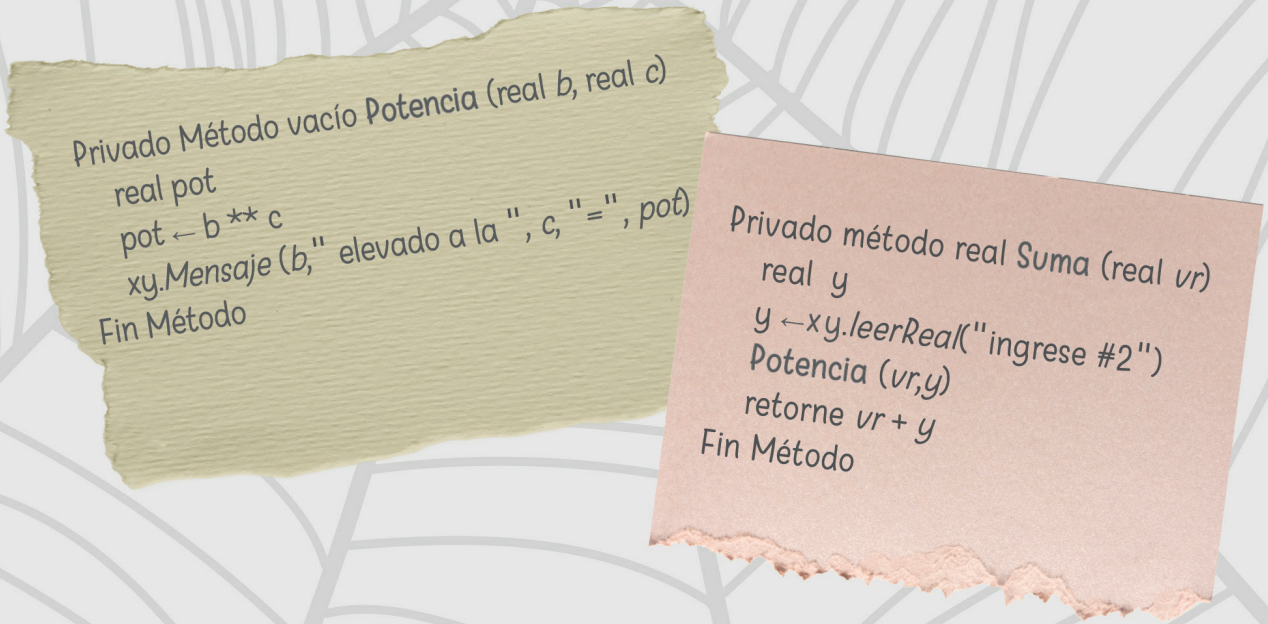
Institución Universitaria
Reacreditada en Alta Calidad

Lógica de programación básica orientada a objetos con ejercicios resueltos



Delio A. Aristizábal Martínez

Sandra M. Quiceno Metaute

LÓGICA DE PROGRAMACIÓN BÁSICA ORIENTADA A OBJETOS CON EJERCICIOS RESUELTOS

LÓGICA DE PROGRAMACIÓN BÁSICA ORIENTADA A OBJETOS CON EJERCICIOS RESUELTOS

Delio A. Aristizábal Martínez
Sandra M. Quiceno Metaute

Institución Universitaria
Reacreditada en Alta Calidad

La lógica te lleva desde A hasta B.
La imaginación te lleva a cualquier parte.

Albert Einstein

La mayoría de los buenos programadores programan
no porque esperen que se les pague o por adulación,
sino porque es divertido programar.

Linus Torvalds

Lógica de programación básica orientada a objetos con ejercicios resueltos

Edición: 2023
ISBN: 978-958-5122-71-0 (PDF)
ISBN: 978-958-5122-70-3 (impreso)

Autores
Delio A. Aristizábal Martínez
©Sandra M. Quiceno Metaute

Directora editorial
Juliana Cardona Quiros
Profesional universitario – Editorial ITM
Sebastián Vásquez Moreno
Corrección de textos
Martha Cecilia Caballero Jerez
Diseño y diagramación
Mauricio Raigosa Álvarez

Sello Editorial ITM
Calle 75 75-101 / Tel: 604 440 51 00 ext. 5197
Editado en Medellín, Colombia
catalogo.itm.edu.co - fondoeditorial@itm.edu.co
www.itm.edu.co

Aristizábal Martínez, Delio A.
Lógica de programación básica orientada a objetos con ejercicios resueltos / Delio A. Aristizábal Martinez, Sandra M. Quiceno Metaute -- Medellín: Institución Universitaria ITM, 2023.

528 p. -- (Teknik)

Incluye referencias bibliográficas

1. Lógica de programación. 2. Programación orientada a objetos. 3. Algoritmos. I. Quiceno Metaute, Sandra M. II. Tít. III. Serie.

005.1 SCDD 21

Catalogación en la publicación - Biblioteca ITM

Este es un Texto de Formación elaborado por profesores de la Facultad de Ingenierías del ITM. Según la OCDE, este libro se inscribe en la gran área de Ingeniería y Tecnología, en las áreas de Ingeniería de Sistemas y Comunicaciones, Telecomunicaciones, Sistemas de Automatización y Sistemas de Control.

CONTENIDO

AGRADECIMIENTOS

Agradezco a Dios y a la vida por todo el tiempo que me dieron para poder realizar este libro; y, por supuesto, a Delio por haber confiado en mí para formar este equipo académico que nos llevó a alcanzar este maravilloso logro. Para mi hija Vanessa tengo todas las más especiales palabras de gratitud por toda la comprensión y el acompañamiento con sus aromáticas para relajarme y permitirme coger energías para continuar cada día y cada noche.

Sandra M. Quiceno M.

Agradezco de todo corazón a mis hermosas hijas por su alegría, entusiasmo y apoyo; sin ellas no hubiera sido posible obtener la fuerza necesaria en este proceso; y, de manera muy especial, a mi coautora Sandra M. Quiceno M. por su tiempo, su conocimiento, paciencia, sensatez y acompañamiento en este reto. Un agradecimiento a mis estudiantes y a mis compañeros por sus enseñanzas y apreciaciones para mejorar día a día en este camino de la educación.

Delio A. Aristizábal M.

Los autores queremos agradecer a la Institución Universitaria ITM por el apoyo brindado para la edición de este libro. A la Editorial ITM por la compañía brindada en todo lo correspondiente a la edición y a la publicación. A la Facultad de Ingenierías de la institución por justificar y promover la escritura de este texto en pro de sus estudiantes y en especial a los del programa Desarrollo de Software. Y a todas las personas que de una u otra forma nos brindaron el aliento para la escritura de este libro

PRESENTACIÓN

Tengo el enorme agrado de presentar este libro cuyos autores no solo son colegas, sino amigos a lo largo de la vida. Si bien aprecio todo el trabajo realizado por estos excelentes autores, considero importante resaltar el aporte que le hacen al fortalecimiento de nuestros futuros profesionales en el área de la lógica de programación. Es notable la sencillez con la que se explican conceptos muy complejos, lo que permite a los lectores sin conocimientos específicos del tema comprenderlos sin mayores dificultades.

Este es un libro que pueden disfrutar estudiantes de secundaria y de educación superior. Confío en que se convertirá en un buen material de estudio y apoyo con el fin de lograr las competencias necesarias para dar solución a los problemas de la cotidianidad aplicando una buena lógica para ello.

Leonel Velásquez Torres

INTRODUCCIÓN

¿Te han dicho que programar es difícil? ¿Te gusta la programación? Si las respuestas son afirmativas, este libro es una puerta para iniciar con entusiasmo el universo de la programación. Pensar, razonar, analizar, cuestionar, programar y actuar son la clave para hacer parte de la transformación en todos los ámbitos.

El libro contempla los pasos para iniciar en el campo de la programación, con conceptos fundamentales como: los elementos para la solución de un problema, entidades primitivas, estructuras algorítmicas: secuenciales, de decisión (condicionales), iterativas (ciclos) y arreglos unidimensionales y bidimensionales, bajo el concepto de la Programación Orientada a Objetos (POO).

Con el objetivo de entregar al lector herramientas para estudiar, analizar y desarrollar el pensamiento lógico, se escribió este libro enfocado en el planteamiento de ejercicios, cada uno con su propuesta de solución algorítmica. Los ejercicios planteados, en su mayoría, han sido creados o adquiridos por medio de las experiencias empresarial y académica (como docentes) en la asignatura de Lógica de Programación o Algoritmos; algunos de los problemas planteados han sido muy genéricos y pueden ser parecidos a los de otros autores. Cada ejercicio es utilizado y aplicado según el tema explicado para permitir la mejor comprensión al lector.

Con el objetivo de proponer un análisis y entendimiento de los ejercicios, la solución inicia con la representación de un diagrama de clase tipo UML y la estructura esencial del diseño de un algoritmo basado en 3 preguntas: ¿qué solicitan? (Entrada) y ¿qué entregan? (Salida), en las que se definen los datos de entrada y salida; y la última: ¿cómo hacerlo? (Proceso), en el que se explica una alternativa de procedimientos a seguir para la solución del problema, reforzando el concepto de modularidad utilizada en la POO.

Para la solución de los ejercicios realizados, se utilizó una sintaxis en el diseño de los algoritmos escritos en un lenguaje natural (pseudocódigo) muy propia de los programadores, con el uso de instrucciones y palabras, de forma sencilla, que ayudan a una mayor comprensión sobre cada tema de estudio. Con el avance del curso se crean 3 clases base para la manipulación de la información: clsGenerales para valores simples y para los estructurados (arreglos), clsVectorGral - unidimensionales (vector), y clsMatrizGral - bidimensionales (matriz).

En el pseudocódigo de este libro se utilizan los tipos de datos Entero, Real, Cadena y Lógico, los cuales representan la naturaleza de los datos en la memoria; no se utilizan palabras acentuadas ni caracteres especiales para los nombres de los componentes de los algoritmos, debido a que existen reglas que establecen cómo nombrar los identificadores; se utilizan el formato de negrita para resaltar los nombres de los métodos, y la cursiva para las variables y constantes; además, se utiliza la notación lowerCamelCase para dichos nombres y el operador de asignación en las fórmulas es ←.

La experiencia ha mostrado que muchos lectores quieren más información de la suministrada. Para brindar este valor agregado, en cada tema se dejan retos y planteamientos de problemas propuestos.

Todas las figuras y tablas son diseños propios de los autores.

Los autores agradecen el envío de cualquier comentario sobre este libro a los correos: delioaristizabal@itm.edu.co y sandraquiceno0780@correo.itm.edu.co

1.
CONCEPTOS BÁSICOS

CAPÍTULO 1. CONCEPTOS BÁSICOS

1.1. ¿Qué es un problema?

Un problema es una cuestión o inquietud que se debe solucionar (resolver). Para solucionar un problema primero se debe identificar en qué consiste (qué solicitan) y con qué contamos para iniciar su solución (qué entregan o qué tenemos); luego, se propone una serie de pasos para dar una solución a dicho problema (cómo hacerlo).

1.2. ¿Qué es un algoritmo?

Un algoritmo es el conjunto de instrucciones o serie de pasos que representa el procedimiento a seguir, con un orden lógico, para dar solución a uno a varios problemas.

Un problema puede tener múltiples formas de solución, de modo que pueden existir múltiples algoritmos para obtener el mismo resultado; todo depende de la concepción o entendimiento del problema por parte del programador.

1.3. Pasos para la solución de un problema mediante algoritmos

Para obtener una solución adecuada u óptima a un problema planteado se debe tener en cuenta una serie de conceptos y condiciones. Véase la Tabla 1 para dar claridad a los datos de *entrada* y *salida* y, finalmente, expresar en un lenguaje natural el procedimiento a emplear para obtener la solución.

Tabla 1.

Pasos para la solución de un problema

Paso	Función	Acción
Escuchar y escribir	Establecer los requerimientos previos	Escuchar y escribir el relato del problema a partir de las necesidades y restricciones.
Entenderlo	Análisis previo del problema	Analizar y comprender el problema antes de realizar el algoritmo.
Plantearlo	Definición de requerimientos	Establecer claramente los requerimientos solicitados por el cliente, teniendo en cuenta las restricciones para obtener la solución al problema.
Estructurarlo	Identificación de los módulos	Establecer el grupo de bloques de código en el algoritmo, para simplificar o especializar tareas, y para el funcionamiento modular del programa.
Escribirlo	Realización de los algoritmos	Escribir la solución algorítmica para que los demás la entiendan, cumpliendo con las características y las restricciones que se indicaron en los requerimientos.

Continúa

Paso	Función	Acción
Codificarlo	Implementación de los algoritmos	Traducción del algoritmo y de sus componentes a cualquier lenguaje de programación logrando el resultado esperado. En este paso se habla de un programa para computadora compuesto por un conjunto finito de instrucciones para dar solución a los requerimientos establecidos a solucionar.

La tarea de un buen programador consiste en analizar bien el problema a partir de sus requerimientos y condiciones, establecer adecuadamente la lógica en el algoritmo, codificar el algoritmo en un programa utilizando un lenguaje de programación, compilar, realizar pruebas y los respectivos correctivos lógicos, ponerlo en funcionamiento y hacer mantenimiento y actualización.

1.4. Estructura de un algoritmo

Todo algoritmo está estructurado en 3 macropartes fundamentales: Entrada, Proceso y Salida que permiten dar la solución a un problema con sus diferentes requerimientos. Véase Figura 1.

Figura 1.

Estructura de un algoritmo

Las partes de un algoritmo son las que permiten, unificadamente, dar un orden lógico; por lo tanto, es fundamental definir cada una de estas partes para establecer una solución con precisión. Véase Tabla 2.

Tabla 2.

Partes de la estructura del algoritmo

Parte	Comportamiento
Inicio	Es la apertura del algoritmo a la solución del problema.
Datos de entrada	Son los datos que se identifican en el análisis de los requerimientos para ser suministrados al algoritmo como valores iniciales para resolver el problema.
Proceso	Es el conjunto de pasos, instrucciones o procedimientos necesarios para establecer los datos de salida, teniendo en cuenta los datos de entrada.

Continúa

Parte	Comportamiento
Datos de salida	Es la información o los resultados del proceso como una solución al problema planteado.
Fin	Indica la culminación del algoritmo.

1.5 Características de un algoritmo

Para realizar un análisis y un diseño adecuado del algoritmo, se deben tener en cuenta las características que se especifican en la Tabla 3.

Tabla 3.

Características de un algoritmo

Característica	Función
Legible	Debe tener buena claridad en su escritura y en sus componentes, que permita entenderlo y leerlo fácilmente.
Finito	Debe tener un principio y un final con una cantidad limitada de instrucciones.
Preciso	Cada instrucción y la acción que realiza deben estar muy bien definidas. No deben ser ambiguas.
Definido (validez)	Al momento de realizar dos o más pruebas a un algoritmo con iguales datos de entrada, se debe presentar una consistencia en los resultados obtenidos ya que debe de ser el mismo resultado en cada ejecución, a excepción de que se manejen valores aleatorios en la entrada o en el proceso.
Eficiente	Debe optimizar la memoria, el recurso del equipo y que el tiempo de respuesta sea eficaz.
Flexible	Permite realizar cambios y mantenimientos preventivo y correctivo.
Modular	Que se pueda fraccionar en tareas o comportamientos especializados, ya sea en métodos o funciones.
Tener datos de entrada	Debe tener datos de entrada como insumo para ser utilizados en las operaciones del proceso; no siempre existe la necesidad de solicitar información al usuario por estar contemplada dentro de él mismo.
Tener datos de salida	Siempre genera como mínimo una repuesta; no es adecuado en un algoritmo que hace algo, no saber si se realizó o no satisfactoriamente; como mínimo tener un mensaje de error o de éxito.

1.6. Entidades primitivas

Son los componentes requeridos en el diseño e implementación de los algoritmos; entre estos componentes se tienen: *tipos de datos, identificadores, operadores, expresiones y palabras reservadas*; además, son las encargadas de especificar las condiciones para modificar expresiones o fórmulas algebraicas o matemáticas a expresiones algorítmicas.

1.7. Tipos de datos

El tipo de dato es una característica o atributo que indica al programador sobre la clase o naturaleza del dato que se va a manejar. En este libro se utilizan dos grupos:

1.7.1. Simples

Los tipos de datos simples se basan en un único valor de una única naturaleza; son atómicos y pueden ser predefinidos por el lenguaje de programación o por el usuario. Entre los tipos de datos simples se tienen:

✓ *Numéricos:* son aquellos signos o símbolos utilizados para designar cantidades, valores o entidades que se comportan como cantidades. Son las expresiones de la relación entre la cantidad y la unidad.

Son datos numéricos:

- *Entero:* es el tipo de dato al que se le puede asignar un valor perteneciente al conjunto de los números naturales enteros positivos, negativos o el cero.
- *Real:* es el tipo de dato al que se le puede asignar un valor perteneciente al conjunto de los números de punto flotante (fraccionario), formado por los números racionales e irracionales; números compuestos de una parte entera y de una decimal separados, generalmente, por una coma.

✓ *Carácter o Cadena (alfanumérico):* el tipo de dato Carácter o Cadena es el atributo al que se le puede asignar información compuesta por letra(s), número(s) o símbolo(s), conocidos como datos alfanuméricos, entre ellas: un carácter, palabras, frases, símbolos, etc.

Cuando una cadena contiene información numérica no es posible realizar operaciones matemáticas.

✓ *Lógico:* es el atributo al que se le puede asignar como único valor: verdadero o falso.

✓ *Vacío:* es un atributo especial para representar un valor nulo o vacío. Se utiliza como parámetro de salida o retorno en algunos de los métodos o funciones con el fin de indicar que el método no retorna nada.

✓ *Otros:* entre ellos tenemos: las fechas, moneda, hora, etc.

Para los algoritmos a desarrollar en este libro se utilizan, básicamente, los tipos de datos: Cadena, Entero, Real y Lógico; además, Vacío, como retorno en algunos procedimientos o métodos.

1.7.2. Estructurados

Los tipos de datos estructurados se basan en una colección de valores homogéneos y no homogéneos bajo un mismo nombre para su posterior manipulación.

Entre los tipos de datos estructurados están los arreglos. Son estructuras que admiten almacenar una colección de valores homogéneos bajo un mismo nombre en memoria. Entre ellos se tienen: los vectores (unidimensionales), las matrices (bidimensionales) y otras dimensiones. Otras estructuras son: pilas, colas, listas, árboles, archivos y grafos.

1.8. Identificadores

Es el nombre que se le asigna a cada elemento, componente o entidad en un algoritmo o programa (clase, método, funciones, variable, etiquetas o constante, entre otros). Para nombrar un identificador deben cumplirse unas reglas de sintaxis como:

- ✓ Ser definido por uno o más caracteres o números.
- ✓ El primer componente debe ser una letra o el carácter de subrayado (_), el resto pueden ser letras, números o el mismo subrayado (_). Las letras pueden ser mayúsculas o minúsculas. El único carácter especial permitido en el nombre de un identificador es subrayado (_).
- ✓ No deben existir dos componentes en un ámbito con el mismo nombre, a no ser que se encuentren en condiciones o ambientes especiales, lo cual no quiere decir que un identificador no pueda aparecer más de una vez en un algoritmo.
- ✓ No puede contener espacios en blanco.
- ✓ El nombre debe ser mnemotécnico, según el dato que almacene o tenga asociado.

Ejemplos de identificadores escritos correctamente:

Num, *cant_Estu*, *_Promedio, nota1*, *Nota_2*, *Nota3*, *subtotal, subTotal2*

Ejemplos de escritura incorrecta en identificadores:

1nota, *%Descuento*, *Cod Est*, *Prom/Tot*, *nota 1*, *$TotalAPagar*

En la escritura de algoritmos y en la programación se utilizan como buena práctica, entendimiento y eficiencia de la memoria por parte del programador, diferentes tipos de convenciones o notaciones para nombrar los identificadores; entre ellas tenemos:

- ✓ La notación *húngara* que consiste en colocar como prefijo el tipo de dato o una simplificación de esta en el identificador para dar claridad del tipo de dato que es al momento de la lectura por parte del programador; por ejemplo: cadenaNombre o strNombre, enteroEdad o intEdad.
- ✓ La notación *Camel Case* se aplica para identificadores con palabras compuestas. Este tipo de escritura está conformado por dos estilos: *UpperCamelCase* y *lowerCamelCase*. El primero consiste en usar la primera letra de cada palabra en mayúscula, y el segundo, *lowerCamelCase* usa la primera letra de la primera palabra en minúscula y la primera letra de las palabras compuestas restantes en mayúscula.

Ejemplos:

UpperCamelCase: ***NotasFinales, TotalCancelado.***
lowerCamelCase: ***notasFinales, totalCancelado.***

Este libro utiliza la notación *lowerCamelCase*.

1.9. Palabras reservadas

Son identificadores *predefinidos* que tienen una función especial dentro del contexto del algoritmo y no se pueden manipular por parte del programador, ya que tienen una tarea específica en el diseño de las estructuras utilizadas en el algoritmo.

Algunas de las palabras reservadas en los algoritmos de este libro son:

Cadena, Real, Lógico, Entero, vacío, Clase, Método, Principal, Si, Entonces, Sino, Fin, Para, Hacer, Mientras, Const, Público, Privado, verdadero, falso, nuevo, Según, Caso, Sea, retorne, interrumpir y Mod.

1.10. Variables y constantes

Las variables y las constantes son identificadores que reservan espacio en la memoria, pero tienen un comportamiento diferente durante la ejecución del algoritmo o programa.

1.10.1. Variable

Es un identificador y puede cambiar de valor durante la ejecución del algoritmo/programa.

Ejemplos:

- ✓ Real *Dscto*, *reteFte*, *sub_Tot* ← 0, *Dscto* ← 0.05, *Renta*
- ✓ Cadena *Nombre*, *estadoCiv* ← "C", *ciu_Dpto*
- ✓ Entero *cantEstud* ← 0, *i*, *j*, *k_Est*

En los ejemplos anteriores se observa que primero va el tipo de dato seguido del identificador (ejemplo: Real *Dscto*); a esto se le conoce como *Declarar una variable*, pero algunas variables tienen asignado un valor al momento de la declaración, esto es *Declarar e inicializar una variable* (ejemplo: Entero *cantEstud* ← 0). Se debe tener en cuenta que al mismo tiempo se pueden declarar e inicializar varias variables, solo basta separarlas con una coma (,). Ejemplo: Entero *cantEstud* ← 0, *i*, *j*, *k_Est*

1.10.2. Constante

Es un identificador y no puede cambiar de valor durante la ejecución de un programa. Para este tipo de identificador se deben tener en cuenta los siguientes criterios:

- ✓ Se debe declarar e inicializar antes de ser usada.
- ✓ Se utiliza la palabra reservada *Const* antes del tipo de dato.
- ✓ Su escritura es en mayúscula. Generalmente se hace así.

En un lenguaje de programación, cuando se intenta cambiar el valor de una constante durante la ejecución del programa, inmediatamente se genera un mensaje de error.

Ejemplos:

Const Real PI ← 3.1415926

Const Real IVA ← 0.19

1.11. Comentarios o documentación de un programa

Son frases o párrafos significativos, explicativos o descriptivos no ejecutables que ayudan a la fácil comprensión del código. Los comentarios constituyen la documentación interna en el código para dar mayor claridad a lo codificado y no afectan el orden del algoritmo, solo pretenden hacer más fácil su lectura. En este libro se utilizan 2 formatos de comentarios:

1.11.1. Comentario de una línea completa

Se utiliza para comentar de forma parcial o total una línea de código, generalmente para dar información de lo que contiene la línea o la siguiente. Los símbolos serán un par de barras inclinadas // (*slash*) que se insertan al inicio del texto comentario, en letra cursiva de color verde.

1.11.2. Comentario multilínea

Se utiliza para comentar un bloque de líneas del código. Para abrir el comentario se utilizan los símbolos /* (*slash* y asterisco), y */ (asterisco y *slash*) para cerrarlo; todo lo

que está dentro de /* */ no afecta el código. En la Tabla 4 se pueden observar algunos ejemplos.

Tabla 4.

Ejemplos de comentarios

Tipo de comentario	Ejemplo
De una línea completa	*// Este es el primer método que se ejecuta* Público Método vacío **Principal** ( ) Entero *Acum* ← 0 *// Se declara e inicializa la variable Acum* Entero *A*, *B* *// B ← A^2* *B ← A ** A // a la variable B se le asigna el cuadrado de A* Fin Método Principal
Multilínea	*/* El algoritmo intercambia los valores de las variables a y b, sin utilizar variables auxiliares para ese proceso */* Público Método vacío **Principal** ( ) Entero *a* ← 8, *b* ← 10 ***/** *a ← a + b* *b ← a – b* *a ← a + b // El signo más no; debe ser el menos* ***/*** *a ← a + b* *b ← a – b* *a ← a – b* Fin Método

1.12. Operadores y expresiones

Los operadores y las expresiones son elementos fundamentales en el diseño de un algoritmo; se utilizan para escribir instrucciones que, generalmente, son de tipo lógico/matemático. En conjunto pueden ser asignados a un identificador de tipo variable: Cadena, Numérico o Lógico, o ser implementados para las decisiones o ciclos en el diseño del algoritmo.

1.12.1. Operador

Es un símbolo (o conjunto de símbolos) que determina la función sobre dos operandos. Los operadores están relacionados con los tipos de datos de los operandos. Las clasificaciones básicas de los operadores se pueden observar en la Tabla 5.

Tabla 5.

Clasificación de los operadores

Clasificación	Operadores
Aritméticos	+ (suma), – (resta), * (multiplicación) **, ^ (potencia), Mod (residuo de una división entera)
Lógicos	v (o, disyunción), ∧ (y, conjunción), ~ (negación lógica)
Relacionales	> (mayor que), < (menor que), >= (mayor o igual que), <= (menor o igual que), == (igual que, comparación)
De asignación	←, = (conocidos como asignación interna) El operador utilizado en este libro es: ←
Entrada y Salida	Entrada: Lea (conocido como asignación externa) Salida: Muestre, Imprima o Escriba El operador utilizado en este libro es: Muestre

1.12.2. Expresión

Es cualquier sentencia compuesta por operadores y operandos en el algoritmo y genera un valor al ser evaluada. Algunos tipos de expresiones:

- ✓ Aritmética: la que utiliza operadores aritméticos; generalmente, el resultado obtenido es un valor numérico.
- ✓ Lógica: es aquella que utiliza operadores lógicos y genera los siguientes valores: falso o verdadero.
- ✓ Relacional: utiliza operadores relacionales al efectuar la operación; el resultado obtenido es un valor lógico, falso o verdadero.
- ✓ Asignación: utiliza operadores de asignación; generalmente, el resultado obtenido guarda relación con los datos y los operadores utilizados.
- ✓ Entrada: utiliza el operador *Lea*; su función es asignar a una variable el valor ingresado por el usuario o por un medio externo.
- ✓ Salida: utiliza los operadores: *Muestre*, *Imprima* o *Escriba* para mostrar por pantalla u otro medio el valor o los valores al usuario, como información o respuesta de una operación o proceso, de una manera simple (individual) o unida (concatenada, utilizando el signo coma) con el valor de otras variables o textos.

En la Tabla 6 se pueden observar algunos ejemplos de expresiones en las que se tienen las siguientes variables con su respectivo valor de asignación:

A ← 10, *B* ← 4, *N* ← verdadero, *M* ← falso. Estos valores se conservan en los ejemplos de la Tabla 6.

Tabla 6.

Ejemplos de expresiones

Operador	Expresión	Resultado
Aritméticos	*X* ← *A* + *B*	*X* ← 14
	Y ← *A* ** *B*	*Y* ← 10000
	R ← *A Mod B*	*R* ← 2
Lógicos	*X* ← *N* ∨ *M* (*Ver tablas de verdad*)	*X* ← *verdadero*
	Y ← *N* ∧ *M*	*Y* ← *falso*
	R ← ~*N*	*R* ← *falso*
Relacionales	*X* ← *A* > *B*	*X* ← *verdadero*
	Y ← *A* <= *B*	*Y* ← *falso*
	R ← *A* == *B*	*R* ← *falso*
Asignación	*R* ← *A* + *B*	*R* ← 14
Entrada y Salida	Muestre "Ingrese el valor numérico de A: "	*A* ← –5
	Lea *A*	
	Muestre "Ingrese el valor numérico de B: "	*B* ← 10
	Lea *B*	
	Muestre "Vr. A = ", *A*, ", Vr. B = ", *B*	Vr. A = –5, Vr. B = 10

1.12.3. Tablas de verdad básicas con operadores lógicos

Se dice que la proposición lógica es una expresión en la que su resultado debe ser verdadero o falso, pero no puede ser bivalente. De modo que una tabla con valores lógicos muestra el resultado de una proposición compuesta por operadores lógicos en la que cada proposición de valores consta de dos operandos y un operador para ser evaluados (en pareja); y de igual manera con los siguientes elementos de la expresión. Cabe anotar que cada expresión es evaluada y se obtiene un valor lógico (verdadero o falso).

Entre las diferentes tablas de verdad este libro utiliza la disyunción y la conjunción.

- ✓ Disyunción: es la operación que relaciona dos o más proposiciones o expresiones utilizando el operador "o" lógico o su equivalente "V". Al evaluar la expresión, el