= 5.0):
  print("Excelente.")
elif (nota < 4.6 and nota >= 3.6):
  print("Buena.")
elif (nota < 3.6 and nota >= 3.0):
  print("Aceptable.")
elif (nota < 3.0 and nota >= 2.0):
  print("Insuficiente.")
elif (nota < 2.0 and nota >= 0):
  print("Deficiente.")
else:
  print("Error en la nota.")
```

```
Ingrese una nota:
3.5
Aceptable.
```

```
Ingrese una nota:
-4.2
Error en la nota.
```

Figura 151. Noveno ejercicio resuelto de estructuras de decisión en Python.

3.3.10 Décimo ejercicio

Una persona no tiene claridad sobre el dispositivo que va a comprar para su computadora. La decisión la tomará de acuerdo con una bonificación que recibirá de parte de la empresa donde labora. Si recibe menos de $ 50 000 de bonificación comprará una cámara web, si recibe entre $ 50 000 y $ 200 000 comprará un *subwoofer*; si recibe más de $ 200 000 y hasta $ 500 000 se comprará un disco externo, si recibe más de $ 500 000 y hasta $ 1 000 000 se comprará una multifuncional y si recibe más de $ 1 000 000 se comprará un proyector.

Hacer un algoritmo que permita ayudarle a esta persona a comprar un dispositivo.

En la figura 152 se muestra el pseudocódigo en PSeInt y en la figura 153 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese la bonificación $: "
    Leer Bon
    Si (Bon < 0) Entonces
        Imprimir "Error en bonificación. Debe ser un valor positivo."
    Sino
        Si (Bon <= 50000) Entonces
            Imprimir "Se recomienda comprar cámara web."
        Sino
            Si (Bon <= 200000) Entonces
                Imprimir "Se recomienda comprar subwoofer."
            Sino
                Si (Bon <= 500000) Entonces
                    Imprimir "Se recomienda comprar disco externo."
                Sino
                    Si (Bon <= 1000000) Entonces
                        Imprimir "Se recomienda comprar multifuncional."
                    Sino
                        Imprimir "Se recomienda comprar proyector."
                    FinSi
                FinSi
            FinSi
        FinSi
    FinSi
FinAlgoritmo
```

Ejecución 2a. Condición Verdadera

```
Ingrese la bonificación $:
> 35000
Se recomienda comprar cámara web.
```

Ejecución Ninguna Condición Verdadera

```
Ingrese la bonificación $:
> 2500000
Se recomienda comprar proyector.
```

Figura 152. Décimo ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```python
print("Ingrese la bonificación $: ")
bon = int(input())
if (bon < 0):
    print("Error en bonificación. Debe ser un valor
    positivo.")
elif (bon <= 50000):
    print("Se recomienda comprar cámara web.")
elif (bon <= 200000):
    print("Se recomienda comprar subwoofer.")
elif (bon <= 500000):
    print("Se recomienda comprar disco externo.")
elif (bon <= 1000000):
    print("Se recomienda comprar multifuncional.")
else:
    print("Se recomienda comprar proyector.")
```

```
Ingrese la bonificación $:
35000
Se recomienda comprar cámara web.
```

```
Ingrese la bonificación $:
2500000
Se recomienda comprar proyector.
```

Figura 153. Décimo ejercicio resuelto de estructuras de decisión en Python.

3.3.11 Décimo primer ejercicio

Desarrolle un algoritmo que permita leer un mes e imprima la estación del año correspondiente. Tenga en cuenta que noviembre, diciembre y enero son estaciones de invierno; febrero, marzo y abril son de primavera; mayo, junio y julio son de verano; agosto, septiembre y octubre son de otoño.

En la figura 154 se muestra el pseudocódigo en PSeInt y en la figura 155 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese el mes del año en minúsculas: "
    Leer Mes
    Mes = minusculas(Mes)
    Si (Mes = "noviembre" O Mes = "diciembre" O Mes = "enero") Entonces
        Imprimir "Estación: invierno."
    Sino
        Si (Mes = "febrero" O Mes = "marzo" O Mes = "abril") Entonces
            Imprimir "Estación: primavera. "
        Sino
            Si (Mes = "mayo" O Mes = "junio" O Mes = "julio") Entonces
                Imprimir "Estación: verano."
            Sino
                Si (Mes = "agosto" O Mes = "septiembre" O Mes="octubre") Entonces
                    Imprimir "Estación: otoño."
                Sino
                    Imprimir "Error al ingresar el mes."
                FinSi
            FinSi
        FinSi
    FinSi
FinAlgoritmo
```

Ejecución 4a. Condición Verdadera

```
Ingrese el mes del año en minúsculas:
> septiembre
Estación: otoño.
```

Ejecución Ninguna Condición Verdadera

```
Ingrese el mes del año en minúsculas:
> lunes
Error al ingresar el mes.
```

Figura 154. Décimo primer ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese el mes del año en minúsculas: ")
mes = input()
mes = str.lower(mes)
if (mes=="noviembre" or mes=="diciembre" or mes=="enero"):
  print("Estación: invierno.")
elif (mes=="febrero" or mes=="marzo" or mes=="abril"):
  print("Estación: primavera. ")
elif (mes=="mayo" or mes=="junio" or mes=="julio"):
  print("Estación: verano.")
elif (mes=="agosto" or mes=="septiembre" or mes=="octubre"):
  print("Estación: otoño.")
else:
  print("Error al ingresar el mes.")
```

```
Ingrese el mes del año en minúsculas:
septiembre
Estación: otoño.
```

```
Ingrese el mes del año en minúsculas:
lunes
Error al ingresar el mes.
```

Figura 155. Décimo primer ejercicio resuelto de estructuras de decisión en Python.

3.3.12 Décimo segundo ejercicio

Una nueva calculadora científica permite hallar un cálculo complejo como el de la siguiente ecuación: . Plantear una solución que haga este cálculo

teniendo en cuenta la restricción de la radicación. Si el resultado de esa operación es positivo, determinar la raíz, de lo contrario, imprimir un mensaje indicando que es una raíz imaginaria.

En la figura 156 se muestra el pseudocódigo en PSeInt y en la figura 157 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Ingrese el valor de X: "
3      Leer X
4      Operacion = 3 * 3.14159265 - X
5      Si (Operacion > 0) Entonces
6          Raiz_cuad = Operacion ^ 0.5
7          Imprimir "Raíz cuadrada: ", Raiz_cuad
8      Sino
9          Imprimir "Es una raíz imaginaria."
10     FinSi
11 FinAlgoritmo
```

Ejecución Condición Verdadera

```
Ingrese el valor de X:
> 3
Raíz cuadrada: 2.5347145697
```

Ejecución Condición Falsa

```
Ingrese el valor de X:
> 15
Es una raíz imaginaria.
```

Figura 156. Décimo segundo ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
1  print("Ingrese el valor de X: ")
2  x = float(input())
3  operacion = 3 * 3.14159265 - x
4  if (operacion > 0):
5      raiz_cuad = operacion ** 0.5
6      print("Raíz cuadrada: ", raiz_cuad)
7  else:
8      print("Es una raíz imaginaria.")
```

```
Ingrese el valor de X:
3
Raíz cuadrada:  2.5347145697
```

```
Ingrese el valor de X:
15
Es una raíz imaginaria.
```

Figura 157. Décimo segundo ejercicio resuelto de estructuras de decisión en Python.

3.3.13 Décimo tercer ejercicio

En las pruebas ICFES se presentan dos tipos pruebas: una de aptitud matemática y otra de lenguaje. Leer los puntajes obtenidos por un estudiante en cada prueba e imprimir en cuál obtuvo el mayor puntaje. Tenga en cuenta que ambos puntajes pueden ser iguales.

En la figura 158 se muestra el pseudocódigo en PSeInt y en la figura 159 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Digite resultado en prueba de matemáticas: "
3      Leer Prueba_Mat
4      Imprimir "Digite resultado en prueba de lenguaje: "
5      Leer Prueba_Leng
6      Si (Prueba_Mat > Prueba_Leng) Entonces
7          Imprimir "Obtuvo el mayor puntaje en matemáticas."
8      Sino
9          Si (Prueba_Leng > Prueba_Mat) Entonces
10             Imprimir "Obtuvo el mayor puntaje en lenguaje."
11         Sino
12             Imprimir "En ambas pruebas obtuvo el mismo puntaje."
13         FinSi
14     FinSi
15 FinAlgoritmo
```

Ejecución 2a. Condición Verdadera

```
Digite resultado en prueba de matemáticas:
> 25
Digite resultado en prueba de lenguaje:
> 30
Obtuvo el mayor puntaje en lenguaje.
```

Ejecución Ninguna Condición Verdadera

```
Digite resultado en prueba de matemáticas:
> 25
Digite resultado en prueba de lenguaje:
> 25
En ambas pruebas obtuvo el mismo puntaje.
```

Figura 158. Décimo tercer ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Digite el resultado en la prueba de matemáticas: ")
prueba_mat = input()
print("Digite el resultado en la prueba de lenguaje: ")
prueba_leng = input()
if (prueba_mat > prueba_leng):
  print("Obtuvo el mayor puntaje en matemáticas.")
else:
  if (prueba_leng > prueba_mat):
    print("Obtuvo el mayor puntaje en lenguaje.")
  else:
    print("En ambas pruebas obtuvo el mismo puntaje.")
```

```
Digite el resultado en la prueba de matemáticas:
25
Digite el resultado en la prueba de lenguaje:
30
Obtuvo el mayor puntaje en lenguaje.
```

```
Digite el resultado en la prueba de matemáticas:
25
Digite el resultado en la prueba de lenguaje:
25
En ambas pruebas obtuvo el mismo puntaje.
```

Figura 159. Décimo tercer ejercicio resuelto de estructuras de decisión en Python.

3.3.14 Décimo cuarto ejercicio

Elaborar un algoritmo que calcule la nota definitiva de un estudiante, teniendo en cuenta que la nota definitiva consta de cuatro notas que valen las dos primeras un 10% y las otras dos un 40%.

Imprimir la nota y un mensaje que diga "Ganó, ¡felicitaciones!", si la nota definitiva es mayor o igual a 3.0; "Perdió, ¡puede habilitar!", si perdió la materia con una nota definitiva menor a 3.0 y mayor o igual a 2.5; y "Perdió, ¡debe repetir!", si la nota definitiva es menor a 2.5. Se deben leer notas del estudiante, las cuales se encuentran entre 0.0 y 5.0.

En la figura 160 se muestra el pseudocódigo en PSeInt y en la figura 161 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite nombre del estudiante: "
    Leer Nom
    Imprimir "Digite nota 1: "
    Leer Not1
    Imprimir "Digite nota 2: "
    Leer Not2
    Imprimir "Digite nota 3: "
    Leer Not3
    Imprimir "Digite nota 4: "
    Leer Not4
    Ndef = Not1 * 0.1 + Not2 * 0.1 + Not3 * 0.4 + Not4 * 0.4
    Si (Ndef >= 3 Y Ndef <=5) Entonces
        Imprimir "Ganó, ¡felicitaciones!. Definitiva: ", Ndef
    Sino
        Si (Ndef >= 2.5) Entonces
            Imprimir "Perdió, ¡puede habilitar!. Definitiva: ", Ndef
        Sino
            Si (Ndef >= 0.0) Entonces
                Imprimir "Perdió, ¡debe repetir. Definitiva: ", Ndef
            FinSi
        FinSi
    FinSi
FinAlgoritmo
```

Ejecución 1a. Condición Verdadera

```
Digite nombre del estudiante: > Miguel
Digite nota 1: > 5.0
Digite nota 2: > 3.5
Digite nota 3: > 4.0
Digite nota 4: > 5.0
Ganó, ¡felicitaciones!. Definitiva: 4.45
```

Ejecución 3a. Condición Verdadera

```
Digite nombre del estudiante: > Luisa
Digite nota 1: > 1.5
Digite nota 2: > 2.3
Digite nota 3: > 1.8
Digite nota 4: > 1.2
Perdió, ¡debe repetir. Definitiva: 1.58
```

Figura 160. Décimo cuarto ejercicio resuelto de estructuras de decisión en PSelnt.

main.py

```
print("Digite nombre del estudiante: ")
nom = input()
print("Digite nota 1: ")
not1 = float(input())
print("Digite nota 2: ")
not2 = float(input())
print("Digite nota 3: ")
not3 = float(input())
print("Digite nota 4: ")
not4 = float(input())
ndef = not1 * 0.1 + not2 * 0.1 + not3 * 0.4 + not4 * 0.4
if (ndef >= 3 and ndef <= 5):
  print("Ganó, ¡felicitaciones!. Definitiva: ", ndef)
else:
  if (ndef >= 2.5):
    print("Perdió, ¡puede habilitar!. Definitiva: ", ndef)
  else:
    if (ndef >= 0.0):
      print("Perdió, ¡debe repetir. Definitiva: ", ndef)
```

```
Digite nombre del estudiante: Miguel
Digite nota 1: 5.0
Digite nota 2: 3.5
Digite nota 3: 4.0
Digite nota 4: 5.0
Ganó, ¡felicitaciones!. Definitiva:  4.45
```

```
Digite nombre del estudiante: Luisa
Digite nota 1: 1.5
Digite nota 2: 2.3
Digite nota 3: 1.8
Digite nota 4: 1.2
Perdió, ¡debe repetir. Definitiva:  1.58
```

Figura 161. Décimo cuarto ejercicio resuelto de estructuras de decisión en Python.

3.3.15 Décimo quinto ejercicio

Leer los tres lados de un triángulo (A, B y C). Imprimir el tipo de triángulo teniendo en cuenta que es un triángulo equilátero (solo si sus tres lados son iguales), es isósceles (si dos de sus lados son iguales) y es un triángulo escaleno (si todos los lados son diferentes). Además, validar que sus lados permitan formar un triángulo: la condición es que cada lado tiene que ser menor que la suma de los otros dos.

En la figura 162 se muestra el pseudocódigo en PSeInt y en la figura 163 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese el lado 1 del triángulo: "
    Leer Lad1
    Imprimir "Ingrese el lado 2 del triángulo: "
    Leer Lad2
    Imprimir "Ingrese el lado 3 del triángulo: "
    Leer Lad3
    Si (Lad1<Lad2+Lad3 Y Lad2<Lad1+Lad3 Y Lad3<Lad1+Lad2) Entonces
        Si (Lad1 = Lad2 Y Lad2 = Lad3) Entonces
            Imprimir "Los lados forman un triángulo equilátero."
        Sino
            Si (Lad1 = Lad2 O Lad2 = Lad3 O Lad1 = Lad3) Entonces
                Imprimir "Los lados forman un triángulo isósceles."
            Sino
                Imprimir "Los lados forman un triángulo escaleno."
            FinSi
        FinSi
    Sino
        Imprimir "Los lados no forman un triángulo."
    FinSi
FinAlgoritmo
```

Ejecución 3a. Condición Verdadera

```
Ingrese el lado 1 del triángulo:
> 3.5
Ingrese el lado 2 del triángulo:
> 1.5
Ingrese el lado 3 del triángulo:
> 3.5
Los lados forman un triángulo isósceles.
```

Niguna Condición Verdadera

```
Ingrese el lado 1 del triángulo:
> 3.5
Ingrese el lado 2 del triángulo:
> 1.5
Ingrese el lado 3 del triángulo:
> 8.5
Los lados no forman un triángulo.
```

Figura 162. Décimo quinto ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese el lado 1 del triángulo: ")
lad1 = float(input())
print("Ingrese el lado 2 del triángulo: ")
lad2 = float(input())
print("Ingrese el lado 3 del triángulo: ")
lad3 = float(input())
if (lad1<lad2+lad3 and lad2<lad1+lad3 and lad3<lad1+lad2):
  if (lad1==lad2 and lad2==lad3):
    print("Los lados forman un triángulo equilátero.")
  else:
    if (lad1==lad2 or lad2==lad3 or lad1==lad3):
      print("Los lados forman un triángulo isósceles.")
    else:
      print("Los lados forman un triángulo escaleno.")
else:
  print("Los lados no forman un triángulo.")
```

```
Ingrese el lado 1 del triángulo:
3.5
Ingrese el lado 2 del triángulo:
1.5
Ingrese el lado 3 del triángulo:
3.5
Los lados forman un triángulo isósceles.
```

```
Ingrese el lado 1 del triángulo:
3.5
Ingrese el lado 2 del triángulo:
1.5
Ingrese el lado 3 del triángulo:
8.5
Los lados no forman un triángulo.
```

Figura 163. Décimo quinto ejercicio resuelto de estructuras de decisión en Python.

3.3.16 Décimo sexto ejercicio

Desarrollar un algoritmo que permita simular un juego de cuatro preguntas de *Falso* y *Verdadero* que se le hacen a un participante. Cada pregunta acertada vale 5 puntos y cada vez que contesta correctamente, se hace la siguiente pregunta; si se equivoca en alguna de las respuestas el juego termina.

Leer estas cuatro preguntas (una a una, no leerlas todas al empezar) e imprimir si gana o pierde; así mismo su puntaje obtenido. Las cuatro preguntas con sus respectivas respuestas usadas en este ejemplo son:

- ¿Cristóbal Colón descubrió a América? Respuesta: (*Verdadera*).
- ¿La moneda de Colombia es el dólar? Respuesta: (*Falsa*).
- ¿Roger Federer es jugador de fútbol americano? Respuesta: (*Falsa*).

- ¿Leonardo Da Vinci fue inventor, pintor y arquitecto? Respuesta: (*Verdadera*).

En la figura 164 se muestra el pseudocódigo en PSeInt y en la figura 165 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "1-¿Cristóbal Colón descubrió a América?: "
    Leer Rta1
    Si (Rta1 = "Verdadera") Entonces
        Imprimir "2-¿La moneda de Colombia es el dólar?: "
        Leer Rta2
        Si (Rta2 = "Falsa") Entonces
            Imprimir "3-¿Roger Federer es jugador de fútbol americano?: "
            Leer Rta3
            Si (Rta3 = "Falsa") Entonces
                Imprimir "4-¿Leonardo Da Vinci fue inventor y pintor?: "
                Leer Rta4
                Si (Rta4 = "Verdadera") Entonces
                    Imprimir "Felicitaciones, ganó. Total puntos: 20."
                Sino
                    Imprimir "Ha perdido en la 4ª pregunta. Total puntos: 15.
                FinSi
            Sino
                Imprimir "Ha perdido en la 3ª pregunta. Total puntos: 10."
            FinSi
        Sino
            Imprimir "Ha perdido en la 2ª pregunta. Total puntos: 5."
        FinSi
    Sino
        Imprimir "Ha perdido en la 1ª pregunta. Total puntos: 0."
    FinSi
FinAlgoritmo
```

Ejecución todas las Condiciones Verdaderas

```
1-¿Cristóbal Colón descubrió a América?:
> Verdadera
2-¿La moneda de Colombia es el dólar?:
> Falsa
3-¿Roger Federer es jugador de fútbol americano?:
> Falsa
4-¿Leonardo Da Vinci fue inventor y pintor?:
> Verdadera
Felicitaciones, ganó. Total puntos: 20.
```

Ejecución 2a. Condición Falsa

```
1-¿Cristóbal Colón descubrió a América?:
> Verdadera
2-¿La moneda de Colombia es el dólar?:
> Verdadera
Ha perdido en la 2ª pregunta. Total puntos: 5.
```

Figura 164. Décimo sexto ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("1-¿Cristóbal Colón descubrió a América?: ")
rta1 = input()
if (rta1=="Verdadera"):
  print("2-¿La moneda de Colombia es el dólar?: ")
  rta2 = input()
  if (rta2=="Falsa"):
    print("3-¿Roger Federer es jugador de fútbol americano?: ")
    rta3 = input()
    if (rta3=="Falsa"):
      print("4-¿Leonardo Da Vinci fue inventor y pintor?: ")
      rta4 = input()
      if (rta4=="Verdadera"):
        print("Felicitaciones, ganó. Total puntos: 20.")
      else:
        print("Ha perdido en la 4ª pregunta. Total puntos: 15.")
    else:
      print("Ha perdido en la 3ª pregunta. Total puntos: 10.")
  else:
    print("Ha perdido en la 2ª pregunta. Total puntos: 5.")
else:
  print("Ha perdido en la 1ª pregunta. Total puntos: 0.")
```

```
1-¿Cristóbal Colón descubrió a América?:
Verdadera
2-¿La moneda de Colombia es el dólar?:
Falsa
3-¿Roger Federer es jugador de fútbol americano?:
Falsa
4-¿Leonardo Da Vinci fue inventor y pintor?:
Verdadera
Felicitaciones, ganó. Total puntos: 20.
```

```
1-¿Cristóbal Colón descubrió a América?:
Verdadera
2-¿La moneda de Colombia es el dólar?:
Verdadera
Ha perdido en la 2ª pregunta. Total puntos: 5.
```

Figura 165. Décimo sexto ejercicio resuelto de estructuras de decisión en Python.

3.3.17 Décimo séptimo ejercicio

El costo de la entrada a un parque de diversión depende de la edad de la persona: si la persona tiene más de 1 año y menos de 4 años entra gratis; si tiene entre 4 y 8 años paga US $ 2, si tiene entre 9 y 16 años paga US $ 5, si tiene entre 17 y 35 años paga US $ 7 y si tiene más de 35 años paga US $ 10.

Hacer un algoritmo que pida la edad de una persona e imprima el valor de entrada en dólares y en pesos colombianos. Tenga en cuenta que se debe leer el TRM actual (tasa representativa del mercado).

En la figura 166 se muestra el pseudocódigo en PSeInt y en la figura 167 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite la edad de la persona: "
    Leer Edad
    Imprimir "Digite el valor de TRM actual: "
    Leer TRM
    Si (Edad > 1 Y Edad < 4) Entonces
        Cto_Ent_Dol = 0
    Sino
        Si (Edad <= 8) Entonces
            Cto_Ent_Dol = 2
        Sino
            Si (Edad <= 16) Entonces
                Cto_Ent_Dol = 5
            Sino
                Si (Edad <= 35) Entonces
                    Cto_Ent_Dol = 7
                Sino
                    Si (Edad > 35) Entonces
                        Cto_Ent_Dol = 10
                    FinSi
                FinSi
            FinSi
        FinSi
    FinSi
    Cto_Ent_Pes = Cto_Ent_Dol * TRM
    Imprimir "Costo de la entrada: US$", Cto_Ent_Dol
    Imprimir "Costo de la entrada: $", Cto_Ent_Pes
FinAlgoritmo
```

Ejecución 2a. Condición Verdadera

```
Digite la edad de la persona:
> 8
Digite el valor de TRM actual:
> 3800
Costo de la entrada: US$2
Costo de la entrada: $7600
```

Ejecución 5a. Condición Verdadera

```
Digite la edad de la persona:
> 45
Digite el valor de TRM actual:
> 3950
Costo de la entrada: US$10
Costo de la entrada: $39500
```

Figura 166. Décimo séptimo ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Digite la edad de la persona: ")
edad = int(input())
print("Digite el valor de TRM actual: ")
trm = float(input())
if (edad > 1 and edad < 4):
  cto_ent_dol = 0
else:
  if (edad <= 8):
    cto_ent_dol = 2
  else:
    if (edad <= 16):
      cto_ent_dol = 5
    else:
      if (edad <= 35):
        cto_ent_dol = 7
      else:
        if (edad > 35):
          cto_ent_dol = 10
cto_ent_pes = cto_ent_dol * trm
print("Costo de la entrada: US$", cto_ent_dol)
print("Costo de la entrada: $", cto_ent_pes)
```

```
Digite la edad de la persona:
8
Digite el valor de TRM actual:
3800
Costo de la entrada: US$ 2
Costo de la entrada: $ 7600.0
```

```
Digite la edad de la persona:
45
Digite el valor de TRM actual:
3950
Costo de la entrada: US$ 10
Costo de la entrada: $ 39500.0
```

Figura 167. Décimo séptimo ejercicio resuelto de estructuras de decisión en Python.

3.3.18 Décimo octavo ejercicio

Desarrolle un algoritmo que lea el salario básico de un empleado, el estado (1: activo, 2: suspendido) y el número de hijos. Calcular el salario neto de un empleado teniendo en cuenta que si está activo y el número de hijos es mayor de 4 le hace un descuento del 10% del salario básico y se le da un auxilio de alimentación de $ 50 000; si está activo y el número de hijos es menor o igual que 4 se le hace un descuento del 15% y un auxilio de $ 25 000; si está suspendido y el número de hijos es mayor de 6 se le

hace un descuento del 5% y se le da un auxilio de $ 40 000; en los demás casos, a cada empleado se le hace un descuento del 3% y se da un auxilio de $ 30 000.

En la figura 168 se muestra el pseudocódigo en PSeInt y en la figura 169 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingrese salario básico: $"
    Leer Sal_Bas
    Imprimir "Ingrese el estado (1: Activo, 2: Suspendido): "
    Leer Estado
    Imprimir "Ingrese el número de hijos: "
    Leer Num_Hijos
    Si (Estado = 1) Entonces
        Si (Num_Hijos > 4) Entonces
            Sal_Neto = Sal_bas * 0.9 + 50000
        Sino
            Sal_Neto = Sal_bas * 0.85 + 25000
        FinSi
    Sino
        Si (Estado = 2) Entonces
            Si (Num_Hijos > 6) Entonces
                Sal_Neto = Sal_bas * 0.95 + 40000
            Sino
                Sal_Neto = Sal_bas * 0.97 + 30000
            FinSi
        FinSi
    FinSi
    Imprimir "Salario Neto: $", Sal_Neto
FinAlgoritmo
```

Ejecución 2a. Condición Falsa

```
Ingrese salario básico: $
> 800000
Ingrese el estado (1: Activo, 2: Suspendido):
> 1
Ingrese el número de hijos:
> 3
Salario Neto: $705000
```

Ejecución todas las Condiciones Falsas

```
Ingrese salario básico: $
> 800000
Ingrese el estado (1: Activo, 2: Suspendido):
> 2
Ingrese el número de hijos:
> 2
Salario Neto: $806000
```

Figura 168. Décimo octavo ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingrese salario básico: $")
sal_bas = float(input())
print("Ingrese el estado (1: Activo, 2: Suspendido): ")
estado = int(input())
print("Ingrese el número de hijos: ")
num_hijos = int(input())
if (estado == 1):
  if (num_hijos > 4):
    sal_neto = sal_bas * 0.9 + 50000
  else:
    sal_neto = sal_bas * 0.85 + 25000
else:
  if (estado == 2):
    if (num_hijos > 6):
      sal_neto = sal_bas * 0.95 + 40000
    else:
      sal_neto = sal_bas * 0.97 + 30000
print("Salario Neto: $",sal_neto)
```

```
Ingrese salario básico: $
800000
Ingrese el estado (1: Activo, 2: Suspendido):
1
Ingrese el número de hijos:
2
Salario Neto: $ 705000.0
```

```
Ingrese salario básico: $
800000
Ingrese el estado (1: Activo, 2: Suspendido):
2
Ingrese el número de hijos:
2
Salario Neto: $ 806000.0
```

Figura 169. Décimo octavo ejercicio resuelto de estructuras de decisión en Python.

3.3.19 Décimo noveno ejercicio

Elabore un algoritmo que lea el nombre, la edad, el sexo (F: Femenino, M: Masculino) y el estado civil (1: Casado, 2: Soltero, 3: Separado, 4: Otro) de una persona que desea participar en las elecciones de este año. Si es menor de edad imprimir un mensaje que diga que no puede votar, de lo contrario imprima un mensaje indicando la mesa en la cual le corresponder votar. Tenga en cuenta las condiciones de la tabla 42.

Tabla 42.
Información votación. Ejercicio 3.3.19.

Edad	Sexo	Estado Civil	Mensaje
Mayor de edad	Femenino	Casada	Vota en la Mesa 1
Mayor de edad	Femenino	Soltera	Vota en la Mesa 2
Mayor de edad	Femenino	Separada	Vota en la Mesa 3
Mayor de edad	Femenino	Otro	Vota en la Mesa 4
Mayor de edad	Masculino	Casado	Vota en la Mesa 5
Mayor de edad	Masculino	Soltero	Vota en la Mesa 6
Mayor de edad	Masculino	Separado	Vota en la Mesa 7
Mayor de edad	Masculino	Otro	Vota en la Mesa 8
Menor de edad	–	–	No puede votar

Cuando un algoritmo es tan extenso y tiene varias condiciones, se recomienda poner números a las preguntas y al cerrar la condición especificar el número de la pregunta a la que corresponde la instrucción de cierre. También, se podría considerar leer la edad y validar si es menor de edad y mostrar el respectivo mensaje; de lo contrario leer la información restante. No tendría sentido solicitarle sexo y estado civil a una persona que no puede votar.

En la figura 170 se muestra el pseudocódigo en PSeInt y en la figura 171 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Ingresar la edad en años: "
    Leer Ed
    Si (Ed < 18) Entonces   ///Pregunta 1
        Imprimir "No puede votar"
    Sino
        Imprimir "¿Nombres completos?: "
        Leer Nom
        Imprimir "¿Sexo [M: Masculino, F: Femenino]?: "
        Leer Sex
        Imprimir "¿Estado civil [1: Casado, 2: Soltero, 3: Separado y 4: Ot
        Leer Ec
        Si (Sex = "F") Entonces   ///Pregunta 2
            Si (Ec = 1) Entonces   ///Pregunta 3
                Imprimir "Usted puede votar en la Mesa 1."
            Sino
                Si (Ec = 2) Entonces   ///Pregunta 4
                    Imprimir "Usted puede votar en la Mesa 2."
                Sino
                    Si (Ec = 3) Entonces    ///Pregunta 5
                        Imprimir "Usted puede votar en la Mesa 3."
                    Sino
                        Si (Ec = 4) Entonces    ///Pregunta 6
                            Imprimir "Usted puede votar en la Mesa 4."
                        FinSi   ///Cierre Pregunta 6
                    FinSi   ///Cierre Pregunta 5
                FinSi   ///Cierre Pregunta 4
            FinSi   ///Cierre Pregunta 3
        Sino
            Si (Sex = "M") Entonces   ///Pregunta 7
                Si (Ec = 1) Entonces   ///Pregunta 8
                    Imprimir "Usted puede votar en la Mesa 5."
                Sino
                    Si (Ec = 2) Entonces   ///Pregunta 9
                        Imprimir "Usted puede votar en la Mesa 6."
                    Sino
                        Si (Ec = 3) Entonces   ///Pregunta 10
                            Imprimir "Usted puede votar en la Mesa 7."
                        Sino
                            Si (Ec = 4) Entonces   ///Pregunta 11
                                Imprimir "Usted puede votar en la Mesa 8."
                            FinSi   ///Cierre Pregunta 11
                        FinSi   ///Cierre Pregunta 10
                    FinSi   ///Cierre Pregunta 9
                FinSi   ///Cierre Pregunta 8
            FinSi   ///Cierre Pregunta 7
        FinSi   ///Cierre Pregunta 2
    FinSi   ///Cierre Pregunta 1
FinAlgoritmo
```

Ejecución 1a. Condición Verdadera

```
Ingresar la edad en años:> 16
No puede votar
```

Ejecución 3a. Condición Verdadera

```
Ingresar la edad en años:> 44
¿Nombres completos?:> Jairo
¿Sexo [M: Masculino, F: Femenino]?:> M
¿Estado civil [1: Casado, 2: Soltero, 3: Separado y 4: Otro]:> 1
Usted puede votar en la Mesa 5.
```

Figura 170. Décimo noveno ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Ingresar la edad en años: ")
ed = int(input())
if (ed<18): #Pregunta 1
  print("No puede votar")
else:
  print("¿Nombres completos?: ")
  nom = input()
  print("¿Sexo [M: Masculino, F: Femenino]?: ")
  sex = input()
  print("¿Estado civil [1: Casado, 2: Soltero, 3: Separado y 4: Otro]: ")
  ec = input()
  if (sex=="F"):  #Pregunta 2
    if (ec==1): #Pregunta 3
      print("Usted puede votar en la Mesa 1.")
    else:
      if (ec==2): #Pregunta 4
        print("Usted puede votar en la Mesa 2.")
      else:
        if (ec==3): #Pregunta 5
          print("Usted puede votar en la Mesa 3.")
        else:
          if (ec==4): #Pregunta 6
            print("Usted puede votar en la Mesa 4.")
  else:
    if (sex=="M"):  #Pregunta 7
      if (ec==1): #Pregunta 8
        print("Usted puede votar en la Mesa 5.")
      else:
        if (ec==2): #Pregunta 9
          print("Usted puede votar en la Mesa 6.")
        else:
          if (ec==3): #Pregunta 10
            print("Usted puede votar en la Mesa 7.")
          else:
            if (ec==4): #Pregunta 11
              print("Usted puede votar en la Mesa 8.")
```

```
Ingresar la edad en años: 16
No puede votar
```

```
Ingresar la edad en años: 44
¿Nombres completos?: Jairo Ramírez
¿Sexo [M: Masculino, F: Femenino]?: M
¿Estado civil [1: Casado, 2: Soltero, 3: Separado y 4: Otro]: 1
Usted puede votar en la Mesa 5.
```

Figura 171. Décimo noveno ejercicio resuelto de estructuras de decisión en Python.

3.3.20 Vigésimo ejercicio

Se requiere de un algoritmo que permita leer tres números y muestre el valor mayor, menor y medio. Además, que imprima mensajes en caso de que los números sean iguales.

En la figura 172 se muestra el pseudocódigo en PSeInt y en la figura 173 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite número 1: "
    Leer N1
    Imprimir "Digite número 2: "
    Leer N2
    Imprimir "Digite número 3: "
    Leer N3
    Si (N1 = N2 Y N2 = N3) Entonces ///Pregunta 1
        Imprimir "Los tres números son iguales."
    Sino
        Si (N1 = N2) Entonces ///Pregunta 2
            Imprimir "El primer número y el segundo son iguales."
        Sino
            Si (N1 = N3) Entonces ///Pregunta 3
                Imprimir "El primer número y el tercero son iguales."
            Sino
                Si (N2 = N3) Entonces ///Pregunta 4
                    Imprimir "El segundo y el tercero son iguales."
                Sino
                    Si (N1 < N2) Entonces ///Pregunta 5
                        Si (N1 < N3) Entonces ///Pregunta 6
                            Si (N2 < N3) Entonces ///Pregunta 7
                                Imprimir "Mayor: ", N3, ". Medio: ", N2, ". Menor: ", N1
                            Sino
                                Imprimir "Mayor: ", N2, ". Medio: ", N3, ". Menor: ", N1
                            FinSi ///Cierra la pregunta 7
                        Sino
                            Imprimir "Mayor: ", N2, ". Medio: ", N1, ". Menor: ", N3
                        FinSi ///Cierra la pregunta 6
                    Sino
                        Si (N1 > N3) Entonces   ///Pregunta 8
                            Si (N2 < N3) Entonces ///Pregunta 9
                                Imprimir "Mayor: ", N1, ". Medio: ", N3, ". Menor: ", N2
                            SiNo
                                Imprimir "Mayor: ", N1, ". Medio: ", N2, ". Menor: ", N3
                            FinSi///Cierra la pregunta 9
                        SiNo
                            Imprimir "Mayor: ", N3, ". Medio: ", N1, ". Menor: ", N2
                        FinSi ///Cierra la pregunta 8
                    FinSi ///Cierra la pregunta 5
                FinSi ///Cierra la pregunta 4
            FinSi ///Cierra la pregunta 3
        FinSi ///Cierra la pregunta 2
    FinSi ///Cierra la pregunta 1
FinAlgoritmo
```

Ejecución 9a. Condición Verdadera

```
Digite número 1: > 8
Digite número 2: > 2
Digite número 3: > 5
Mayor: 8. Medio: 5. Menor: 2
```

Ejecución 1a. Condición Verdadera

```
Digite número 1: > 5
Digite número 2: > 5
Digite número 3: > 5
Los tres números son iguales.
```

Figura 172. Vigésimo ejercicio resuelto de estructuras de decisión en PSeInt.

main.py

```
print("Digite número 1: ", end="")
n1 = int(input())
print("Digite número 2: ", end="")
n2 = int(input())
print("Digite número 3: ", end="")
n3 = int(input())
if (n1==n2 and n2==n3): #Pregunta 1
  print("Los tres números son iguales.")
else:
  if (n1==n2):  #Pregunta 2
    print("El primer número y el segundo son iguales.")
  else:
    if (n1==n3):  #Pregunta 3
      print("El primer número y el tercero son iguales.")
    else:
      if (n2==n3):  #Pregunta 4
        print("El segundo y el tercero son iguales.")
      else:
        if (n1<n2): #Pregunta 5
          if (n1<n3): #Pregunta 6
            if (n2<n3): #Pregunta 7
              print("Mayor: ",n3,". Medio: ",n2,". Menor: ",n1)
            else:
              print("Mayor: ",n2,". Medio: ",n3,". Menor: ",n1)
          else:
            print("Mayor: ",n2,". Medio: ",n1,". Menor: ",n3)
        else:
          if (n1>n3): #Pregunta 8
            if (n2<n3): #Pregunta 9
              print("Mayor: ",n1,". Medio: ",n3,". Menor: ",n2)
            else:
              print("Mayor: ",n1,". Medio: ",n2,". Menor: ",n3)
          else:
            print("Mayor: ",n3,". Medio: ",n1,". Menor: ",n2)
```

```
Digite número 1: 5
Digite número 2: 5
Digite número 3: 5
Los tres números son iguales.
```

```
Digite número 1: 8
Digite número 2: 2
Digite número 3: 5
Mayor:  8 . Medio:  5 . Menor:  2
```

Figura 173. Vigésimo ejercicio resuelto de estructuras de decisión en Python.

3.4 Ejercicios propuestos

Para poner en práctica lo aprendido, se recomienda realizar los siguientes ejercicios propuestos de estructuras de decisión y selección.

3.4.1 Primer ejercicio

Hacer un algoritmo que permita leer la edad de una persona y valide si puede votar o no en las elecciones para elegir al presidente de la acción comunal del barrio en el que vive. Tenga en cuenta que solo lo pueden hacer las personas tengan 18 años o más.

3.4.2 Segundo ejercicio

Diseñar un algoritmo que lea un número e imprima si es positivo, negativo o neutro. Para esto se usan condiciones de mayor, menor o igual que cero.

3.4.3 Tercer ejercicio

Se requiere de un algoritmo que lea un número, lo imprima acompañado de un mensaje indicando si es par o impar. Para hacer la validación tenga en cuenta la operación módulo.

3.4.4 Cuarto ejercicio

En una cita de control de peso saludable se solicita a un paciente su estatura (en metros) y su peso (en kilogramos). Se debe implementar un algoritmo que permita mostrar si tiene el peso ideal, si tiene sobrepeso o tiene obesidad dependiendo de su índice de masa corporal (IMC), el cual es un indicador que sirve para dar respuesta a este requerimiento. Validar que el peso y la estatura sean valores positivos.

El IMC se calcula con la siguiente fórmula: $IMC = \frac{Peso}{Estatura^2}$

Por ejemplo, si el peso es igual 55 kilogramos y la estatura es igual 1.65 metros. IMC = 55 / 1.65 ^ 2 = 55 / 2.75 = 20.20, esta persona tendría el peso ideal. Para llegar a esta conclusión se debe conocer los siguientes rangos permitidos para el IMC:

- Entre 19 y 24: ¡Excelente! Tienes el peso ideal.
- Entre 25 y 39: ¡Cuidado! Tienes sobrepeso.
- Más de 30: ¡Urgente! Tienes obesidad.

3.4.5 Quinto ejercicio

Desarrollar un algoritmo que permita leer el nombre y las iniciales del estado civil (S: Soltero, C: Casado, V: Viudo, U: Unión Libre y O: Otro) de una persona. Imprimir el nombre y el estado civil completo.

3.4.6 Sexto ejercicio

Desarrollar un algoritmo que calcule el salario neto de un empleado sabiendo que se le hace una retención del 25% de su salario básico por concepto de un préstamo. Imprima un mensaje que diga si gana más del mínimo o no. El valor del salario básico y el salario mínimo deben ser leídos.

3.4.7 Séptimo ejercicio

Imprimir el costo total de una factura la cual tiene un costo original que sufre un aumento cuando el material solicitado requiere transporte. Hacer un algoritmo que facilite realizar este proceso ingresando el costo de la factura original y si se requiere transporte o no. Tenga en cuenta que si requiere transporte tendrá un incremento sobre el valor de la factura. Este valor deberá ser leído.

3.4.8 Octavo ejercicio

Encuentre la forma de resolver la siguiente ecuación a través de un algoritmo y validando que la operación que va en el divisor sea diferente de cero.

$$\text{Ecuación} = \frac{L\,(n-3)}{n^2 + x}$$

3.4.9 Noveno ejercicio

Una persona desea realizar un viaje a alguna de las siguientes ciudades colombianas (tabla 43). Desarrolle un algoritmo que permita ingresar el

nombre de la ciudad y visualice el estado del clima, el valor del pasaje, el valor del descuento y el valor después de aplicado el descuento. Tenga en cuenta la tabla 43.

Tabla 43.
Ciudades, climas y pasajes. Ejercicio 3.4.9.

Ciudad	Clima	Valor pasaje ($)	Descuento (%)
Barranquilla	Caliente	100 000	0
Bogotá	Frío	75 000	10
Cali	Templado	80 000	15
Cartagena	Caliente	90 000	5
Pasto	Frío	50 000	40

3.4.10 Décimo ejercicio

Hacer un algoritmo que lea el año y el mes de nacimiento de una persona e imprima su edad y la cantidad de meses aproximados que ha vivido. En caso de haber nacido en el mes de febrero en un año bisiesto mostrar un mensaje: "Usted es un caso especial". Un año es bisiesto si divisible entre 4 excepto los que sean divisibles entre 100 y omitiendo los que sean divisibles entre 400.

3.4.11 Décimo primer ejercicio

Elabore un algoritmo que calcule el valor pagado al Instituto de Seguros Sociales (ISS), el valor de retención en la fuente, el salario básico y el salario neto mensual de un empleado. Se debe leer cédula, nombre, salario básico hora y las horas trabajadas en el mes. Para calcular el valor pagado al ISS y la retención en la fuente tenga en cuenta la tabla 44.

Tabla 44.
Salarios y porcentajes. Ejercicio 3.4.11.

Rango salarial ($)	Retención (%)	ISS (%)
Menor a 480 000	4	6
Entre 480 000 y 560 000	7	8
Entre 560 001 y 750 000	10	11
Entre 750 001 y 900 000	20	30
Mayor a 900 000	25	35

3.4.12 Décimo segundo ejercicio

En una cooperativa de ahorro se pone una cantidad de dinero a una tasa de interés mensual cambiante en un periodo determinado. A través de un algoritmo, se permite leer el plazo y la cantidad de dinero que se desea ahorrar para poder calcular el interés obtenido y el valor total al cabo de 6, 12, 24 y 36 meses. La tasa de interés para cada uno de los plazos se resume en la tabla 45.

Tabla 45.
Tasas de interés. Ejercicio 3.4.12.

Plazo (meses)	Tasa de interés (%)
6	1.2
12	1.5
24	1.7
36	1.8

3.4.13 Décimo tercer ejercicio

Hacer un algoritmo que lea el precio de fábrica de un artículo y el país de procedencia. E Imprima el valor neto del artículo y el valor del recargo.

El precio de un artículo tiene un recargo dependiendo de su país de origen. Si el producto es de Colombia tiene un recargo del 1%; si es de Venezuela, Ecuador, Bolivia o Perú tiene el 8%; si es de Chile, Paraguay, Uruguay o Argentina tienen el 13%; y si es de otro país tiene el 18%.

3.4.14 Décimo primer ejercicio

La categoría de un neumático se determina por su desgaste y su tracción. Se requiere de un algoritmo que permita leer de un neumático estos dos datos y muestre su categoría (número entre 1 y 6). Tener en cuenta la tabla 46.

Tabla 46.
Categorías, desgaste y tracción de neumáticos. Ejercicio 3.4.14.

Desgaste	Tracción	Categoría
Inferior a 100	Buena	1
Inferior a 100	Regular	2
Inferior a 100	Mala	3
Superior o igual a 100	Buena	4
Superior o igual a 100	Regular	5
Superior o igual a 100	Mala	6

3.4.15 Décimo quinto ejercicio

El gerente de una empresa requiere de un estudiante de programación que le ayude a liquidar la nómina de uno de sus empleados en la última quincena. Al empleado se le solicita la cantidad de horas, el valor por hora y el horario en el que laboró en la quincena (1: Diurno y 2: Nocturno).

Se pide implementar un algoritmo que imprima el salario básico y el recargo, teniendo en cuenta que el valor de cada hora en horario nocturno es el 1.75 con respecto al valor de la hora leído inicialmente.

3.4.16 Décimo sexto ejercicio

Un cliente de un concesionario necesita saber si las nuevas llantas de su vehículo cumplen con el diámetro externo requerido. Para lo anterior debe comparar el diámetro de las dos llantas, y la diferencia entre ellas no debe ser superior a 6 milímetros.

Hacer un algoritmo que lea el diámetro de ambas llantas en milímetros e imprima si cumple o no con el requisito; en caso de no cumplir con lo esperado, imprimir la cantidad de milímetros que está por encima o por debajo la nueva llanta con respecto a la anterior.

3.4.17 Décimo séptimo ejercicio

Una persona desea comprar una de tres máquinas. Si opta por una máquina plana tendrá un descuento del 11%, si opta por una recubridora el 17% y si opta por una fileteadora el 23%.

Hacer un algoritmo que permita ingresar la máquina que desea comprar esta persona y dependiendo su elección pida el precio. Calcular e imprimir el valor del descuento y el precio de venta de la máquina elegida.

3.4.18 Décimo octavo ejercicio

Hacer un algoritmo que lea: la marca del vehículo y el precio de venta del vehículo e imprima: *módico* si el precio de venta está dentro de los rangos establecidos por marca; *alto* si el precio de venta supera el precio máximo y *bajo* si el precio de venta es inferior al precio mínimo. Imprimir el mensaje dependiendo la tabla 47.

Tabla 47.
Precios y marcas de vehículos. Ejercicio 3.4.18.

Marca	Precio mínimo ($)	Precio máximo ($)
Chevrolet	20 000 000	30 000 000
Mazda	45 000 000	60 000 000
Toyota	40 000 000	95 000 000
Renault	50 000 000	70 000 000
Audi	60 000 000	90 000 000
BMW	95 000 000	200 000 000

3.4.19 Décimo noveno ejercicio

Para participar en una competencia se tienen que cumplir los siguientes requisitos: ser mayor de edad, pesar más de 50 kilogramos y medir más de 1.5 metros. Se pide leer la edad, el peso y la estatura de una persona e imprimir si puede o no participar en una competencia. Para cada aspecto que no cumpla imprimirle lo que le hace falta para llegar al límite exigido.

3.4.20 Vigésimo ejercicio

A una estación de gasolina se acerca un cliente para tanquear o lavar sus vehículos. Al momento de llegar se llena un registro con la siguiente información: nombre del cliente, cédula del cliente, placa del vehículo, tipo de vehículo (1-Particular, 2-Servicio público, 3-Remolques y 4-Camiones) y tipo de servicio (L-Lavada, T-Tanqueo y A-Ambos servicios).

Hacer un algoritmo que muestre el costo de servicio teniendo en cuenta que si el vehículo es particular o de servicio público la lavada tiene un costo de $ 30 000; mientras que en los otros casos tiene un costo de $ 50 000. Si el cliente desea tanquear, se debe solicitar el número de galones que desea y el tipo de combustible (1: Corriente y 2: ACPM). El valor de un galón de gasolina corriente es de $ 8000 y el valor de uno de ACPM es de $ 7500. Si el cliente realiza ambos servicios se le hace un descuento del 5% sobre el valor del servicio que incluye el valor de la lavada y el valor del tanqueo.

3.4.21 Vigésimo primer ejercicio

Desarrollar un pseudocódigo que lea las iniciales de la clasificación de una película en Estados Unidos, e imprima el significado de las iniciales relacionadas y su edad en la clasificación. Tenga en cuenta la tabla 48.

Tabla 48.
Clasificación de películas. Ejercicio 3.4.21.

Iniciales	Significado	Edades
G	Todos los públicos.	Todas las edades
PG	Guía paternal sugerida.	+10 años
PG–13	Guía paternal estricta.	+13 años
R	Restringido.	+17 años
NC–17	Prohibido para audiencia <= 17 años	+18 años
NR	Sin clasificar.	

3.4.22 Vigésimo segundo ejercicio

Una persona requiere de un *software* al cual se le pueda ingresar la fecha de nacimiento de una persona y validar si la fecha es correcta. Hacer el respectivo algoritmo para este *software*. Teniendo en cuenta que para que esto se cumpla, el año tiene que ser mayor que 1990 y menor que el año actual (el cual debe ser leído); el mes tiene que ser un número entre 1 y 12; y los días tienen que estar entre 1 y 31, los cuales se determinan con las estas consideraciones:

- Los meses de enero, marzo, mayo, julio, agosto, octubre y diciembre tienen 31 días.
- Los meses abril, junio, septiembre y noviembre tienen 30 días.
- El mes de febrero tiene 28 o 29 días dependiendo si el año es bisiesto o no.

3.4.23 Vigésimo tercer ejercicio

La contadora de una corporación es llamada para hacer la liquidación de la nómina del último mes. Para determinar el salario neto se debe leer el estado del empleado (1: Laborando, 2: Incapacitado y 3: Jubilado). Si está laborando pedirle el valor de una hora normal, el número de horas normales, número de horas festivas, número de horas extras diurnas, número de horas extras nocturnas, número de horas extras diurnas festivas y número de horas extras nocturnas festivas. Si está incapacitado o jubilado, pedirle su salario mensual.

Se requiere hacer un algoritmo que sistematice la situación anterior e imprima el salario neto de ese empleado y todos los cálculos realizados. Tenga en cuenta que las horas festivas se pagan al 1.25 de una hora normal, las horas extras diurnas al 1.5; las horas extras nocturnas al 1.75; las extras diurnas festivas al doble y las extras nocturnas festivas al 2.25. Finalmente, para la persona incapacitada, su pago será el 66.66% de su salario mensual y para la persona jubilada su pago será el 92% del salario mensual.

3.4.24 Vigésimo cuarto ejercicio

Cuatro personas a quienes les gustan las apuestas participan en un juego en el cual cada uno hace un aporte de dinero determinado para ver quien adivina un número secreto que aparece al sacar una balota. La casa de apuesta siempre duplica el valor jugado por los participantes. El ganador se lleva el 70% de todo lo apostado (gana quien adivine el número secreto) y el 30% queda para la casa de apuestas. En caso de no presentarse ningún ganador la casa se lleva lo apostado por los cuatro jugadores.

Hacer un algoritmo que permita leer lo apostado por cada jugador, el número secreto y el número de balota de cada uno de los cuatro jugadores (no se repiten número de balotas). Imprimir el valor apostado (el doble de las apuestas de los 4 jugadores), si hay ganador o no. En caso de presentarse un ganador imprimir el valor apostado, el valor recibido por el ganador (después de realizarle el cálculo a la casa de apuestas) y el dinero recibido por la casa. Si ninguno gana mostrar el valor que recibirá la casa por estas apuestas.

3.4.25 Vigésimo quinto ejercicio

Un empresario necesita saber si sigue pagando o no horas extras a trabajadores por la producción que realiza el domingo. Todo depende de la siguiente condición: si las ganancias obtenidas por una producción son como mínimo el doble de lo pagado en horas extras entonces continuará trabajando; en caso contrario suspenderá labores el domingo y no continuará pagando horas extras. Hacer un algoritmo que lea el valor de la hora normal, el número de horas extras e imprima el mensaje correspondiente teniendo en cuenta lo siguiente:

- El valor de una hora extra es equivalente al 1.75 del valor de una hora normal.
- La ganancia de la producción se halla restando el costo de venta y el costo de fabricación.

3.5 Práctica final estructuras de decisión

Con este tipo de ejercicio, se pretende consolidar todos los conceptos aprendidos en esta unidad. A pesar de lo extenso, recuerde que los algoritmos y la programación siempre buscarán dividir un problema en fracciones más pequeñas que faciliten llegar a una solución clara, coherente y eficiente.

Una librería vende libros de acuerdo con especificaciones resumidas en las tablas 39 y 50.

Tabla 49.
Información libros. Práctica final.

Nacionalidad	Número de páginas	Valor hoja	IVA (%)
Argentina	Menor que 100	$ 0.36 (Peso argentino)	21
Argentina	Entre 100 y 300	$ 0.54 (Peso argentino)	21
Argentina	Mayor a 300	$ 0.98 (Peso argentino)	21
Americana	Menor que 100	US $ 2 (Dólar)	9
Americana	Entre 100 y 300	US $ 4 (Dólar)	9
Americana	Mayor a 300	US $ 6 (Dólar)	9
Europea	Menor que 100	€ 0.4 (Euro)	25
Europea	Entre 100 y 300	€ 0.8 (Euro)	25
Europea	Mayor a 300	€ 0.10 (Euro)	25

Tabla 50.
Porcentajes de rebajas. Práctica final.

# Mes	Rebaja (%)	Estación del año
12-1-2	17.85	Invierno
3-4-5	12.35	Primavera
6-7-8	10.05	Verano
9-10-11	8.75	Otoño

Realice un algoritmo que permita leer el número de páginas de un libro que se desea comprar, la nacionalidad del autor del libro (dependiendo la nacionalidad pida la equivalencia de esa moneda en pesos colombianos. Tenga en cuenta que un peso argentino equivale a $ 144.3238) y el número del mes del año en el cual se hace la compra.

Mostrar el siguiente reporte:

- Valor del libro en la moneda que corresponde, dependiendo del número y el valor página (tener en cuenta que una hoja tiene dos páginas).
- Valor del IVA en la moneda que corresponde.
- Valor de rebaja en la moneda que corresponde (basado en la tabla 50).
- El precio final del libro en la moneda que corresponde.

- Valor devuelto por la cajera entregado al momento de pagar (en la moneda que corresponde). Tenga en cuenta que sí alcance el dinero; en caso de no ser suficiente imprimir un mensaje.
- Valor del libro en pesos colombianos.
- Valor del IVA en pesos colombianos (19%).
- Valor de rebaja en pesos colombianos.
- Valor final del libro en pesos colombianos.
- La estación del año en el cual se compró el libro (basado en la tabla 50).

3.6 Estructuras de selección múltiple

Son estructuras conocidas con el nombre de estructuras caso o estructuras Segun en PSeInt, son similares a las estructuras de decisión múltiple, pero más compactas, ahorran espacio y son mucho más simples de utilizar. Pueden reemplazar estructuras de decisión múltiple que tienen preguntas usando el signo igual (=) con sola una variable que toma valores conocidos y finitos. Esta estructura, normalmente, tiene en cuenta el error del usuario en caso de digitarse un valor erróneo.

La estructura *Segun* usada en PSeInt no existe en Python, sino que fue reemplazada por la sentencia *elif*, mientras que en *C++* o *Java* se conoce como *switch* y la opción *De otro modo* se escribe *default* en estos lenguajes de programación.

La sintaxis es la siguiente:

Segun (Variable)

Caso Valor_1: Instrucciones que ejecuta si la variable es igual al Valor_1

Caso Valor_2: Instrucciones que ejecuta si la variable es igual al Valor_2

Caso Valor_3: Instrucciones que ejecuta si la variable es igual al Valor_3

Caso Valor_N: Instrucciones que ejecuta si la variable es igual al Valor_N

De otro modo: Ejecuta esto si la variable no es igual a ninguna de las anteriores

FinSegun

En lugar de *Valor_1*, *Valor_2*, hasta el V*alor_N*, se pone el primer valor que toma la variable, luego el segundo y así sucesivamente hasta el último valor.

La sintaxis de la estructura de selección en diagramación libre se puede ver en la figura 174.

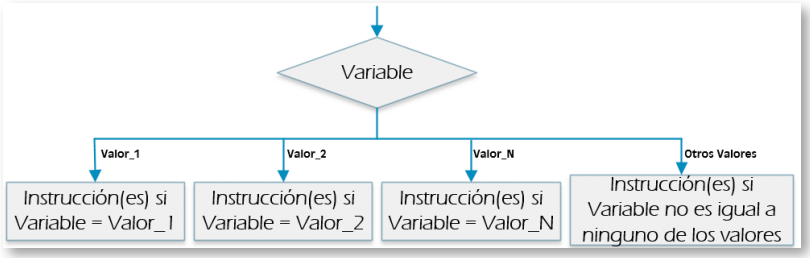


Figura 174. Sintaxis de la estructura de selección múltiple en diagramación libre.

Las estructuras de selección múltiple tienen algunas limitaciones:

- En la estructura solo puede evaluarse condiciones que usen una sola variable con el operador igual (=).
- En la estructura solo puede evaluarse condiciones que usen una sola variable separadas por el operador de la disyunción (*O*). Por ejemplo, la condición: Si (Dia=”viernes” O Dia = “sábado”); pero no se puede usar con varias condiciones separadas con el operador de la conjunción (*Y*). Por ejemplo, la condición: Si (Dia >= 1 Dia <=7), no se podría implementar en una estructura *Segun*.
- En lenguajes de programación como C++ o Java, solo son permitidos los tipos de variable entero (*int*) y carácter (*char*).
- En el lenguaje de programación Python, esta estructura no existe; ya que fue reemplazada por la estructura de decisión *if*/*elif*.

Ejemplo 1. Hacer un algoritmo que permita ingresar el número de un día de la semana (número entre 1 y 7) e imprimir el nombre del día al que corresponde. Tenga en cuenta que el número 1 corresponde al lunes, el 2 al martes y, así sucesivamente, hasta el domingo. En caso de ingresar un número diferente mostrar un mensaje de error.

En la figura 175 se muestra el pseudocódigo en PSeInt y en la figura 176 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite el número del día: "
    Leer Numdia
    Segun (Numdia)
        Caso 1: Imprimir "Número corresponde al: lunes."
        Caso 2: Imprimir "Número corresponde al: martes."
        Caso 3: Imprimir "Número corresponde al: miércoles."
        Caso 4: Imprimir "Número corresponde al: jueves."
        Caso 5: Imprimir "Número corresponde al: viernes."
        Caso 6: Imprimir "Número corresponde al: sábado."
        Caso 7: Imprimir "Número corresponde al: domingo."
        De Otro Modo: Imprimir "Error. Número no existe."
    FinSegun
FinAlgoritmo
```

```
Digite el número del día:
> 2
Número corresponde al: martes.
```

```
Digite el número del día:
> 9
Error. Número no existe.
```

Figura 175. Primer ejemplo de estructura de selección múltiple en PSeInt.

main.py

```
print("Digite el número del día: ")
numdia = int(input())
if numdia == 1:
  print("Número corresponde al: lunes.")
elif numdia == 2:
  print("Número corresponde al: martes.")
elif numdia == 3:
  print("Número corresponde al: miércoles.")
elif numdia == 4:
  print("Número corresponde al: jueves.")
elif numdia == 5:
  print("Número corresponde al: viernes.")
elif numdia == 6:
  print("Número corresponde al: sábado.")
elif numdia == 7:
  print("Número corresponde al: domingo.")
else:
  print("Error. Número  no existe.")
```

```
Digite el número del día:
2
Número corresponde al: martes.
```

```
Digite el número del día:
9
Error. Número no existe
```

Figura 176. Primer ejemplo de estructura de selección múltiple en Python.

En Python no existe esta estructura de selección múltiple, la cual es reemplazada por la estructura de decisión múltiple *if/elif*. Mirar el ejemplo anterior en ese lenguaje de programación. En la figura 177 se muestra el ejercicio anterior en diagramación libre.

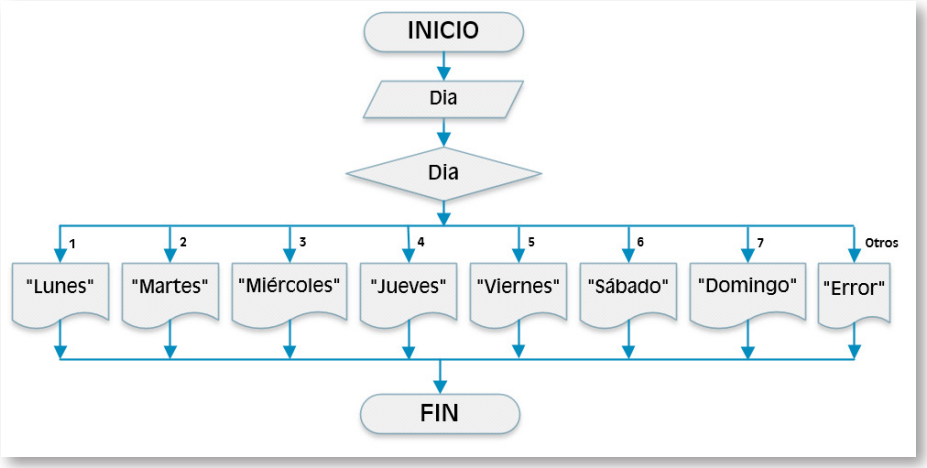


Figura 177. Estructura de selección múltiple en diagramación libre.

Ejemplo 2. Una nutricionista necesita la implementación de un *software* que le permita ingresar un determinado alimento para visualizar el índice glicémico (IG) correspondiente. Para este proceso, tenga en cuenta los valores de la tabla 51.

Tabla 51.
Índices glicémicos. Segundo ejemplo estructura de selección.

Alimento	**IG**	**Alimento**	**IG**
Cacahuete	15	Arroz	44
Cereza	22	Uva	46
Lenteja	22	Queso	46
Pera	38	Helado	60
Manzana	38	Pasa	60
Mango	38	Sandía	70
Banana	44	Galleta	70
Naranja	44	Cerveza	110

En la figura 178 se muestra el pseudocódigo en PSeInt y en la figura 179 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite el alimento: "
    Leer Alimento
    Segun (Alimento)
        "Cacahuete": Indice = 15
        "Cereza", "Lenteja": Indice = 22
        "Pera", "Manzana", "Mango": Indice = 38
        "Banana", "Naranja", "Arroz": Indice = 44
        "Uva", "Queso": Indice = 46
        "Helado", "Pasa": Indice = 60
        "Sandía", "Galleta": Indice = 70
        "Cerveza": Indice = 110
        De otro modo: Indice = 0
    FinSegun
    Si (Indice = 0) Entonces
        Imprimir "Error. Alimento no existe en esta tabla."
    SiNo
        Imprimir "Índice Glicémico: ", Indice
    FinSi
FinAlgoritmo
```

Ejecución Caso 8

```
Digite el alimento:
> Cerveza
Índice Glicémico: 110
```

Ejecución De otro modo

```
Digite el alimento:
> Banano
Error. Alimento no existe en esta tabla.
```

Figura 178. Segundo ejemplo de estructura de selección múltiple en PSeInt.

main.py

```
print("Digite el alimento: ")
alimento = input()
if (alimento)=="Cacahuete":
    indice = 15
elif (alimento)=="Cereza" or (alimento)=="Lenteja":
    indice = 22
elif (alimento)=="Pera" or (alimento)=="Manzana" or (alimento)=="Mango":
    indice = 38
elif (alimento)=="Banana" or (alimento)=="Naranja" or (alimento)=="Arroz":
    indice = 44
elif (alimento)=="Uva" or (alimento)=="Queso":
    indice = 46
elif (alimento)=="Helado" or (alimento)=="Pasa":
    indice = 60
elif (alimento)=="Sandía" or (alimento)=="Galleta":
    indice = 70
elif (alimento)=="Cerveza":
    indice = 110
else:
    indice = 0
if (indice==0):
    print("Error. Alimento no existe en esta tabla.")
else:
    print("Índice Glicémico: ",indice)
```

```
Digite el alimento:
Cerveza
Indice Glicémico:  110
```

```
Digite el alimento:
Banano
Error. Alimento no existe en esta tabla.
```

Figura 179. Segundo ejemplo de estructura de selección múltiple en Python.

La palabra *Caso* puede ser omitida en PSeInt, a partir de este ejercicio no se colocará más, en otros lenguajes como Java o C++ se usa la palabra *case*. Cuando se encuentra en una estructura de selección múltiple en la cual varios casos realizan la misma instrucción, como en el anterior ejemplo, simplemente se agrupan los casos separándolos con comas dejándolos en una sola línea. Por ejemplo, en el caso de las cerezas y las lentejas se asigna el número 22, ya que los dos tienen el mismo índice glicémico, así: *"Cereza", "Lenteja": Indice = 22*.

Algo importante en las estructuras de selección múltiple es el control del error del usuario, en este ejemplo anterior, se usó el índice glicémico igual a cero para controlarlo. Así, después de cerrar la estructura se pregunta por este valor y se muestra el mensaje de error, de lo contrario se muestra el índice glicémico correspondiente a cada alimento.

Ejemplo 3. En un supermercado se hacen promociones de acuerdo con un número que sale al poner a girar una ruleta con números entre 0 y 9. El cliente llega a la caja registradora, pone a girar la ruleta y de acuerdo con este número obtiene los siguientes descuentos resumidos en la tabla 52.

Tabla 52.
Balotas y descuentos. Tercer ejemplo estructura de selección.

Número	Descuento (%)
0 o 1	5
2 o 3	13
4 o 5	22
6 o 7	30
8 o 9	50

Realizar un algoritmo que permita leer el número obtenido y el valor de compra. Imprimir el valor total que debe pagar un cliente determinado.

En la figura 180 se muestra el pseudocódigo en PSeInt y en la figura 181 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite número obtenido: "
    Leer Num
    Imprimir "Digite valor de la compra: $"
    Leer Val_Compra
    Segun (Num)
        0, 1: Total_Pag = Val_Compra * 0.95
        2, 3: Total_Pag = Val_Compra * 0.87
        4, 5: Total_Pag = Val_Compra * 0.78
        6, 7: Total_Pag = Val_Compra * 0.7
        8, 9: Total_Pag = Val_Compra * 0.5
        De otro modo: Imprimir "Error. Número no existe."
            Total_Pag = Val_Compra
    FinSegun
    Imprimir "Valor total a pagar: $", Total_Pag
FinAlgoritmo
```

Ejecución Caso 5

```
Digite número obtenido: > 8
Digite valor de la compra: $ > 758000
Valor total a pagar: $379000
```

Ejecución De otro modo

```
Digite número obtenido: > 11
Digite valor de la compra: $ > 780000
Error. Número no existe.
Valor total a pagar: $780000
```

Figura 180. Tercer ejemplo de estructura de selección múltiple en PSeInt.

main.py

```
print("Digite número obtenido: ")
num = int(input())
print("Digite valor de la compra: $")
val_compra = float(input())
if (num)==0 or (num)==1:
  total_pag = val_compra * 0.95
elif (num)==2 or (num)==3:
  total_pag = val_compra * 0.87
elif (num)==4 or (num)==5:
  total_pag = val_compra * 0.78
elif (num)==6 or (num)==7:
  total_pag = val_compra * 0.7
elif (num)==8 or (num)==9:
  total_pag = val_compra * 0.5
else:
  print("Error. Número no existe.")
  total_pag = val_compra
print("Valor total a pagar: $",total_pag)
```

```
Digite número obtenido:8
Digite valor de la compra: $758000
Valor total a pagar: $ 379000.0
```

```
Digite número obtenido:11
Digite valor de la compra: $780000
Error. Número no existe.
Valor total a pagar: $ 780000.0
```

Figura 181. Tercer ejemplo de estructura de selección múltiple en Python.

Ejemplo 4.

Los doce signos animales del zodiaco según el calendario chino son: rata, buey, tigre, conejo, dragón, serpiente, caballo, cabra, mono, gallo, perro y cerdo. Para explicar su origen existen varias leyendas, resumimos las dos más populares. La primera cuenta que a la rata se la encomendó la tarea de invitar a los animales para que se presentaran al banquete del Emperador de Jade para seleccionarlos como signos del zodiaco. Otra leyenda cuenta que fue el propio Emperador de Jade el que organizó una carrera y el orden de llegada de los animales fue lo que marco su posición en el zodiaco (Díaz, 2018).

El calendario chino se resume en la tabla 53.

Tabla 53.
Calendario chino. Cuarto ejemplo estructura de selección.

Años	Signo zodiacal	Animal
1924-1936-1948-1960-1972-1984-1996-2008	Sagitario	Rata
1925-1937-1949-1961-1973-1985-1997-2009	Capricornio	Buey
1926-1938-1950-1962-1974-1986-1998-2010	Acuario	Tigre
1927-1939-1951-1963-1975-1987-1999-2011	Piscis	Conejo
1928-1940-1952-1964-1976-1988-2000-2012	Aries	Dragón
1929-1941-1953-1965-1977-1989-2001-2013	Tauro	Serpiente
1930-1942-1954-1966-1978-1990-2002-2014	Géminis	Caballo
1931-1943-1955-1967-1979-1991-2003-2015	Cáncer	Cabra
1932-1944-1956-1968-1980-1992-2004-2016	Leo	Mono
1933-1945-1957-1969-1981-1993-2005-2017	Virgo	Gallo
1934-1946-1958-1970-1982-1994-2006-2018	Libra	Perro
1935-1947-1959-1971-1983-1995-2007-2019	Escorpión	Cerdo

Hacer un algoritmo que permita ingresarle el año de nacimiento de una persona (entre 1924 y 2019) y este le muestre el signo zodiacal y el animal correspondiente.

En la figura 182 se muestra el pseudocódigo en PSeInt y en la figura 183 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite año de nacimiento [entre 1.924 y 2.019]: "
    Leer A_Nac
    Segun (A_Nac)
        1924, 1936, 1948, 1960, 1972, 1984, 1996, 2008:
            Imprimir "Signo: Sagitario. Animal: Rata."
        1925, 1937, 1949, 1961, 1973, 1985, 1997, 2009:
            Imprimir "Signo: Capricornio. Animal: Buey."
        1926, 1938, 1950, 1962, 1974, 1986, 1998, 2010:
            Imprimir "Signo: Acuario. Animal: Tigre."
        1927, 1939, 1951, 1963, 1975, 1987, 1999, 2011:
            Imprimir "Signo: Piscis. Animal: Conejo."
        1928, 1940, 1952, 1964, 1976, 1988, 2000, 2012:
            Imprimir "Signo: Aries. Animal: Dragón."
        1929, 1941, 1953, 1965, 1977, 1989, 2001, 2013:
            Imprimir "Signo: Tauro. Animal: Serpiente."
        1930, 1942, 1954, 1966, 1978, 1990, 2002, 2014:
            Imprimir "Signo: Géminis. Animal: Caballo."
        1931, 1943, 1955, 1967, 1979, 1991, 2003, 2015:
            Imprimir "Signo: Cáncer. Animal: Cabra."
        1932, 1944, 1956, 1968, 1980, 1992, 2004, 2016:
            Imprimir "Signo: Leo. Animal: Mono."
        1933, 1945, 1957, 1969, 1981, 1993, 2005, 2017:
            Imprimir "Signo: Virgo. Animal: Gallo."
        1934, 1946, 1958, 1970, 1982, 1994, 2006, 2018:
            Imprimir "Signo: Libra. Animal: Perro."
        1935, 1947, 1959, 1971, 1983, 1995, 2007, 2019:
            Imprimir "Signo: Escorpión. Animal: Cerdo."
        De otro modo: Imprimir "Año no está en el rango."
    FinSegun
FinAlgoritmo
```

Ejecución Caso 4

```
Digite año de nacimiento [entre 1.924 y 2.019]:
> 1975
Signo: Piscis. Animal: Conejo.
```

Ejecución De otro modo

```
Digite año de nacimiento [entre 1.924 y 2.019]:
> 1920
Año no está en el rango.
```

Figura 182. Cuarto ejemplo de estructura de selección múltiple en PSeInt.

main.py

```
print("Digite año de nacimiento [entre 1.924 y 2.019]: ")
a_nac = float(input())
if a_nac==1924 or a_nac==1936 or a_nac==1948 or a_nac==1960
or a_nac==1972 or a_nac==1984 or a_nac==1996 or a_nac==2008:
  print("Signo: Sagitario. Animal: Rata.")
elif a_nac==1925 or a_nac==1937 or a_nac==1949 or a_nac==1961
or a_nac==1973 or a_nac==1985 or a_nac==1997 or a_nac==2009:
  print("Signo: Capricornio. Animal: Buey.")
elif a_nac==1926 or a_nac==1938 or a_nac==1950 or a_nac==1962
or a_nac==1974 or a_nac==1986 or a_nac==1998 or a_nac==2010:
  print("Signo: Acuario. Animal: Tigre.")
elif a_nac==1927 or a_nac==1939 or a_nac==1951 or a_nac==1963
or a_nac==1975 or a_nac==1987 or a_nac==1999 or a_nac==2011:
  print("Signo: Piscis. Animal: Conejo.")
elif a_nac==1928 or a_nac==1940 or a_nac==1952 or a_nac==1964
or a_nac==1976 or a_nac==1988 or a_nac==2000 or a_nac==2012:
  print("Signo: Aries. Animal: Dragón.")
elif a_nac==1929 or a_nac==1941 or a_nac==1953 or a_nac==1965
or a_nac==1977 or a_nac==1989 or a_nac==2001 or a_nac==2013:
  print("Signo: Tauro. Animal: Serpiente.")
elif a_nac==1930 or a_nac==1942 or a_nac==1954 or a_nac==1966
or a_nac==1978 or a_nac==1990 or a_nac==2002 or a_nac==2014:
  print("Signo: Géminis. Animal: Caballo.")
elif a_nac==1931 or a_nac==1943 or a_nac==1955 or a_nac==1967
or a_nac==1979 or a_nac==1991 or a_nac==2003 or a_nac==2015:
  print("Signo: Cáncer. Animal: Cabra.")
elif a_nac==1932 or a_nac==1944 or a_nac==1956 or a_nac==1968
or a_nac==1980 or a_nac==1992 or a_nac==2004 or a_nac==2016:
  print("Signo: Leo. Animal: Mono.")
elif a_nac==1933 or a_nac==1945 or a_nac==1957 or a_nac==1969
or a_nac==1981 or a_nac==1993 or a_nac==2005 or a_nac==2017:
  print("Signo: Virgo. Animal: Gallo.")
elif a_nac==1934 or a_nac==1946 or a_nac==1958 or a_nac==1970
or a_nac==1982 or a_nac==1994 or a_nac==2006 or a_nac==2018:
  print("Signo: Libra. Animal: Perro.")
elif a_nac==1935 or a_nac==1947 or a_nac==1959 or a_nac==1971
or a_nac==1983 or a_nac==1995 or a_nac==2007 or a_nac==2019:
  print("Signo: Escorpión. Animal: Cerdo.")
else:
  print("Año no está en el rango.")
```

```
Digite año de nacimiento [entre 1.924 y 2.019]:
1975
Signo: Piscis. Animal: Conejo.
```

```
Digite año de nacimiento [entre 1.924 y 2.019]:
1920
Año no está en el rango.
```

Figura 183. Cuarto ejemplo de estructura de selección múltiple en Python.

3.7 Ejercicios resueltos

3.7.1 Primer ejercicio

En una institución de educación básica primaria se maneja un equivalente entre calificaciones cuantitativas (enteras) y calificaciones cualitativas. La calificación 5 es equivalente a excelente, 4 es bueno, 3 es aceptable, 2 es insuficiente y 1 es deficiente. Hacer un algoritmo que permita leer esa calificación cuantitativa y muestre la nota cualitativa.

En la figura 184 se muestra el pseudocódigo en PSeInt y en la figura 185 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Digite nota [valor entero]: "
3      Leer Nota
4      Segun (Nota)
5          5: Equivalente = "Excelente"
6          4: Equivalente = "Bueno"
7          3: Equivalente = "Aceptable"
8          2: Equivalente = "Insuficiente"
9          1: Equivalente = "Deficiente"
10         De Otro Modo: Equivalente = ""
11     FinSegun
12     Si (Equivalente = "") Entonces
13         Imprimir "Nota incorrecta"
14     Sino
15         Imprimir "Nota Equivale a: ", Equivalente
16     FinSi
17 FinAlgoritmo
```

Ejecución Caso 1

```
Digite nota [valor entero]:
> 5
Nota Equivale a: Excelente
```

Ejecución De otro modo

```
Digite nota [valor entero]:
> 6
Nota incorrecta
```

Figura 184. Primer ejercicio resuelto estructura de selección múltiple en PSeInt.

main.py

```
print("Digite nota [valor entero]: ")
nota = int(input())
if (nota)==5:
  equivalente = "Excelente"
elif (nota)==4:
  equivalente = "Bueno"
elif (nota)==3:
  equivalente = "Aceptable"
elif (nota)==2:
  equivalente = "Insuficiente"
elif (nota)==1:
  equivalente = "Deficiente"
else:
  equivalente = ""
if (equivalente==""):
  print("Nota incorrecta")
else:
  print("Nota Equivale a: ", equivalente)
```

```
Digite nota [valor entero]:
5
Nota Equivale a:  Excelente
```

```
Digite nota [valor entero]:
6
Nota incorrecta
```

Figura 185. Primer ejercicio resuelto estructura de selección múltiple en Python.

3.7.2 Segundo ejercicio

En una central telefónica se tiene información con los indicativos de cada una de las ciudades de Colombia y sus tarifas, resumidas en la siguiente tabla 54.

Tabla 54.
Ciudades y tarifas. Ejercicio 3.7.2.

Indicativo	Ciudad	Tarifa por minuto ($)
1	Bogotá	50
2	Cali	70
4	Medellín	100
5	Barranquilla	160
6	Pereira	180
7	Cúcuta	190

Desarrolle un algoritmo que acepte como entrada un indicativo y el número de minutos que dura una llamada. Mostrar la ciudad, la tarifa por minuto y el valor total que se debe pagar un usuario por dicha llamada.

En la figura 186 se muestra el pseudocódigo en PSeInt y en la figura 187 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite el indicativo: "
    Leer Indicativo
    Imprimir "Digite # de minutos: "
    Leer Num_Min
    Segun (Indicativo)
        Caso 1: Val = Num_Min * 50
            Ciudad = "Bogotá"
            Tarifa = 50
        Caso 2: Val = Num_Min * 70
            Ciudad = "Cali"
            Tarifa = 70
        Caso 4: Val = Num_Min * 100
            Ciudad = "Medellín"
            Tarifa = 100
        Caso 5: Val = Num_Min * 160
            Ciudad = "Barranquilla"
            Tarifa = 160
        Caso 6: Val = Num_Min * 180
            Ciudad = "Pereira"
            Tarifa = 180
        Caso 7: Val = Num_Min * 190
            Ciudad = "Cúcuta"
            Tarifa = 190
        De Otro Modo: Imprimir "Indicativo no existe. "
            Val = 0
            Ciudad = "Ninguna"
            Tarifa = 0
    FinSegun
    Imprimir "Ciudad a la que marca: ", Ciudad
    Imprimir "Tarifa: $", Tarifa
    Imprimir "Valor llamada: $", Val
FinAlgoritmo
```

Ejecución Caso 3

```
Digite el indicativo: > 4
Digite # de minutos: > 8
Ciudad a la que marca: Medellín
Tarifa: $100
Valor llamada: $800
```

Ejecución De otro modo

```
Digite el indicativo: > 3
Digite # de minutos: > 2
Indicativo no existe.
Ciudad a la que marca: Ninguna
Tarifa: $0
Valor llamada: $0
```

Figura 186. Segundo ejercicio resuelto estructura de selección múltiple en PSeInt.

main.py

```
print("Digite el indicativo: ")
indicativo = int(input())
print("Digite # de minutos: ")
num_min = int(input())
if indicativo == 1:
  val = num_min * 50
  ciudad = "Bogotá"
  tarifa = 50
elif indicativo == 2:
  val = num_min * 70
  ciudad = "Cali"
  tarifa = 70
elif indicativo == 4:
  val = num_min * 100
  ciudad = "Medellín"
  tarifa = 100
elif indicativo == 5:
  val = num_min * 160
  ciudad = "Barranquilla"
  tarifa = 160
elif indicativo == 6:
  val = num_min * 180
  ciudad = "Pereira"
  tarifa = 180
elif indicativo == 7:
  val = num_min * 190
  ciudad = "Cúcuta"
  tarifa = 190
else:
  print("Indicativo no existe. ")
  val = 0
  ciudad = "Ninguna"
  tarifa = 0
print("Ciudad a la que marca: ", ciudad)
print("Tarifa: $", tarifa)
print("Valor llamada: $", val)
```

```
Digite el indicativo: 4
Digite # de minutos: 8
Ciudad a la que marca:  Medellín
Tarifa: $ 100
Valor llamada: $ 800
```

```
Digite el indicativo: 3
Digite # de minutos: 2
Indicativo no existe.
Ciudad a la que marca:  Ninguna
Tarifa: $ 0
Valor llamada: $ 0
```

Figura 187. Segundo ejercicio resuelto estructura de selección múltiple en Python.

Una buena práctica de programación es evitar que se muestren mensajes en cero como la última salida en PSeInt; porque lo lógico es que muestre los resultados, solo en caso de haber ingresado a alguna de las opciones. Luego de cerrar la estructura de selección múltiple *(FinSegun)*, se pregunta si la tarifa es igual a cero, imprimir un mensaje mostrando que se presentó un error en el indicativo ingresado; de lo contrario, se imprime toda la información.

3.7.3 Tercer ejercicio

En una empresa se implementó un *software* en el cual se ingresa una extensión telefónica de alguna de las oficinas y carga en pantalla la información de la dependencia y la persona encargada. Diseñar un algoritmo que simule este proceso. Se debe tener en cuenta la siguiente estructura: extensión-dependencia (encargado).

Los datos son los siguientes: 750-Decanatura (Alejandro Álvarez), 751-Sistemas (Alexander García), 752-Laboratorio Redes (Beatriz Flórez), 753-Humanidades (Felipe Álvarez), 754-Dirección (Gustavo Moreno), 755-Coordinación Ingeniería (Iván Zapata), 756-Virtualidad (Jairo Ramírez), 757-Desarrollo (Marco Caramma), 758-Decanatura Artes (Mauricio Guzmán), 759-Investigación (Oscar Mesa), 760-Programador (Joseph Raigoza) y 780.Ciencias Básicas (Óscar Saavedra).

En la figura 188 se muestra el pseudocódigo en PSeInt y en la figura 189 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Digite el número de la extensión: "
3      Leer Extension
4      Segun(Extension)
5          750: Imprimir "Decanatura (Alejandro Álvarez)."
6          751: Imprimir "Sistemas (Alexander García)."
7          752: Imprimir "Laboratorio Redes (Ledy Cañaveral)."
8          753: Imprimir "Humanidades (Felipe Álvarez)."
9          754: Imprimir "Dirección (Gustavo Moreno)."
10         755: Imprimir "Coordinación Ingeniería (Iván Zapata)."
11         756: Imprimir "Virtualidad (Jairo Ramírez)."
12         757: Imprimir "Desarrollo (Miguel Ángel Ramírez)."
13         758: Imprimir "Decanatura Artes (Mauricio Guzmán)."
14         759: Imprimir "Investigación (Oscar Mesa)."
15         760: Imprimir "Programador (Joseph Raigoza)."
16         780: Imprimir "Ciencias Básicas (Oscar Saavedra)."
17         De Otro Modo: Imprimir "Error. Extensión no existe."
18     FinSegun
19 FinAlgoritmo
```

Ejecución Caso 11

```
Digite el número de la extensión:
> 760
Programador (Joseph Raigoza).
```

Ejecución De otro modo

```
Digite el número de la extensión:
> 770
Error. Extensión no existe.
```

Figura 188. Tercer ejercicio resuelto estructura de selección múltiple en PSeInt.

main.py

```
print("Digite el número de la extensión: ")
extension = int(input())
if extension == 750:
  print("Decanatura (Alejandro Álvarez).")
elif extension == 751:
  print("Sistemas (Alexander García).")
elif extension == 752:
  print("Laboratorio Redes (Ledy Cañaveral).")
elif extension == 753:
  print("Humanidades (Felipe Álvarez).")
elif extension == 754:
  print("Dirección (Gustavo Moreno).")
elif extension == 755:
  print("Coordinación Ingeniería (Iván Zapata).")
elif extension == 756:
  print("Virtualidad (Jairo Ramírez).")
elif extension == 757:
  print("Desarrollo (Miguel Ángel Ramírez).")
elif extension == 758:
  print("Decanatura Artes (Mauricio Guzmán).")
elif extension == 759:
  print("Investigación (Oscar Mesa).")
elif extension == 760:
  print("Programador (Joseph Raigoza).")
elif extension == 780:
  print("Ciencias Básicas (Oscar Saavedra).")
else:
  print("Error. Extensión no existe.")
```

```
Digite el número de la extensión:
760
Programador (Joseph Raigoza).
```

```
Digite el número de la extensión:
770
Error. Extensión no existe.
```

Figura 189. Tercer ejercicio resuelto estructura de selección múltiple en Python.

3.7.4 Cuarto ejercicio

Hacer un algoritmo que lea el signo zodiacal de una persona e imprima el elemento al que corresponde, de acuerdo con la tabla 55.

Tabla 55.
Signos zodiacales. Ejercicio 3.7.4.

Signo zodiacal	Elemento
Aries-Leo-Sagitario	Fuego
Tauro-Virgo-Capricornio	Tierra
Géminis-Libra-Acuario	Aire
Cáncer-Escorpión-Piscis	Agua

En la figura 190 se muestra el pseudocódigo en PSeInt y en la figura 191 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
1  Algoritmo Inicio
2      Imprimir "Digite el signo zodiacal: "
3      Leer Signo_Zod
4      Signo_Zod = Minusculas(Signo_Zod)
5      Segun(Signo_Zod)
6          "aries", "leo", "sagitario": El = "Fuego"
7          "tauro", "virgo", "capricornio": El = "Tierra"
8          "géminis", "libra", "acuario":  El = "Aire"
9          "cáncer", "escorpión", "piscis": El = "Agua"
10         De Otro Modo:  El = ""
11     FinSegun
12     Si (El = "") Entonces
13         Imprimir "Signo zodiacal no existe."
14     SiNo
15         Imprimir "El elemento es: ",  El
16     FinSi
17 FinAlgoritmo
```

Ejecución Caso 3

```
Digite el signo zodiacal:
> Libra
El elemento es: Aire
```

Ejecución De otro modo

```
Digite el signo zodiacal:
> Cancer
Signo zodiacal no existe.
```

Figura 190. Cuarto ejercicio resuelto estructura de selección múltiple en PSeInt.

main.py

```
print("Digite el signo zodiacal: ")
signo = input()
signo = str.lower(signo)
if (signo)=="aries" or (signo)=="leo" or (signo)=="sagitario":
  el = "Fuego"
elif (signo)=="tauro" or (signo)=="virgo" or (signo)=="capricornio":
  el = "Tierra"
elif (signo)=="géminis" or (signo)=="libra" or (signo)=="acuario":
  el = "Aire"
elif (signo)=="cáncer" or (signo)=="escorpión" or (signo)=="piscis":
  el = "Agua"
else:
  el = ""
if (el == ""):
  print("Signo zodiacal no existe.")
else:
  print("El elemento es: ", el)
```

```
Digite el signo zodiacal:
Libra
El elemento es:  Aire
```

```
Digite el signo zodiacal:
Cancer
Signo zodiacal no existe.
```

Figura 191. Cuarto ejercicio resuelto estructura de selección múltiple en Python.

3.7.5 Quinto ejercicio

Hacer un algoritmo que le permita a un cliente solicitar un crédito en un almacén del centro de la ciudad y mostrarle el artículo que debe llevar dependiendo del plazo que tenga para pagarlo. Tenga en cuenta la tabla 56.

Tabla 56.
Plazos y artículos. Ejercicio 3.7.5.

Plazo en meses	Artículo
48	Computadora
36	TV 42 pulgadas
24	BlackBerry
12	Cámara digital
6	Impresora

En la figura 192 se muestra el pseudocódigo en PSeInt y en la figura 193 se muestra la solución codificada en Python, cada una con sus respectivas ejecuciones.

```
Algoritmo Inicio
    Imprimir "Digite el plazo otorgado: "
    Leer Plazo
    Segun(Plazo)
        48: Art = "un computador."
        36: Art = "un TV de 42 pulgadas."
        24: Art = "un BlackBerry."
        12: Art = "una cámara digital."
        6:  Art = "una impresora."
        De Otro Modo: Art = ""
    FinSegun
    Si (Art = "") Entonces
        Imprimir "Error. Plazo no existe."
    SiNo
        Imprimir "Podría llevar ", Art
    FinSi
FinAlgoritmo
```

Ejecución Caso 2

```
Digite el plazo otorgado:
> 36
Podría llevar un TV de 42 pulgadas.
```

Ejecución De otro modo

```
Digite el plazo otorgado:
> 8
Error. Plazo no existe.
```

Figura 192. Quinto ejercicio resuelto estructura de selección múltiple en PSeInt.

main.py

```python
print("Digite el plazo otorgado: ")
plazo = int(input())
if (plazo)==48:
    art = "un computador."
elif (plazo)==36:
    art = "un TV de 42 pulgadas."
elif (plazo)==24:
    art = "un BlackBerry."
elif (plazo)==12:
    art = "una cámara digital."
elif (plazo)==6:
    art = "una impresora."
else:
    art = ""
if (art==""):
    print("Error. Plazo no existe.")
else:
    print("Podría llevar ",art)
```

```
Digite el plazo otorgado:
36
Podría llevar  un TV de 42 pulgadas.
```

```
Digite el plazo otorgado:
8
Error. Plazo no existe.
```

Figura 193. Quinto ejercicio resuelto estructura de selección múltiple en Python.

3.8 Ejercicios propuestos

3.8.1 Primer ejercicio

Diseñar un pseudocódigo que permita leer la masa atómica de un elemento químico y luego imprimir su nombre, teniendo en cuenta la tabla 57.

Tabla 57.
Elementos y masas atómicas. Ejercicio 3.8.1.

Masa atómica	Elemento
1.01	Hidrógeno
16	Oxígeno
6.94	Litio
9.01	Berilio
14.01	Nitrógeno
19	Flúor
26.98	Aluminio
22.99	Sodio

3.8.2 Segundo ejercicio

Hacer un algoritmo que permita ingresar el número de un mes del año (número entre 1 y 12) e imprima el nombre del mes al que corresponde este número. Tenga en cuenta que el número 1 es para enero, el 2 es para febrero, y así sucesivamente hasta el número 12 que corresponde a diciembre. En caso de ingresarse un número diferente mostrar un mensaje de error.

3.8.3 Tercer ejercicio

Una persona desea viajar a una de las siguientes ciudades de Colombia. Los códigos, las ciudades y los departamentos se encuentran resumidos en la tabla 58.

Tabla 58.
Códigos, ciudades y departamentos. Ejercicio 3.8.3.

Código	Ciudad	Departamento
101	Medellín	Antioquia
102	Bogotá	Cundinamarca
103	Cali	Valle del Cauca
104	Bucaramanga	Santander
105	Cartagena	Bolívar
106	Armenia	Quindío
107	Santa Marta	Magdalena
108	Tunja	Boyacá

Desarrollar un algoritmo que permita leer el código de una de esas ciudades y muestre el nombre de la ciudad y el nombre del departamento correspondiente.

3.8.4 Cuarto ejercicio

En un peaje creado en las afueras de la ciudad se requiere hacer prueba a un sensor ubicado en la carretera. El sensor captura un código de barras donde se obtiene el número de la placa, el peso y el tipo de vehículo.

Hacer un algoritmo que permita leer, por separado, cada uno de los campos del código de barras e imprima el tipo de vehículo al cual corresponde. Tenga en cuenta lo siguientes tipos: 1-Escolar, 2-Familiar, 3-Blindado, 4-Servicio público, 5-Ambulancia, 6-Funerario, 7-Grúas y 8-Vehículos pesados.

3.8.5 Quinto ejercicio

Hacer un algoritmo que lea dos números y una opción. Dependiendo la opción (letra inicial de cada operación) realice las operaciones que corresponda. Apoyarse en estos datos donde se cuenta con lo siguiente (tener en cuenta que se mostrarán *C***ódigo**-*Operación*), por ejemplo, en la primera opción "S", se debe realizar la operación de suma y así, sucesivamente con las demás.

Las opciones con sus respectivas operaciones son: S-Suma, R-Resta, M-Multiplicación, D-División, E-El primer número elevado al segundo y G-El segundo número elevado al primero.

3.8.6 Sexto ejercicio

Hacer un algoritmo que permita leer el nombre de una de las asignaturas matriculadas por un estudiante de la Institución Universitaria Salazar y Herrera y muestre el nombre del profesor, horario, número de créditos y valor de la asignatura. Para lo anterior tenga en cuenta que cada crédito tiene un valor que debe ser leído y los datos son tomados de la tabla 59.

Tabla 59.
Asignaturas, docentes, horarios y créditos. Ejercicio 3.8.6.

Asignatura	Docente	Horario	Créditos
Competencias Comunicativas	Natalia Rojas	Lunes 6-8 a. m.	4
Matemáticas I	Óscar Saavedra	Martes 6-8 a. m.	3
Lógica de Programación	Jairo Ramírez	Martes 8-10 a. m.	3
Direccionamiento	Carlos Pérez	Miércoles 10-12 m.	2
Competencias Informáticas	Sandra Berrio	Jueves 8-10 a. m.	2
Introducción	Andrés Arias	Viernes 10-12 m.	2
Física I	Germán Vélez	Viernes 12–2 p. m.	4

3.8.7 Séptimo ejercicio

Diseñar un algoritmo que permita leer el código de la carrera a la cual desea inscribirse una persona y el número de materias que desea cursar. Imprima el nombre de la carrera y el valor total que debe pagar por las materias teniendo en cuenta la tabla 60.

Tabla 60.
Carreras y valores de materias. Ejercicio 3.8.7.

Código	Carrera	Valor materia ($)
01	Sistemas	400 000
02	Administración	380 000
03	Contaduría	390 000
04	Agronomía	200 000

3.8.8 Octavo ejercicio

En un anuario mundial de deportistas, en la tabla 61, aparece el resumen de año de nacimiento, nombre del deportista, mes de nacimiento, deporte destacado y nacionalidad.

Tabla 61.
Resumen de deportistas. Ejercicio 3.8.8.

Año	Deportista	Nacimiento	Deporte	Nacionalidad
1981	Javier Pedro Saviola	Diciembre	Fútbol	Argentina
1977	Emanuel Ginóbili	Julio	Baloncesto	Argentina
1975	Tiger Woods	Diciembre	Golf	Estadounidense
1971	Lance Armstrong	Septiembre	Ciclismo	Estadounidense
1969	Michael Schumacher	Enero	Automovilismo	Alemana
1967	Michael Johnson	Septiembre	Atletismo	Estadounidense
1963	Gari Kaspárov	Abril	Ajedrez	Rusa
1956	Martina Navrátilová	Octubre	Tenis	Checoslovaca

Se requiere de un algoritmo que permita leer el año de nacimiento de alguno de los deportistas e imprima todos sus datos.

3.8.9 Noveno ejercicio

Hacer un algoritmo que permita ingresar el número de un día de la semana y el tipo de calendario (gregoriano o juliano). Imprimir el nombre del día al que corresponde este número. Tenga en cuenta que en el calendario gregoriano el número 1 es para el lunes, el 2 es para el martes, y así sucesivamente, hasta el número 7 que corresponde al domingo; mientras que en calendario juliano el 0 es para el domingo, el 1 es para el lunes y así sucesivamente hasta el número 6 que corresponde al sábado. En caso de ingresarse un número diferente (dependiendo el calendario) mostrar un mensaje de error.

3.8.10 Décimo ejercicio

En un laboratorio se requiere implementar un algoritmo que permita leer el número atómico de alguno de los elementos químicos registrados en la tabla 62. Mostrar mensajes con el nombre del elemento, el símbolo químico y la masa atómica.

Tabla 62.
Elementos, símbolos, masas y números atómicas. Ejercicio 3.8.10.

# Atómico	Nombre elemento	Símbolo químico	Masa atómica
1	Hidrógeno	H	1.0079
2	Helio	He	4.0026
3	Litio	Li	6.941
4	Berilio	Be	9.0121
5	Boro	B	10.811
6	Carbono	C	12.01107
7	Nitrógeno	N	14.0067
8	Oxígeno	O	15.9994
9	Flúor	F	18.9984
10	Neón	Ne	20.1797

3.8.11 Décimo primer ejercicio

En un montallantas antioqueño se tomó la decisión de abrir sucursales en varios países del mundo debido a la firma del Tratado de Libre Comercio (TLC). Dentro de su publicidad electrónica debe hacerse un cambio; debido a que el término *llanta* es usado en Colombia y algunos países vecinos, pero en otros no existe o tienen otro significado. Se pide plantear una solución donde se ingrese de forma manual el código del país al que llegue uno de los tableros electrónicos e imprima el término correcto para "llanta" dependiendo de la tabla 63.

Tabla 63.
Países y términos. Ejercicio 3.8.11.

Código	País	Término usado
063	Argentina	Cubierta
211	Chile	Neumático
169	Colombia	Llanta
199	Cuba	Goma
245	España	Neumático
586	Paraguay	Cubierta
611	Puerto Rico	Goma
647	República Dominicana	Goma
845	Uruguay	Cubierta
850	Venezuela	Caucho

3.8.12 Décimo segundo ejercicio

Un presentador colombiano de televisión fue trasladado a un programa de deportes en Estados Unidos. Al tener tantas dificultades con las siglas más usadas en estos deportes, solicita la realización de un algoritmo que reciba una sigla y muestre su significado en inglés y la traducción al español. El listado que entrega el presentador se presenta en la tabla 64.

Tabla 64.
Siglas, significado y traducción. Ejercicio 3.8.12.

Sigla	Inglés	Español
NBA	National Basketball Association	Asociación Nacional de Baloncesto
NFL	National Football League	Liga Nacional de Fútbol Americano
MLS	Major League Soccer	Liga Mayor de Fútbol
MLB	Major League Baseball	Ligas Mayores de Béisbol
NHL	National Hockey League	Liga Nacional de Hockey
PGA	Professional Golf Associate	Asociación Profesional de Golf

3.8.13 Décimo tercer ejercicio

Determinar el salario neto de un empleado sabiendo que este cálculo se realiza con base al tipo de empleado (55-Temporal, 66-Vinculado, 77-Administrativo y 88-Gerencia). Si es temporal recibe un salario de $ 550 000, si es vinculado recibe un salario de $ 1 000 000, si es administrativo recibe un salario de $ 1 400 000 y si es de gerencia recibe un salario de $ 2 000 000. A todos los empleados se les hace una retención del 5% para ahorros.

3.8.14 Décimo cuarto ejercicio

La dureza de los materiales se mide por la escala de Mohs, la cual tiene una relación de 10 materiales clasificados de menor a mayor en relación con su dureza. Hacer un algoritmo que permita ingresar el número de dureza de un mineral y se muestre el nombre y las observaciones de este. Para esta solución, tomar como base la tabla 65.

Tabla 65.
Minerales y durezas. Ejercicio 3.8.14.

Dureza	Nombre mineral	Observación del mineral
1	Talco	Se raya fácilmente con la uña
2	Yeso	Se raya con la uña con más dificultad
3	Calcita	Se raya con una moneda de cobre
4	Fluorita	Se raya con un cuchillo de acero
5	Apatito	Se raya difícilmente con un cuchillo
6	Ortoclasa	Se raya con una lija para el acero
7	Cuarzo	Puede rayar el vidrio
8	Topacio	Rayado por herramientas de carburo de wolframio
9	Corindón	Rayado por herramientas de carburo de silicio
10	Diamante	Es el más duro. Rayado solo por otro diamante

3.8.15 Décimo quinto ejercicio

Desarrollar un algoritmo que lea el nombre de una compañía y muestre el producto insignia que fabrica. Tenga que cuenta que Bridgestone, Firestone y Goodyear son compañías que fabrican neumáticos. Fabricato, Coltejer y Leonisa fabrican telas. Coca-Cola, Pepsi y Postobón producen gaseosas. Grupo Nutresa, Colanta, Parmalat y Alpina producen alimentos.

3.8.16 Décimo sexto ejercicio

Las tarifas de vehículos que pasan por el túnel de Occidente se encuentran consolidas en la tabla 66. Hacer un algoritmo que lea la categoría del vehículo e imprima la descripción del vehículo y el valor de la tarifa.

Tabla 66.
Categorías y tarifas de vehículos. Ejercicio 3.8.16.

Categoría	Descripción	Valor tarifa ($)
1	Auto, campero y camionetas	15 000
2	Bus, buseta y microbús	18 000
3	Camiones de 3 ejes	35 000
4	Camiones de 5 ejes	46 000
5	Camiones de 6 ejes	57 000
6	Camiones con más de 6 ejes	57 000 + 5000 por eje adicional

3.8.17 Décimo séptimo ejercicio

En un almacén venden varios dispositivos de almacenamiento para computadoras, pero siempre se ha tenido dificultad con la información relacionada con los diferentes CD y DVD.

Se pide hacer un algoritmo que permita digitar la referencia de un dispositivo que pertenezca a la tabla 67 y muestre toda la información pertinente.

Tabla 67.
Dispositivos y capacidades. Ejercicio 3.8.17.

Referencia	Dispositivo	Capacidad	Duración máxima audio	Duración máxima video
21011	Disco compacto (CD)	650 Mb	1 hora 18 min	15 minutos
21012	DVD 1 cara/1 capa	4.7 Gb	9 horas 30 min	2 horas 15 min
21013	DVD 1 cara/2 capas	8.5 Gb	17 horas 30 min	4 horas
10401	DVD 2 caras/1 capa	9.4 Gb	19 horas	4 horas 30 min
10402	DVD 2 caras/2 capas	17 Gb	35 horas	8 horas

3.8.18 Décimo octavo ejercicio

En un centro automotriz se venden vehículos de varias marcas (1: Toyota, 2: Mazda, 3: Chevrolet y 4: Otro) y dos tipos (1: Nacional y 2: Importado). Se pide hacer un algoritmo que lea la marca y el tipo de un vehículo e imprima su valor comercial y su valor real (el valor real es igual al valor comercial más $ 1 000 000 si es nacional y $ 2 500 000 si es importado). Realizar las operaciones teniendo en cuenta la tabla 68.

Tabla 68.
Elementos y masas atómicas. Ejercicio 3.8.18.

Marca	Tipo	Valor Comercial ($)
Toyota	Nacional	40 000 000
Mazda	Nacional	45 000 000
Chevrolet	Nacional	30 000 000
Toyota	Importado	55 000 000
Mazda	Importado	65 000 000
Chevrolet	Importado	38 000 000
Otro	Nacional o Importado	30 000 000

Nota: para este ejercicio se debe utilizar estructuras de selección múltiple para una de las variables y estructuras de decisión para la otra.

3.8.19 Décimo noveno ejercicio

El trabajo con funciones en Microsoft Excel genera algunos errores que muchas veces no se comprenden, los cuales se resumen en la tabla 69. Diseñar un algoritmo que ingrese el código del error y muestre el error al que corresponde y una explicación de este.

Tabla 69.
Códigos y errores de Excel. Ejercicio 3.8.19.

Código	Error	Explicación
ERRO231	######	Valor supera el ancho de la columna.
ERRO235	#¿NOMBRE?	Nombre de una fórmula no existe.
ERRO241	#¡REF!	Referencia que no existe.
ERRO251	¡DIV/0!	División por cero o por una celda vacía.
ERRO255	#¡VALOR!	Argumento de fórmula no es el esperado.
ERRO265	#¡NUM!	Valor numérico no válido en una fórmula.
ERRO275	#¡NULO!	Intersección de dos áreas equivocada.
ERRO299	#N/A	Valor no está disponible para una fórmula.

3.8.20 Vigésimo ejercicio

Un arqueólogo ha presentado un proyecto de investigación de culturas antiguas al ministerio de cultura de su país; este siempre que recibe propuestas las evalúa y responde con un valor encriptado (clave) que le hace llegar a las personas que se inscriben. Esta clave les permite a las personas conocer el tipo de presupuesto, el valor de presupuesto entregado y la cultura que le aprueban investigar. Todo se resume en la tabla 70.

Tabla 70.
Claves y presupuestos. Ejercicio 3.8.20.

Clave	Tipo presupuesto	Valor presupuesto	Cultura
QWRO1	Excelente	$99 000 000 000	Cultura egipcia
H1JEPD	Muy bueno	$80 000 000 000	Imperio maya
RWQC9K	Bueno	$40 000 000 000	Imperio romano
GTNGFT	Regular	$10 000 000 000	Cultura inca
JZR61A	Malo	$5 000 000 000	Cultura tairona

Hacer un algoritmo que le permita a este arqueólogo traducir su clave recibida.

3.9 Práctica final de estructuras de decisión y selección

Nota: con este tipo de ejercicio, se pretende consolidar todos los conceptos aprendidos en esta unidad; a pesar de lo extenso, recuerde que los algoritmos y la programación siempre buscarán dividir un problema en fracciones más pequeñas que faciliten llegar a una solución clara, coherente y eficiente. El enunciado es el siguiente:

Decamerón y Avianca a través de un convenio ofrecen planes de viajes a una serie de empresas en Colombia brindándoles deferentes alternativas. Al momento de tomar un plan por parte de alguna empresa se le pide a dicha empresa el código del plan, el tipo de empresa (Privada o Pública), el total de dinero destinado para cubrir el valor total del plan (en pesos) y el mes en el que se solicita el mismo. A la agencia se le solicita el valor de cada kilómetro y el valor actual del dólar (TRM).

Sacar un informe con la ciudad de destino y la distancia del viaje en kilómetros teniendo en cuenta el código del plan y la tabla 71.

Tabla 71.
Información planes. Práctica final.

Código plan	Ciudad destino	Kilómetros
P_1	San Andrés	974
P_2	Cartagena	460.74
P_3	Barranquilla	535.41
P_4	Aruba	559.28
P_5	Tolú	364.52
P_6	Santa Marta	578.4

Tener en cuenta las siguientes equivalencias y la siguiente tabla: 1 pie equivale a 0.3084 metros; 1 braza equivale a 6 Pies; 1 legua equivale a 5572.7 metros y 1 milla equivale a 1.609 kilómetros.

El trimestre actual dependiendo el mes en el que se solicita el plan. Mostrar PRIMER TRIMESTRE para los meses enero, febrero o marzo, SEGUNDO TRIMESTRE para abril, mayo o junio, TERCER TRIMESTRE para julio, agosto o septiembre o CUARTO TRIMESTRE para octubre, noviembre y diciembre.

El valor inicial del viaje (multiplicación del número de kilómetros por el valor de cada kilómetro) y el valor descuento dependiendo del tipo de empresa y el número de grupos que van a tomar el plan (véase tabla 72).

Tabla 72.
Información de descuentos. Práctica final.

Tipo empresa	# de grupos	Descuento (%)
Privada	Entre 1 y 5	0
Privada	Entre 5 y 10	3.5
Privada	Más de 10	8.5
Pública	Entre 1 y 5	2.55
Pública	Entre 5 y 10	7.5
Pública	Más de 10	8

El valor real del viaje en pesos (valor después del descuento), el valor real del viaje en dólares, el valor entregado por la empresa y el valor devuelto a la empresa luego del pago (validar si alcanza o no el dinero).

4. Solución a los ejercicios propuestos

A continuación, la solución de los ejercicios del numeral 1.3.3.

a. 2 + 3 * 4
   2 + 12
   14

b. 10 ^ 2 ^ 2 / 7
   100 ^ 2 / 7
   10000 / 7
   1428.5714285714 ≈ 1428.57

c. 42 / 2 / 3 / 7 ^ 3
   42 / 2 / 3 / 343
   21 / 3 / 343
   7 / 343
   0.0204081632 ≈ 0.02

d. 8 + 7 * 3 + 4 * 6
   8 + 21 + 4 * 6
   8 + 21 + 24
   29 + 24
   53

e. 5 ^ 3 + 120 – 3 ^ 5
   125 + 120 – 3 ^ 5
   125 + 120 – 243
   245 – 243
   2

f. 12 + 3 * 7 + 5 * 4
12 + 21 + 5 * 4
12 + 21 + 20
33 + 20
53

g. 4 + 5 ^ 2 – 500 + 20 ^ 2 * 10
4 + 25 – 500 + 20 ^ 2 * 10
4 + 25 – 500 + 400 * 10
4 + 25 – 500 + 4000
29 – 500 + 4000
–471 + 4000
3529

h. (4 ^ 3 + 2 * 40 – 140) ^ 2
(64 + 2 * 40 – 140) ^ 2
(64 + 80 – 140) ^ 2
(144 – 140) ^ 2
4 ^ 2
16

i. 5 + (25 * 2 + 5 * 8 / 2 – 10) * 2
5 + (50 + 5 * 8 / 2 – 10) * 2
5 + (50 + 40 / 2 – 10) * 2
5 + (50 + 20 – 10) * 2
5 + (70 – 10) * 2
5 + 60 * 2
5 + 120
125

j. 2 * 25 / 5 + 5 ^ 2 – 5
2 * 25 / 5 + 25 – 5
2 * 5 + 25 – 5
10 + 25 – 5
35 – 5
30

k. 15 – ((4 * 5 + (8 / 2 – 5) * 2) + 8 * 10 / 2)
15 – ((4 * 5 + (4 – 5) * 2) + 8 * 10 / 2)
15 – ((4 * 5 + (–1) * 2) + 8 * 10 / 2)
15 – ((4 * 5 –1 * 2) + 8 * 10 / 2)
15 – ((20 – 1 * 2) + 8 * 10 / 2)
15 – ((20 – 2) + 8 * 10 / 2)
15 – (18 + 8 * 10 / 2)
15 – (18 + 80 / 2)
15 – (18 + 40)
15 – 58
–43

l. 2 – 5 * 12 / 2 + 8
2 – 60 / 2 + 8
2 – 30 + 8
–28 + 8
–20

m. (8 ^ 2 + 3 * 10 + 5) – (6 ^ 2 – 2 * 5 – 6)
(64 + 3 * 10 + 5) – (6 ^ 2 – 2 * 5 – 6)
(64 + 30 + 5) – (6 ^ 2 – 2 * 5 – 6)
(64 + 5) – (6 ^ 2 – 2 * 5 – 6)

99 – (6 ^ 2 – 2 * 5 – 6)
99 – (36 – 2 * 5 – 6)
99 – (36 – 10 – 6)
99 – (26 – 6)
99 – 20
79

n. (33 + 5 * 3 ^ 5 + 11 * 3 + 14) / (3 + 2)
(33 + 5 * 243 + 11 * 3 + 14) / (3 + 2)
(33 + 1215 + 11 * 3 + 14) / (3 + 2)
(33 + 1215 + 33 + 14) / (3 + 2)
(1248 + 33 + 14) / (3 + 2)
(1281 + 14) / (3 + 2)
1295 / (3 + 2)
1295 / 5
259

o. (2 * 5 * 3 * 4 ^ 2) / (48 / 2 ^ 2 – 20 + 2 ^ 2) / 10
(2 * 5 * 3 * 16) / (48 / 2 ^ 2 – 20 + 2 ^ 2) / 10
(10 * 3 * 16) / (48 / 2 ^ 2 – 20 + 2 ^ 2) / 10
(30 * 16) / (48 / 2 ^ 2 – 20 + 2 ^ 2) / 10
480 / (48 / 2 ^ 2 – 20 + 2 ^ 2) / 10
480 / (48 / 4 – 20 + 2 ^ 2) / 10
480 / (48 / 4 – 20 + 4) / 10
480 / (12 – 20 + 4) / 10
480 / (–8 + 4) / 10
480 / (–4) / 10
–120 / 10
–12

p. 4 ^ 2 + 3 ^ 2 * 2 ^ 2 + 5 ^ 3 – 4 ^ ((7 * 3 mod 9) mod 25)
4 ^ 2 + 3 ^ 2 * 2 ^ 2 + 5 ^ 3 – 4 ^ ((21 mod 9) mod 25)
4 ^ 2 + 3 ^ 2 * 2 ^ 2 + 5 ^ 3 – 4 ^ (3 mod 25)
4 ^ 2 + 3 ^ 2 * 2 ^ 2 + 5 ^ 3 – 4 ^ 3
16 + 3 ^ 2 * 2 ^ 2 + 5 ^ 3 – 4 ^ 3
16 + 9 * 2 ^ 2 + 5 ^ 3 – 4 ^ 3
16 + 9 * 4 + 5 ^ 3 – 4 ^ 3
16 + 9 * 4 + 125 – 4 ^ 3
16 + 9 * 4 + 125 – 64
16 + 36 + 125 – 64
52 + 125 – 64
177 – 64
113

q. (4 – (9 * 8 * 5 mod 3) * 3) / (100 mod 8 mod 3)
(4 – (72 * 5 mod 3) * 3) / (100 mod 8 mod 3)
(4 – (360 mod 3) * 3) / (100 mod 8 mod 3)
(4 – 0 * 3) / (100 mod 8 mod 3)
(4 – 0) / (100 mod 8 mod 3)
4 / (100 mod 8 mod 3)
4 / (4 mod 3)
4 / 1
4

r. 208 / 4 / (455 * 2 / 5 mod 4 + 50)
208 / 4 / (910 / 5 mod 4 + 50)
208 / 4 / (182 mod 4 + 50)
208 / 4 / (2 + 50)
208 / 4 / 52

52 / 52
1

s. (17 / 7 * 2) + (4 * 3 / 5 – 2) – (40 / 3 / 2 – 5)
(2.428 * 2) + (4 * 3 / 5 – 2) – (40 / 3 / 2 – 5)
4.85 + (4 * 3 / 5 – 2) – (40 / 3 / 2 – 5)
4.85 + (12 / 5 – 2) – (40 / 3 / 2 – 5)
4.85 + (2.4 – 2) – (40 / 3 / 2 – 5)
4.85 + 0.4 – (40 / 3 / 2 – 5)
4.85 + 0.4 – (13.33 / 2 – 5)
4.85 + 0.4 – (6.66 – 5)
4.85 + 0.4 – 1.66
5.25 – 1.66
3.59

t. 4 ^ 2 * 5 * 3 – 8 – 152 – (5 ^ 3 * 10 \ 4) mod 2 * 4
4 ^ 2 * 5 * 3 – 8 – 152 – (125 * 10 \ 4) mod 2 * 4
4 ^ 2 * 5 * 3 – 8 – 152 – (1250 \ 4) mod 2 * 4
4 ^ 2 * 5 * 3 – 8 – 152 – 312 mod 2 * 4
16 * 5 * 3 – 8 – 152 – 312 mod 2 * 4
80 * 3 – 8 – 152 – 312 mod 2 * 4
240 – 8 – 152 – 312 mod 2 * 4
240 – 8 – 152 – 312 mod 8
240 – 8 – 152 – 0
232 – 152 – 0
80 – 0
80

u. 8 * (4 – 3 ^ 2) ^ 2 – 9 * (3 ^ 4 – 4 ^ 3) ^ 3
8 * (4 – 9) ^ 2 – 9 * (3 ^ 4 – 4 ^ 3) ^ 3
8 * (–5) ^ 2 – 9 * (3 ^ 4 – 4 ^ 3) ^ 3
8 * 25 – 9 * (3 ^ 4 – 4 ^ 3) ^ 3
8 * 25 – 9 * (81 – 4 ^ 3) ^ 3
8 * 25 – 9 * (81 – 64) ^ 3
8 * 25 – 9 * 17 ^ 3
8 * 25 – 9 * 4913
200 – 9 * 4913
200 – 44217
–44017

v. (4 + 2 * 9 / 5) – (2 ^ 2 * 8) / 2 ^ 2
(4 + 18 / 5) – (2 ^ 2 * 8) / 2 ^ 2
(4 + 3.6) – (2 ^ 2 * 8) / 2 ^ 2
7.6 – (2 ^ 2 * 8) / 2 ^ 2
7.6 – (4 * 8) / 2 ^ 2
7.6 – 32 / 2 ^ 2
7.6 – 32 / 4
7.6 – 8
–0.4

w. 8 ^ 2 / 4 ^ 3 mod 10 ^ 2 / 5 + 100
64 / 4 ^ 3 mod 10 ^ 2 / 5 + 100
64 / 64 mod 10 ^ 2 / 5 + 100
64 / 64 mod 100 / 5 + 100
1 mod 100 / 5 + 100
1 mod 20 + 100
1 + 100
101

x. (80 / 8) ^ 2 – 4 / 2 + (20 / 4) ^ 3
10 ^ 2 – 4 / 2 + (20 / 4) ^ 3
10 ^ 2 – 4 / 2 + 5 ^ 3
100 – 4 / 2 + 5 ^ 3
100 – 4 / 2 + 125
100 – 2 + 125
98 + 125
223

y. 89 + (66 ^ 2 mod 2.011 + 400) ^ (2 mod 8) / 3 ^ (3 + 2)
89 + (4356 mod 2011 + 400) ^ (2 mod 8) / 3 ^ (3 + 2)
89 + (334 + 400) ^ (2 mod 8) / 3 ^ (3 + 2)
89 + 734 ^ (2 mod 8) / 3 ^ (3 + 2)
89 + 734 ^ 2 / 3 ^ (3 + 2)
89 + 734 ^ 2 / 3 ^ 5
89 + 538756 / 3 ^ 5
89 + 538756 / 243
89 + 2217.102
2306.102

z. 14 – (7 + 4 * 3 – (–2) ^ 2 * 2 – 6) + (2 ^ 2 + 6 – 5 * 3)
14 – (7 + 4 * 3 – 4 * 2 – 6) + (2 ^ 2 + 6 – 5 * 3)
14 – (7 + 12 – 4 * 2 – 6) + (2 ^ 2 + 6 – 5 * 3)
14 – (7 + 12 – 8 – 6) + (2 ^ 2 + 6 – 5 * 3)
14 – (19 – 8 – 6) + (2 ^ 2 + 6 – 5 * 3)
14 – (11 – 6) + (2 ^ 2 + 6 – 5 * 3)
14 – 5 + (2 ^ 2 + 6 – 5 * 3)
14 – 5 + (4 + 6 – 5 * 3)
14 – 5 + (4 + 6 – 15)
14 – 5 + (10 – 15)

14 – 5 + (–5)
14 – 5 – 5
9 – 5
4

A continuación, la solución de los ejercicios del numeral 1.4.4.

a. $\frac{2}{3} + \frac{2}{4} + \frac{3}{5}$ → 2/3 + 2/4 + 3/5

b. $y + \frac{x}{z+5}$ → y + x/(z + 5)

c. $\frac{x^2}{a^2} + \frac{y^2}{b^2}$ → x^2/a^2 + y^2/b^2

d. $\frac{\frac{x^2}{a^2}+y^2}{b^2}$ → (x^2/a^2 + y^2)/b^2

e. $\frac{x^2+y^2}{a^2+b^2}$ → (x^2 + y^2)/(a^2 + b^2)

f. $\frac{a^2\,(c^2-2m+y^2)}{2^2}$ → a^2 * (c^2- 2*m + y^2)/2^2

g. $\frac{\frac{x^2+3xy}{a^2+y^2}}{b^2}$ → ((x ^ 2 + 3*x*y)/(a^2 + y^2))/b^2

h. $\frac{a^2(a^2-c^2)}{z}$ → a ^2* (a^2 – c^2)/z

i. $\left(\frac{a^2+bx+c^3}{x}\right)^2$ → ((a ^2+b* x+ c^3)/x)^2

j. $\frac{b.h}{2} + \frac{y^2}{2} - x^2 \,.\, \frac{8x\,.4}{2}$ → b*h/2 + y^2/2 – x^2* (8*x*4/2)

k. $\left(\frac{a^2-2mn^2-n^3}{pqr}\right)^2$ → ((a ^2 – 2*m*n^2 – n^3)/(p*q*r))^2

l. $\frac{a}{b^2} + \left(\frac{a}{b}\right)^2 - \left(\frac{m^2n^3}{4x}\right)^3$ → a/b^2 + (a/b)^2 – ((m^2* n^3)/(4*x))^3

Convertir las siguientes expresiones algorítmicas a expresiones matemáticas:

m. 7 * (1 + y) → 7 (1+y)

n. a ^ 3 + b ^ 3 → $a^3 + b^3$

o. (x + y) / (u + w / a) → $\frac{x+y}{u+\frac{w}{a}}$

p. a ^ 3 + 3 * a ^ 2 * b + 3 * a * b ^ 2 + b ^ 3 → $a^3 + 3a^2 b + 3ab^2 + b^3$

q. (a + b) ^ 2 – (a – b ^ 2) → $(a+b)^2 – (a – b^2)$

r. x – y ^ 2 * (x – y) ^ (2 – n ^ 3) → $x – y^2 \ (x – y)^{2 – n^3}$

s. (a / b) + (c / a) + c → $\frac{a}{b} + \frac{c}{a} + c$

t. a / (b + c) / (a / b + c) → $\frac{\frac{a}{b+c}}{\frac{a}{b}+c}$

A continuación, la solución de los ejercicios del numeral 1.7.5.

a. ‘A’ → Alfabético – Carácter (Un solo carácter comillas sencillas)

b. –89 → Numérico – Entero (Número sin decimales)

c. “125” → Alfabético – Cadena (Varios caracteres comillas dobles)

d. “–9” → Alfabético – Cadena (Varios caracteres comillas dobles)

e. –5 → Numérico – Entero (Número sin decimales)

f. 0 → Numérico – Entero o Lógico (Número sin decimales o booleano)

g. 125.00 → Numérico – Real (Número con decimales)

h. 4.000 → Numérico – Entero (Número sin decimales)

i. '+' → Alfabético – Carácter (Un solo carácter comillas sencillas)

j. "9.12" → Alfabético – Cadena (Varios caracteres comillas dobles)

k. 325 → Numérico – Entero (Número sin decimales)

l. "Lógica" → Alfabético – Cadena (Varios caracteres comillas dobles)

m. –5698.2 → Numérico – Real (Número con decimales)

n. V → Lógico o booleano (Valor booleano *Verdadero)*

o. 2,333 → Numérico – Real (Número con decimales)

p. Falso → Lógico o booleano (Valor booleano *Falso)*

q. 0.0005 → Numérico – Real (Número con decimales)

r. 30000 → Numérico – Entero (Número sin decimales)

s. "Sí" → Alfabético – Cadena (Varios caracteres comillas dobles)

t. –500.00 → Numérico – Real (Número con decimales)

A continuación, la solución de los ejercicios del numeral 1.7.6

a. 3 < 6 → *Verdadero* (3 es mayor que 6).

b. 5 >= 10 → *Falso* (5 no es mayor o igual que 10).

c. 'A' > 'Z' → *Falso* (La letra 'A' no es mayor que la 'Z').

d. ‘8’ > ‘700’ → *Verdadero* (‘8’ es mayor que ‘7’. Cuando se comparan cadena de caracteres, se hace una comparación carácter por carácter, no la cadena completa).

e. 8 <> 8 → *Falso* (8 no es diferente de 8. Es igual).

f. 64 >= 64 → *Verdadero* (64 es igual a 64. Cuando se pone el operador >=, se evalúa si es mayor, si no se cumpla, pasa a evaluar si es igual).

g. ‘8’ > ‘7’ → *Verdadero* (‘8’ es mayor que ‘7’).

h. 45 mod 2 = 0 → *Falso* (45 mod 2 = 0 ➔ 1 = 0 ➔ *Falso*).

i. 2 ^ 4 <> 84 → *Verdadero* (2 ^ 4 <> 84 ➔ 16 <> 84 ➔ *Verdadero*).

j. 85.36 = 85.34 V 235 = 114 → *Falso* (85.36=85.34 V 235=114 ➔ *Falso* V *Falso* ➔ *Falso*.

k. ~ (‘A’ >= ‘Z’ V ‘8’ <= ‘0’) → *Verdadero* (~ (‘A’ >= ‘Z’ V ‘8’ <= ‘0’) ➔ ~ (*Falso* V *Falso*) ➔ *~ Falso* ➔ *Verdadero)*.

l. 12 <> 15 Λ 111<= 114 → *Verdadero (*12 <> 15 Λ 111<= 114 ➔ *Verdadero* Λ *Verdadero* ➔ *Verdadero)*

m. ~ (17 < 698 Λ ‘J’ <> ‘K’) → *Falso* (~ (17 < 698 Λ ‘J’ <> ‘K’) ➔ ~ (*Verdadero* Λ *Verdadero*) ➔ ~ (*Verdadero*) ➔ *Falso*).

n. 158 >= 158 Λ 256 <= 256 → *Verdadero* (158 >= 158 Λ 256 <= 256 ➔ *Verdadero* Λ *Verdadero* ➔ *Verdadero*).

o. 25 > 12 V 0 > 14 → *Verdadero* (25 > 12 V 0 > 14 ➔ *Verdadero* V *Falso* ➔ *Verdadero*).

p. p. 25 > 12 Λ 0 > 14 → *Falso* (25 > 12 Λ 0 > 14 ➔ *Verdadero* Λ *Falso* ➔ *Falso*).

q. q. 3 ^ 2 = 36 / 6 + 3 → *Verdadero* (3 ^ 2 = 36 / 6 + 3 ➔ 9 = 36 / 6 + 3 ➔ 9 = 6 + 3 ➔ 9 = 9 ➔ *Verdadero*).

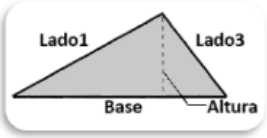
r. r. 5 * 4 + 2 > 45 mod 13 → *Verdadero* (5 * 4 + 2 > 45 mod 13 ➔ 20 + 2 > 45 mod 13 ➔ 20 + 2 > 6 ➔ 22 > 6 ➔ *Verdadero*).

A continuación, la solución de los ejercicios del numeral 1.7.10.

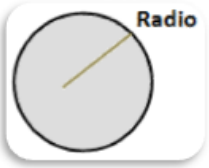
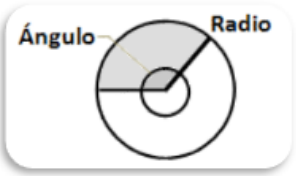
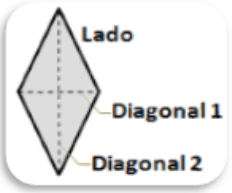
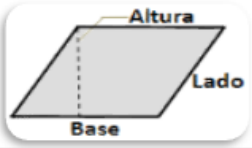
a. $Sueldo → Incorrecta (*$*)
b. A → Correcta
c. cElUlAr → Correcta
d. Dir_casa → Correcta
e. Edad → Correcta
f. Notas → Correcta
g. N–Tel → Incorrecta (–)
h. SB512 → Correcta
i. Tel → Correcta
j. c@rreo → Incorrecta (@)
k. Apellido paterno → Incorrecta ( )
l. Nombre → Correcta
m. 2Salarios → Incorrecta (*2*)
n. Pensión → Incorrecta (Ó)
o. N1 → Correcta
p. Primera/Nota → Incorrecta (*/*)
q. 1Nota → Incorrecta (*1*)
r. Año → Incorrecta (*Ñ*)

Anexos

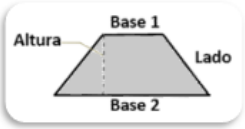
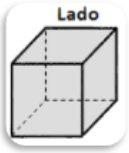
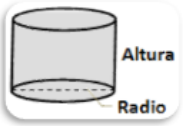
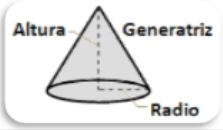
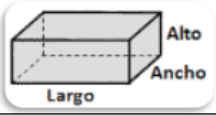
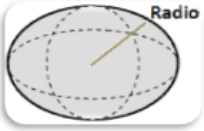
Anexo 1. Fórmulas generales de figuras geométricas.

Figura	Fórmula
Cuadrado 	$\textbf{Área} = Lado^2$ o $\textbf{Área} = Lado \,.\, Lado$ $\textbf{Perímetro} = 4 \,.\, Lado$ $\textbf{Diagonal} = Lado \,.\, \sqrt{2}$
Rectángulo 	$\textbf{Área} = Base \,.\, Altura$ $\textbf{Perímetro} = 2\,(Base + Altura)$ $\textbf{Diagonal} = \sqrt{Base^2 + Altura^2}$
Triángulo equilátero 	$\textbf{Área} = \frac{Lado \,.\, Altura}{2}$ o $\textbf{Área} = \frac{\sqrt{3}}{4} \,.\, Lado^2$ $\textbf{Perímetro} = 3 \,.\, Lado$ $\textbf{Altura} = \frac{\sqrt{3}}{2} \,.\, Lado$
Triángulo isósceles 	$\textbf{Área} = \frac{Base \,.\, Altura}{2}$ $\textbf{Perímetro} = 2 \,.\, Altura + Base$
Triángulo escaleno 	$\textbf{Área} = \frac{Base \,.\, Altura}{2}$ $\textbf{Perímetro} = Lado1 + Lado2 + Lado3$ *Nota*: $Base = Lado2$

Fórmulas de figuras geométricas 1.
Fuente: Spiegel (1991).

Figura	Fórmula
Círculo 	$\acute{A}rea = \pi \cdot Radio^2$ $Perímetro = 2 \cdot \pi \cdot Radio$
Arco de circunferencia 	$\acute{A}rea = \frac{\pi \cdot radio^2 \cdot Angulo}{360°}$ $Perímetro = \frac{2 \cdot \pi \cdot radio \cdot Angulo}{360°}$
Corona circular 	$\acute{A}rea = \pi \cdot (Radio1^2 - Radio2^2)$ $Perímetro = 2 \cdot \pi \cdot (Radio1 + Radio2)$ *Nota. Radio1 es el Radio de la circunferencia externa y Radio2 el de la circunferencia más interna.*
Rombo 	$\acute{A}rea = \frac{Diagonal1 \cdot Diagonal2}{2}$ $Perímetro = 4 \cdot Lado$ $4 \cdot Lado^2 = Diagonal1^2 + Diagonal2^2$
Romboide o paralelogramo 	$\acute{A}rea = Base \cdot Altura$ $Perímetro = 2\ (Lado + Base)$

Fórmulas de figuras geométricas 2.
Fuente: Spiegel (1991).

Figura	Fórmula
Trapecio 	$\text{Área} = \frac{(\text{Base1} \cdot \text{Base2})}{2} \cdot \text{Altura}$ $\text{Perímetro} = Lado + Base1 + Base2 + \text{Altura}$
Cubo 	$\text{Área} = 6 . Lado^2$ $\text{Volumen} = Lado^3$
Cilindro 	$\text{Área} = 2 . \pi . Radio\ (Altura + Base)$ $\text{Volumen} = \pi . Radio^2 . Altura$
Cono 	$\text{Área} = \pi . Radio\ (Generatriz + Radio)$ $\text{Volumen} = \frac{\pi . \text{Radio}^2 . \text{Altura}}{3}$
Ortoedro 	$\text{Área} = 2\ (Largo . Ancho + Largo . Alto + Ancho . Alto)$ $\text{Volumen} = Largo . Ancho . Alto$
Esfera 	$\text{Área} = 4 . \pi . Radio^2$ $\text{Volumen} = \frac{4}{3} . \pi . \text{Radio}^3$
Polígono regular 	$\text{Área} = \frac{5 . Lado . Apotema}{2}$ o $\text{Área} = \frac{5}{8} . Radio^2$ $\text{Perímetro} = 5 . Lado$

Fórmulas de figuras geométricas 3.
Fuente: Spiegel (1991).

Anexo 2. Fórmulas de conversión de temperaturas.

Equivalencias → ↑	Centígrados	Fahrenheit	Kevin
Centígrados (°C)	1	1,8 °C + 32	°C + 273,15
Fahrenheit (°F)	(°F – 32) / 1,8	1	(°F + 459,67) / 1,8
Kelvin (K)	K – 273,15	1.8K – 459,67	1
Rankine (Ra)	1,8 (Ra – 491,67)	Ra – 459,67	1,8 (Ra – 491.67) + 273.15
Réaumur (°Re)	1,25 °Re	2,25 °Re + 32	1,25 °Re + 273.15

Equivalencias → ↑	Rankine	Réaumur
Centígrados (°C)	1,8 °C + 491,67	0,8 °C
Fahrenheit (°F)	°F + 459,67	4 / 9 (°F - 32)
Kelvin (K)	1.8 (K – 273.15) + 491,67	0.8° (K – 273.15)
Rankine (Ra)	1	4 / 9 (Ra – 491.67)
Réaumur (°Re)	2,25 °Re + 491.67	1

Importante:

- La temperatura Rankine no es muy usada actualmente.
- La escala de temperatura Kelvin no tiene el símbolo grados (°), debido a que es una escala de temperatura termodinámica (absoluta).
- Punto de ebullición agua: 100 º C (212 ° F).
- Punto de congelación agua: 0° C (32° F).
- Temperatura corporal promedio del cuerpo humano: 37° C (98.6° F).
- Temperatura ambiente confortable: de 20° C a 25° C (de 68° Fa 77° F).

Anexo 3. Equivalencias de longitud.

Equivalencias → ↑	Kilómetro (Km)	Hectómetro (Hm)	Decámetro (Dm)	Metro (m)	Decímetro (dm)	Centímetro (cm)	Milímetro (mm)
Kilómetro (Km)	1	10	100	1.000	10^4	10^5	10^6
Hectómetro (Hm)	0,1	1	10	100	1.000	10^4	10^5
Decámetro (Dm)	0,01	0,1	1	10	100	1.000	10^4
Metro (m)	0,001	0,01	0,1	1	10	100	1.000
Decímetro (dm)	10^{-4}	0,001	0,01	0,1	1	10	100
Centímetro (cm)	10^{-5}	10^{-4}	0,001	0,01	0,1	1	10
Milímetro (mm)	10^{-6}	10^{-5}	10^{-4}	0,001	0,01	0,1	1
Miriámetro	10	100	1.000	10^4	10^5	10^6	10^7
Legua Americana (Lea)	4,828	48,28	482,8	4.828,03	48.280,3	482.803,2	4.828.03
Milla (mi)	1,609	16,09	160,93	1.609,34	16.093,4	160.934,4	1.609.34
Braza Inglesa	$1{,}829\times10^{-3}$	$1{,}829\times10^{-2}$	0,1829	1,829	18,29	182,9	1.829
Braza Española	$1{,}672\times10^{-3}$	$1{,}672\times10^{-2}$	0,1672	1,672	16,72	167,2	1.672
Yarda (yd)	$9{,}14\times10^{-4}$	$9{,}14\times10^{-3}$	$9{,}14\times10^{-2}$	0,9144	9,144	91,44	914,4
Pie (ft – pie)	$3{,}048\times10^{-4}$	$3{,}048\times10^{-3}$	$3{,}048\times10^{-2}$	0,3048	3,048	30,48	300,48
Pulgada (in – pulg)	$2{,}54\times10^{-5}$	$2{,}54\times10^{-4}$	$2{,}54\times10^{-3}$	0,0254	0,254	2,54	25,4

Equivalencias → ↑	Legua Americana	Milla Americana	Braza Inglesa	Yarda (yd)	Pie (ft – pie)	Pulgada (in – pulg)
Kilómetro (Km)	0,207	0,621	546,81	1.093,6	3.280,84	39.370,08
Metro (m)	$2{,}07\times10^{-4}$	$6{,}21\times10^{-4}$	0,54681	1,0936	3,28	39,37
Centímetro (cm)	$2{,}07\times10^{-6}$	$6{,}21\times10^{-6}$	$5{,}468\times10^{-3}$	$1{,}09\times10^{-2}$	0,0328	0,3937
Milímetro (mm)	$2{,}07\times10^{-7}$	$6{,}21\times10^{-7}$	$5{,}468\times10^{-4}$	$1{,}09\times10^{-3}$	$3{,}28\times10^{-3}$	$3{,}937\times10^{-2}$
Miriámetro	2,07	6,21	5.468,1	10.936,1	32.808,4	393.700,8
Legua Americana (Lea)	1	3	2.640	5.280	15.840	190.080
Milla Americana (mi)	0,333	1	880	1.760	5.280	63.360
Braza Inglesa	$3{,}79\times10^{-4}$	$1{,}136\times10^{-3}$	1	2	6	72
Yarda (yd)	$1{,}89\times10^{-4}$	$5{,}68\times10^{-4}$	0,5	1	3	36
Pie (ft – pie)	$6{,}31\times10^{-5}$	$1{,}89\times10^{-4}$	0,167	0,333	1	12
Pulgada (in – pulg)	$5{,}26\times10^{-6}$	$1{,}578\times10^{-5}$	0,0139	0,028	0,083	1

Anexo 4. Equivalencias de masa.

Equivalencias → ↑	TM (t)	TAL (t)	TAC (t)	Kilogramo	Gramo	Stone (st)
Tonelada Métrica (t)	1	0,984	1,102	1.000	1.000.000	157,473
Tonelada America Larga (t)	1,016	1	1,12	1.016,046	1.016.046,94	160
Tonelada Americana Corta (t)	0,907	0,892	1	907,185	907.185	142,857
Kilogramo	0,001	$9{,}842\times10^{-4}$	$1{,}102\times10^{-3}$	1	1.000	0,157
Gramo (g)	10^{-6}	$9{,}842\times10^{-7}$	$1{,}1\times10^{-6}$	10^{-2}	1	$1{,}577\times10^{-4}$
Stone (st)	$6{,}35\times10^{-3}$	$6{,}25\times10^{-3}$	7×10^{-3}	6,35	6.350,3	1
Libra Americana	$4{,}536\times10^{-4}$	$4{,}464\times10^{-4}$	0,0005	0,454	453,593	$7{,}14\times10^{-2}$
Libra Inglesa	$3{,}732\times10^{-4}$	$3{,}674\times10^{-4}$	$4{,}143\times10^{-4}$	0,373	373,24	$5{,}88\times10^{-2}$
Quintal Americano Largo	$5{,}08\times10^{-2}$	0,05	56	50,802	50.802,37	8
Quintal Americano Corto	$4{,}536\times10^{-2}$	$4{,}464\times10^{-2}$	0,05	45,36	45.359,27	7,143
Onza Americana	$2{,}835\times10^{-5}$	$2{,}79\times10^{-5}$	$3{,}125\times10^{-5}$	2,835	28,35	$4{,}464\times10^{-3}$
Onza Inglesa	$3{,}11\times10^{-5}$	$3{,}061\times10^{-5}$	$3{,}429\times10^{-5}$	$3{,}11\times10^{-2}$	31,103	$4{,}898\times10^{-3}$

Equivalencias → ↑	Libra Americana	Libra Inglesa	Quintal Largo	Quintal Corto	Onza Americana	Onza Inglesa
Tonelada Métrica (t)	2.204,62	2.679,23	19,684	22,046	35.273,94	32.150,74
Tonelada America Larga (t)	2.240	2.722,22	20	22,4	35.840	32.666,67
Tonelada Americana Corta (t)	2.000	2.430,56	17,857	20	32	29.166,67
Kilogramo	2,205	2,68	$1{,}968\times10^{-2}$	$2{,}205\times10^{-2}$	35,27394	32,15
Gramo (g)	$2{,}205\times10^{-3}$	$2{,}68\times10^{-3}$	2×10^{-5}	$2{,}205\times10^{-5}$	$3{,}527\times10^{-2}$	$0{,}321\times10^{-2}$
Libra Americana	1	0,823	$8{,}93\times10^{-3}$	0,01	16	14,58
Libra Inglesa	0,823	1	$7{,}347\times10^{-3}$	$8{,}229\times10^{-3}$	13,166	12
Quintal Americano Largo	112	136,11	1	1,12	1.792	1.633,33
Quintal Americano Corto	100	121,53	0,893	1	1.600	1.458
Onza Americana	0,0625	0,076	$5{,}58\times10^{-4}$	$6{,}25\times10^{-4}$	-	0,912
Onza Inglesa	0,0686	0,083	$6{,}122\times10^{-4}$	$6{,}857\times10^{-4}$	1,097	-
Stone (st)	14	17,014	0,125	0,14	224	204,167

Anexo 5. Equivalencias de volumen.

Equivalencias → ↑	Kilolitro (kL ⇔ m^3)	Hectolitro (hL)	Decalitro (DL)	Litro (L ⇔ dm^3)
Kilolitro (kL)	1	10	100	1.000
Hectolitro (hL)	0,1	1	10	100
Decalitro (DL)	0,01	0,1	1	10
Litro (L)	0,001	0,01	0,1	1
Decilitro (dl)	10^{-4}	0,001	0,01	0,1
Centilitro (cl)	10^{-5}	10^{-4}	0,001	0,01
Mililitro (ml)	10^{-6}	10^{-5}	10^{-4}	0,001
Microlitro (µl)	10^{-9}	10^{-8}	10^{-7}	0^{-6}
Barril Inglés	0,164	1,636	16,365	163,65
Barril Americano	0,1589	1,589	15,89	159
Galón Inglés (gal)	$4{,}546\times10^{-3}$	$4{,}546\times10^{-2}$	0,4546	4,546
Galón Americano (gal)	$3{,}785\times10^{-3}$	$3{,}785\times10^{-2}$	0,3785	3,785
Cuarto Galón Inglés (qt)	$1{,}137\times10^{-3}$	$1{,}137\times10^{-2}$	0,1137	1,137
Cuarto Galón Americano (qt)	$9{,}46\times10^{-4}$	$9{,}46\times10^{-3}$	$9{,}46\times10^{-2}$	0,946
Pinta Inglesa	$5{,}68\times10^{-4}$	$5{,}68\times10^{-3}$	$5{,}68\times10^{-2}$	0,568
Pinta Americana	$4{,}73\times10^{-4}$	$4{,}73\times10^{-3}$	$4{,}73\times10^{-2}$	0,473
Pie Cúbico Americano	$2{,}83\times10^{-2}$	0,283	2,83	28,317
Onza Liquida Inglesa	$2{,}8\times10^{-5}$	$2{,}8\times10^{-4}$	$2{,}8\times10^{-3}$	$2{,}8\times10^{-2}$
Onza Liquida Americana	3×10^{-5}	3×10^{-4}	3×10^{-3}	3×10^{-2}
Pulgada Cúbica ($Pulg^3$)	164×10^{-5}	$1{,}64\times10^{-4}$	$1{,}64\times10^{-3}$	$1{,}64\times10^{-2}$

Equivalencias → ↑	Decilitro (dl)	Centilitro (cl)	Mililitro (ml⇔ cm^3)	Microlitro (µl ⇔ mm^3)
Kilolitro (kL)	10^4	10^5	10^6	10^9
Hectolitro (hL)	1.000	10^4	10^5	10^8
Decalitro (DL)	100	1.000	10^4	10^7
Litro (L)	10	100	1.000	10^6
Decilitro (dl)	1	10	100	100.000
Centilitro (cl)	0,1	1	10	10.000
Mililitro (ml)	0,01	0,1	1	1.000
Microlitro (µl)	10^{-5}	10^{-4}	10^{-3}	1
Barril Inglés	1.636,5	16.365	163.650	163.650
Barril Americano	1.589,87	15.898,72	158.987,24	158.987.239
Galón Inglés (gal)	45,46	454,61	4.546,01	4.546.099,26
Galón Americano (gal)	37,85	378,541	3.785,41	3.785.409,67
Cuarto Galón Inglés (qt)	11,365	113,653	1.136,52	1.136.524,95
Cuarto Galón Americano (qt)	9,464	94,635	946,353	946.353,134
Pinta Inglesa	5,683	56,826	568,262	568.262,214
Pinta Americana	4,732	47,318	473,177	473.176,57
Pie Cúbico Americano	283,168	2.831,68	28.316,84	28.316.835,9
Onza Liquida Inglesa	0,284	2,841	28,413	28.413,12
Onza Liquida Americana	0,3	2,96	29,574	29.573,532
Pulgada Cúbica ($Pulg^3$)	0,163	1,639	16,387	16.387,06

Equivalencias de volumen 1.

Equivalencias → ↑	Barril Inglés	Barril Americano	Galón Inglés (gal)	Galón Americano (gal)	Cuarto Galón Inglés (qt)	Cuarto Galón Americano (qt)
Kilolitro (kL)	6,11	6,29	220	264,17	888	1.056,7
Hectolitro (hL)	0,611	0,629	22	26,417	88	105,6
Decalitro (DL)	$6{,}11\times10^{-2}$	$6{,}29\times10^{-2}$	2,2	2,6417	8,8	10,6
Litro (L)	$6{,}11\times10^{-3}$	$6{,}29\times10^{-3}$	0,22	0,26417	0,88	1,06
Decilitro (dl)	$6{,}11\times10^{-4}$	$6{,}29\times10^{-4}$	$2{,}2\times10^{-2}$	$2{,}642\times10^{-2}$	$8{,}8\times10^{-2}$	0,106
Centilitro (cl)	$6{,}11\times10^{-5}$	$6{,}29\times10^{-5}$	$2{,}2\times10^{-3}$	$2{,}642\times10^{-3}$	$8{,}8\times10^{-3}$	$1{,}06\times10^{-2}$
Mililitro (ml)	$6{,}11\times10^{-6}$	$6{,}29\times10^{-6}$	$2{,}2\times10^{-4}$	$2{,}642\times10^{-4}$	$8{,}8\times10^{-4}$	$1{,}06\times10^{-3}$
Microlitro (µl)	$6{,}11\times10^{-9}$	$6{,}29\times10^{-9}$	$2{,}2\times10^{-7}$	$2{,}642\times10^{-7}$	$8{,}8\times10^{-9}$	$1{,}06\times10^{-6}$
Barril Inglés	1	1,029	36	43,23	143,99	172,927
Barril Americano	0,972	1	34,97	42	139,89	168
Galón Inglés (gal)	$2{,}8\times10^{-2}$	0,029	1	1,2	4	4,80
Galón Americano (gal)	$2{,}3\times10^{-2}$	0,0238	0,833	1	3,33	4
Cuarto Galón Inglés (qt)	7×10^{-3}	$7{,}2\times10^{-3}$	0,25	0,3	1	1,2
Cuarto Galón Americano (qt)	6×10^{-3}	6×10^{-3}	0,208	0,25	0,833	1
Pinta Inglesa	$3{,}5\times10^{-3}$	4×10^{-3}	0,125	0,15	0,5	0,6
Pinta Americana	$2{,}9\times10^{-3}$	3×10^{-3}	0,104	0,125	0,416	0,5
Pie Cúbico Americano	0,173	0,178	6,229	7,48	24,915	29,92
Onza Liquida Inglesa	$1{,}73\times10^{-4}$	$1{,}78\times10^{-4}$	$6{,}25\times10^{-3}$	$7{,}51\times10^{-3}$	$2{,}5\times10^{-2}$	3×10^{-2}
Onza Liquida Americana	$1{,}8\times10^{-4}$	$1{,}9\times10^{-4}$	$6{,}5\times10^{-3}$	$7{,}8\times10^{-3}$	$2{,}6\times10^{-2}$	$3{,}13\times10^{-2}$
Pulgada Cúbica ($Pulg^3$)	$1{,}13\times10^{-4}$	10^{-4}	$3{,}6\times10^{-3}$	$4{,}3\times10^{-3}$	$1{,}44\times10^{-2}$	$1{,}7\times10^{-2}$

Equivalencias → ↑	Pinta Inglesa	Pinta Americana	Pie Cúbico Americano	Onza Liquida Inglesa	Onza Liquida Americana	Pulgada Cúbica ($Pulg^3$)
Kilolitro (kL)	1.759,75	2.113,4	35,31	35.195,01	33.814,02	61.023,76
Hectolitro (hL)	176	211,34	3,531	3.519,5	3.381,4	6.102,376
Decalitro (DL)	17,6	21,16	0,3531	352	338,14	610,2
Litro (L)	1,76	2,11	$3{,}531\times10^{-2}$	35,2	33,8	61,02
Decilitro (dl)	0,176	0,211	$3{,}531\times10^{-3}$	3,52	3,38	6,102
Centilitro (cl)	$1{,}76\times10^{-2}$	$2{,}11\times10^{-2}$	$3{,}531\times10^{-4}$	0,352	0,338	0,6102
Mililitro (ml)	$1{,}76\times10^{-3}$	$2{,}11\times10^{-3}$	$3{,}531\times10^{-5}$	$3{,}52\times10^{-2}$	$3{,}38\times10^{-2}$	$6{,}102\times10^{-2}$
Microlitro (µl)	$1{,}76\times10^{-6}$	$2{,}11\times10^{-6}$	$3{,}51\times10^{-8}$	$3{,}52\times10^{-5}$	338×10^{-5}	$6{,}102\times10^{-5}$
Barril Inglés	287,98	345,85	5,780	5.759,66	5.533,66	9.986,54
Barril Americano	279,78	336	5,615	5.595,56	5.376	9.702
Galón Inglés (gal)	8	9,61	0,161	160	153,722	277,42
Galón Americano (gal)	6,661	8	0,134	133,23	128	231
Cuarto Galón Inglés (qt)	2	2,4	4×10^{-2}	40	38,43	69,355
Cuarto Galón Americano (qt)	1,67	2	$3{,}3\times10^{-2}$	33,3	32	57,75
Pinta Inglesa	1	1,2	2×10^{-2}	20	19,215	34,677
Pinta Americana	0,832	1	$1{,}67\times10^{-2}$	16,653	16	28,875
Pie Cúbico Americano	49,83	59,844	1	996,611	957,50	1.728
Onza Liquida Inglesa	5×10^{-2}	6×10^{-2}	$1{,}003\times10^{-3}$	1	0,96	1,734
Onza Liquida Americana	$5{,}2\times10^{-2}$	$6{,}25\times10^{-2}$	$1{,}04\times10^{-3}$	1,04	1	1,805
Pulgada Cúbica (Pulg3)	$2{,}9\times10^{-2}$	$3{,}5\times10^{-2}$	$5{,}8\times10^{-4}$	0,577	0,554	1

Equivalencias de volumen 2.

Anexo 6. Equivalencias de tiempo.

Equivalencias → ↑	**Milenio**	**Siglo**	**Decada**	**Año**	**Mes**	**Semana**	**Día**
Milenio	1	10	100	1.000	12.000	52.180	365.250
Siglo	0,1	1	10	100	1.200	5214,29	36.525
Decada	0,01	0,1	1	10	120	521,8	3.652,5
Año	0,001	0,01	0,1	1	12	52,18	365,25
Mes	$8{,}33\times10^{-5}$	$8{,}33\times10^{-4}$	$8{,}33\times10^{-3}$	0,083	1	4,348	30,4375
Semana	$1{,}918\times10^{-5}$	$1{,}918\times10^{-4}$	$1{,}918\times10^{-3}$	0,019	0,23	1	7
Día	$2{,}74\times10^{-6}$	$2{,}74\times10^{-5}$	$2{,}74\times10^{-4}$	$2{,}74\times10^{-3}$	0,0328	0,1428	1
Hora	$1{,}14\times10^{-7}$	$1{,}14\times10^{-6}$	$1{,}14\times10^{-5}$	$1{,}14\times10^{-4}$	$1{,}37\times10^{-3}$	0,0059	0,0417
Minuto	$1{,}903\times10^{-9}$	$1{,}903\times10^{-8}$	$1{,}903\times10^{-7}$	$1{,}903\times10^{-6}$	$2{,}28\times10^{-5}$	$9{,}92\times10^{-5}$	$6{,}94\times10^{-4}$
Segundo (s)	$3{,}17\times10^{-11}$	$3{,}17\times10^{-10}$	$3{,}17\times10^{-9}$	$3{,}17\times10^{-8}$	$3{,}8\times10^{-7}$	$1{,}65\times10^{-6}$	$1{,}16\times10^{-5}$
Milisegundo (ms)	$3{,}17\times10^{-14}$	$3{,}17\times10^{-13}$	$3{,}17\times10^{-12}$	$3{,}17\times10^{-11}$	$3{,}8\times10^{-10}$	$1{,}65\times10^{-9}$	$1{,}16\times10^{-8}$
Microsegundo (µs)	$3{,}17\times10^{-17}$	$3{,}17\times10^{-16}$	$3{,}17\times10^{-15}$	$3{,}17\times10^{-14}$	$3{,}8\times10^{-13}$	$1{,}65\times10^{-12}$	$1{,}16\times10^{-11}$
Nanosegundo (ns)	$3{,}17\times10^{-20}$	$3{,}17\times10^{-19}$	$3{,}17\times10^{-18}$	$3{,}17\times10^{-17}$	$3{,}8\times10^{-16}$	$1{,}65\times10^{-15}$	$1{,}16\times10^{-14}$

Equivalencias → ↑	**Hora**	**Minuto**	**Segundo (s)**	**Milisegundo (ms)**	**Microsegundo (µs)**	**Nanosegundo (ns)**
Milenio	$8{,}76\times10^{6}$	$5{,}256\times10^{8}$	$3{,}154\times10^{10}$	$3{,}154\times10^{13}$	$3{,}154\times10^{16}$	$3{,}154\times10^{19}$
Siglo	876.600	$5{,}256\times10^{7}$	$3{,}154\times10^{9}$	$3{,}154\times10^{12}$	$3{,}154\times10^{15}$	$3{,}154\times10^{18}$
Decada	87.660	5,256×106	$3{,}154\times10^{10}$	$3{,}154\times10^{11}$	$3{,}154\times10^{14}$	$3{,}154\times10^{17}$
Año	8.766	525.960	31.557.600	$3{,}154\times10^{10}$	$3{,}154\times10^{13}$	$3{,}154\times10^{16}$
Mes	730,5	43.830	2.629.800	$2{,}63\times10^{9}$	$2{,}63\times10^{12}$	$2{,}63\times10^{15}$
Semana	168	10.080	604.800,02	$6{,}05\times10^{8}$	$6{,}05\times10^{11}$	$6{,}05\times10^{14}$
Día	24	1.440	86.400	$8{,}64\times10^{7}$	$8{,}64\times10^{10}$	$8{,}64\times10^{13}$
Hora	1	60	3.600	$3{,}6\times10^{6}$	$3{,}6\times10^{9}$	$3{,}6\times10^{12}$
Minuto	0,0167	1	60	60.000	6×10^{7}	6×10^{10}
Segundo (s)	$2{,}78\times10^{-4}$	0,0167	1	1.000	10^{6}	10^{9}
Milisegundo (ms)	$2{,}78\times10^{-7}$	$1{,}67\times10^{-5}$	0,001	1	1.000	10^{6}
Microsegundo (µs)	$2{,}78\times10^{-10}$	$1{,}67\times10^{-8}$	10^{-6}	10^{-3}	1	1.000
Nanosegundo (ns)	$2{,}78\times10^{-13}$	$1{,}67\times10^{-11}$	10^{-9}	10^{-6}	10^{-3}	1

Anexo 7. Equivalencias de superficie.

Equivalencias → ↑	Kilometro Cuadrado (km^2)	Hectárea (ha)	Área (a)	Metro Cuadrado (m^2)	Centimetro Cuadrado (cm^2)	Milimetros Cuadrado (mm^2)	Micrometro Cuadrado (μm^2)
Kilometro Cuadrado (km^2)	1	100	10.000	1.000.000	10^{10}	10^{12}	10^{18}
Hectárea (ha)	0,01	1	100	10.000	10^{8}	10^{10}	10^{16}
Área (a)	10^{-4}	0,01	1	100	10^{6}	10^{8}	10^{14}
Metro Cuadrado (m^2)	10^{-6}	10^{-4}	0,01	1	10.000	10^{6}	10^{12}
Decimetro Cuadrado (dm^2)	10^{-8}	10^{-6}	10^{-4}	0,01	100	10^{4}	10^{10}
Milimetros Cuadrado (mm^2)	10^{-13}	10^{-10}	10^{-8}	10^{-6}	0,01	1	10^{6}
Micrometro Cuadrado (μm^2)	10^{-16}	10^{-16}	10^{-15}	10^{-13}	10^{-8}	10^{-6}	1
Milla Cuadrada	2,59	259	25.900	$2,59\times10^{6}$	$2,59\times10^{10}$	$2,59\times10^{12}$	$2,59\times10^{18}$
Acre	$4,05\times10^{-3}$	0,405	40,47	4.046,85	$4,05\times10^{7}$	$4,05\times10^{9}$	$4,05\times10^{15}$
Yarda Cuadrada (yr^2)	$8,36\times10^{-7}$	$8,36\times10^{-5}$	$8,36\times10^{3}$	0,836	$8,36\times10^{3}$	$8,36\times10^{5}$	$8,36\times10^{11}$
Pie Cuadrado (ft^2)	$9,29\times10^{-8}$	$9,29\times10^{-6}$	$9,29\times10^{-4}$	0,093	$9,29\times10^{2}$	$9,29\times10^{4}$	$9,29\times10^{10}$
Pulgada Cuadrada (in^2)	$6,45\times10^{-10}$	$6,45\times10^{-8}$	$6,45\times10^{-6}$	$6,45\times10^{-4}$	6,4516	645,16	$6,45\times10^{8}$

Equivalencias → ↑	Milla Cuadrada	Acre	Yarda Cuadrada (yr^2)	Pie Cuadrado (ft^2)	Pulgada Cuadrada (in^2)
Kilometro Cuadrado (km^2)	0,386	247,11	1.195.990	10.763.910	$1,55\times10^{9}$
Hectárea (ha)	$3,86\times10^{-3}$	2,471	11.960	107.639,1	$1,55\times10^{7}$
Área (a)	$3,86\times10^{-5}$	0,0247	119,6	1.076,39	$1,55\times10^{6}$
Metro Cuadrado (m^2)	$3,86\times10^{-7}$	$2,47\times10^{-4}$	1,196	10,76	$1,55\times10^{3}$
Decimetro Cuadrado (dm^2)	$3,86\times10^{-9}$	$2,47\times10^{-6}$	0,0119	0,1076	15,5
Milimetros Cuadrado (mm^2)	$3,86\times10^{-13}$	$2,47\times10^{-10}$	$1,2\times10^{-6}$	$1,08\times10^{-5}$	$1,55\times10^{-3}$
Micrometro Cuadrado (μm^2)	$3,86\times10^{-19}$	$2,47\times10^{-16}$	$1,2\times10^{-12}$	$1,08\times10^{-11}$	$1,55\times10^{-9}$
Milla Cuadrada	1	640	3.097.600	27.878.396	$4,01\times10^{9}$
Acre	$1,56\times10^{-3}$	1	4.840	43.560	6.272.636,6
Yarda Cuadrada (yr^2)	$3,23\times10^{-7}$	$2,07\times10^{-4}$	1	9	1.296
Pie Cuadrado (ft^2)	$3,59\times10^{-8}$	$2,3\times10^{-5}$	0,111	1	144
Pulgada Cuadrada (in^2)	$2,49\times10^{-10}$	$1,59\times10^{-7}$	$7,72\times10^{-4}$	0,0069	1

Anexo 8. Equivalencias de computación.

Equivalencias → ↑	Bit (b)	Kilobit (Kb)	Megabit (Mb)	Gigabit (Gb)	Terabit (Tb)	Petabit (Pb)	Exabit (Eb)
Bit (b)	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$	$9{,}09\times10^{-13}$	$8{,}88\times10^{-16}$	$8{,}67\times10^{-19}$
Kilobit (Kb)	1.024	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$	$9{,}09\times10^{-13}$	$8{,}88\times10^{-16}$
Megabit (Mb)	1.048.576	1.024	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$	$9{,}09\times10^{-13}$
Gigabit (Gb)	1.073.741.824	1.048.576	1.024	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$
Terabit (Tb)	$1{,}1\times10^{12}$	$1{,}07\times10^{9}$	1.048.575,94	1.024	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$
Petabit (Pb)	$1{,}13\times10^{15}$	$1{,}1\times10^{12}$	$1{,}07\times10^{9}$	1.048.575,94	1.024	1	$9{,}77\times10^{-4}$
Exabit (Eb)	$1{,}15\times10^{18}$	$1{,}13\times10^{15}$	$1{,}1\times10^{12}$	$1{,}07\times10^{9}$	1.048.576,01	1.024	1
Byte (B)	8	0,0078	$7{,}63\times10^{-6}$	$7{,}45\times10^{-9}$	$7{,}28\times10^{-12}$	$7{,}11\times10^{-15}$	$6{,}94\times10^{-18}$
Kilobyte (KB)	8.192	8	0,0078	$7{,}63\times10^{-6}$	$7{,}45\times10^{-9}$	$7{,}28\times10^{-12}$	$7{,}11\times10^{-15}$
Megabyte (MB)	8.388.608	8.192	8	0,0078	$7{,}63\times10^{-6}$	$7{,}45\times10^{-9}$	$7{,}28\times10^{-12}$
Gigabyte (GB)	8.589.934.592	8.388.608	8.192	8	0,0078	$7{,}63\times10^{-6}$	$7{,}45\times10^{-9}$
Terabyte (TB)	$8{,}8\times10^{12}$	$8{,}59\times10^{9}$	8.388.607,26	8.192	8	0,0078	$7{,}63\times10^{-6}$
Petabyte (PB)	$9{,}01\times10^{15}$	$8{,}8\times10^{12}$	$8{,}59\times10^{9}$	8.388.606,87	8.192	8	0,0078
Exabyte (EB)	$9{,}22\times10^{18}$	$9{,}01\times10^{15}$	$8{,}8\times10^{12}$	$8{,}59\times10^{9}$	8.388.607,61	8.192	8

Equivalencias → ↑	Byte (B)	Kilobyte (KB)	Megabyte (MB)	Gigabyte (GB)	Terabyte (TB)	Petabyte (PB)	Exabyte (EB)
Bit (b)	0,125	$1{,}22\times10^{-4}$	$1{,}19\times10^{-7}$	$1{,}16\times10^{-10}$	$1{,}14\times10^{-13}$	$1{,}11\times10^{-16}$	$1{,}08\times10^{-19}$
Kilobit (Kb)	128	0,125	$1{,}22\times10^{-4}$	$1{,}19\times10^{-7}$	$1{,}16\times10^{-10}$	$1{,}14\times10^{-13}$	$1{,}11\times10^{-16}$
Megabit (Mb)	131.072	128	0,125	$1{,}22\times10^{-4}$	$1{,}19\times10^{-7}$	$1{,}16\times10^{-10}$	$1{,}14\times10^{-13}$
Gigabit (Gb)	134.217.728	131.072	128	0,125	$1{,}22\times10^{-4}$	$1{,}19\times10^{-7}$	$1{,}16\times10^{-10}$
Terabit (Tb)	$1{,}37\times10^{11}$	134.217.719,75	131.071,99	128	0,125	$1{,}22\times10^{-4}$	$1{,}19\times10^{-7}$
Petabit (Pb)	$1{,}41\times10^{14}$	1,37×1011	134.217.720,75	131.071,99	128	0,125	$1{,}22\times10^{-4}$
Exabit (Eb)	$1{,}44\times10^{17}$	1,41×1014	$1{,}37\times10^{11}$	134.217.720,88	131.072	128	0,125
Byte (B)	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$	$9{,}09\times10^{-13}$	$8{,}88\times10^{-16}$	$8{,}67\times10^{-19}$
Kilobyte (KB)	1.024	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$	$9{,}09\times10^{-13}$	$8{,}88\times10^{-16}$
Megabyte (MB)	1.048.576	1.024	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$	$9{,}09\times10^{-13}$
Gigabyte (GB)	1.073.741.824	1.048.576	1.024	1	$9{,}77\times10^{-4}$	$9{,}54\times10^{-7}$	$9{,}31\times10^{-10}$
Terabyte (TB)	$1{,}1\times10^{12}$	$1{,}07\times10^{9}$	1.048.575,91	1.024	1	$9{,}77\times10^{-4}$	$9{,}537\times10^{-7}$
Petabyte (PB)	$1{,}13\times10^{15}$	$1{,}1\times10^{12}$	1,07×109	1.048.575,86	1.024	1	9,77×10-4
Exabyte (EB)	$1{,}15\times10^{18}$	$1{,}13\times10^{15}$	$1{,}1\times10^{12}$	$1{,}07\times10^{9}$	1.048.575,98	1.024	1

Anexo 9. Tabla de caracteres ASCII.

Caracteres ASCII de control			Caracteres ASCII imprimibles									ASCII extendido (Página de código 437)											
00	NULL	(carácter nulo)	32	espacio	64	@	96	`				128	Ç	160	á	192	└	224	Ó				
01	SOH	(inicio encabezado)	33	!	65	A	97	a				129	ü	161	í	193	┴	225	ß				
02	STX	(inicio texto)	34	"	66	B	98	b				130	é	162	ó	194	┬	226	Ô				
03	ETX	(fin de texto)	35	#	67	C	99	c				131	â	163	ú	195	├	227	Ò				
04	EOT	(fin transmisión)	36	$	68	D	100	d				132	ä	164	ñ	196	─	228	õ				
05	ENQ	(consulta)	37	%	69	E	101	e				133	à	165	Ñ	197	┼	229	Õ				
06	ACK	(reconocimiento)	38	&	70	F	102	f				134	å	166	ª	198	ã	230	µ				
07	BEL	(timbre)	39	'	71	G	103	g				135	ç	167	º	199	Ã	231	þ				
08	BS	(retroceso)	40	(	72	H	104	h				136	ê	168	¿	200	╚	232	Þ				
09	HT	(tab horizontal)	41	)	73	I	105	i				137	ë	169	®	201	╔	233	Ú				
10	LF	(nueva línea)	42	*	74	J	106	j				138	è	170	¬	202	╩	234	Û				
11	VT	(tab vertical)	43	+	75	K	107	k				139	ï	171	½	203	╦	235	Ù				
12	FF	(nueva página)	44	,	76	L	108	l				140	î	172	¼	204	╠	236	ý				
13	CR	(retorno de carro)	45	-	77	M	109	m				141	ì	173	¡	205	═	237	Ý				
14	SO	(desplaza afuera)	46	.	78	N	110	n				142	Ä	174	«	206	╬	238	¯				
15	SI	(desplaza adentro)	47	/	79	O	111	o				143	Å	175	»	207	¤	239	´				
16	DLE	(esc.vínculo datos)	48	0	80	P	112	p				144	É	176	░	208	ð	240	≡				
17	DC1	(control disp. 1)	49	1	81	Q	113	q				145	æ	177	▒	209	Ð	241	±				
18	DC2	(control disp. 2)	50	2	82	R	114	r				146	Æ	178	▓	210	È	242	‗				
19	DC3	(control disp. 3)	51	3	83	S	115	s				147	ô	179	│	211	Ë	243	¾				
20	DC4	(control disp. 4)	52	4	84	T	116	t				148	ö	180	┤	212	È	244	¶				
21	NAK	(conf. negativa)	53	5	85	U	117	u				149	ò	181	Á	213	ı	245	§				
22	SYN	(inactividad sínc)	54	6	86	V	118	v				150	û	182	Â	214	Í	246	÷				
23	ETB	(fin bloque trans)	55	7	87	W	119	w				151	ù	183	À	215	Î	247	¸				
24	CAN	(cancelar)	56	8	88	X	120	x				152	ÿ	184	©	216	Ï	248	°				
25	EM	(fin del medio)	57	9	89	Y	121	y				153	Ö	185	╣	217	┘	249	¨				
26	SUB	(sustitución)	58	:	90	Z	122	z				154	Ü	186	║	218	┌	250	·				
27	ESC	(escape)	59	;	91	[	123	{				155	ø	187	╗	219	█	251	¹				
28	FS	(sep. archivos)	60	<	92	\	124	|			156	£	188	╝	220	▄	252	³				
29	GS	(sep. grupos)	61	=	93	]	125	}				157	Ø	189	¢	221	¦	253	²				
30	RS	(sep. registros)	62	>	94	^	126	~				158	×	190	¥	222	Ì	254	■				
31	US	(sep. unidades)	63	?	95	_						159	ƒ	191	┐	223	▀	255	nbsp				
127	DEL	(suprimir)																					

Fuente: El Código ASCII (s. f.).

Referencias

El Código ASCII. (s. f.). *El código ASCII*. Recuperado de https://elcodigoascii.com.ar/

Corona, M. A., y Ancona, M. D. (2011). *Diseño de Algoritmos y su codificación en Lenguaje C* (1ª ed.). México D.F: Mc Graw Hill.

Díaz, R. (2 de febrero de 2018). Descubre qué animal eres en el horóscopo chino. *Cadena SER*. Recuperado de http://cadenaser.com/ser/2015/02/19/sociedad/1424345838_727677.html

Disfruta Las Matemáticas. (2011). *Ecuaciones y Fórmulas.* Recuperado de http://www.disfrutalasmatematicas.com/algebra/ecuaciones-formulas.html

EcuRed. (s. f.). *Tablas de la verdad*. Recuperado de https://www.ecured.cu/Tablas_de_la_verdad

Edukativos. Apuntes para universitarios. (2013). *Organización física de una computadora (hardware)*. Recuperado de https://www.edukativos.com/apuntes/archives/3772

Fernández, J. (2017). *Juegos de matemáticas II (Soluciones).* Recuperado de https://soymatematicas.com/soluciones–juegos–de–matematicas–ii/

Sauceda, E. (2019). *Diseño de algoritmos en la programación de computadoras.* Recuperado de: http://www.academia.edu/download/32424199/Diseno_de_algoritmos_en_la_programacion_de_computadoras.docx. Visitado el 2 de noviembre de 2.019.

Gilpérez, J. M. (2015). *Operadores en Python.* Recuperado de http://www.josemanuelgilperez.com/programacion/python/operadores-en-python/

Gómez, S. R. y Moraleda, E. (2015). *Aproximación a la ingeniería del software.* Madrid: Universitaria Ramón Areces.

Guerrero, H. E. (2016). *La culpa es del programmer.* Popayán, Colombia: Auto–Edición.

Guzmán, E. y López, W. (2019). *Implementación de una estrategia didáctica de programación para la formación de habilidades de resolución de problemas en niños* (Tesis de maestría). Universidad Distrital Francisco José de Caldas, Bogotá. Recuperado de http://repository.udistrital.edu.co/bitstream/11349/14875/1/Guzm%C3%A1nTiqueElizabeth2019.pdf

Joyanes, L. (2008). *Fundamentos de Programación. Algoritmos, estructura de datos y objetos* (4ª ed.). Aravaca, España: Mc Graw Hill.

Juganaru, M. (2014). *Introducción a la programación.* México D.F.: Grupo Editorial Patria.

López, J. L. y Gutiérrez, Á. (2014). *Programación orientada a objetos con C++ y Java.* México, D.F.: Grupo Editorial Patria.

Neoreason, I. (2019). *Repl*. Recuperado de https://repl.it/languages

Niño, M. Á., Cobos, C. A. y Mendoza, M. E. (2010). *Introducción a la Informática. Algoritmos y Porgramación en Lenguaje C*. Popayán, Colombia: Universidad del Cauca. Recuperado de http://www.academia.edu/32719144/Introducción_a_la_Informática_Algoritmos_y_Porgramación_en_Lenguaje_C

Python Software Foundation. (2017). *Ttutorial de Python.* Recuperado de http://docs.org.ar/tutorial

Rey, J. P. (2018). *TECNOEXPLORA*. Recuperado de https://www.lasexta.com/tecnologia-tecnoxplora/ciencia/ecologia/cuantos-mosquitos-harian-falta-beberse-toda-sangre_2016090157fd15950cf2fd8cc6b1c255.html

Ruiz, M., Llorente, J., González, C., Aparicio, A. y Arribas, F. (2019). *Matemáticas I 1º Bachillerato*. Editorial Editex.

Sientes, B. (2017). *Elementos de un programa de Python.* Recuperado de http://www.mclibre.org/consultar/python/lecciones/python–elementos.html

Spiegel, M. (1991). *Manual de fórmulas y tablas matemáticas.* México: Mc Graw Hill.

Lista de tablas

Lista de figuras

Glosario

A

Algoritmo

Serie finita de pasos no ambiguos, que realiza una tarea concreta en un tiempo finito, previendo todas las situaciones posibles. Define el conjunto de instrucciones que sirven para ejecutar una tarea o resolver un problema. · 37, 40, 43, 48, 49, 50, 51, 52, 53

Algoritmo cualitativo

Pasos o instrucciones descritos por medio de palabras que sirven para llegar a la respuesta o solución de un problema. También conocido como algoritmo no computacional. · 40

Algoritmo cuantitativo

Pasos o instrucciones que involucran cálculos numéricos para llegar a la respuesta o dar solución a un problema. También conocido como algoritmo computacional. · 40

ASP .NET

Es un entorno para aplicaciones web desarrollado y comercializado por Microsoft. Es usado por programadores y diseñadores para construir sitios web dinámicos, aplicaciones web y servicios web XML. Apareció en enero de 2002 con la versión 1.0 del .NET Framework, y es la tecnología sucesora de la tecnología Active Server Pages (ASP). ASP.NET está construido sobre el Common Language Runtime, permitiendo a los programadores escribir código ASP.NET usando cualquier lenguaje admitido por el .NET Framework. · 41

B

Bigdata

Es un término que describe el gran volumen de datos, tanto estructurados como no estructurados, que inundan los negocios cada día. Pero no es la cantidad de datos lo que es importante. Lo que importa con el Big Data es lo que las organizaciones hacen con los datos. Big Data se puede analizar para obtener ideas que conduzcan a mejores decisiones y movimientos de negocios estratégicos. · 92

Booleano

Esta es la lógica que las computadoras usan para determinar si una declaración es falsa o verdadera. · 15, 16, 55

C

C#

Es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET. C# es uno de los lenguajes de programación diseñados para la infraestructura de lenguaje común. · 41

C++

C++ es un lenguaje que abarca tres paradigmas de la programación la progra-

143, 144, 146, 148, 150, 151, 153, 157, 165, 176, 209, 210, 212

Diagramación rectangular

También conocida con el nombre de diagrama estructurado N–S (Nassi–Shneiderman). En esta diagramación se utilizan una serie de rectángulos consecutivos, que son de diferentes tipos, y representan diferentes partes de un algoritmo. · 48

Disyunción

Relación entre dos o más elementos por la que cada uno de ellos excluye a los demás. · 16, 18

E

Elif

Es una contracción de else if. La estructura de control if / elif/ else permite encadenar varias condiciones. · 212

Entera

Variable que permite almacenar datos numéricos enteros (sin decimales) · 21, 22, 55

Entrada

Lugar donde se relaciona la información que va a ser leída. Aquí van todos los datos que se desconocen y se requieren para realizar los cálculos. · 39

Estructura secuencial

La estructura de secuencia hace referencia al orden de ejecución de instrucciones que se hace de forma secuencial, o sea, una instrucción después de la otra. · 63, 64

estructuras de decisión

Son estructuras que se usan en algoritmos que requieren evaluar algún tipo de condición o que tenga que elegir entre varias alternativas para llegar a una solución más precisa. · 4, 6, 46, 48, 137, 138, 139, 141, 156, 158, 159, 161, 207, 209, 237, 239

Estructuras de selección

Son estructuras conocidas con el nombre de estructuras caso, son similares a las estructuras de decisión múltiple pero son más compactas, ahorran espacio y son mucho más simples de utilizar. · 4, 48, 137, 210, 237

Expresiones algorítmicas

Expresiones manejadas dentro de lógica de programación que usan una sintaxis que permite ser "entendida" por la computadora. · 30, 32, 34, 35, 63, 74, 77, 250

Expresiones matemáticas

Expresiones que constan de un conjunto de símbolos del alfabeto, que en una expresión matemática incluyen constantes y variables. · 30, 31, 34, 35, 74, 250

F

Funciones

Se crearon para evitar tener que repetir constantemente fragmentos de código. Una función podría considerarse como una variable que encierra código dentro de si. Por lo tanto cuando accedemos a dicha variable (la función) en realidad lo que estamos haciendo es ordenar al programa que ejecute un

Lógica

. · 37

Manera de razonar o discurrir. · 1, 2, 4, 5, 6, 13, 14, 15, 16, 19, 30, 37, 42, 45, 55, 63, 70, 91, 104, 138, 139, 141

Lógica matemática

También llamada lógica simbólica, lógica teorética, lógica formal, o logística,1 es parte tanto de la lógica como de las matemáticas, y consiste en el estudio matemático de la lógica, y en la aplicación de dicho estudio a otras áreas de la matemática y de las ciencias. La lógica matemática tiene estrechas conexiones con las ciencias de la computación y con la lógica filosófica. · 15, 16, 139

M

Microsoft Excel

Es una aplicación de hojas de cálculo que forma parte de la suite de oficina Microsoft Office. Es una aplicación utilizada en tareas financieras y contables, con fórmulas, gráficos y un lenguaje de programación. · 238

O

Operadores

Símbolos que relacionan valores, variables o constantes dando un resultado determinado según el tipo de dato procesado. · 13, 14, 15, 16, 19, 23, 24, 29, 30, 36, 56, 60, 63, 69, 70, 105, 106, 138, 139

Operadores aritméticos

Operadores que permiten la realización de operaciones matemáticas con variables y constantes. · 14, 15, 23, 30, 60, 63, 69, 70

Operadores lógicos

Operadores que permiten hacer comparaciones entre variables y/o valores. Estos operadores se dividen en operadores lógicos relacionales y operadores lógicos booleanos. · 15, 16, 138

P

PHP

Acrónimo recursivo en inglés de PHP Hypertext Preprocessor (preprocesador de hipertexto), es un lenguaje de programación de propósito general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en un documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera el HTML resultante. · 41

Porcentaje

Cantidad correspondiente a una proporción de un total dividido en cien partes. · 114

Proceso

Lugar donde se realizan las operaciones y cálculos necesarios para alcanzar el resultado esperado. · 39, 66, 105, 108

Programador

Es aquella persona que escribe, depura y mantiene el código fuente de un pro-

de 70.2 o el valor de una retención de 0.08. · 55, 57, 251

Repl.it

Es una plataforma ideal para programar, tanto si eres un programador experimentado como si estás dando tus primeros pasos en un lenguaje. Sus puntos fuertes son que no necesitarás instalar absolutamente nada ni pagar para utilizarla. · 74

S

Salida

Lugar donde se describe la información que se va a imprimir, la cual dará respuesta a los requerimientos o peticiones que tenía el algoritmo al empezar. · 39, 108

Sintaxis

Es la parte de la gramática que estudia las reglas y principios que gobiernan la combinatoria de constituyentes sintácticos y la formación de unidades superiores a estos, como los sintagmas y las oraciones gramaticales. La sintaxis, por tanto, estudia las formas en que se combinan las palabras, así como las relaciones sintagmáticas y paradigmáticas existentes entre ellas. · 4, 5, 30, 41, 43, 45, 104, 141, 142, 145, 146, 149, 150, 156, 157, 158, 209

Software

Es todo programa o aplicación programado para realizar tareas específicas. · 1, 3, 4, 5, 134, 135, 205, 212, 223

V

Variable

Es un espacio de memoria reservado para almacenar un valor determinado que corresponde a un tipo de dato soportado por el lenguaje de programación en el cual se trabaja. · 16, 20, 39, 54, 55, 59, 60, 63, 65, 66, 67, 68, 72, 75, 76, 77, 84, 85, 108, 127, 170, 209, 210

VBA

Visual Basic for Aplication (Visual Basic para aplicaciones) es una implementación del lenguaje de programación controlado por eventos discontinuo de Microsoft , Visual Basic 6, y su entorno de desarrollo integrado asociado (IDE). · 27

Visual Basic

Lenguaje de programación de Microsoft orientado a eventos y utilizado principalmente para realizar consultas a bases de datos de Microsoft como Fox Pro, SQL, etc. que funcionan en servidores Windows. · 27, 41

Del Autor

Ingeniero de sistemas de la Institución Universitaria Salazar y Herrera

Magíster en software libre de la UNAB

Docente de tiempo completo de la Escuela de Ingeniería de la Institución Universitaria Salazar y Herrera

Coordinador de semilleros de programación y coach de los equipos de programación competitiva de la Institución Universitaria Salazar y Herrera

Fundamentos iniciales de lógica de programación I.
Algoritmos en PSeInt y Python

Tipografía: Times New Roman