CIENCIAS NATURALES E INGENIERÍA

Pensamiento algorítmico

ABEL GARCÍA NÁJERA

KAREN SAMARA MIRANDA CAMPOS

SAÚL ZAPOTECAS MARTÍNEZ

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA

MATEMÁTICAS APLICADAS Y SISTEMAS

COLECCIÓN LIBRO DE TEXTO

UNIVERSIDAD AUTÓNOMA METROPOLITANA
Casa abierta al tiempo
Unidad Cuajimalpa

Esta obra es una de las ganadoras de la “Convocatoria para libros de texto como apoyo en la impartición de los programas de estudios”, 2023. Fue evaluada para su publicación por el Consejo Editorial de la UAM Unidad Cuajimalpa, con base en los dictámenes solicitados a pares académicos mediante un esquema que preserva el anonimato mutuo. Estos dictámenes resultaron favorables.

Diseño de portada: Aldo Juárez Herrera

Universidad Autónoma Metropolitana
Unidad Cuajimalpa
Av. Vasco de Quiroga 4871, col. Santa Fe Cuajimalpa
Alcaldía Cuajimalpa de Morelos
C.P. 05348, Ciudad de México

www.cua.uam.mx

ISBN: 978-607-28-3140-7
ISBN: 978-607-28-3020-2 (Colección)

UAMC
EDITORIAL

Pensamiento Algorítmico

Abel García Nájera
Karen Samara Miranda Campos
Saúl Zapotecas Martínez

Universidad Autónoma Metropolitana Unidad Cuajimalpa
División de Ciencias Naturales e Ingeniería
Departamento de Matemáticas Aplicadas y Sistemas

Casa abierta al tiempo
UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Cuajimalpa

Índice general

Prólogo

A finales del siglo XX, la computación y las tecnologías de la información se desarrollaron a pasos agigantados. Actualmente, vivimos en la llamada "sociedad de la información" donde muchas de las tareas que las personas necesitan o desean hacer pueden realizarse con una computadora o con ayuda de ellas. Esto se acentuó con la pandemia derivada del coronavirus SARS-CoV-2 desde inicios del año 2020, donde prácticamente todas las actividades productivas encontraron una alternativa para continuar su labores a través de soluciones basadas en tecnologías de la información y la comunicación. No obstante, debido a la familiaridad con la que utilizamos dispositivos electrónicos cotidianamente, muchas veces es fácil olvidar que las computadoras carecen de inteligencia o entendimiento real. De la misma manera, cuando se habla de computación se piensa en las computadoras como la herramienta principal. La computación se concibe entonces como el arte de utilizar las computadoras para realizar una tarea específica. Sin embargo, detrás de este arte de utilizar las máquinas se esconde un vasto campo científico, del cual sus ramificaciones sobrepasan ampliamente a las computadoras y su funcionamiento. En este documento, queremos explorar los conceptos básicos de computación y programación sin utilizar las computadoras, regresando a los fundamentos, es decir, **cómo resolver problemas**.

La percepción de "inteligencia" proviene de las personas que han resuelto los problemas y que les han "mostrado" a las computadoras cómo resolverlos, es decir, las computadoras pueden realizar aquellas tareas para las cuales las personas han desarrollado algoritmos para resolverlas. Así, se espera que las y los profesionistas de la computación sean capaces de crear algoritmos que las computadoras puedan ejecutar. Para este fin, es indispensable que las Ingenieras y los Ingenieros en Computación desarrollen el llamado *pensamiento algorítmico*[1], que es una forma de pensar procedimentalmente para resolver problemas, es decir, siguiendo una secuencia paso a paso. El pensamiento algorítmico se concibe como el conjunto de habilidades que permiten entender, crear, ejecutar y evaluar algoritmos [5].

Entre estas habilidades se encuentran:

- especificar problemas,
- analizar problemas,

[1] También se encuentra el término como *pensamiento computacional* o *problem solving skill*.

- encontrar los procesos básicos de un problema dado,
- construir un algoritmo basado en esos procesos básicos,
- pensar acerca de los casos (especiales y normales) que involucra el problema y
- encontrar y proponer mejoras a un algoritmo.

El proceso del pensamiento algorítmico se puede dividir en descomposición, reconocimiento de patrones y abstracción. Ya que un algoritmo documenta el *cómo* llevar acabo y completar una tarea, si un algoritmo está bien escrito, debería poder utilizarse no solamente para realizar una única tarea sino un grupo de tareas similares que requieran los mismos pasos. La existencia de un algoritmo significa que la tarea puede ser potencialmente **automatizada**, es decir, que la puede llevar a cabo una computadora [18].

La motivación para crear la unidad de enseñanza-aprendizaje (UEA) Taller de Algoritmos, y en particular este libro, es ofrecerle al alumnado la información necesaria para que desarrollen el pensamiento algorítmico. Esto como resultado de las observaciones hechas durante los primeros quince años de vida de las licenciaturas impartidas en la División de Ciencias Naturales e Ingeniería y de la experiencia acumulada impartiendo esta UEA y UEA relacionadas, pero sobre todo de los valiosos comentarios realizados por alumnas y alumnos. Adicionalmente, es muy importante destacar que las habilidades desarrolladas durante esta UEA no son exclusivas de las áreas de conocimiento de Computación y Matemáticas Aplicadas. Las habilidades, los conceptos y el razonamiento lógico que forman parte del pensamiento algorítmico son igualmente necesarios para el alumnado de otras licenciaturas [15], incluso de otras áreas diferentes a la ingeniería, ya que permiten plantear soluciones sin dar nada por hecho o considerar que se sabe algo previamente. Por ejemplo, si se quisiera hacer un análisis de texto, el análisis es un procedimiento algorítmico [9]. Por lo anterior, se recomienda que el alumnado de otras licenciaturas cursen esta UEA, cuyo objetivo subyacente es que las personas aprendan a analizar problemas, a pensar en soluciones a dichos problemas y expresarlas en términos en los que cualquier otra persona pueda entenderlas para que también los puedan resolver.

Este libro forma parte del componente pedagógico del modelo educativo de la Universidad Autónoma Metropolitana Unidad Cuajimalpa al seguir los cuatro principios: constructivismo, pensamiento crítico, aprender a aprender y aprendizaje como proceso social. Se busca que el alumnado aprenda las bases para que desarrollen algoritmos, es decir, encuentren soluciones a problemas. Asimismo, utiliza como estrategia didáctica el aprendizaje basado en problemas.

El propósito de este libro es ofrecer, no solo a la juventud lectora sino también al personal docente, un conjunto de problemas de los cuales se describe la solución detalladamente como guía para desarrollar las habilidades que conforman el pensamiento algorítmico, como pensamiento crítico, pensamiento creativo, razonamiento crítico y estructurado y habilidades de comunicación. Para ello, los problemas estudiados van más allá de los clásicos presentados en diversos

libros de algoritmos asociados a la programación, ya que muchos de ellos están basados en situaciones reales y actuales. Hasta donde sabemos, este libro es único en su clase, pues se desprende del componente tecnológico para privilegiar el desarrollo cognitivo.

Relación del contenido con los programas de estudio

El objetivo principal de la UEA Taller de Algoritmos refiere a que el alumnado sea capaz de diseñar soluciones a diversos problemas, utilizando las estructuras básicas de control, sin la necesidad de aprender un lenguaje de programación. Es decir, se busca que el alumnado empiece a desarrollar el pensamiento lógico, abstracto y ordenado para converger en un algoritmo, adquiriendo las habilidades necesarias para, eventualmente, implementar sus algoritmos en algún lenguaje de programación. El contenido de este libro cubre en su totalidad el Programa de Estudios, perteneciente a los Planes de Estudio de la Licenciatura de Ingeniería en Computación y de la Licenciatura de Matemáticas Aplicadas. Sin embargo, la inclusión de temas adicionales y el desarrollo de cada capítulo obedece a la experiencia y a la mejora continua del material didáctico durante los últimos siete años que la UEA se ha impartido.

Objetivo General de la UEA Taller de Algoritmos:

Al final de la UEA el alumnado será capaz de diseñar algoritmos para resolver problemas, mediante diagramas de flujo y pseudocódigo, en donde se utilicen las estructuras básicas de control.

Objetivos parciales:

1. Comprender el concepto de algoritmo y su representación mediante diagramas de flujo y pseudocódigo.
2. Diseñar algoritmos utilizando estructuras secuenciales, selectivas y repetitivas.
3. Describir la solución de diferentes tipos de problemas de manera algorítmica.

Contenido sintético:

1. Elementos básicos y estructuras para el diseño de algoritmos.
2. Estructura de control secuencial.
3. Estructura de control selectiva.
4. Estructura de control iterativa.

Es importante aclarar que, si bien este libro se basa en el contenido de la UEA Taller de Algoritmos, la cual es parte de los Planes de Estudios de la Licenciatura en Ingeniería en Computación y de la Licenciatura en Matemáticas Aplicadas, ambas ofrecidas por la División de Ciencias Naturales e Ingeniería, también es pertinente para otras licenciaturas de la misma División, así como para otras licenciaturas de otras Divisiones y de otras Unidades. Por ejemplo, Ingeniería Biológica y Biología Molecular de la misma División, la Licenciatura en Tecnologías y Sistemas de Información y la Licenciatura en Diseño, de la División de Ciencias de la Comunicación y Diseño, de la UAM Cuajimalpa, la Licenciatura en Ingeniería en Computación y Telecomunicaciones y la Licenciatura en Ingeniería en Sistemas Mecatrónicos Industriales, de la División de Ciencias Básica e Ingeniería, de la Unidad Lerma, y la Ingeniería Química, de la División de Ciencias Básica e Ingeniería, de la Unidad Iztapalapa.

¿Cómo leer este libro?

A lo largo de los diferentes Capítulos, presentamos definiciones y notas importantes, dependiendo de lo que deseamos enfatizar. Por esta razón, clasificamos estas cajas de la siguiente forma.

Cajas naranjas

Definiciones.

Cajas moradas

Información relevante a considerar.

Asimismo, incluimos ejemplos de cada tema estudiado, para los cuales presentamos su resolución detalladamente. También, al final de cada Capítulo, proponemos una serie de ejercicios para que quienes lean este libro los resuelvan y pongan a prueba los conocimientos adquiridos.

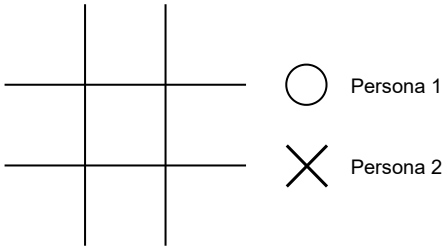
Con el fin de apoyar a la labor docente, queremos también compartir las presentaciones y ejercicios que acompañan al libro y que nos han sido de mucha utilidad para impartir la UEA Taller de Algoritmos. Las presentaciones y series de ejercicios, ambos en formato PDF, están disponibles para descarga en: https://github.com/ic-mx/pensamiento-algoritmico.

1 Algoritmos *desconectados*

Antes de describir el concepto formal de algoritmo, es conveniente introducir a los algoritmos desde el punto de vista de la vida cotidiana. Los algoritmos se suelen asociar directamente con el uso de computadoras; sin embargo, utilizamos algoritmos diariamente para resolver diversos tipos de problemas, como trasladarnos a la Universidad, ir a ver una película al cine, cocinar, inscribirse a un curso, entre muchas otras actividades. Por esta razón, es que el término *algoritmo* no debería sernos ajeno.

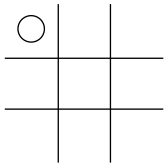
1.1. Juego de gato

Para introducir una primera aproximación del concepto de algoritmo, utilizaremos el ejemplo del juego de gato, o tres en línea, que es ampliamente conocido. En este juego se tiene una rejilla de tres por tres espacios, en donde dos personas que juegan, una representada con el símbolo $\times$ y la otra con $\bigcirc$, buscan colocar su símbolo en tres casillas contiguas, o en línea, antes que la persona oponente. La configuración inicial es dada como sigue:

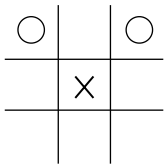


Lo que buscamos, por supuesto, es una manera de ganar siempre que sea posible o, por lo menos, no perder. Para ello, podemos describir unas cuantas instrucciones que, al seguirlas, nos aseguren llegar a nuestra meta. Por ejemplo, usando el símbolo $\bigcirc$, la estrategia para ganar tirando primero sería:

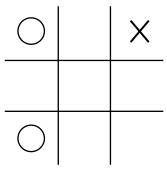
1. Coloca tu primer símbolo en una esquina.



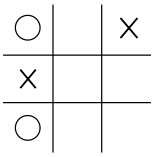
2. Si la otra persona coloca su símbolo en el centro, coloca el tuyo en una esquina contigua y tendrías que esperar a que la otra persona cometa un error.



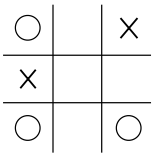
3. En cambio, si la otra persona coloca su símbolo en cualquier otra casilla, ya sea en otra esquina o entre dos esquinas, coloca tu segundo símbolo en alguna de las esquinas contiguas y la partida está prácticamente ganada por ti.



4. La otra persona debería intentar bloquear tu línea.



5. Finalmente, estaría una de las esquinas libres para que coloques tu tercer símbolo, de manera que tengas dos posibles maneras de colocar tu siguiente y último símbolo. ¡Ganaste!



1.2. Estrategia para ganar

Al igual que el ejemplo del juego de gato, se puede decir que un algoritmo es una estrategia para ganar, que permite encontrar una solución a un problema dado.

Algoritmo (primera aproximación)

Un algoritmo describe paso a paso la solución a un problema dado, de tal forma que cualquiera que siga esos pasos pueda resolver el mismo problema. Es decir, es simplemente un método sistemático, o receta, para hacer algo.

Ejemplo 1.1. Para iniciar, escribiremos en español el algoritmo para hacer una llamada telefónica desde un dispositivo móvil inteligente.

1. Sacar al dispositivo móvil del estado de inactividad para poder marcar la contraseña de acceso o que identifique nuestros datos biométricos (huella o rostro).
2. Oprimir el ícono de la función "teléfono" y marcar el número al cual deseamos llamar.
3. Oprimir el ícono de llamar.

Ejemplo 1.2. Receta para hacer magdalenas:

1. Cernir la harina y el polvo para hornear en un tazón.
2. Fundir la mantequilla en una cacerola y dejar enfriar.
3. Obtener la ralladura del limón.
4. Verter los huevos en un recipiente junto con el azúcar y batir durante 5 minutos. Agregar la harina poco a poco, después la mantequilla y la ralladura de limón sin dejar de revolver.
5. Precalentar el horno a 200°C.
6. Untar mantequilla sobre el molde de las magdalenas y verter la mezcla con un espesor de 2 cm.
7. Meter al horno durante 5 minutos a una temperatura de 200°C y después otros 10 minutos a 200°C.
8. Sacar del molde y dejar enfriar.

1.3. Ejercicios

Para cada ejercicio descrito a continuación, escribe un algoritmo en español para resolverlo.

Ejercicio 1.1. Viajar en metro desde una estación a otra.

Ejercicio 1.2. Publicar una foto en la red social de tu preferencia.

Ejercicio 1.3. Buscar un libro en la Biblioteca Digital de la UAM[2].

Ejercicio 1.4. Descargar una aplicación en un dispositivo móvil inteligente.

Ejercicio 1.5. Preparar tu bebida favorita.

[2] Biblioteca Digital. Universidad Autónoma Metropolitana. URL: https://bidi.uam.mx/. Fecha de última visita: 22/09/23.

2 Elementos básicos para el diseño de algoritmos

Los algoritmos pueden ser muy generales, como en los ejemplos anteriores. Al hacer esto damos por sentado una serie de características que describen el contexto y que nos permiten relajar la descripción de los pasos. A lo largo de este capítulo, veremos la importancia de ser específicos al momento de describir instrucciones junto con otras propiedades y cualidades que debe cumplir la descripción de las instrucciones en un algoritmo.

De acuerdo con el diccionario de la lengua española[3], la definición de *algoritmo* es la siguiente.

Algoritmo (segunda aproximación)

Conjunto ordenado y finito de instrucciones que permite hallar la solución a un problema.

De esta definición, podemos apreciar dos propiedades básicas de las instrucciones que se describen en los algoritmos: *ordenadas* y *finitas*.

La primera propiedad, ordenadas, se refiere a que las instrucciones deben darse, describirse o listarse en el orden o secuencia en que se deben llevar a cabo.

Retomemos el Ejemplo 1.1, el algoritmo para hacer una llamada telefónica desde un dispositivo móvil inteligente. Es indispensable que, para oprimir el botón de "llamar", primero debimos haber marcado el número telefónico, ya que no podríamos hacerlo a la inversa.

En cuanto a la segunda propiedad, finitas, se refiere a que, sin importar cuán grande sea el algoritmo, en algún momento debe terminar y devolver o mostrar la solución al problema. En otras palabras, no puede describir instrucciones indefinidamente.

Aunado a estas dos propiedades, las instrucciones deben ser *precisas*, de manera que cualquiera que lea las instrucciones del algoritmo pueda llegar al resultado esperado. Esta propiedad es fácilmente olvidada dado que, por nuestra naturaleza cognitiva, asumimos o damos por hecho que la persona que lee el algoritmo es capaz de entender lo "obvio" y que, por lo tanto, no necesita una mayor explicación o detalle. Pero esto nos lleva a la pregunta ***¿qué es lo obvio?***

[3] algoritmo. Dicionario de la lengua española. URL: https://dle.rae.es/algoritmo. Fecha de última visita: 22/09/2023.

Al existir más de una respuesta, lo obvio se convierte en un objeto de interpretación, que muy posiblemente nos lleve a obtener un resultado erróneo.

Para evitar que las personas que utilicen nuestro algoritmo caigan en errores, debemos acordar la definición de un algoritmo, sus propiedades y la manera en que debemos expresar las instrucciones que lo componen. Evidentemente, no existe un estándar universal, por lo cual seguiremos los acuerdos a continuación descritos.

2.1. Proceso de resolución de un problema

El objetivo principal de un algoritmo es comunicar una manera de solucionar un problema. Lo más importante es especificar claramente lo que se está intentando hacer, en otras palabras, el algoritmo. La característica esencial de los algoritmos es que permiten la realización de procedimientos, gracias a la formalización y a la descripción de secuencias lógicas a un nivel más abstracto y, por tanto, más general.

De lo anterior, podemos identificar varios conceptos a tomar en cuenta durante el diseño de un algoritmo. Sí, un algoritmo es una serie de pasos que, al seguirlos, nos permiten resolver un problema, pero esos pasos deben cumplir con propiedades específicas y deben definir el contexto en el que se desarrollan.

Retomando el Ejemplo 1.2, la receta para hacer magdalenas, podemos fácilmente identificar los componentes de la definición anterior.

Estado inicial Son las condiciones necesarias para que exista y se pueda resolver el problema. En el caso de la receta para hacer magdalenas, suponemos que la persona que preparará las magdalenas tiene un horno a su disposición y los utensilios necesarios, como los moldes o los tazones. Sin esto, aunque quisiera prepararlas, no podría. Podemos ver al estado inicial como el conjunto de restricciones que existen para resolver el problema.

Datos de entrada Es el conocimiento mínimo necesario para poder resolver el problema. En nuestro ejemplo, son los ingredientes de la receta y que, por supuesto, la persona tiene al momento de preparar las magdalenas. Para identificar los datos de entrada podríamos preguntarnos ¿qué debo conocer que es necesario para resolver el problema?

Datos de salida Es el resultado de seguir las instrucciones del algoritmo que representa la solución al problema planteado. Nos debemos preguntar ¿cuáles son las incógnitas o qué es lo que estoy buscando? Al final, los datos de salida son la respuesta a la pregunta que nos planteamos para resolver el problema.

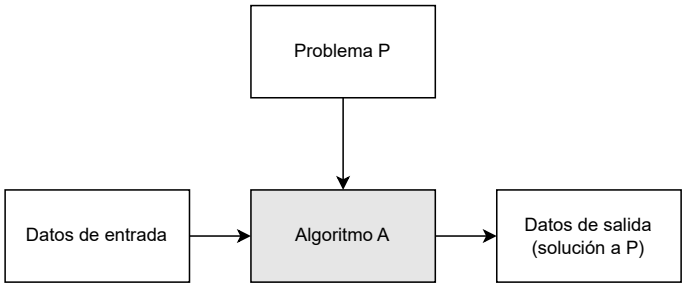


Figura 2.1: Proceso de resolución de un problema.

Ejemplo 2.1. Un trabajador por honorarios tiene un salario por hora y le pagan por horas trabajadas al día. Determinar la cantidad que recibirá el trabajador a la semana.

- **Datos de entrada:** En este caso, la información necesaria para poder determinar la cantidad que recibirá el trabajador es el salario por hora y las horas que trabajó.
- **Datos de salida:** La cantidad que el trabajador efectivamente recibirá como pago.

Ejemplo 2.2. Calcular el área de un triángulo.

- **Datos de entrada:** En este caso, la información necesaria es la base y la altura del triángulo del cual se quiere conocer el área.
- **Datos de salida:** El área del triángulo.

Hablando formalmente, definiremos un algoritmo de la siguiente manera.

Algoritmo

Conjunto ordenado, preciso, claro y finito de instrucciones que permite hallar la solución a un problema de manera sistemática, de tal forma que, dados un estado inicial y los datos de entrada, siguiendo las instrucciones podemos obtener datos de salida, es decir, la solución al problema.

Esta definición la podemos ver de forma esquemática en la Figura 2.1, en donde el Algoritmo A está diseñado para resolver un Problema P, y al cual se le proporcionan datos de entrada. Como resultado de la ejecución del Algoritmo A, obtenemos los datos de salida, consecuentemente, la solución al Problema P.

2.2. Propiedades

Para que podamos considerar una serie de instrucciones como un algoritmo, estas en su conjunto deben cumplir con las siguientes propiedades:

Precisas Se debe especificar cuáles son los datos de entrada y de salida en términos de cómo se representan los datos. Por ejemplo, si se espera que la salida de un algoritmo sea una fecha, ¿cuál será el formato de esa fecha? Se debe dejar muy claro cuáles y cuántos son los datos esperados como entradas o como salidas.

Ordenadas Se deben enlistar las instrucciones conforme la secuencia en que se deben realizar o ejecutar.

Finitas Un algoritmo debe considerar un número finito de pasos, es decir, debe terminar en algún momento.

Claras Las instrucciones se deben expresar de forma clara y que no den pie a ambigüedades sobre las acciones que se deben realizar. Es decir, las instrucciones no deben dar lugar a interpretaciones.

Correctas Un algoritmo debe llegar a la solución adecuada del problema. En un ejemplo general, cuando estamos horneando un pastel de chocolate, no vamos a obtener un pastel de fresa al final. Este punto es sumamente importante porque, aunque las instrucciones cumplan con las propiedades anteriores, si el resultado no es correcto, en consecuencia, el algoritmo tampoco lo será.

Cada una de estas propiedades se ejemplificarán más adelante, de tal manera que se consideren todos los términos involucrados en la conceptualización y en el diseño de un algoritmo.

2.3. Prueba y depuración de un algoritmo

Una vez que hemos descrito los pasos a seguir para resolver un problema, naturalmente surge la pregunta **¿cómo sé si mi solución realmente resuelve el problema?** Por el momento, el único mecanismo que tenemos para verificar que nuestros algoritmos son correctos es la *prueba*. Probar un algoritmo consiste en:

- Considerar un caso conocido del problema, es decir, para un estado inicial y datos de entrada dados, cuál debería ser la salida (resultado).
- Seguir los pasos tal y como se describen en el algoritmo.
- Comparar si el resultado obtenido es igual al resultado esperado.

Por ejemplo, si estamos horneando un pastel de chocolate, al finalizar los pasos de la receta debemos obtener un pastel de chocolate. En caso de obtener un resultado distinto al esperado, se considera que el algoritmo es incorrecto.

Si no obtuvimos el resultado esperado, debemos identificar la razón: puede ser que hacen falta instrucciones, que alguna de las instrucciones no es la correcta, el orden de las instrucciones es incorrecto o que una de las instrucciones es ambigua, entre otras.

Una vez que hemos identificado la razón por la cual no obtuvimos el resultado esperado, debemos *depurar* el algoritmo. De acuerdo con el diccionario de la lengua española[4], una de las acepciones de *depurar* es: Limpiar, purificar. En otras palabras y en nuestro contexto, podemos dar la siguiente definición.

Depurar

Corregir los errores y las inconsistencias en los algoritmos.

El proceso de prueba y depuración se debe repetir hasta que nuestro algoritmo obtenga los resultados que se esperan para los casos conocidos del problema. Es importante recalcar que un algoritmo debe funcionar para todo el conjunto de casos de un problema y no solamente para un subconjunto de ellos. Por ejemplo, la fórmula para calcular el área de un triángulo nos permite saber el área de un triángulo de cualesquiera longitudes y no solo para algunas longitudes específicas.

2.4. Diccionario y gramática

Al utilizar nuestro idioma como medio de expresión de los algoritmos, muchas veces obviamos u omitimos detalles que nos parecen irrelevantes, generalmente porque pensamos directamente en un contexto específico. Como mencionamos en las Secciones 2.1 y 2.2, al diseñar una solución a un problema y expresarla de alguna forma, debemos asegurarnos de que cualquier persona que siga los pasos descritos pueda resolver el problema. En muchas ocasiones, para asegurarnos de que no existan ambigüedades, necesitaremos usar un *diccionario*. El diccionario de la lengua española[5] define *diccionario* como: Repertorio en forma de libro o en soporte electrónico en el que se recogen, según un orden determinado, las palabras o expresiones de una o más lenguas, o de una materia concreta, acompañadas de su definición, equivalencia o explicación.

[4] depurar. Diccionario de la lengua española. URL: https://dle.rae.es/depurar. Fecha de última visita: 22/09/2023

[5] diccionario. Diccionario de la lengua española. URL: https://dle.rae.es/diccionario. Fecha de última visita: 22/09/2023.

Diccionario

Agrupación de palabras, expresiones o símbolos de una materia concreta con su respectiva definición, equivalencia o explicación.

Ejemplo 2.3. Vamos a suponer que colaboras para ACME Satellites, Inc. En días pasados, el satélite Quetzalcóatl I salió de órbita y te piden que traces una trayectoria para que regrese a orbitar la Tierra. Para esto, el Quetzalcóatl I tiene definidas las siguientes instrucciones:

Avanzar: Avanza exactamente dos unidades hacia el frente.

Girar: Gira 90° hacia la derecha.

Regresar: Retrocede exactamente una unidad hacia la dirección opuesta del frente.

Utilizando secuencias de estas tres instrucciones, debes escribir un algoritmo para que el Quetzalcóatl I regrese a orbitar la Tierra dado el escenario que se muestra en la Figura 2.2. Toma en cuenta que los asteroides impiden que el Quetzalcóatl I avance.

Ahora, observemos que tenemos un conjunto de operaciones que puede llevar a cabo el Quetzalcóatl I, el cual representa un diccionario. Es decir, tenemos cuáles son las palabras o expresiones válidas para proponer una trayectoria para el Quetzalcóatl I y, además, existe una definición concreta del significado de cada instrucción. Por ejemplo, en el caso de la instrucción **Avanzar**, el Quetzalcóatl I avanzará exactamente dos unidades, no más y no menos, y tampoco es necesario decir en que dirección debe avanzar, puesto que en la definición está especificado que avanzará hacia el frente.

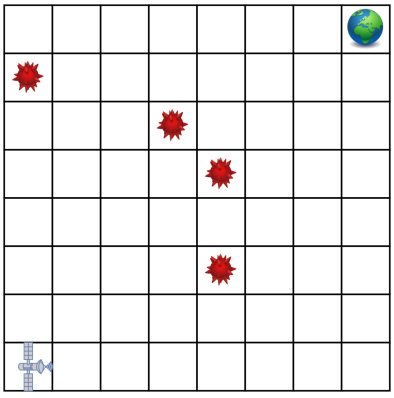


Figura 2.2: Escenario del Quetzalcóatl I.

Por otro lado, el diccionario de la lengua española[6] da como una acepción de *gramática*, la siguiente: Parte de la lingüística que estudia los elementos de una lengua, así como la forma en que estos se organizan y se combinan. Sin embargo, en nuestro caso, adoptaremos la siguiente definición.

Gramática

Arte de hablar y escribir correctamente un idioma.

Es decir, tiene que ver con la propiedad de claridad de los algoritmos. Retomando el Ejemplo 2.3, debemos utilizar las instrucciones del Quetzalcóatl I tal cual están dadas y seguir las reglas. Por lo tanto, si al estar trazando una trayectoria escribimos las instrucciones "Avanzar hacia atrás" o "Girar hacia la izquierda", estas no cumplen con las reglas del Quetzalcóatl I, por lo que generarían confusión y, lo más importante, ¡el algoritmo no sería correcto!

2.5. Ejercicios

Para cada uno de los siguientes ejercicios, especifica cuáles son los datos de entrada y cuáles son los datos de salida.

Ejercicio 2.1. Calcular el promedio de los tres números a, b, c.

Ejercicio 2.2. Receta para cocinar arroz blanco.

Ejercicio 2.3. Cambiar la llanta de un carro.

Ejercicio 2.4. Comprar comida en el comedor de la UAM Cuajimalpa.

Ejercicio 2.5. Comprar comida en el comedor de la UAM Lerma.

Para los siguientes ejercicios, que ya resolviste en el Capítulo 1, aplica los conceptos de las propiedades de los algoritmos, de prueba y depuración, y de diccionario y gramática.

Ejercicio 2.6. Viajar en metro desde una estación a otra.

Ejercicio 2.7. Publicar una foto en la red social de tu preferencia.

[6] gramática. Diccionario de la lengua española. URL: https://dle.rae.es/gramático. Fecha de última visita: 22/09/2023.

Ejercicio 2.8. Buscar un libro en la Biblioteca Digital de la UAM[7].

Ejercicio 2.9. Descargar una aplicación para un dispositivo móvil inteligente.

Ejercicio 2.10. Preparar tu bebida favorita.

[7] Biblioteca Digital. Universidad Autónoma Metropolitana. URL: https://bidi.uam.mx/. Fecha de última visita: 22/09/23.

3 Convenciones y expresiones

Si bien es cierto que podemos expresar algoritmos en un lenguaje coloquial, en forma de recetas o de pictogramas, es imperativo que las instrucciones descritas en un algoritmo cumplan cabalmente las propiedades descritas en el Capítulo anterior. Expresar los algoritmos de tal forma que cualquier persona que los lea y los siga llegue al mismo resultado que la persona que lo escribió, es primordial para decir que un algoritmo es correcto. Como mencionamos, para este objetivo es necesario seguir un estándar que permita esta propiedad.

El lenguaje algorítmico es una convención que permite expresar a una persona la idea o los pasos para la resolución de un problema. Asimismo, permite probar la solución y analizar su complejidad. A continuación, presentaremos las convenciones básicas para la construcción y representación de los algoritmos que utilizaremos en este libro.

3.1. Datos, tipos de datos y rango de datos

Nuestra definición de algoritmo habla de datos de entrada y de datos de salida, pero ¿qué son los datos? Los *datos* son símbolos que representan objetos, eventos y propiedades de observaciones del mundo real hechas por personas o instrumentos. Por ejemplo, el alfabeto tiene símbolos que están asociados a sonidos con los cuales se pueden formar palabras. Otro ejemplo son las notas musicales que se convierten en sonidos al interpretarlas con algún instrumento.

En nuestra vida cotidiana usamos diferentes tipos de datos de diversas formas. El número 35, por ejemplo, puede representar la edad de una persona o el número de horas que alguien trabajó en una semana y el número 22.5 puede ser la temperatura actual o el precio de una bebida. En estos casos, tanto 35 como 22.5 son datos, pero de diferente tipo.

Para la construcción básica de algoritmos, podemos utilizar diferentes tipos de datos, cuya definición dependerá de lo que se desee representar para la solución de un problema. Así, el *tipo de datos* es un atributo de los datos que indica la clase de datos que se va a manejar. Esto incluye imponer algunas restricciones como:

- ¿Qué valores pueden tomar los datos?
- ¿Qué operaciones se pueden realizar sobre los datos?

Algunos tipos de datos que se utilizan recurrentemente son:

Entero	Son todos los números enteros, positivos y negativos, sin parte fraccionaria. Ejemplos: -57, 0, 22, 109, 123,543.
Real	Son los números decimales, es decir, positivos y negativos que consideran una parte fraccionaria. Ejemplos: -197.3333, 0.0, 3.1416, 27.8, 3,257.89
Secuencia de caracteres	Son los símbolos alfanuméricos que incluyen letras, números y carácteres especiales. Ejemplos: "a", "B", "5", "?", "Karen", "Emiliano", "Emma".
Lógico	Son aquellos datos que solo pueden tener uno de los dos valores representados por el álgebra de Bool: **Verdadero** (**V**) y **Falso** (**F**).

Como veremos más adelante, el tipo de dato también define el conjunto de operaciones o acciones que se pueden aplicar a esos datos.

Es conveniente también definir el *rango* de datos, el cual engloba a un subconjunto de valores del tipo de datos que son válidos para un dato. Por ejemplo, la edad de una persona se puede representar como un tipo de datos entero, pero no hay edades negativas y tampoco puede exceder por mucho a 100.

Un ejemplo de datos, tipo de datos y rango de datos, es "el color distintivo de las Unidades de la UAM". Los valores válidos de los colores que se puede utilizar son azul, morado, naranja, rojo y verde, y aunque existen más colores, únicamente estos cinco están asociados a las Unidades de la UAM, por lo que el color café no es válido. De hecho, los colores distintivos de las Unidades no son de cualquier tonalidad, los colores corresponden a las Reglas de Aplicación de los Elementos de Identidad Institucional[8].

3.2. Identificadores

Los identificadores se utilizan en los algoritmos para denominar a los datos o valores de manera general. Dichos identificadores son parecidos a las literales en el álgebra, que son letras que representan números. Por ejemplo:

[8] Reglas de Aplicación de los Elementos de Identidad Institucional. Universidad Autónoma Metropolitana. URL: https://www.comunicacionsocial.uam.mx/identidaduam/html/lineamientos/6-0-colores.html. Fecha de última visita: 22/09/2023

Aritmética $8 + 12 = 20$

Álgebra a + b = c, en donde a = 8, b = 12 y c = 20

Algoritmos $total \leftarrow costo1 + costo2$, en donde $costo1 \leftarrow 8$ y $costo2 \leftarrow 12$

Dado que los identificadores son nombres simbólicos para la representación de los datos, se recomienda que estén asociados a las características del dato.

Nombre de identificadores

El identificador ss puede representar el salario semanal de un empleado, pero no asocia ninguna característica del dato. Por otro lado, $SalarioSemanal$ nos da una idea inmediata del dato que representa. Algunas buenas prácticas para nombrar identificadores son las siguientes:

- Debe comenzar con una letra (mayúscula o minúscula).
- No debe contener caracteres especiales.
- Nombres cortos.
- Debe guardar relación con el dato asociado.

Ejemplo 3.1. Representar el dato que da respuesta a la pregunta **¿cuál es tu nombre?**

- **Identificador**: $nombre$, $nombre\_pila$, $PrimerNombre$, son algunos ejemplos.
- **Tipo de dato**: Secuencia de caracteres alfanuméricos.
- **Rango**: [cualquier nombre que exista en el mundo].

Ejemplo 3.2. Representar el dato **color distintivo de alguna unidad de la UAM.**

- **Identificador**: $ColorUnidad$, $color\_unidad$, por ejemplo.
- **Tipo de dato**: Secuencia de caracteres alfanuméricos.
- **Rango**: [“azul”, “morado”, “naranja”, “rojo”, “verde”].
- **Diccionario**: El que se muestra en la Tabla 3.1.

Color	Unidad	Pantone	CMYK	RGB	Hexadecimal
	Azcapotzalco	186 C	10, 100, 84, 3	205, 3, 46	#CD032E
	Cuajimalpa	144 C	0, 60, 100, 0	240, 130, 0	#F08200
	Iztapalapa	369 C	70, 5, 100, 0	87, 165, 25	#57A519
	Lerma	253 C	45, 90, 0, 0	173, 37, 168	#AD25A8
	Xochimilco	285 C	85, 50, 0, 0	0, 114, 206	#0072CE

Tabla 3.1: Colores distintivos de las Unidades de la UAM.

Identificadores vs. secuencia de caracteres

Para evitar confusiones entre los nombres de los identificadores y las secuencias de caracteres, consideraremos que todas las secuencias de caracteres se escriben entre comillas, por ejemplo, "unidad", mientras que para los identificadores utilizaremos una tipografía itálica, por ejemplo, $unidad$.

3.3. Asignación

La asignación es la operación de dar un valor específico a un identificador dado. La asignación se representa mediante una flecha apuntando hacia la izquierda:

$$\leftarrow$$

En general, la forma de representar la asignación de un valor a un identificador es como sigue:

$$identificador \leftarrow \textbf{valor}$$

Es decir, un **valor** se asigna ($\leftarrow$) a un $identificador$.

Ejemplo 3.3. Vamos a describir el significado de las siguientes asignaciones:

$edad \leftarrow 18$

Al identificador $edad$ se le asigna el valor 18. Esto es significa que ahora $edad$ tiene un valor de 18 o $edad$ representa el valor 18.

$suma \leftarrow 5 + 10$

Al identificador $suma$ se le asigna el resultado de sumar 5 más 10, por lo que ahora $suma$ tiene un valor de 15.

Es importante destacar que toda asignación es de naturaleza *destructiva*. Esto significa que cada vez que se asigna un nuevo valor a un identificador, se *borra* cualquier valor asignado anteriormente al mismo identificador.

Ejemplo 3.4. Supongamos que asignamos secuencialmente los siguientes valores al identificador $edad$:

$edad \leftarrow 18$
$edad \leftarrow 21$
$edad \leftarrow 26$

El valor final del identificador $edad$ es 26, ya que los valores 18 y 21 han sido destruidos.

Ejemplo 3.5. Para las siguientes preguntas, definiremos un identificador con un nombre apropiado y le asignaremos el valor que responda a la pregunta.

¿Cuál es tu nombre?
$nombre \leftarrow$ "Emiliano"

¿Cuál es tu edad?
$edad \leftarrow 12$

¿Cuándo es tu cumpleaños?
$cumple \leftarrow$ "18-06"
o
$cumple \leftarrow$ "18/06"
o
$cumple \leftarrow$ "18 de junio"

En este último caso, los valores asignados a $cumple$ son una secuencia de caracteres que representan una fecha, por lo que el rango de datos deberá ser una secuencia de caracteres válida para una fecha.

3.4. Expresiones

Una expresión es una combinación de operadores y operandos que, al actuar los primeros sobre los segundos, arroja un resultado. Así como al estudiar el español aprendimos a construir oraciones para comunicarnos, las expresiones nos permiten transmitir las instrucciones del lenguaje habitual al lenguaje algorítmico.

Los operandos pueden ser identificadores u otras expresiones. Por otro lado, los operadores son símbolos que indican determinadas acciones sobre los operandos. Existen varios tipos de operadores: aritméticos, relacionales y lógicos.

Es importante señalar que, el resultado de una expresión generalmente se asigna a un identificador para su uso posterior. Estas asignaciones siguen las reglas que se expusieron anteriormente. La asignación está directamente ligada a los tipos de datos, esto es debido a que a los identificadores únicamente se les pueden asignar valores del mismo tipo.

Ejemplo 3.6. Consideremos que el identificador $resultado$ almacena datos de tipo entero. Veamos qué sucede cuando intentamos hacer las siguientes asignaciones.

$resultado \leftarrow 98765$

Es una asignación correcta, porque se le está asignando directamente un número entero.

$resultado \leftarrow 9876 - 543$

Es una asignación correcta, porque el resultado de la resta es un número entero.

$resultado \leftarrow$ "18 de junio"

Es una asignación incorrecta, puesto que se intenta asignar un dato de tipo secuencia de caracteres a un identificador de tipo entero.

3.4.1. Expresiones aritméticas

Una expresión aritmética está compuesta por operadores aritméticos que actúan sobre operandos numéricos, es decir, tipos de datos enteros o reales, de la cual se obtiene un resultado numérico.

Los operadores aritméticos que utilizaremos por convención son:

Símbolo	Operación
+	Suma
−	Resta
*	Multiplicación
/	División
`MOD`	Módulo

Ejemplo 3.7. Algunas expresiones aritméticas son las siguientes:

Español	Notación matemática	Expresión aritmética
El doble de un número	$2x$	$2 * x$
La mitad de un número	$\frac{x}{2}$	$x/2$
Longitud de una circunferencia	$2\pi r$	$2 * pi * r$
Área de un triángulo	$\frac{bh}{2}$	$(b * h)/2$
Ecuación de segundo grado	$ax^2 + bx + c$	$a * x * x + b * x + c$
Conversión de horas y minutos a segundos	$3600h + 60m$	$3600 * h + 60 * m$
El triple de un número menos dos	$3x - 2$	$3 * x - 2$
La mitad de un número menos cinco	$\frac{x-5}{2}$	$(x - 5)/2$

En los dos últimos ejemplos anteriores, la redacción es ambigüa, porque el triple de un número menos dos también puede ser $3 * (x - 2)$ y la mitad de un número menos cinco también se puede expresar como $(x/2) - 5$. De hecho, esta es una de las razones por las que preferimos utilizar expresiones en lugar del espanõl.

Divisiones y módulo

Es importante destacar los casos de la división y el módulo. La división (/), en general, puede ser entre dos operandos de tipo real, entero o una combinación de ellos, cuyo resultado dependerá del tipo de datos de los operandos sobre los que actúa. En cambio, el módulo (`MOD`) es el residuo o resto que se obtiene de una división entera, por lo que siempre actúa entre operandos de tipo entero.

Las operaciones aritméticas se evalúan de acuerdo con la siguiente precedencia:

1. ( )
2. $*$, /, `MOD`
3. +, −

Esta precedencia nos indica que, primero, se deben evaluar los operadores que aparecen dentro de un par de paréntesis. Después, se evalúan los operadores de multiplicación, división y módulo. Finalmente, se evaluarán las sumas y las restas.

Propiedad asociativa

Es importante considerar que, por la propiedad asociativa por la izquierda, cuando en una expresión aritmética existen operadores con la misma precedencia, éstos se evalúan en el orden en que aparecen de izquierda a derecha.

A continuación, se presentan algunos ejemplos de evaluación de expresiones aritméticas. En cada línea, el operador aritmético que tiene precedencia es el que se muestra en color anaranjado y el valor resaltado en **negritas y subrayado** es el resultado de evaluar la operación que tenía precedencia en la línea anterior.

Ejemplo 3.8. Evaluemos la siguiente expresión aritmética para encontrar el valor que se asignará a f.

$$
\begin{aligned}
f &\leftarrow (3+2+1+8+4)/6 \\
f &\leftarrow (3+2+1+8+4)/6 \\
f &\leftarrow (\underline{\mathbf{5}}+1+8+4)/6 \\
f &\leftarrow (\underline{\mathbf{6}}+8+4)/6 \\
f &\leftarrow (\underline{\mathbf{14}}+4)/6 \\
f &\leftarrow \underline{\mathbf{18}}\ /\ 6 \\
f &\leftarrow \underline{\mathbf{3}}
\end{aligned}
$$

Ejemplo 3.9. Evaluemos la siguiente expresión aritmética para encontrar el valor que se asignará a f.

$$
\begin{aligned}
f &\leftarrow 3 * (2+1) + (8+4)/6 \\
f &\leftarrow 3 * (2+1) + (8+4)/6 \\
f &\leftarrow 3 * \underline{\mathbf{3}} + (8+4)/6 \\
f &\leftarrow 3 * 3 + (8+4)/6 \\
f &\leftarrow 3 * 3 + \underline{\mathbf{12}}/6 \\
f &\leftarrow \underline{\mathbf{9}} + 12 \;/\; 6 \\
f &\leftarrow 9 + \underline{\mathbf{2}} \\
f &\leftarrow \underline{\mathbf{11}}
\end{aligned}
$$

3.4.2. Expresiones relacionales

De forma similar a las expresiones aritméticas, las expresiones relacionales manifiestan, como su nombre lo indica, una *relación* entre los operandos y cuyo resultado es un valor *lógico*, es decir, **Falso** o **Verdadero**. Los operadores relacionales son los siguientes:

Símbolo	Operación
$=$	Igual que
$\neq$	Diferente que
$<$	Menor que
$\leq$	Menor o igual que
$>$	Mayor que
$\geq$	Mayor o igual que

Los operadores relacionales son binarios, es decir, comparan dos operandos, el que está a la izquierda del operador con el que está a la derecha, para definir el resultado. Para que se pueda establecer un relación entre los dos operandos, estos deben pertenecer al mismo rango de datos, de otra forma, la expresión no será válida y no se podrá evaluar. Estos operadores también tienen una precedencia establecida y es la siguiente:

1. ( )
2. $<$, $\leq$, $>$, $\geq$
3. $=$, $\neq$

Al igual que con los operadores aritméticos, primero se evaluarán los operadores que estén dentro de un par de paréntesis. Después, todos los operadores menor que, menor o igual que, mayor que y mayor o igual que. Al final, se evaluarán los operadores igual que y diferente que.

De forma similar que en las expresiones aritméticas, cuando en una expresión existen operadores relacionales con la misma precedencia, se aplica la propiedad asociativa por la izquierda y éstos se evalúan en el orden en que aparecen de izquierda a derecha.

Ejemplo 3.10. Consideremos la siguiente tabla, en donde los identificadores a y b toman valores de diferentes tipos de datos, los cuales son comparados con los operadores relacionales.

Tipo	a	b	$a = b$	$a \neq b$	$a < b$	$a \leq b$	$a > b$	$a \geq b$
entero	2	3	**F**	**V**	**V**	**V**	**F**	**F**
real	4.4	4.4	**V**	**F**	**F**	**V**	**F**	**V**
carácter	"y"	"x"	**F**	**V**	**F**	**F**	**V**	**V**
carácter	"15/03"	"14-09"	**F**	**V**	**V**	**V**	**F**	**F**
	"rojo"	3.1416	*operandos de diferente tipo*, ¡opearción inválida!					

En el tercer caso, los identificadores a y b toman valores de secuencia de caracteres, "y" y "x", respectivamente. En esta caso, la relación que podemos considerar es un orden lexicográfico (alfabético), es decir, "x" antes que "y".

En el cuarto caso, los identificadores también toman valores de secuencia de caracteres, pero podemos pensar que estas secuencias representan fechas, por lo que la relación está basada en un orden temporal y, como estas secuencias de caracteres representan fechas válidas, 15 de marzo para la primera y 14 de septiembre para la segunda, las expresiones se pueden evaluar.

En el último caso, a tiene asignado una cadena de caracteres, que posiblemente representa el color rojo, y b tiene asignado un número real. Es evidente que, como estos identificadores tienen asignados datos de diferentes tipos, no se puede establecer una relación entre ellos, por lo que la expresión no se puede evaluar y, en consecuencia, no es una expresión válida.

A continuación, se presenta un ejemplo de evaluación de expresiones relacionales. En cada línea, el operador relacional que tiene precedencia es el que se muestra en color anaranjado y el valor resaltado en **negritas y subrayado** es el resultado de evaluar la operación que tenía precedencia en la línea anterior.

Ejemplo 3.11. Evaluemos la siguiente expresión relacional para encontrar el valor que se asignará a f.

$$
\begin{aligned}
f &\leftarrow 4 < 5 = (\mathbf{F} \neq 8 \geq 5) \\
f &\leftarrow 4 < 5 = (\mathbf{F} \neq 8 \geq 5) \\
f &\leftarrow 4 < 5 = (\mathbf{F} \neq \underline{\mathbf{V}}) \\
f &\leftarrow 4 < 5 = \underline{\mathbf{V}} \\
f &\leftarrow \underline{\mathbf{V}} = \mathbf{V} \\
f &\leftarrow \underline{\mathbf{V}}
\end{aligned}
$$

Es importante mencionar que los operadores aritméticos tienen precedencia sobre los operadores relacionales. Es decir, si en una expresión aparecen operadores aritméticos y relacionales, primero se evalúan las operaciones aritméticas y después se evalúan los operadores relacionales, de acuerdo con la precedencia de ambos tipos de operadores y considerando la propiedad asociativa por la izquierda.

Ejemplo 3.12. Evaluemos la siguiente expresión, que contiene operadores aritméticos y relacionales, considerando $a \leftarrow 6, b \leftarrow 1, x \leftarrow 3, y \leftarrow 2$:

$$
\begin{aligned}
f &\leftarrow a + b - 1 < x * y \\
f &\leftarrow 6 + 1 - 1 < 3 * 2 \\
f &\leftarrow 6 + 1 - 1 < \underline{\mathbf{6}} \\
f &\leftarrow \underline{\mathbf{7}} - 1 < 6 \\
f &\leftarrow \underline{\mathbf{6}} < 6 \\
f &\leftarrow \underline{\mathbf{F}}
\end{aligned}
$$

3.4.3. Expresiones lógicas

Una expresión lógica es un conjunto de operadores *lógicos* actuando sobre operandos de tipo lógico. Estas expresiones arrojan un resultado lógico, es decir, **Verdadero** (**V**) o **Falso** (**F**). Los operadores lógicos son los siguientes:

Símbolo	Operación
$\wedge$	**Y**
$\vee$	**O**
$\neg$	**NO**

El operador **Y** lógico ($\wedge$) arrojará un resultado **Verdadero** cuando ambos operandos sean **Verdadero**. El resultado de la operación **O** lógica ($\vee$) será **Verdadero** cuando alguno de los operandos o ambos tomen el valor **Verdadero**. El operador **NO** lógico ($\neg$) es unario, es decir, solo actúa sobre un operando, cambiando el valor de este. La siguiente tabla muestra la definición de estos operadores.

a	b	$a \wedge b$	$a \vee b$	$\neg a$	$\neg b$
F	**F**	**F**	**F**	**V**	**V**
F	**V**	**F**	**V**	**V**	**F**
V	**F**	**F**	**V**	**F**	**V**
V	**V**	**V**	**V**	**F**	**F**

La precedencia de los operadores lógicos es la que se muestra a continuación:

1. ( )
2. $\neg$
3. $\wedge$
4. $\vee$

De forma similar a los operadores aritméticos y relacionales, primero se evaluarán todos las operadores que estén dentro de un par de paréntesis. Después, se evaluarán las operaciones **NO**, **Y** y **O**, en ese orden.

Al igual que en las expresiones aritméticas y relacionales, cuando en una expresión existen operadores lógicos con la misma precedencia, se aplica la propiedad asociativa por la izquierda y éstos se evalúan en el orden en que aparecen de izquierda a derecha.

A continuación, se presentan unos ejemplos de evaluación de expresiones lógicas. En cada línea, el operador lógico que tiene precedencia es el que se muestra en color anaranjado y el valor resaltado en **negritas y subrayado** es el resultado de evaluar la operación que tenía precedencia en la línea anterior.

Ejemplo 3.13. Evaluemos la siguiente expresión lógica considerando $p \leftarrow \mathbf{F}$, $q \leftarrow \mathbf{V}$, $s \leftarrow \mathbf{F}$:

$$
\begin{aligned}
f &\leftarrow \neg(p \vee q) \wedge s \\
f &\leftarrow \neg(\mathbf{F} \vee \mathbf{V}) \wedge \mathbf{F} \\
f &\leftarrow \neg(\mathbf{F} \vee \mathbf{V}) \wedge \mathbf{F} \\
f &\leftarrow \neg\, \underline{\mathbf{V}} \wedge \mathbf{F} \\
f &\leftarrow \underline{\mathbf{F}} \wedge \mathbf{F} \\
f &\leftarrow \underline{\mathbf{F}}
\end{aligned}
$$

Ejemplo 3.14. Evaluemos la siguiente expresión lógica considerando $p \leftarrow \mathbf{F}$, $q \leftarrow \mathbf{V}$, $s \leftarrow \mathbf{F}$:

$$
\begin{aligned}
f &\leftarrow \neg p \vee (q \wedge s) \\
f &\leftarrow \neg \mathbf{F} \vee (\mathbf{V} \wedge \mathbf{F}) \\
f &\leftarrow \neg \mathbf{F} \vee (\mathbf{V} \wedge \mathbf{F}) \\
f &\leftarrow \neg \mathbf{F} \vee \underline{\mathbf{F}} \\
f &\leftarrow \underline{\mathbf{V}} \vee \mathbf{F} \\
f &\leftarrow \underline{\mathbf{V}}
\end{aligned}
$$

La precedencia de todos los operadores, aritméticos, relacionales y lógicos, es la siguiente:

1. ( )
2. $*$, /, `MOD`
3. $+, -$
4. $<, \leq, >, \geq$
5. $=, \neq$
6. $\neg$
7. $\wedge$
8. $\vee$

Es decir, primero se evalúan las operaciones dentro de un par de paréntesis, a continuación los operadores artiméticos, luego los relacionales y, al final, los operadores lógicos, considerando la precedencia dentro de cada tipo de operador.

Ejemplo 3.15. Evaluemos la siguiente expresión, en donde intervienen los tres tipos de operadores, para $x \leftarrow 3, y \leftarrow 6$:

$$
\begin{aligned}
f &\leftarrow \neg(x+1<5) \wedge \neg(y-1 \geq 7) \\
f &\leftarrow \neg(3+1<5) \wedge \neg(6-1 \geq 7) \\
f &\leftarrow \neg(3+1<5) \wedge \neg(6-1 \geq 7) \\
f &\leftarrow \neg(\underline{4}<5) \wedge \neg(6-1 \geq 7) \\
f &\leftarrow \neg\underline{\mathbf{V}} \wedge \neg(6-1 \geq 7) \\
f &\leftarrow \neg\mathbf{V} \wedge \neg(6-1 \geq 7) \\
f &\leftarrow \neg\mathbf{V} \wedge \neg(\underline{5} \geq 7) \\
f &\leftarrow \neg\mathbf{V} \wedge \neg\underline{\mathbf{F}} \\
f &\leftarrow \underline{\mathbf{F}} \wedge \neg\mathbf{F} \\
f &\leftarrow \mathbf{F} \wedge \underline{\mathbf{V}} \\
f &\leftarrow \underline{\mathbf{F}}
\end{aligned}
$$

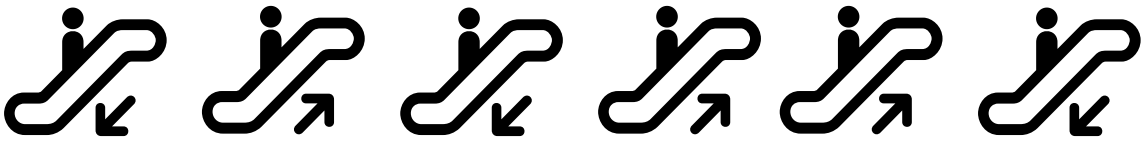
Ejemplo 3.16. Evaluemos la siguiente expresión, en donde intervienen los tres tipos de operadores, para $x \leftarrow 3, y \leftarrow 6$:

$$
\begin{aligned}
f &\leftarrow \neg((y-2>4) \vee (x+2 \leq 4)) \\
f &\leftarrow \neg((6-2>4) \vee (3+2 \leq 4)) \\
f &\leftarrow \neg((6-2>4) \vee (3+2 \leq 4)) \\
f &\leftarrow \neg((6-2>4) \vee (3+2 \leq 4)) \\
f &\leftarrow \neg((\underline{\mathbf{4}}>4) \vee (3+2 \leq 4)) \\
f &\leftarrow \neg(\underline{\mathbf{F}} \vee (3+\ 2 \leq 4)) \\
f &\leftarrow \neg(\mathbf{F} \vee (3+2 \leq 4)) \\
f &\leftarrow \neg(\mathbf{F} \vee (\underline{\mathbf{5}} \leq 4)) \\
f &\leftarrow \neg(\mathbf{F} \vee \underline{\mathbf{F}}) \\
f &\leftarrow \neg\ \underline{\mathbf{F}} \\
f &\leftarrow \underline{\mathbf{V}}
\end{aligned}
$$

3.5. Ejercicios

Ejercicio 3.1. Observa tu credencial de la Universidad. Enlista los datos y sus correspondientes tipos de datos que puedas identificar. Escribe los identificadores para representar cada dato.

Ejercicio 3.2. Observa la siguiente figura. Se puede ver que hay personas que están subiendo y bajando las escaleras como indican las flechas. Escribe cómo y con qué tipo de datos representarías si la persona está subiendo o bajando.



Ejercicio 3.3. Encuentra el valor asignado a f en las siguientes expresiones aritméticas, considerando $a \leftarrow 4, b \leftarrow 5, c \leftarrow 1$:

1. $f \leftarrow b * a - b * b/5 * c$
2. $f \leftarrow a * b/3 * 3$
3. $f \leftarrow (((b+c)/2 * a + 10) * 3 * b) - 6$

Ejercicio 3.4. Escribe la expresión aritmética que representa a cada enunciado siguiente:

1. Hace 10 años, la edad del padre era el triple de la edad de su hijo y hoy es el doble. ¿Cuáles son las edades actuales del padre y del hijo?
2. Juan le dice a Pedro: "si me das una oveja, voy a tener el doble de ovejas que tú". Pedro le contesta: "no seas tan listo, dámela tú a mí y así los dos tendremos igual número de ovejas". ¿Cuántas ovejas tiene cada uno?

Ejercicio 3.5. Obtener el valor de un número elevado a una potencia dada.

Ejercicio 3.6. Encuentra el valor asignado a f en las siguientes expresiones relacionales:

1. $f \leftarrow 7 < 5 = (\mathbf{V} = 5 \geq 5)$
2. $f \leftarrow (2 \geq 1 = \mathbf{V}) \neq 9 \geq 10$
3. $f \leftarrow 3 > 4 = 9 < 2 \neq 6 < 8$

Ejercicio 3.7. Evalúa las siguientes expresiones, que contienen operadores aritméticos y relacionales, para los valores $a \leftarrow 4$, $b \leftarrow 2$, $x \leftarrow 1$, $y \leftarrow 3$:

1. $f \leftarrow a * b \geq x * y + 2$
2. $f \leftarrow y - x - 2 \leq a - b$
3. $f \leftarrow a/b + x * y \neq 5$

Ejercicio 3.8. Evalúa las siguientes expresiones lógicas para los valores $a \leftarrow \mathbf{V}$, $b \leftarrow \mathbf{F}$, $x \leftarrow \mathbf{V}$, $y \leftarrow \mathbf{F}$:

1. $f \leftarrow (a \vee b) \wedge (x \vee y)$
2. $f \leftarrow a \vee b \wedge x \vee y$
3. $f \leftarrow a \wedge b \vee x \wedge y$

Ejercicio 3.9. Escribe la expresión lógica que representa a cada enunciado siguiente:

1. Los lados a, b y c de un triángulo equilátero tienen la misma longitud.
2. El horario de visita del Zoológico de Chapultepec es de 9:00 a 16:30 horas.
3. El Zoológico de Chapultepec cierra sus puertas los días lunes.

Ejercicio 3.10. Evalúa las siguientes expresiones, que combinan los tres tipos de operadores, para los valores $a \leftarrow 6$, $b \leftarrow 4, c \leftarrow 2$:

1. $f \leftarrow \neg(a - b = c) \vee (c \neq 0) \wedge (b + c \geq 3)$
2. $f \leftarrow (a > b + c) \wedge (a - c > 0) \vee (b < 3 - a)$
3. $f \leftarrow (a > b - c) \vee (a - c < 0) \wedge (b < 3 + a)$

Ejercicio 3.11. Escribe la expresión lógica que representa a cada enunciado siguiente:

1. Los lados a, b y c de un triángulo escaleno tienen diferente longitud.
2. Para que una persona pueda entrar a The Bright Day Coaster en cierto parque de atracciones, debe medir al menos 1.05 metros e ir acompañado de un adulto o puede ingresar sola si mide al menos 1.20 metros.
3. El horario de visita del Zoológico de Chapultepec es de 9:00 a 16:30 horas y cierra sus puertas los días lunes, el 1 de enero y el 25 de diciembre.

4 Diseño de algoritmos

Como hemos mencionado, un algoritmo representa las indicaciones que permiten solucionar un problema. Evidentemente, para poder expresar una solución, primero debemos saber cómo resolver el problema. Es así que aprender a resolver problemas es indispensable para el diseño de algoritmos. La resolución de problemas implica desarrollar un proceso ordenado y sistemático de pensar que se conoce como *pensamiento algorítmico*.

Pensamiento algorítmico

Habilidad de formular problemas, pensar creativamente sobre las soluciones y expresar la solución clara y concretamente.

A continuación describiremos algunos conceptos básicos para la resolución de problemas. Cabe resaltar que existen diferentes tipos de pensamientos y de maneras de observar y resolver problemas. Nuestro objetivo es que la persona que está leyendo este libro aprenda a identificar la información necesaria, para que posteriormente pueda diseñar sus propios algoritmos.

4.1. Resolución de problemas

Antes de continuar definiremos brevemente qué es un problema. Coloquialmente, usamos la palabra *problema* para expresar una dificultad o algo que nos cuesta trabajo resolver. Entre otras acepciones, el Diccionario de la Lengua Española define *problema*[9] como: *planteamiento de una situación cuya respuesta desconocida debe obtenerse a través de métodos científicos*. A partir de aquí nos basaremos en esta definición, puesto que nuestro objetivo es el diseño de algoritmos para resolver problemas.

La definición anterior de *problema* indica que la respuesta debe obtenerse a través de métodos científicos. Para ello, existen diversos modelos bien establecidos para la resolución de problemas que se clasifican en modelos matemáticos o psicológicos. La persona interesada en profundizar sobre dichos modelos puede consultar la tesis de Pino Ceballos [12]. En nuestro caso, utilizaremos una versión del modelo de Polya y Conway [13], el cual es ampliamente conocido, como

[9] problema. Diccionario de la lengua española. URL: https://dle.rae.es/problema. Fecha de última visita: 22/09/2023.

una primera aproximación a la resolución sistemática de problemas [16]. Así, presentamos las etapas y algunas preguntas que nos ayudan al desarrollo de la solución:

1. Analizar y comprender el problema.
    - ¿Cuáles cosas o qué información (datos de entrada) conozco?
    - ¿Qué espero obtener (datos de salida) al resolver el problema?
    - ¿Conozco o identifico las reglas (restricciones o contexto) para resolverlo?
    - ¿Qué conocimiento o herramientas (recursos) tengo a mi alcance?
2. Planear una estrategia.
    - Pensar en algunos ejemplos de las cosas que sé (datos de entrada) y de los resultados que debería obtener (datos de salida).
    - ¿Existe alguna relación entre lo que sé y el resultado deseado? Si es así, ¿cuál?
    - ¿Cómo puedo aplicar las reglas para llegar al resultado?
    - Identificar información adicional que me ayude a obtener el resultado.
    - Describir los pasos para solucionar el problema.
3. Probar la estrategia.
    - Si sigo exactamente los pasos que describí, ¿qué resultado obtengo?
    - ¿El resultado es el que esperaba?
    - Si no es el que esperaba, ¿cuál es la razón?
4. Depurar la estrategia.
    - ¿Hay algo que deba corregir?
    - ¿Hay algo que pueda mejorar?
    - ¿Hay algo que se pueda plantear de otra forma?

Cabe destacar que estas etapas son dinámicas y cíclicas, es decir, podemos repetir estas etapas más de una vez antes de tener la solución final y al realizar una etapa podemos regresar a etapas previas para reformular nuestras respuestas. En este capítulo, ejemplificamos la parte del entendimiento del problema. Conforme vayamos avanzando, nos basaremos en las siguientes etapas para describir y verificar nuestros algoritmos.

Ejemplo 4.1. Un grupo de 25 personas adultas debe cruzar un río ancho, profundo y con una corriente muy fuerte, con ningún puente a la vista. Se dan cuenta de que hay dos niños jugando con un bote de remos en la orilla. El bote es tan pequeño que solo puede transportar a lo más a dos niños o a una persona adulta. ¿Cómo pueden las personas adultas usar el bote para cruzar el río y dejar a los niños en posesión del bote?

Comencemos con las preguntas para analizar y comprender el problema:

¿Cuáles cosas o qué información (datos de entrada) conozco? Se sabe, por la descripción del problema, que hay 25 personas adultas que deben cruzar el río.

¿Qué espero obtener (datos de salida) al resolver el problema? El objetivo es que las 25 personas adultas crucen el río.

¿Conozco o identifico las reglas (restricciones o contexto) para resolverlo? Como el río es ancho, profundo y con una corriente muy fuerte, no pueden nadar hasta el otro lado y tampoco hay puentes por los que puedan cruzar. Por lo tanto, se necesita otro medio para cruzarlo.

¿Qué conocimiento o herramientas (recursos) tengo a mi alcance? Se sabe que se puede utilizar un bote con ayuda de dos niños.

Estas podrían ser las respuestas iniciales en esta etapa de la resolución del problema. Sin embargo, como podemos observar, faltaría indicar algunas restricciones, por ejemplo, la capacidad del bote o que el bote y los niños deben regresar a la orilla donde originalmente se encontraban. Estas observaciones seguramente las notaremos después de leer un par de veces el problema. Con la práctica seremos capaces de encontrar esta información con mayor facilidad.

Ejemplo 4.2. Si se conoce el radio de una esfera, encontrar su volumen.

Si bien en este ejemplo la formulación del problema es corta, veremos que se pueden responder las preguntas correspondientes al análisis y comprensión del problema gracias al conocimiento previo que tenemos sobre cómo calcular el volumen de una esfera.

¿Cuáles cosas o qué información (datos de entrada) conozco? El problema indica que debemos basarnos en el radio de una esfera para encontrar su volumen, entonces, debemos conocer el valor del radio.

¿Qué espero obtener (datos de salida) al resolver el problema? Se espera un número que representa el volumen de la esfera.

¿Conozco o identifico las reglas (restricciones o contexto) para resolverlo? En este caso, la esfera es una figura geométrica.

¿Qué conocimiento o herramientas (recursos) tengo a mi alcance? Se puede recordar o buscar la fórmula para calcular el volumen de una esfera, la cual es: $V = \frac{4}{3}\pi r^3$.

Ejemplo 4.3. Recordemos el ejemplo de la receta para hacer magdalenas, la cual describe los pasos que debemos seguir para prepararlas. En dicha receta, también se incluyen los ingredientes y los utensilios, pero, ¿cómo se relacionan con el análisis del problema, el cual es, preparar unas magdalenas?

¿Cuáles cosas o qué información (datos de entrada) conozco? En nuestro ejemplo, son los ingredientes de la receta y que, por supuesto, la persona que cocinará tiene al momento de preparar las magdalenas.

¿Qué espero obtener (datos de salida) al resolver el problema? Al finalizar los pasos, esperamos tener las magdalenas horneadas y listas para comer.

¿Conozco o identifico las reglas (restricciones o contexto) para resolverlo? En el caso de la receta, suponemos que la persona que preparará las magdalenas tiene un horno a su disposición y los utensilios necesarios, como los moldes o los tazones. Sin esto, aunque quisiéramos prepararlas, no podríamos hacerlo.

¿Qué conocimiento o herramientas (recursos) tengo a mi alcance? En este caso, podríamos pensar en que tenemos un recetario a la mano para ir leyendo la receta.

Análisis del problema

Es imposible escribir un algoritmo para solucionar un problema si no entendemos el problema.

4.2. Representación de los algoritmos

Un algoritmo es un conjunto ordenado, preciso, claro y finito de instrucciones que permite hallar la solución a un problema de manera sistemática, de tal forma que, dados un estado inicial y los datos de entrada, siguiendo las instrucciones podemos obtener datos de salida, es decir, la solución al problema. Estos pasos o, hablando formalmente, este algoritmo se puede representar o implementar de diferentes maneras. Entre las representaciones más utilizadas

se encuentran: algún idioma, como el español, diagramas de flujo, pseudocódigo, fórmulas e, incluso, programas de computadora.

Cada representación tiene sus propias peculiaridades y muchas veces se utilizan para diferentes fines. Esto no quiere decir que una es mejor que otra, significa que cada una tiene sus propias ventajas y desventajas y que, además, muchas veces son complementarias, ya que nos podrían ayudar a aclarar el mismo algoritmo representado de otra manera.

Implementación de un algoritmo

Se dice que un algoritmo se implementa o está implementado cuando se elige una forma de representarlo.

A continuación se introducirán las tres formas de representación de algoritmos que utilizamos en este libro: español, diagrama de flujo y pseudocódigo.

4.2.1. Español

La representación elemental de los algoritmos es la lengua natal de la persona que escribe el algoritmo, en nuestro caso, el español. Como lo vimos en el Capítulo 1, cuando describimos la solución a un problema, lo pensamos inicialmente en español. Esta representación consiste en describir instrucciones de la manera más precisa o exacta posible.

4.2.2. Diagramas de flujo

Los diagramas de flujo son una representación gráfica de las instrucciones que se deben seguir. Surgen para mejorar la calidad de la comunicación entre personas sobe los aspectos esenciales del procesamiento de la información. Una de las principales ventajas de representar los algoritmos mediante diagramas de flujo es que se puede apreciar cómo los datos cambian o *fluyen* cuando son procesados en un algoritmo. En general, se utilizan los símbolos normalizados por el Instituto Nacional Estadounidense de Estándares (ANSI, por sus siglas en inglés) [10]. Cada símbolo representa una acción que se debe realizar y los símbolos están unidos a través de flechas que se dirigen de uno hacia otro para indicar el orden en que la siguiente acción se debe realizar [3]. En la Figura 4.1 se muestran los principales símbolos usados en los diagramas de flujo.

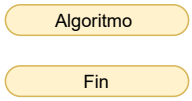
Símbolo	Significado
	Inicio/fin
	Datos de entrada
	Datos de salida
	Instrucción
	Flujo

Tabla 4.1: Principales símbolos usando en los diagramas de flujo.

4.2.3. Pseudocódigo

El pseudocódigo es una mezcla del lenguaje natural, en nuestro caso el español, y de expresiones como las que estudiamos en el Capítulo 3. El pseudocódigo permite estructurar y organizar las instrucciones de manera que se facilite su lectura y seguimiento para la resolución de problemas. Aunque no existe un estándar universal para la descripción de las instrucciones en pseudocódigo, generalmente nos podemos encontrar con las siguientes características:

- Las instrucciones deben estar escritas en el idioma natal.
- Al escribirlo, se indican las acciones que se tienen que realizar para solucionar un problema.
- Se usan los símbolos específicos para descripción de las diferentes operaciones o acciones.
- Los nombres simbólicos, identificadores, se usan para representar las cantidades procesadas por el algoritmo.
- Pueden usarse palabras clave para representar acciones comunes que deben realizarse. Por ejemplo, **leer** o **introducir** para las operaciones de entrada y **escribir** o **arrojar** para las operaciones de salida.
- Se utilizan las sangrías para identificar los bloques de instrucciones.

El objetivo del pseudocódigo es permitir que la persona que diseña el algoritmo se centre en los aspectos lógicos de la solución. Cabe destacar que el pseudocódigo no es un lenguaje formal,

Palabra clave	Significado
Algoritmo	El nombre y el inicio del algoritmo
Fin	Fin del algoritmo
leer	Recibe los datos de entrada
escribir	Arroja los datos de salida

Tabla 4.2: Algunas palabras clave usadas en pseudocódigo.

sin embargo, sí debe poder comprenderse por todos, de tal forma que se pueda reproducir la solución del problema. En la Tabla 4.2 podemos encontrar algunas de las palabras clave usadas en pseudocódigo.

En los siguientes capítulos detallaremos el uso de cada una de estas representaciones, así como ejemplos de su uso en la resolución de problemas.

Ejemplo 4.4. Para ilustrar lo que significa implementar o representar un algoritmo, consideremos el cálculo del índice de masa corporal (IMC), el cual es un método utilizado para estimar la cantidad de grasa corporal que tiene una persona respecto a la estatura y al peso de dicha persona.

El IMC se calcula de la siguiente forma:

$$IMC = \frac{peso}{estatura^2} \ .$$

Encuentra el IMC de una persona.

Retomando lo que hemos aprendido en este capítulo, vamos a responder las preguntas para identificar el estado inicial, los datos de entrada, los datos de salida, las restricciones y los tipos de datos.

¿Cuáles cosas o qué información (datos de entrada) conozco? El problema indica que el cálculo del IMC se basa en la estatura y en el peso de una persona, por lo que debemos conocer estos valores que se representan con un número cada uno.

¿Qué espero obtener (datos de salida) al resolver el problema? Se espera un número que representa el IMC de una persona.

¿Conozco o identifico las reglas (restricciones o contexto) para resolverlo? En el enunciado del problema se describe cómo puedo utilizar la estatura y el peso para calcular el IMC.

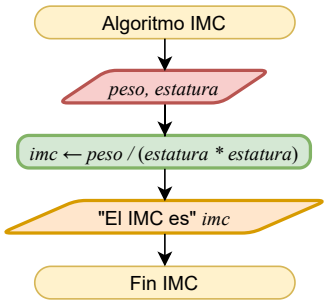


Figura 4.1: Diagrama de flujo para calcular el IMC de una persona.

Algoritmo 4.1: IMC

```
1 leer peso, estatura
2 imc ← peso/(estatura ∗ estatura)
3 escribir "El IMC es" imc
```

¿Qué conocimiento o herramientas (recursos) tengo a mi alcance? En la descripción del problema se hace referencia a la fórmula para calcular el IMC.

El algoritmo para calcular el IMC representado en español es:

1. Registrar el peso y la estatura de una persona.
2. Multiplicar la estatura por ella misma.
3. Dividir el peso por el resultado de la multiplicación anterior.
4. El IMC será el resultado de la división anterior.

El mismo algoritmo, pero representado como diagrama de flujo y en pseudocódigo, se presentan en la Figura 4.1 y en el Algoritmo 4.1, respectivamente. En este momento no nos preocuparemos por entender qué quiere decir cada elemento del diagrama de flujo y del pseudocódigo, solamente queremos mostrar las diferencias entre las formas de representación.

Otras representaciones

En el Ejemplo 4.4, si observamos atentamente, veremos que estamos representado el algoritmo para el cálculo del IMC como una fórmula. Una fórmula también es una manera de representar un algoritmo.

4.3. Estructuras de control

En 1966, Corrado Böhm y Giuseppe Jacopini [2] demostraron que cualquier solución se puede expresar en términos de tres estructuras de control. Estas estructuras nos permiten controlar las rutas de acción que se pueden tomar en los algoritmos, la agrupación de instrucciones en bloques y la manera de modelar la solución de un problema sistemáticamente. Las estructuras básicas de control son:

Secuencial Una estructura secuencial es aquella en la que las instrucciones están una a continuación de la otra, siguiendo una secuencia única, sin cambios de flujo. Los pasos se ejecutan de manera estrictamente secuencial y cada uno de ellos se ejecuta exactamente una vez.

Selectiva La estructura de selección permite ejecutar una acción o un grupo de acciones solo si se cumple una determinada condición. Se selecciona y se ejecuta una de entre varias acciones posibles.

Iterativa En este tipo de estructuras, una sentencia o grupos de sentencias se ejecutan repetidamente varias veces.

En los siguientes capítulos resolveremos problemas utilizando cada una de estas estructuras, así como su uso y representación en diagramas de flujo y pseudocódigo.

Antes de continuar

Es importante enfatizar que el diseño de algoritmos es un proceso cognitivo que recae en la síntesis, abstracción, comunicación, expresión y creatividad de la persona que diseña los algoritmos. Estas capacidades cognitivas únicamente se pueden cultivar al tratar resolver un problema mediante los propios medios y conocimientos. Lo anterior se traduce simple y llanamente en practicar, intentar resolver la mayor cantidad de problemas usando los conceptos adquiridos.

4.4. Ejercicios

Identificar el estado inicial, los datos de entrada, los datos de salida, las restricciones y los tipos de datos de los siguientes problemas.

Ejercicio 4.1. Doña Lore tiene una cocina económica que vende comidas de tres tiempos. En el primer tiempo ofrece sopa de lentejas a $20, crema de elote a $25 y sopa de fideos a $15. En el segundo tiempo, tiene arroz a $15, espagueti a $20 y ensalada a $10. En el tercer tiempo

tiene pollo con papas a $35, bistec en morita a $45 y chile relleno a $30. ¿Cuánto tiene que pagar un comensal que elige una opción de cada tiempo?

Ejercicio 4.2. Dada una fecha, calcular la fecha del siguiente día. Por ejemplo, si la fecha es 2023-10-30, entonces la fecha del siguiente día es 2023-10-31. Si la fecha es 2023-10-31, entonces el día siguiente es 2023-11-01. Si la fecha es 2023-12-31 el día siguiente es 2024-01-01. Considera que la fecha tiene formato con tres valores numéricos: año, mes y día.

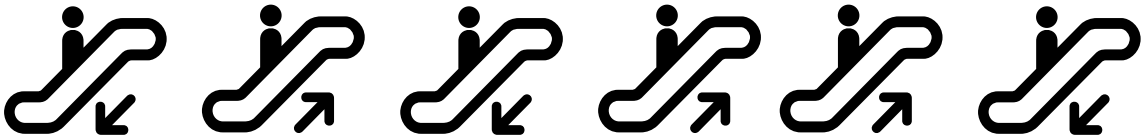
Ejercicio 4.3. Comprar un café de una máquina expendedora.

Ejercicio 4.4. Un granjero tiene un zorro, una gallina y un saco de maíz, pero necesita cruzar un río. El granjero tiene un bote de remos, sin embargo, es muy pequeño y tiene espacio para el granjero y uno solo de los tres objetos (zorro, gallina o saco de maíz). Desafortunadamente, el zorro y la gallina tienen hambre. El zorro no se puede quedar solo con la gallina porque se la comería, tampoco la gallina se puede quedar sola con el saco de maíz por la misma razón.

Ejercicio 4.5. Una persona desea publicar un contenido con la aplicación de TikTok.

Ejercicio 4.6. Comprar un producto de las máquinas expendedoras en el 4o piso de la Unidad Cuajimalpa de la UAM.

Ejercicio 4.7. Observa la siguiente figura. Se puede ver que hay personas que están subiendo y bajando las escaleras como indican las flechas. Se desea contar la cantidad de personas que están subiendo.



Ejercicio 4.8. En la UAM, las calificaciones finales de un curso se expresan como MB, B, S y NA. Sin embargo, una evaluación basada en letras no es fácil de realizar, por lo que el personal docente prefiere hacer una equivalencia entre una calificación numérica y la letra. Por ejemplo:

Calificación numérica	**Calificación final**
Menos de 7	NA
De 7 a 8	S
Más de 8 y hasta a 9	B
Más de 9	MB

Al final del curso, el personal docente asigna la calificación final correspondiente a la calificación numérica. Dada una calificación numérica, ¿cuál será la calificación alfabética?

5 Estructura de control secuencial

Una estructura secuencial es aquella en la que las instrucciones están una a continuación de la otra, siguiendo una secuencia única, sin bifurcaciones o toma de decisiones. Asímismo, cada uno de los pasos se ejecuta exactamente una vez. Es la estructura de control más simple y permite que las instrucciones que la constituyen se ejecuten una tras otra en el orden en que están enlistadas.

5.1. Implementación

Como comentamos en el Capítulo anterior, nos enfocaremos en dos formas de implementar algoritmos: mediante diagramas de flujo y como pseudocódigo.

En un diagrama de flujo, cada instrucción contenida en una estructura de control secuencial se representa mediante un rectángulo. La secuencia de las instrucciones se indica mediante una flecha que sale de un rectángulo y llega a otro, como se muestra en el diagrama de la Figura 5.1. Además, el diagrama de flujo tiene un inicio y un fin, representados por un óvalo antes de la primera instrucción y otro óvalo después de la última. Estos óvalos deben indicar el nombre del algoritmo.

La implementación de esta estructura en pseudocódigo muestra las instrucciones, en el orden en que se van a ejecutar, en una lista numerada y una instrucción por línea, como se muestra en el Algoritmo 5.1. En este pseudocódigo podemos ver que la primera línea indica el nombre del algoritmo, similar al diagrama de flujo. Cabe señalar que la numeración del algoritmo (5.1) no es parte del pseudocódigo, pero la utilizaremos en este libro pare hacer referencia a los mismos.

Algoritmo 5.1: Secuencial

```
1 instrucción 1
2 instrucción 2
  ⋮
  instrucción n
```

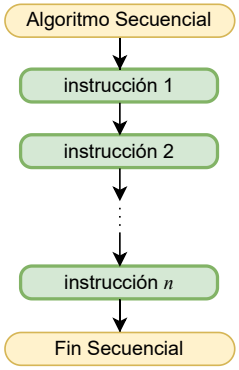


Figura 5.1: En un diagrama de flujo, la estructura de control secuencial se representa mediante rectángulos, que corresponden a las operaciones, unidos mediante flechas, que simbolizan el orden de ejecución de las operaciones.

Ejemplo 5.1. Retomemos el Ejemplo 1.1: Escribir en español el algoritmo para hacer una llamada telefónica desde un dispositivo móvil inteligente. Originalmente, describimos los pasos en español de la siguiente manera:

1. Sacar al dispositivo móvil del estado de inactividad para poder marcar la contraseña de acceso o que identifique nuestros datos biométricos (huella o rostro).
2. Oprimir el ícono de la función "teléfono" y marcar el número al cual deseamos llamar.
3. Oprimir el ícono de llamar.

Si observamos bien, en este ejemplo encontramos las instrucciones en forma de una lista numerada, con la descripción de los pasos a realizar. Sin embargo, las dos primeras líneas de estas instrucciones contienen más de una actividad, por lo que, para escribir este algoritmo de forma apropiada, debemos separarlas con más detalle.

Por ejemplo, la primera instrucción contiene dos actividades: activar el teléfono y desbloquearlo con una contraseña (o con datos biométricos). La segunda instrucción también indica dos tareas: seleccionar la aplicación de "teléfono" y marcar el número al cual se desea llamar.

En este punto llegamos a una pregunta, ¿cuál es el número al que debemos llamar? Dado que el problema no lo indica, en realidad puede ser ¡a cualquier número! Es decir, estas instrucciones son útiles para que una persona llame al número telefónico que desee. Entonces, ¿cómo saber el número? Generalmente, los algoritmos se escriben de forma

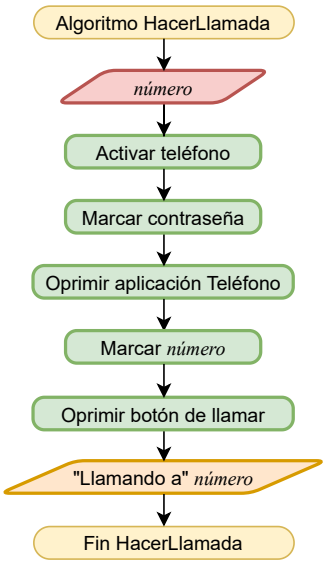


Figura 5.2: Diagrama de flujo para hacer una llamada telefónica desde un dispositivo móvil inteligente.

tal que puedan ser aplicados a diferentes datos cada vez que se ejecute. En este caso, el número al que una persona desee llamar debería ser un dato de entrada, para el cual utilizaremos el identificador $número$ que es de tipo entero.

Este mismo algoritmo lo podemos representar a través de un diagrama de flujo como el que se muestra en la Figura 5.2. Como podemos ver, las instrucciones originales están incluidas como las detallamos anteriormente. Además, este diagrama incluye dos romboides, uno rojo y uno anaranjado, los cuales indican los datos de entrada y los datos de salida, respectivamente.

Por otro lado, el pseudocódigo se presenta en el Algoritmo 5.2. De igual forma, cada instrucción está detallada, una por línea. Además, este pseudocódigo incluye dos palabras *clave* indicadas en negritas, **leer** y **escribir**, que se utilizan para recibir los datos de entrada y para arrojar los datos de salida, respectivamente.

Algoritmo 5.2: HacerLlamada

```
1 leer número
2 Activar teléfono
3 Marcar contraseña
4 Oprimir aplicación Teléfono
5 Oprimir botón de llamar
6 Marcar número
7 escribir "Llamando a" número
```

Ejemplo 5.2. El índice de masa corporal (IMC) sirve para medir la relación entre el peso y la talla de una persona, lo que permite identificar delgadez o sobrepeso en adultos. El IMC se calcula dividiendo el peso de una persona en kilogramos entre el cuadrado de su estatura en metros:

$$IMC = \frac{peso}{estatura^2} \ .$$

De acuerdo con la definición anterior, para poder calcular el IMC de una persona, necesitamos conocer su peso en kilogramos y su estatura en metros. Para estos dos datos utilizaremos los identificadores $peso$ y $estatura$, respectivamente, ambos de tipo real. Dados estos datos de entrada, podemos proceder inmediatamente a dividir el $peso$ entre la $estatura$ elevada al cuadrado y el resultado asignarlo al identificador imc, el cual también es de tipo real. El resultado de estas operaciones se puede arrojar como dato de salida para que la persona sepa su condición.

El diagrama de flujo que se presenta en la Figura 5.3 corresponde a la solución a este problema y el Algoritmo 5.3 es el pseudocódigo de la misma solución. En estas representaciones, los dos datos de entrada requeridos, $peso$ y $estatura$, se reciben en la misma instrucción.

Algoritmo 5.3: IMC

```
1 leer peso, estatura
2 imc ← peso/(estatura ∗ estatura)
3 escribir "El IMC es" imc
```

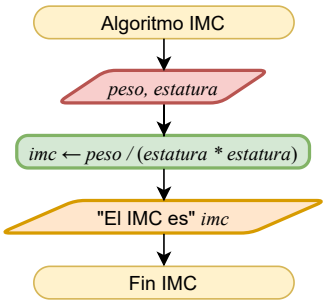


Figura 5.3: Diagrama de flujo para calcular el IMC de una persona.

Recordatorio

Para continuar estudiando más ejemplos, es muy importante recordar que no existe una manera única de resolver un problema dado. Siempre habrán diferentes aproximaciones como puntos de vista. Sin embargo, aquí exponemos solo algunas de esas soluciones, para permitir que las y los lectores propongan las propias.

Ejemplo 5.3. Consideremos ahora el problema de calcular el área y el perímetro de un rectángulo. Analicemos un poco para que podamos diseñar un algoritmo que realice estos cálculos.

Recordando las características de los algoritmos presentadas en el Capítulo 2, es necesario identificar los datos de entrada del problema. En este ejemplo, para poder calcular el área y el perímetro de un rectángulo, se deben conocer su largo y su ancho, por lo tanto, estos deben ser considerados como datos de entrada. Para ello, utilizaremos los identificadores $ancho$ y $largo$ que son de tipo real. De manera similar, utilizaremos los identificadores $area$ y $perimetro$ para los resultados de las expresiones.

El diagrama de flujo de la Figura 5.4 ilustra la solución a este problema y el Algoritmo 5.4 muestra el pseudocódigo correspondiente. Notemos que estas representaciones, tanto el diagrama de flujo como el pseudocódigo, consideran dos instrucciones para los datos de salida, una para el área y otra para el perímetro. Esto en general no es necesario, pero se muestran de esta forma para que el algoritmo tenga mucha mayor claridad.

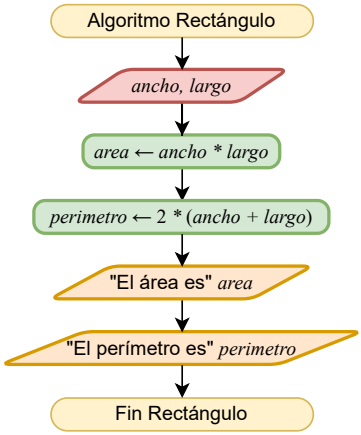


Figura 5.4: Diagrama de flujo para calcular el área y perímetro de un rectángulo.

Algoritmo 5.4: Rectángulo

```
1 leer ancho, largo
2 area ← ancho * largo
3 perimetro ← 2 * (ancho + largo)
4 escribir "El área es" area
5 escribir "El perímetro es" perimetro
```

Ejemplo 5.4. En una competencia atlética de velocidad, el tiempo se mide en minutos y segundos, estos últimos incluyen centésimas. La distancia recorrida se mide en metros. Diseña un algoritmo para determinar la velocidad de una atleta en km/h.

Comencemos con identificar los datos de entrada del problema, que son el tiempo, dado en minutos (min, de tipo entero) y segundos (seg, de tipo real), y la distancia, dada en metros ($metros$, de tipo real). Para poder calcular la velocidad de la atleta en km/h, primero debemos convertir los minutos y segundos a horas ($tiempo$, de tipo real) y los metros a kilómetros ($distancia$, de tipo real). Después, tenemos que dividir la $distancia$ entre el $tiempo$ para calcular la $velocidad$ (tipo real) en las unidades requeridas.

La solución descrita arriba está ilustrada en el diagrama de flujo de la Figura 5.5 y el Algoritmo 5.5 muestra el pseudocódigo correspondiente. Este algoritmo, primeramente, recibe los datos de entrada min, seg, $metros$. Después, se convierten los minutos a horas y se asigna el resultado a $tiempo$. La siguiente instrucción es una expresión aritmética con dos operadores, de los cuales, la división tiene precedencia. Esta división se evalúa para convertir los segundos a horas y el resultado se suma a $tiempo$, que tenía el

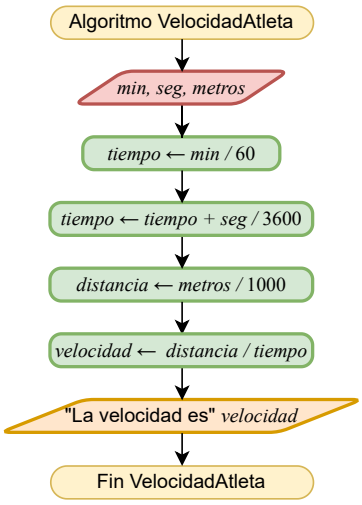


Figura 5.5: Diagrama de flujo para calcular la velocidad de una atleta en km/h.

Algoritmo 5.5: VelocidadAtleta

```
1 leer min, seg, metros
2 tiempo ← min/60
3 tiempo ← tiempo + seg/3600
4 distancia ← metros/1000
5 velocidad ← distancia/tiempo
6 escribir "La velocidad es" velocidad
```

equivalente de los minutos a horas. De esta forma, al evaluar la suma, tenemos la conversión de los minutos y los segundos a horas. Este resultado se asigna al mismo identificador *tiempo*, por lo que el valor previamente asignado es destruido. La siguiente instrucción convierte los metros a kilómetros mediante la división de metros entre 1000 y se asigna el resultado al identificador *distancia*. Finalmente, se calcula la velocidad en unidades km/h dividiendo distancia entre tiempo y se asigna el resultado al identificador *velocidad*, el cual se arroja como dato de salida.

Ejemplo 5.5. Una persona que trabaja por honorarios tiene un salario por hora y se le paga semanalmente por las horas trabajadas a la semana. Para conocer el sueldo neto que recibirá, conocido como total facturado, se debe aumentar el 16 % de IVA al salario semanal. Además, al total facturado se le debe restar la retención del IVA (dos

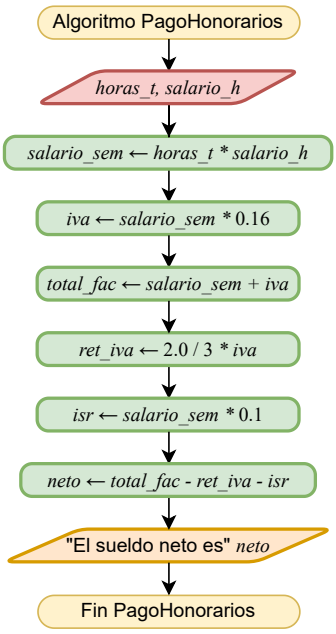


Figura 5.6: Diagrama de flujo para calcular sueldo neto que recibirá una persona que trabaja por honorarios.

terceras partes del IVA) y el ISR (10 % del salario semanal). Determinar el sueldo neto que recibirá la persona a la semana.

Los datos de entrada para este problema son: las horas trabajadas ($horas\_t$, de tipo entero) y el salario por hora ($salario\_h$, de tipo real). Para calcular el sueldo neto que recibirá la persona, procedemos tal cual lo especifica el problema.

Primero, se calcula el salario semanal ($salario\_sem$, de tipo real) correspondiente a la semana trabajada, el IVA respectivo (iva, de tipo real) y el total facturado ($total\_fac$, de tipo real). Posteriormente, se calculan las deducciones, es decir, las retenciones por IVA (ret_iva, de tipo real) y por ISR (isr, de tipo real). Finalmente, para calcular el sueldo neto ($neto$, de tipo real), al total facturado ($total\_fac$) se le restan las dos deducciones (ret_iva e isr).

La Figura 5.6 muestra el diagrama de flujo del algoritmo descrito arriba para resolver el problema y el pseudocódigo correspondiente es el que aparece en el Algoritmo 5.6.

Algoritmo 5.6: PagoHonorarios

```
1 leer horas_t, salario_h
2 salario_sem ← horas_t * salario_h
3 iva ← salario_sem * 0.16
4 total_fac ← salario_sem + iva
5 ret_iva ← 2.0/3 * iva
6 isr ← salario_sem * 0.1
7 neto ← total_fac – ret_iva – isr
8 escribir "El sueldo neto es" neto
```

Ejemplo 5.6. Dada una cantidad de segundos, encontrar a cuántas horas, minutos y segundos son equivalentes.

En este problema nos dan, como datos de entrada, cierta cantidad de $segundos$ (tipo entero), para los cuales debemos encontrar su equivalente en $horas$, $minutos$ y $segundos$, los tres de tipo entero. Sabemos que un minuto es equivalente a 60 segundos y que una hora es equivalente a 3600 segundos. De esta forma, podemos hacer una división entera de los $segundos$ entre 3600 para conocer la equivalencia en $horas$. Los segundos restantes (seg_rest, de tipo entero), que son el residuo de la división anterior, los podemos dividir entre 60 para conocer su equivalencia en $minutos$. Los segundos que vuelvan a quedar como residuo complementarán la equivalencia.

Recordando los operadores aritméticos vistos en el Capítulo 3, sabemos que hay una operación que nos puede ayudar a conocer el residuo de una división entera, que es la operación módulo (`MOD`).

El diagrama de flujo de la Figura 5.7 y el pseudocódigo del Algoritmo 5.7 presentan la solución a este problema.

Algoritmo 5.7: EquivalenciaSegundos

```
1 leer segundos
2 horas ← segundos/3600
3 seg_rest ← segundos MOD 3600
4 minutos ← seg_rest/60
5 seg_rest ← seg_rest MOD 60
6 escribir "Equivalente" horas ":" minutos ":" seg_rest
```

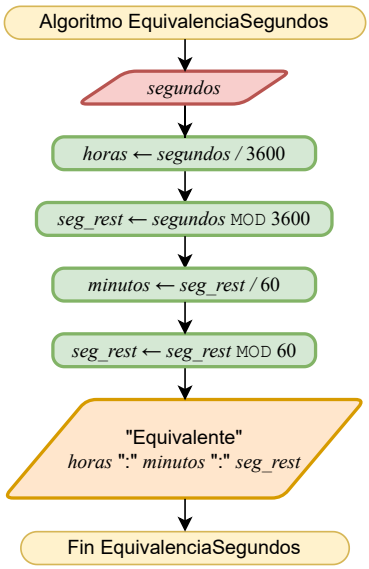


Figura 5.7: Diagrama de flujo para encontrar la equivalencia de cierta cantidad de segundos en horas, minutos y segundos.

División entera

En el Ejemplo 5.6 se requirió calcular dos divisiones enteras, sin embargo, utilizamos el mismo operador de división (/). Entonces, ¿cómo saber cuándo es una división entera? La respuesta nos la van a dar los tipos de datos de los operados de la división. Cuando ambos operandos sean de tipo entero, entonces evaluaremos una división entera.

Ejemplo 5.7. Dadas tres longitudes, es posible formar un triángulo con ellas cuando la suma de cada par de longitudes es mayor que la tercera. En este ejemplo, describiremos la solución para determinar si las tres longitudes dadas pueden formar un triángulo.

Como el problema menciona, tenemos como datos de entrada las tres longitudes $l1$, $l2$ y $l3$. Para saber si estas longitudes pueden formar un triángulo, debemos verificar que la suma de cada par de ellas es mayor que la otra, es decir, tenemos que comprobar que $l1 + l2 > l3$, que $l1 + l3 > l2$ y que $l2 + l3 > l1$. Sabemos que estas expresiones son relacionales y que cada una arrojará un resultado lógico, es decir, **Verdadero** o **Falso**, y que el resultado de las tres expresiones debe ser **Verdadero** para que se pueda formar un triángulo. Entonces, las tres expresiones anteriores se pueden unir mediante una operación **Y** lógica ($\wedge$). Este algoritmo está representado como diagrama de flujo en la Figura 5.8 y como pseudocódigo en el Algoritmo 5.8.

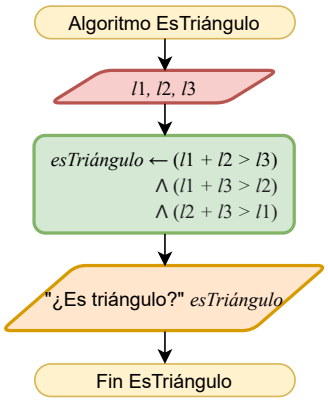


Figura 5.8: Diagrama de flujo para determinar si tres longitudes forman un triángulo.

Algoritmo 5.8: EsTriángulo

1 **leer** $l1$, $l2$, $l3$
2 $esTriángulo \leftarrow (l1 + l2 > l3) \wedge (l1 + l3 > l2) \wedge (l2 + l3 > l1)$
3 **escribir** "¿Es triángulo?" $esTriángulo$

5.2. Ejercicios

Resuelve los siguientes ejercicios con un algoritmo que utilice una estructura de control secuencial.

Ejercicio 5.1. Calcular el promedio de los tres números a, b y c.

Ejercicio 5.2. Una persona compra tres artículos con precios x, y y z, respectivamente. El primer artículo tiene un descuento del 10 %, el segundo artículo tiene un descuento del 5 % y el tercer artículo tiene un descuento del 15 %. ¿Cuál es la cantidad de dinero que ahorrará la persona que compró los artículos?

Ejercicio 5.3. La tarjeta de metrobús tiene un costo de $21 e incluye un viaje (que tiene un costo de $6). Las tarjetas se pueden comprar en las máquinas expendedoras que hay en las estaciones. Estas máquinas no dan cambio, por lo que si se pagan más de $21, el excedente se suma al saldo de la tarjeta. Una persona compra una tarjeta con $30. Después de hacer dos viajes, recarga $20 a la tarjeta. ¿Cuántos viajes puede hacer después de hacer esta última recarga?

Ejercicio 5.4. Encuentra la calificación final de un curso que se calcula con el promedio de las calificaciones de cinco exámenes parciales.

Ejercicio 5.5. Encuentra el volumen de una esfera dado su radio.

Ejercicio 5.6. Una persona se encuentra a la orilla de un río con un lobo, una cabra y una col. La persona tiene que transportar a los otros tres al otro lado del río en su bote. Sin embargo, el bote tiene capacidad sólo para ella y otro más (ya sea el lobo, la cabra o la col). En ausencia de la persona, el lobo se comería a la cabra y la cabra se comería la col. Muestra cómo la persona puede transportar a todos estos "pasajeros" al otro lado.

Ejercicio 5.7. Una persona instala fibra óptica para una compañía. Por cada instalación, hay una tarifa mínima por servicio de $500 y una tarifa adicional de $50 por cada pie de fibra instalada. El total de fibra instalada se da en yardas. Calcula el total de ingresos que obtuvo la persona.

Ejercicio 5.8. En los grandes campos de cultivo se suele fumigar utilizando avionetas. Para que un avión fumigador pueda despegar, la temperatura debe estar por arriba de los 22°C, la humedad relativa debe ser mayor que 15 % y menor que 35 %, y la velocidad del aire debe ser menor que 40 km/h. Verifica que las mediciones sean propicias para el despegue.

Ejercicio 5.9. Supongamos que colaboras para ACME Satellites, Inc. En días pasados, el satélite Quetzalcóatl I salió de órbita y te piden que traces una trayectoria para que regrese a orbitar la tierra. Para esto, el Quetzalcóatl I tiene definidas las siguientes instrucciones:

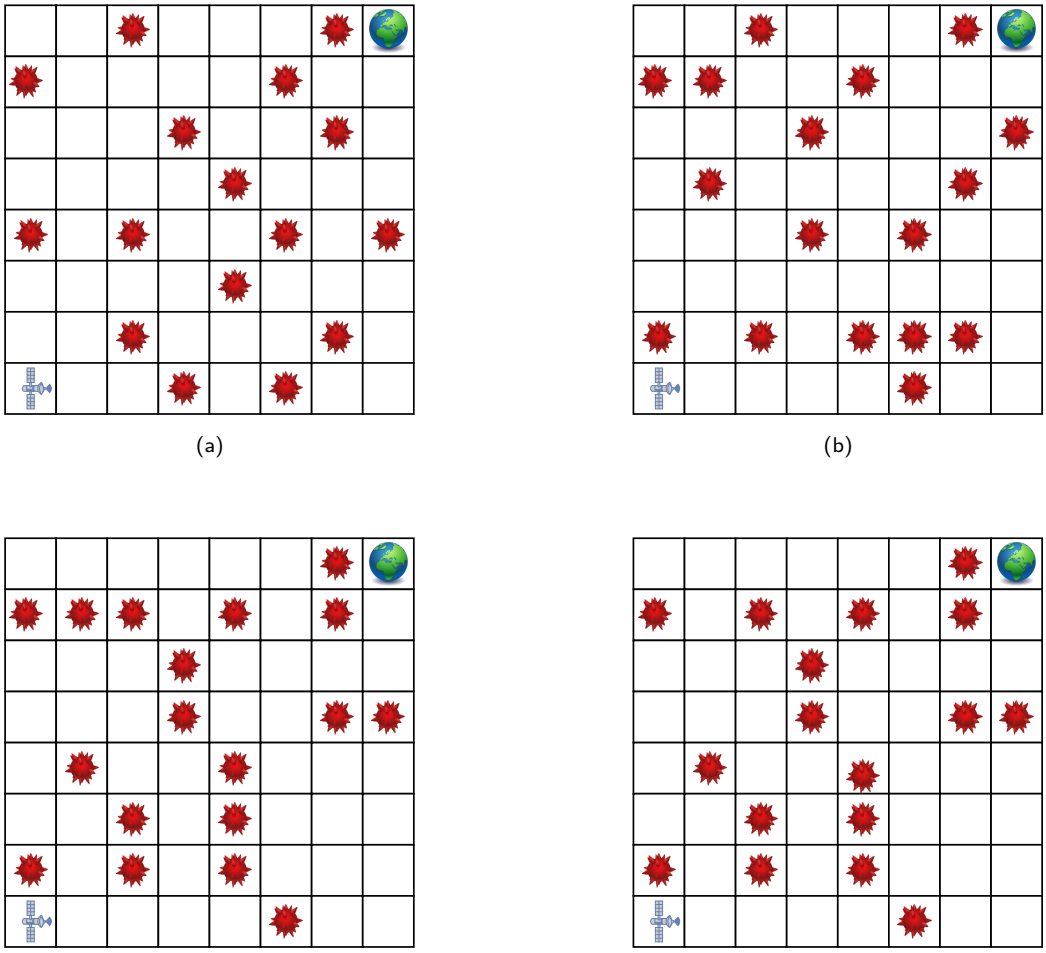
Avanzar: Avanza exactamente dos unidades hacia el frente.

Girar: Gira 90° hacia la derecha.

Regresar: Retrocede exactamente una unidad en dirección opuesta al frente.

Utiliza secuencias de estas tres instrucciones para que el Quetzalcóatl I regrese a orbitar la Tierra en cada uno de los escenarios que se presentan en la Figura 5.9. Toma en cuenta que los asteroides impiden que el Quetzalcóatl I avance.

Ejercicio 5.10. Dada cierta hora del día en horas, minutos y segundos, determina cuánto tiempo falta, en horas, minutos y segundos, para que termine el día.



(a) (b)

(c) (d)

Figura 5.9: Escenarios del Quetzalcóatl I para el Ejercicio 5.9.

6 Estructura de control selectiva

Es común encontrar problemas para los cuales tenemos que realizar ciertas actividades o instrucciones de acuerdo con el cumplimiento de alguna condición. Por ejemplo, si queremos preparar café, primero tenemos que verificar si hay café molido. Si hay, podemos prepararlo inmediatamente, pero si no hay, habrá que molerlo antes de ponerlo en la cafetera.

Para este tipo de problemas es útil considerar la estructura de control selectiva, la cual, precisamente, nos permite lograr este objetivo. En general, al considerar una estructura de control, el flujo del algoritmo se puede dividir de acuerdo con las condiciones que se establezcan.

En un diagrama de flujo, una estructura de control selectiva se representa mediante un rombo y un símbolo de unión (⊗), como los que se muestran en la Figura 6.0. Cuando el flujo del algoritmo alcanza el rombo, este indica que se tiene que evaluar una expresión, la cual está indicada dentro del rombo, a la que llamaremos *expresión de decisión*. En este punto, el flujo del algoritmo tiene dos opciones para continuar de acuerdo con el resultado de la evaluación de la expresión: si es **Falso**, el flujo continuará por la flecha indicada con **F** y si es **Verdadero**, el flujo continuará por la flecha indicada con **V**. Cualquiera que sea el resultado de la evaluación de la expresión, el flujo de ambas opciones se encuentran posteriormente en el símbolo de unión, que indica el final de la estructura de control selectiva.

Podemos clasificar a esta estructura de control en tres variantes, de acuerdo con la cantidad de opciones que tenemos para elegir y a las dependencias que existen entre ellas: estructura de control selectiva simple, estructura de control selectiva múltiple y estructura de control selectiva anidada.

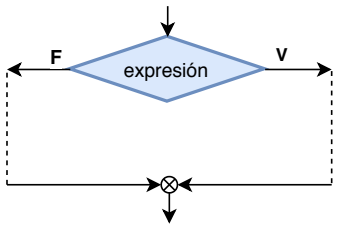


Figura 6.0: En un diagrama de flujo, la estructura de control selectiva se representa mediante un rombo y un símbolo de unión.

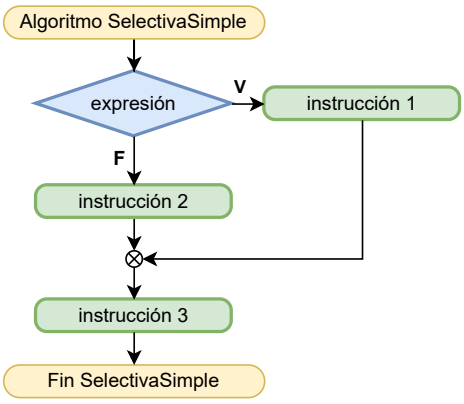


Figura 6.1: Diagrama de flujo que representa la estructura de control selectiva simple.

6.1. Estructura de control selectiva simple

Esta variante de la estructura de control selectiva verifica si una condición se cumple, mediante la evaluación de una expresión relacional o lógica. Si el resultado de esta evaluación es **Verdadero**, se ejecuta un bloque de instrucciones, en caso contrario, se podría ejecutar un bloque diferente. La estructura de control selectiva simple se muestra en el diagrama de flujo de la Figura 6.1.

Como podemos observar en el diagrama de flujo de la Figura 6.1, dentro del rombo existe una expresión que se debe evaluar. En caso de que la evaluación resulte **Verdadero**, se ejecuta la instrucción 1, en caso contrario, es decir, si la evaluación resulta **Falso**, se ejecuta la instrucción 2. Posteriormente, ya sea que se haya ejecutado la instrucción 1 o la instrucción 2, el flujo del algoritmo llega al símbolo de unión (⊗), para finalmente ejecutar la instrucción 3.

Esta variante de la estructura de control selectiva se escribe en pseudocódigo como muestra el Algoritmo 6.1. El rombo del diagrama de flujo se traduce a pseudocódigo con la palabra reservada **si**. Después, se escribe la expresión de decisión y el enunciado termina con la palabra **entonces**. Enseguida, se escribe el bloque de instrucciones que se tiene que ejecutar en caso de que la expresión de decisión evalúe **Verdadero**. Este bloque de instrucciones debe tener un margen mayor para identificar perfectamente que son las instrucciones que se ejecutarán para el caso **Verdadero** de la estructura selectiva. A continuación, se escriben las palabras **si no**, que corresponde a la opción **Falso** del rombo, y se incluye, también con un margen más amplio, el bloque de instrucciones que se ejecutarán en caso de que la expresión de decisión evalúe **Falso**. Finalmente, la estructura de control termina con las palabras **fin si**.

Algoritmo 6.1: SelectivaSimple

```
1 si expresión evalúa Verdadero entonces
2 |   instrucción 1
3 si no
4 |   instrucción 2
5 fin si
6 instrucción 3
```

Ejemplo 6.1. Diseña un algoritmo para saber cuánto se debe pagar por cierta cantidad de sobres de estampas del mundial, si cada sobre cuesta $18, pero si se compran más de 100 sobres, el costo es de $15.

En este caso, la cantidad de sobres que se van a comprar ($cant\_sobres$, de tipo entero) es el dato de entrada y podemos utilizar el identificador $costo$, de tipo real, para almacenar el costo de cada sobre, que depende de la cantidad $cant\_sobres$. Además, utilizaremos el identificador $total$, de tipo real, para almacenar el costo total por los $cant\_sobres$. Inicialmente, se recibe la cantidad de sobres a comprar ($cant\_sobres$) como dato de entrada. Después, se evalúa si la cantidad de sobres es mayor que 100. En caso de que si, el $costo$ de cada sobre será $15 y si no, el $costo$ de cada sobre será $18. Finalmente, se calcula la cantidad a pagar por los $cant\_sobres$ y se almacena en $total$.

La representación en diagrama de flujo de este algoritmo se muestra en la Figura 6.2. En este diagrama, podemos ver que el rombo contiene la expresión de decisión $cant\_sobres > 100$. Al evaluar esta expresión obtendremos un resultado lógico, **Verdadero** o **Falso**, dependiendo del valor de $cant\_sobres$. En caso de que el resultado sea **Verdadero**, el flujo del algoritmo seguirá la flecha indicada con **V**, asignando 15 al identificador $costo$ y el flujo se encontrará con el símbolo de unión. En caso de que la expresión de decisión evalúe **Falso**, entonces el flujo seguirá la flecha indicada con **F**, asignando 18 a $costo$ y posteriormente encontrarse con el símbolo de unión. Finalmente, se calculará el costo total a pagar por los sobres y se arrojará el resultado.

La representación en pseudocódigo se muestra en el Algoritmo 6.2. Podemos ver que la expresión de decisión $cant\_sobres > 100$ se encuentra entre las palabras clave **si** y **entonces**, indicando que se trata de una estructura selectiva. En caso de que la expresión evalúe **Verdadero**, se ejecutará el bloque de instrucciones entre esta línea y la indicada con **si no**, que en este caso solo es la línea 3. Cuando se termine de ejecutar este bloque, el flujo continuará en la línea que sigue el final de esta estructura (**fin si**), en la línea 7. Si el resultado de la expresión es **Falso**, entonces el flujo se saltará todo el bloque anterior hasta encontrar la palabra clave **si no**, línea 4, y se ejecutará el bloque de instrucciones entre esta línea y la indicada por **fin si**. Después, el flujo seguirá con la línea 7.

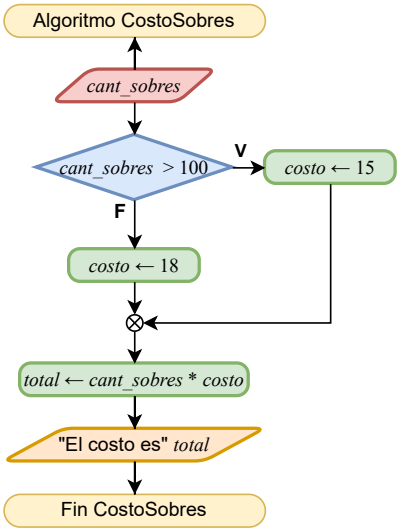


Figura 6.2: Diagrama de flujo para calcular el costo de cierta cantidad de sobres de estampas del mundial.

Algoritmo 6.2: CostoSobres

```
1 leer cant_sobres
2 si cant_sobres > 100 entonces
3 |   costo ← 15
4 si no
5 |   costo ← 18
6 fin si
7 total ← cant_sobres * costo
8 escribir "El costo es" total
```

Ejemplo 6.2. Dados los dos números a y b, encontrar el máximo de estos utilizando una estructura de control selectiva simple.

El problema nos dice que tenemos los números a y b como datos de entrada. Para encontrar cuál de los dos números es el máximo (utilizaremos el identificador max para asignarle este valor), debemos establecer una relación entre ellos. Es decir, tenemos que compararlos mediante un operador relacional, por ejemplo, $a > b$. Si al evaluar esta expresión el resultado es **Verdadero**, entonces sabemos que a es el máximo ($max \leftarrow a$). En caso contrario, b es el máximo ($max \leftarrow b$).

El diagrama de flujo que se muestra en la Figura 6.3 y el pseudocódigo del Algoritmo 6.3 representan esta solución.

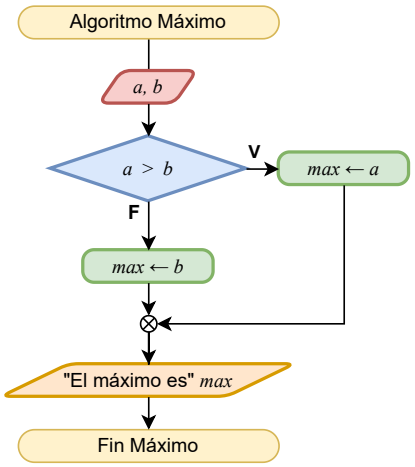


Figura 6.3: Diagrama de flujo para encontrar el máximo de dos números.

Algoritmo 6.3: Máximo

```
1 leer a, b
2 si a > b entonces
3 |   max ← a
4 si no
5 |   max ← b
6 fin si
7 escribir "El máximo es" max
```

Ejemplo 6.3. Una frutería vende manzanas a $10 cada una y naranjas a $5 la pieza. Si una persona quiere comprar cierta cantidad de piezas de alguna de las frutas, ¿cuánto tiene que pagar?

Para saber cuánto tiene que pagar una persona que quiere comprar alguna de las frutas, necesitamos saber cuántas $piezas$ quiere y cuál $fruta$ es. Entonces, estos dos datos se requieren como entrada para el algoritmo. Como cada fruta tiene diferente precio, debemos saber si la $fruta$ es manzana o es naranja. Esto lo podemos saber mediante una expresión relacional, por ejemplo $fruta$ = "manzana". Con esta expresión podremos conocer cuál es la fruta que se compra y, consecuentemente, el $costo$ de cada pieza. Una vez que conocemos el $costo$, sólo nos resta multiplicarlo por la cantidad de $piezas$ para conocer el $total$.

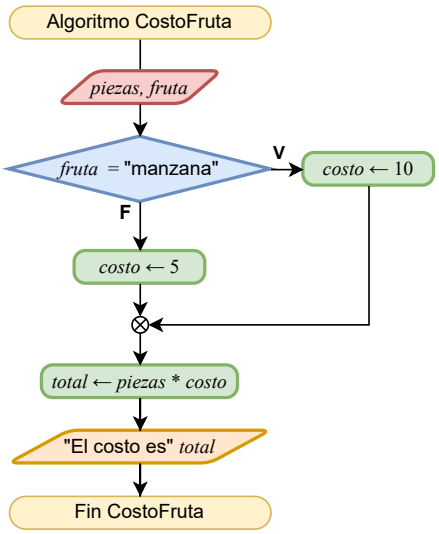


Figura 6.4: Diagrama de flujo para calcular el total de las frutas que quiere comprar un cliente.

Algoritmo 6.4: CostoFruta

```
1 leer piezas, fruta
2 si fruta = "manzana" entonces
3 |  costo ← 10
4 si no
5 |  costo ← 5
6 fin si
7 total ← costo * piezas
8 escribir "El total es" total
```

Este algoritmo se muestra en el diagrama de flujo de la Figura 6.4 y el pseudocódigo correspondiente en el Algoritmo 6.4. En este algoritmo, si la expresión de decisión $fruta$ = "manzana" es **Verdadero**, asigna a $costo$ el valor de 10, porque una manzana cuesta $10. Si la expresión resulta en **Falso**, entonces sabemos que la $fruta$ es naranja, por lo tanto se asigna 5 a $costo$.

Ejemplo 6.4. Utiliza una estructura de control selectiva simple para saber si un triángulo, cuyos lados tienen longitudes a, b y c, es equilátero.

Sabemos que un triángulo es equilátero si las longitudes de sus tres lados son iguales. Por lo tanto, tenemos que verificar esto mediante una expresión que combine operadores relacionales y lógicos. Por ejemplo, si queremos saber si dos lados tienen la misma longitud, podemos expresarlo como $a = b$ y si además requerimos que el otro lado también tenga la misma longitud, entonces $b = c$ no dará la respuesta. Por lo tanto, si estas dos expresiones son **Verdadero**, el triángulo es equilátero, por lo que debemos unirlas con un operador **Y** lógico. Notemos que, por la propiedad transitiva del operador =, no necesitamos verificar $a = c$. El algoritmo para decidir si un triángulo es equilátero dadas las longitudes de sus lados se muestra en el diagrama de flujo de la Figura 6.5. Primeramente, el algoritmo recibe las longitudes a, b y c de los tres lados del triángulo. Después, se evalúa si los tres lados tienen la misma longitud mediante la expresión $a = b \wedge b = c$ y el resultado se almacena en el identificador $esEquilatero$. En caso de que $esEquilatero$ sea **Verdadero**, a la respuesta $resp$ se le asigna que el triángulo es equilátero, si no, la respuesta será que no es un triángulo equilátero. El pseudocódigo correspondiente se muestra en el Algoritmo 6.5.

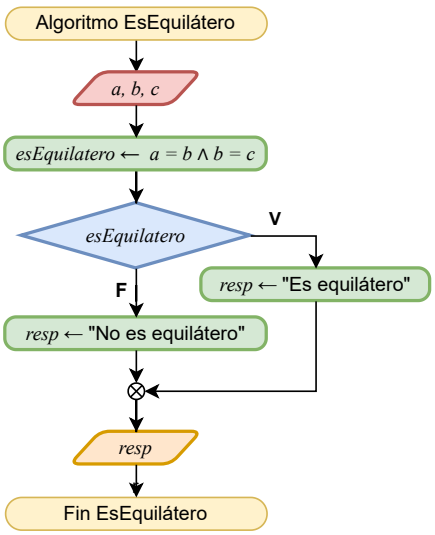


Figura 6.5: Diagrama de flujo para saber si un triángulo es equilátero.

Algoritmo 6.5: EsEquilátero

```
1 leer a, b, c
2 esEquilatero ← a = b ∧ b = c
3 si esEquilatero entonces
4 |   resp ← "Es equilátero"
5 si no
6 |   resp ← "No es equilátero"
7 fin si
8 escribir resp
```

6.2. Estructura de control selectiva múltiple

A diferencia de la estructura de control selectiva simple, la cual considera solamente una expresión de decisión para elegir entre dos posibles opciones para continuar con el flujo del algoritmo, la estructura de control selectiva múltiple evalúa más de una expresión para poder elegir de entre más de dos opciones.

Por ejemplo, si queremos comprar un helado en un lugar que ofrece tres diferentes tipos de sabores, tradicional, premium y de temporada, tenemos que conocer a qué categoría de las tres pertenece el sabor que pedimos para saber cuánto tenemos que pagar.

Esta estructura de control está ilustrada en el diagrama de flujo de la Figura 6.6. Como podemos observar, el diagrama de flujo contiene más de un rombo, cada uno con una expresión de decisión y, para cada una de ellas, se ejecutan instrucciones específicas en caso de que evalúen **Verdadero**. La instrucción $n+1$ se ejecutará cuando ninguna de las expresiones evalúe **Verdadero**. En algunos casos, ésta no es una opción, es decir, si ninguna de las expresiones evalúa **Verdadero**, se terminará la estructura de control.

La traducción a pseudocódigo de este diagrama de flujo se muestra en el Algoritmo 6.6. En este caso, las diferentes expresiones a partir de la segunda, se traducen a pseudocódigo con las palabras **si no si**, que indican que si no evaluó **Verdadero** la expresión anterior, se tiene que evaluar la que está enseguida.

Relación de las expresiones de decisión

Todas las expresiones de decisión que intervienen en una estructura de control selectiva múltiple deben estar relacionadas entre sí para evaluar todas las posibles opciones dentro del mismo contexto de decisión. Por ejemplo, para saber cuánto debemos pagar por un helado de alguna de las categorías de sabor tradicional, premium y de temporada, se deben considerar estas categorías y no el tamaño del helado.

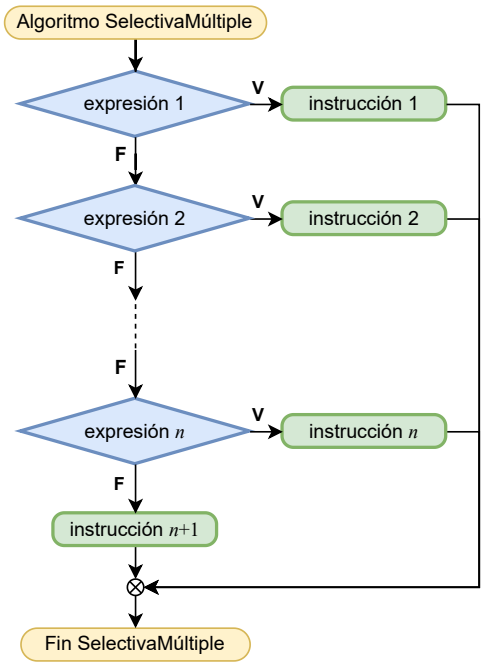


Figura 6.6: La estructura de control selectiva múltiple representada mediante un diagrama de flujo.

Algoritmo 6.6: SelectivaMúltiple

```
si expresión 1 evalúa Verdadero entonces
    instrucción 1
si no si expresión 2 evalúa Verdadero entonces
    instrucción 2
⋮
si no si expresión n evalúa Verdadero entonces
    instrucción n
si no
    instrucción n + 1
fin si
```

Ejemplo 6.5. Una heladería vende distintos sabores de helado y sus precios varían de acuerdo con la categoría de éstos: helado tradicional $70, helado premium $90 y helado de temporada $100. Estos precios son para el tamaño sencillo. Si una persona compra un helado de algún sabor de una de las tres categorías, ¿cuánto tiene que pagar?

Para resolver este problema tenemos que considerar, primero, que existen tres categorías de sabor de helado y que la categoría debe ser una de estas tres. Por lo tanto, al tener tres opciones para la $categoría$, utilizaremos una estructura de control selectiva múltiple. El diagrama de flujo de la Figura 6.7 muestra la solución a este problema.

El algoritmo comienza recibiendo el dato de entrada, en este caso $categoría$, la categoría de sabor del helado. Después, se verifica a qué categoría de sabor pertenece el helado mediante dos expresiones de decisión: $categoría$ = "tradicional" y $categoría$ = "premium". Si alguna de estas expresiones evalúa **Verdadero**, se asigna a $total$ el costo correspondiente del helado: 70 si es tradicional y 90 si es premium. En caso de que ninguna de las dos expresiones anteriores evalúe **Verdadero**, a $total$ se le asigna 100, porque sólo hay una tercera opción, que es la categoría de sabor de temporada.

El pseudocódigo correspondiente a este diagrama de flujo es el que se presenta en el Algoritmo 6.7. Al igual que en el diagrama de flujo, en el pseudocódigo se evalúan las dos expresiones para saber si la categoría del sabor es tradicional o premium. Notemos que, al no haber más opciones que evaluar, la única categoría de sabor que resta es la de temporada, por eso, en la línea 6 sólo escribimos un **si no**.

Algoritmo 6.7: CostoHelado

```
1 leer categoría
2 si categoría = "tradicional" entonces
3 |   total ← 70
4 si no si categoría = "premium" entonces
5 |   total ← 90
6 si no
7 |   total ← 100
8 fin si
9 escribir "El total es" total
```

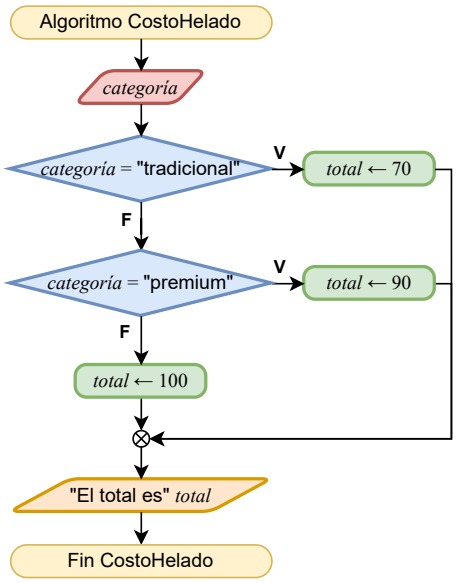


Figura 6.7: Diagrama de flujo para saber cuánto se tiene que pagar por un helado.

Ejemplo 6.6. Encontrar el máximo de los tres números a, b y c.

En el Ejemplo 6.2 resolvimos el problema de encontrar el máximo de dos números y la solución que propusimos fue comparar esos dos números directamente. Ahora, el problema considera tres números, pero la idea de la solución es la misma: comparar pares de números mediante algún operador relacional. Para resolverlo utilizando una estructura de control selectiva múltiple, necesitamos comparar cada par de números e identificar, mediante estas comparaciones, cuál es el máximo, como se muestra en el diagrama de flujo la Figura 6.8.

El algoritmo comienza recibiendo los valores de los tres números a, b y c. Después, para identificar cuál de los tres números es el máximo, se evalúan dos expresiones de decisión: $a \geq b \wedge a \geq c$ y $b \geq a \wedge b \geq c$. La primera evalúa si a es mayor o igual que b y que c, en caso afirmativo a sería el máximo. La segunda evalúa si b es mayor o igual que a y que c, de ser así, b es el máximo. Finalmente, si ninguna de las dos expresiones de decisión evaluaron **Verdadero**, es decir, si ni a ni b fueron el máximo, como tercera y última opción, se determina que c es el máximo. El pseudocódigo correspondiente a esta solución es el que se presenta en el Algoritmo 6.8.

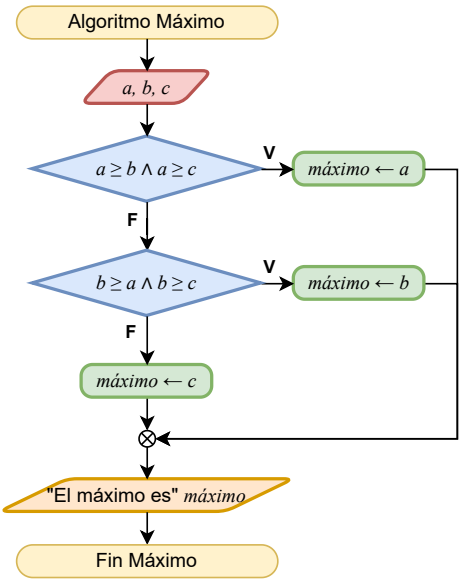


Figura 6.8: Diagrama de flujo para encontrar el máximo de tres números.

Algoritmo 6.8: Máximo

```
1 leer a, b, c
2 si a ≥ b ∧ a ≥ c entonces
3 |   máximo ← a
4 si no si b ≥ a ∧ b ≥ c entonces
5 |   máximo ← b
6 si no
7 |   máximo ← c
8 fin si
9 escribir "El máximo es" máximo
```

Ejemplo 6.7. Si los lados de un triángulo miden a, b y c, ¿qué tipo de triángulo es?

Sabemos que los triángulos se pueden clasificar, de acuerdo con las longitudes de sus lados, en: equilátero, que sus lados tienen la misma longitud; isósceles, que tiene dos lados con la misma longitud y la del tercero es diferente; y escaleno, que todos sus lados tienen diferentes longitudes. Por lo tanto, para saber qué tipo de triángulo es, podemos evaluar expresiones relacionales para identificar a cada tipo de triángulo. Es decir, si las longitudes de los lados del triángulo son iguales, entonces se debería cumplir que $a = b$ y que $a = c$. Por lo tanto esta condición nos servirá para saber que el triángulo

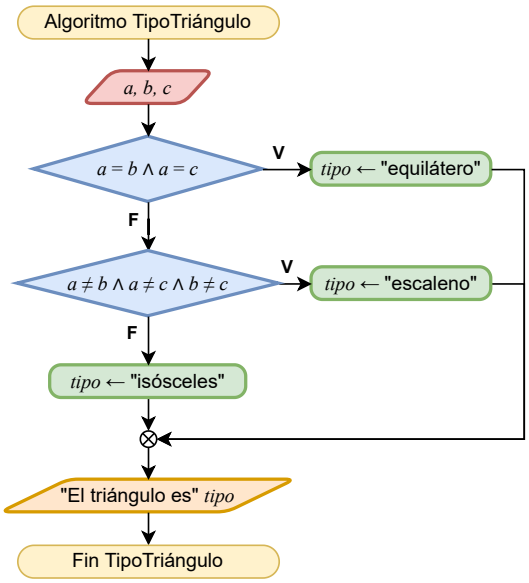


Figura 6.9: Diagrama de flujo para saber qué tipo de triángulo es dadas las longitudes de sus lados.

es equilátero. Notemos que, por la propiedad transitiva del operador =, no necesitamos saber el resultado de $b = c$. Por otro lado, si las longitudes de los lados del triángulo son diferentes, entonces, las expresiones $a \neq b$, $a \neq c$ y $b \neq c$ deben evaluar **Verdadero**. Finalmente, si las longitudes no cumplen con las condiciones anteriores y dado que sólo hay tres tipos de triángulos, entonces, podemos decir que se trata de un triángulo isósceles. Las condiciones anteriores se muestran en el diagrama de flujo de la Figura 6.9.

El algoritmo recibe las longitudes a, b y c de los lados del triángulo y posteriormente evalúa, primero, si los tres lados miden lo mismo mediante la expresión de decisión $a = b \wedge a = c$. En caso de que la expresión evalúe **Verdadero**, el triángulo es equilátero. En caso contrario, se verifica si las tres longitudes son diferentes, mediante la expresión $a \neq b \wedge a \neq c \wedge b \neq c$. En caso de que esta expresión evalúe **Verdadero**, el triángulo es escaleno. Si la segunda expresión también evaluó **Falso**, como última opción, el triángulo es isósceles.

El pseudocódigo correspondiente al diagrama de flujo de la Figura 6.9 se muestra en el Algoritmo 6.9.

Algoritmo 6.9: TipoTriángulo

```
1 leer a, b, c
2 si a = b ∧ a = c entonces
3 |   tipo ← "equilátero"
4 si no si a ≠ b ∧ a ≠ c ∧ b ≠ c entonces
5 |   tipo ← "escaleno"
6 si no
7 |   tipo ← "isósceles"
8 fin si
9 escribir "El triángulo es" tipo
```

Ejemplo 6.8. Dados dos números a y b, decir si a es divisor de b, si b es divisor de a o si no son divisores entre sí.

Para saber si un número es divisor de otro, la división del segundo entre el primero debe ser exacta, es decir, el residuo de la división debe ser cero. Entonces, debemos verificar si el residuo de alguna de las divisiones b/a y a/b es cero. Como sabemos, la operación módulo nos devuelve el residuo de una división, por lo tanto, podemos utilizarla para saber si un número es divisor de otro.

El diagrama de flujo de la Figura 6.10 considera dos expresiones de decisión. Como primer caso, se verifica de que a sea divisor de b, mediante la expresión b `MOD` $a = 0$, y en el segundo caso se evalúa si b es divisor de a, mediante la expresión a `MOD` $b = 0$. Si las dos expresiones evalúan **Falso**, quiere decir que los números no son divisores entre sí. El pseudocódigo correspondiente a este algoritmo es el que se muestra en al Algoritmo 6.10.

Algoritmo 6.10: Divisores

```
1 leer a, b
2 si b MOD a = 0 entonces
3 |   resp ← "a es divisor de b"
4 si no si a MOD b = 0 entonces
5 |   resp ← "b es divisor de a"
6 si no
7 |   resp ← "No son divisores"
8 fin si
9 escribir resp
```

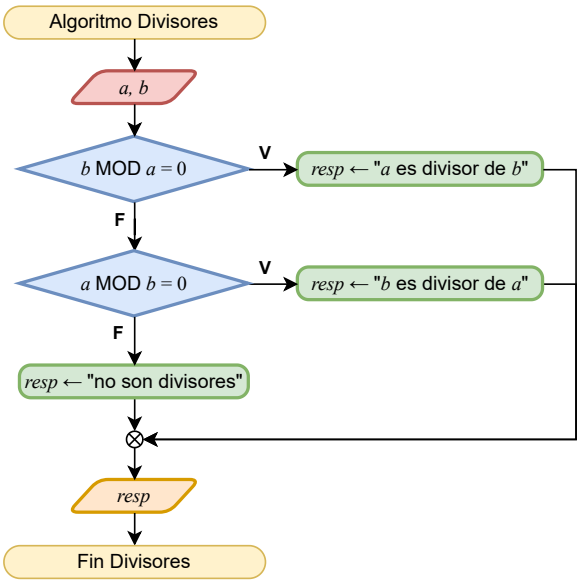


Figura 6.10: Diagrama de flujo para saber si "a es divisor de b", si "b es divisor de a" o si "no son divisores".

6.3. Estructura de control selectiva anidada

Existen problemas para los que se deben tomar diferentes decisiones, una después de otra, y cada una de estas considera las decisiones hechas anteriormente. Para resolver este tipo de problemas es conveniente utilizar la estructura de control selectiva anidada.

Por ejemplo, para saber cuál es el precio que debemos pagar por una computadora personal, primero, debemos saber la marca, después, seleccionar el modelo y, finalmente, las mejoras opcionales como el procesador, memoria, disco duro, etc.

Esta estructura de control está ilustrada en el diagrama de flujo de la Figura 6.11. Como podemos ver, a la izquierda del diagrama se muestran más de una expresión de decisión y, para cada una de ellas, hay una expresión de decisión adicional, para la cual se ejecutan instrucciones específicas en caso de que evalúen **Verdadero** o **Falso**. Por ejemplo, la expresión A considera que la expresión 1 evaluó **Verdadero**, la expresión B toma en cuenta que la expresión 2 evaluó **Verdadero** y así sucesivamente. La instrucción z se ejecutará cuando ninguna de las expresiones 1 a n evalúen **Verdadero**. En algunos casos, ésta no es una opción, es decir, si ninguna de las expresiones evalúa **Verdadero**, se terminará la estructura de control.

La representación de este diagrama de flujo en pseudocódigo se muestra en el Algoritmo 6.11. En este caso, es muy claro que, por cada opción de la estructura selectiva principal, se ejecuta

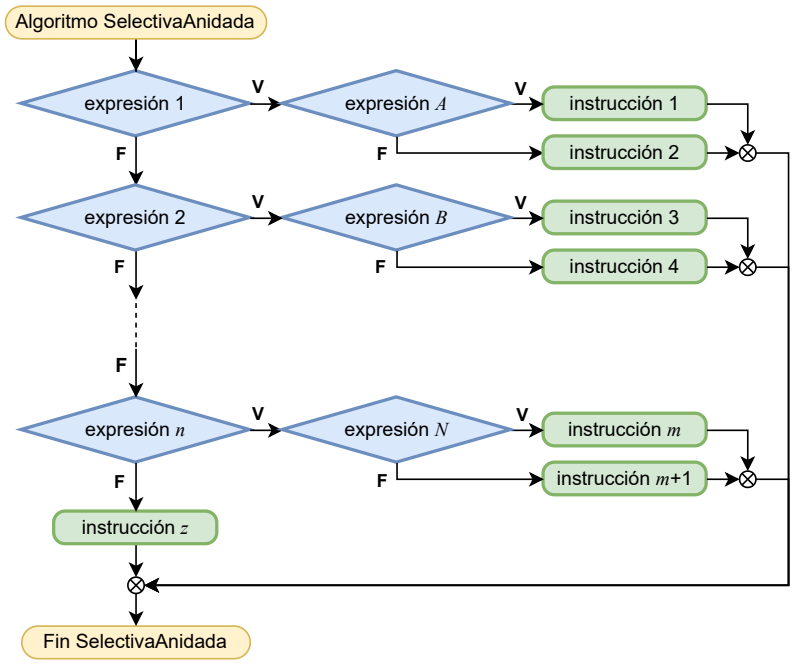


Figura 6.11: La estructura de control selectiva anidada representada mediante un diagrama de flujo.

un bloque diferente, el cual corresponde a otra estructura selectiva. Cuando se termina de ejecutar uno de estos bloques secundarios, el flujo debe continuar al finalizar la estructura selectiva principal. Por ejemplo, si la expresión 1 evaluó **Verdadero**, se debe ejecutar el bloque de instrucciones en las líneas 2-6 y continuar el flujo del algoritmo en la línea 22.

Ejemplo 6.9. Una tienda vende dos marcas de computadoras: la marca "D" y la marca "M". Cada computadora tiene dos modelos diferentes: el modelo "Ultra" y el modelo "Pro". El modelo "Ultra" de la marca "D" cuesta $20.000 y el de la marca "M" $25,000.00, mientras que el modelo "Pro" de la marca "D" cuesta $35,000 y el de la marca "M" cuesta $40,000. Diseña un algoritmo para saber cuánto tiene que pagar una persona que compra una de las computadoras.

Dado que ambas marcas consideran los mismos nombres para sus dos modelos de computadoras, para saber cuánto cuesta una de ellas, es necesario que debamos tomar dos decisiones: saber la marca y cuál es el modelo. Hay dos formas de resolver este problema: primero saber a qué marca pertenece la computadora que la persona desea comprar, es decir, si $marca$ es "D" o "M". Una vez que conozcamos la marca, procedemos a verificar cuál es el $modelo$ que se está llevando la persona. La segunda forma de resolverlo con-

Algoritmo 6.11: SelectivaAnidada

```
1  si expresión 1 evalúa Verdadero entonces
2  |  si expresión A evalúa Verdadero entonces
3  |  |  instrucción 1
4  |  si no
5  |  |  instrucción 2
6  |  fin si
7  si no si expresión 2 evalúa Verdadero entonces
8  |  si expresión B evalúa Verdadero entonces
9  |  |  instrucción 3
10 |  si no
11 |  |  instrucción 4
12 |  fin si
13 ⋮
14 si no si expresión n evalúa Verdadero entonces
15 |  si expresión N evalúa Verdadero entonces
16 |  |  instrucción m
17 |  si no
18 |  |  instrucción m + 1
19 |  fin si
20 si no
21 |  instrucción z
22 fin si
```

sidera, primero, identificar el modelo y despuésla marca. En este ejemplo, utilizaremos el primer enfoque.

Primero debemos identificar la marca de la computadora, para lo cual se evalúa la expresión $marca$ = "D". Si el resultado de esta evaluación es **Verdadero**, sabremos que la computadora es de esta marca y nos restaría conocer el modelo, lo cual podemos saber evaluando la expresión $modelo$ = "Ultra". Si evaluando esta segunda expresión obtenemos **Verdadero**, entonces se trata de la computadora marca "D" y modelo "Ultra", por lo tanto se asigna 20,000 a un identificador $costo$, pero si el resultado es **Falso**, entonces el modelo es "Pro" y se asigna 35,000 a $costo$. Por otro lado, si el resultado de evaluar la primera expresión ($marca$ = "D") es **Falso**, entonces es una computadora de la marca "M" y enseguida procedemos de forma similar a como se hizo con el caso de la marca "D". Este algoritmo se muestra en el diagrama de flujo de la Figura 6.12. Al utilizar pseudocódigo, este algoritmo se ve como se muestra en el Algoritmo 6.12

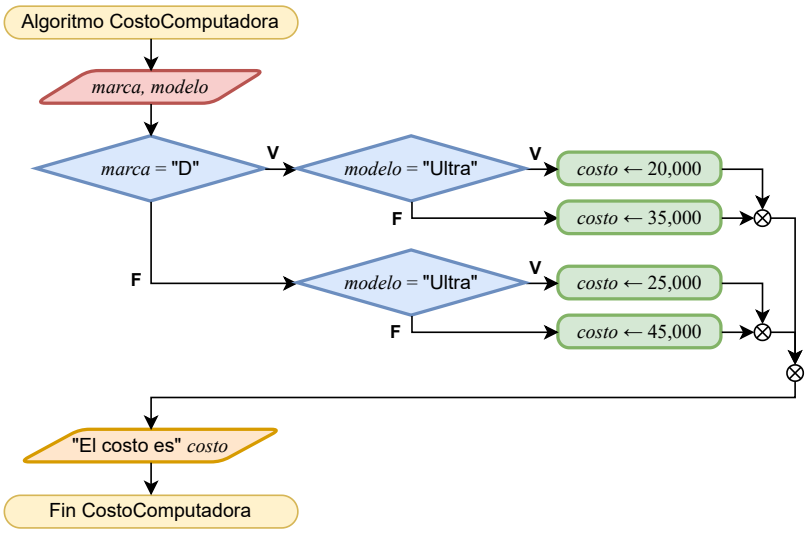


Figura 6.12: Diagrama de flujo para saber cuánto tiene que pagar un cliente que compra una computadora.

Algoritmo 6.12: CostoComputadora

```
1  leer marca, modelo
2  si marca = "D" entonces
3  |  si modelo = "Ultra" entonces
4  |  |  costo ← 20,000
5  |  si no
6  |  |  costo ← 30,000
7  |  fin si
8  si no
9  |  si modelo = "Ultra" entonces
10 |  |  costo ← 25,000
11 |  si no
12 |  |  costo ← 40,000
13 |  fin si
14 fin si
15 escribir "El costo es" costo
```

Ejemplo 6.10. De acuerdo con el clima y la humedad del día, se pueden practicar las actividades indicadas en la siguiente tabla.

		HUMEDAD	
		Alta	Baja
CLIMA	Cálido	Nadar	Tenis
	Frío	Básquetbol	Ver TV

Dado el clima y la humedad actuales, indicar cuál es la actividad recomendada.

En este caso, para poder recomendar una actividad, tenemos que conocer el $clima$ y la $humedad$ presentes. Podemos comenzar evaluando el $clima$ para saber si es cálido ($clima$ = "cálido"). En caso de que si, verificamos ahora si la $humedad$ es alta ($humedad$ = "alta"). Si es el caso, la actividad recomendada es nadar, pero si no, entonces se recomienda el tenis. En caso de que el $clima$ no sea cálido, es decir, el clima es frío, verificamos si la $humedad$ es alta. En caso de que si, se recomienda el básquetbol, pero si no, se recomienda ver TV. Este algoritmo está representado en el diagrama de flujo de la Figura 6.13 y en el pseudocódigo del Algoritmo 6.13.

Algoritmo 6.13: RecomendarActividad

```
1  leer cima, humedad
2  si clima = "cálido" entonces
3  |   si humedad = "alta" entonces
4  |   |   actividad ← "nadar"
5  |   si no
6  |   |   actividad ← "tenis"
7  |   fin si
8  si no
9  |   si humedad = "alta" entonces
10 |   |   actividad ← "básquetbol"
11 |   si no
12 |   |   actividad ← "ver TV"
13 |   fin si
14 fin si
15 escribir "Recomiendo" actividad
```

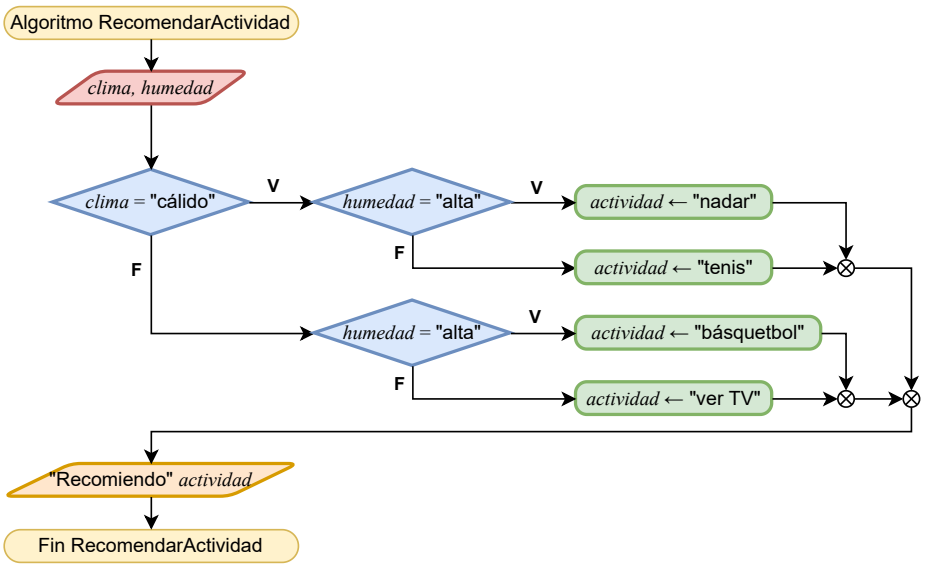


Figura 6.13: Diagrama de flujo para recomendar una actividad de acuerdo con el clima y la humedad.

Ejemplo 6.11. El año se divide en cuatro estaciones: primavera, verano, otoño e invierno. Debido a que las fechas exactas en que cambian las estaciones varían un poco de un año a otro debido a la forma en que se construye el calendario, utilizaremos las siguientes fechas para este ejercicio:

Estación	Fecha de inicio
Primavera	20 de marzo
Verano	21 de junio
Otoño	22 de septiembre
Invierno	21 de diciembre

Dada una fecha en mes y día, ¿a qué estación está asociada la fecha?

Como lo establece el problema, como datos de entrada tenemos un mes y un $día$ que representan una fecha. A pesar de que el mes se puede representar como un tipo entero, en este ejemplo lo representaremos como un tipo cadena de caracteres, siendo el rango válido todas las cadenas que correspondan a un mes del año. Por otro lado, el $día$ debe ser un tipo entero. Para saber a qué estación corresponde esta fecha, debemos tomar

en cuenta los períodos de cada estación. Es decir, la primavera va del 20 de marzo al 20 de junio, el verano va del 21 de junio al 21 de septiembre, el otoño va del 22 de septiembre al 20 de diciembre y el invierno va del 21 de diciembre al 19 de marzo. Con estos períodos podemos ver que hay meses completos que pertenecen a alguna de las estaciones: abril y mayo a la primavera, julio y agosto al verano, octubre y noviembre al otoño, y enero y febrero al invierno. Los cuatro meses que faltan corresponden a dos estaciones cada uno: marzo al invierno y a la primavera, junio a la primavera y al verano, septiembre al verano y al otoño, y diciembre al otoño y al invierno. Por lo tanto, para poder decir a qué estación corresponde una fecha, primero debemos verificar el mes. Si el mes no está dividido entre dos estaciones, sabremos inmediatamente la estación, pero si el mes es uno de los cuatro que corresponde a dos estaciones, entonces debemos verificar el día y compararlo con el día del inicio de la estación.

La solución descrita anteriormente está representada en los algoritmos mostrados en la Figura 6.14 y en el Algoritmo 6.14. Estos algoritmos consideran una estructura selectiva múltiple principal, la cual considera ocho opciones: cuatro para los ocho meses que corresponden a una sola estación y cuatro más para los meses que corresponden a dos estaciones. Para estos cuatro meses se considera una estructura selectiva anidada, en donde se verifica el día del mes para decidir a cuál estación corresponde.

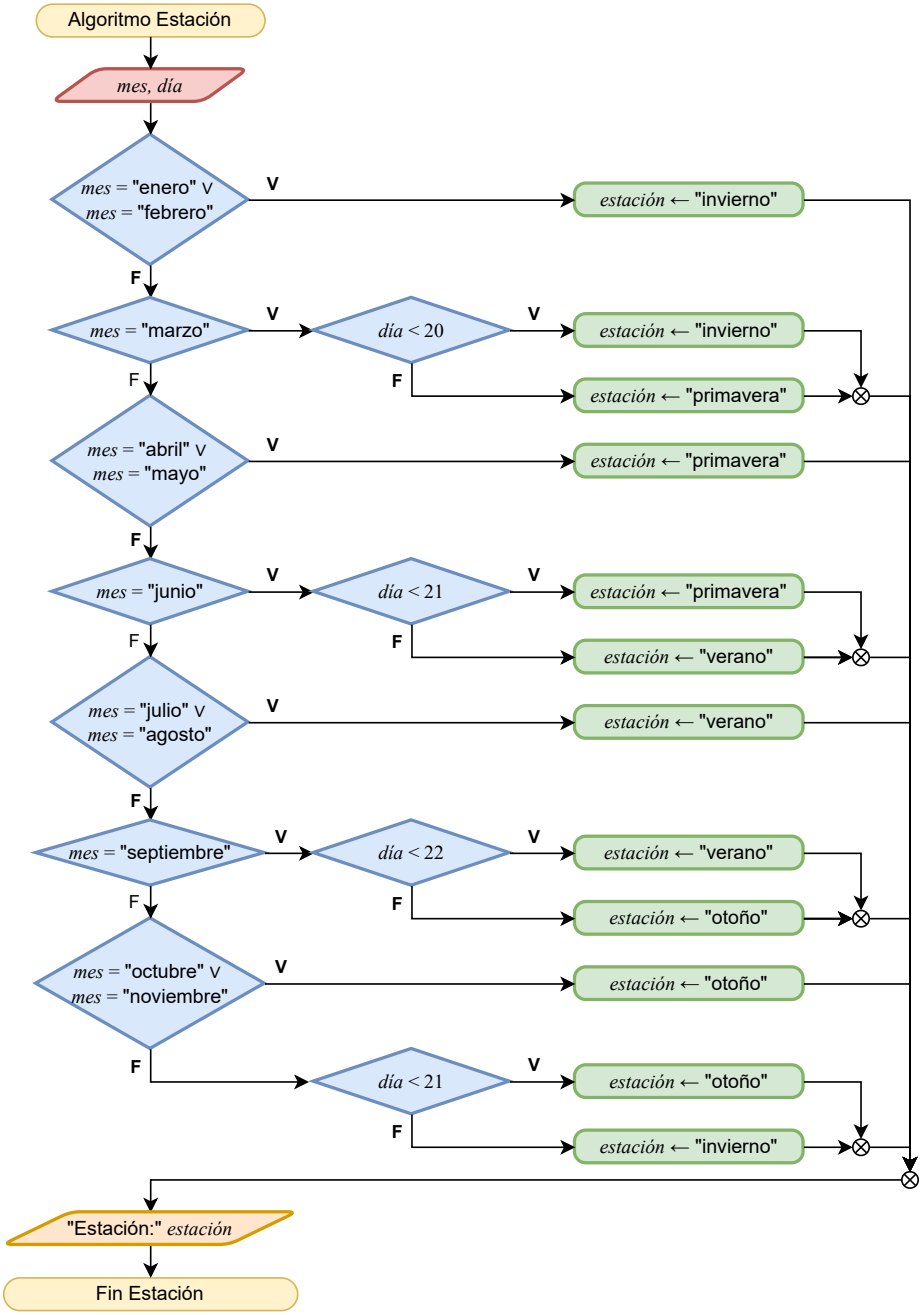


Figura 6.14: Diagrama de flujo para saber a qué estación corresponde una fecha.

Algoritmo 6.14: Estación

```
1  leer mes, día
2  si mes = "enero" ∨ mes = "febrero" entonces
3  |  estación ← "invierno"
4  si no si mes = "marzo" entonces
5  |  si día < 20 entonces
6  |  |  estación ← "invierno"
7  |  si no
8  |  |  estación ← "primavera"
9  |  fin si
10 si no si mes = "abril" ∨ mes = "mayo" entonces
11 |  estación ← "primavera"
12 si no si mes = "junio" entonces
13 |  si día < 21 entonces
14 |  |  estación ← "primavera"
15 |  si no
16 |  |  estación ← "verano"
17 |  fin si
18 si no si mes = "julio" ∨ mes = "agosto" entonces
19 |  estación ← "verano"
20 si no si mes = "septiembre" entonces
21 |  si día < 22 entonces
22 |  |  estación ← "verano"
23 |  si no
24 |  |  estación ← "otoño"
25 |  fin si
26 si no si mes = "octubre" ∨ mes = "noviembre" entonces
27 |  estación ← "otoño"
28 si no si mes = "diciembte" entonces
29 |  si día < 22 entonces
30 |  |  estación ← "otoño"
31 |  si no
32 |  |  estación ← "invierno"
33 |  fin si
34 escribir "Estación" estación
```

6.4. Ejercicios

Resuelve los siguientes ejercicios utilizando estructuras de control selectivas.

Ejercicio 6.1. Una frutería vende manzanas a $45 el kilo y naranjas a $25 el kilo. Si un cliente compra cierta cantidad de kilos de alguna de las dos frutas, ¿cuánto tiene que pagar?

Ejercicio 6.2. La calificación final de un curso es el promedio de las calificaciones parciales $calif1$ y $calif2$. Para que se tenga derecho a calificación final, ambas calificaciones parciales deben ser aprobatorias (mayores o iguales que 6) y el porcentaje de asistencia a clase debe ser al menos de 80 %. ¿Cuál es la calificación final de una persona que asistió a cierto $porcentaje$ de las clases del curso?

Ejercicio 6.3. Determinar si un número x es positivo, negativo o cero.

Ejercicio 6.4. La tienda departamental "Manchester" tiene la siguiente promoción: todas las corbatas que tienen un precio superior a $1,000.00 tienen un descuento del 15 % y todas los demás tienen un descuento del 10 %. ¿Cuál es el precio final que debe pagar una persona que compra una corbata y de cuánto es el descuento que obtendrá?

Ejercicio 6.5. En la tienda "Mascotas Felices" venden diferentes marcas de alimento para perro y todas las semanas ofrecen descuentos. Por ejemplo, esta semana, las marcas "Ciencia de la Dieta", "Rey Perro" y "Plan Profesional" tienen 20 % de descuento. Si una persona compra alimento de alguna marca, ¿cuál es el descuento que le hacen?

Ejercicio 6.6. Comúnmente se dice que un año humano es equivalente a 7 años caninos. Esta simple conversión no reconoce que los perros alcanzan la edad adulta en aproximadamente dos años y que esto también depende de su raza. Sin embargo, el Club Canino Americano[10] lo analiza de la siguiente manera:

- El primer año de vida de un perro de una raza mediana equivale a 15 años humanos.
- El segundo año para un perro equivale a nueve años para un humano.
- Después de eso, cada año humano sería aproximadamente cinco años para un perro.

Tomando en cuenta la conversión descrita, ¿a cuántos años caninos equivalen x años humanos?

[10] How to Calculate Dog Years to Human Years. American Kennel Club. URL: https://www.akc.org/expert-advice/health/how-to-calculate-dog-years-to-human-years/. Fecha de última visita: 22/09/2023.

Ejercicio 6.7. Nuestro planeta gira 365.24219 veces sobre su eje durante una órbita completa alrededor del sol, por lo tanto un año dura 365 días, 5 horas, 48 minutos y 56 segundos, y no únicamente 365 días. Con el fin de corregir este error, al emperador Julio César[11] se le ocurrió crear el año bisiesto. Si cada año nosotros sólo contamos 365 días, perdemos esas más de 5 horas, que de alguna forma debemos recuperar. Durante tres años contamos 365 y al cuarto, el año bisiesto, recuperamos el día que falta, el 29 de febrero.

Fue en el año 44 a. C., al adaptarse al calendario juliano, cuando los años pasaron a tener 365 días, divididos en doce meses de 30 o 31 días, salvo febrero con 28. Siendo conscientes los romanos de que los 365 días no eran un cálculo exacto, cada cuatro años añadían un día más al calendario.

Posteriormente, en el año 1582, el calendario gregoriano sustituyó al juliano y ajustó un poco más el desfase que todavía existía con el calendario juliano, añadiendo excepciones a los años bisiestos: no lo serán los años múltiplos de 100, salvo si son también divisibles entre 400. Por este motivo, el año 1900, que debería haber sido año bisiesto, no lo fue (es múltiplo de 100 y no es divisible entre 400). Y el año 2000, que es múltiplo de 100 y también es divisible entre 400, sí lo fue. Del mismo modo, los años 2100 y 2200 no serán años bisiestos.

Siguiendo las reglas anteriores, indicar si un año es bisiesto.

Ejercicio 6.8. Una frutería vende manzanas a \$45 el kilo, peras a \$40 el kilo y naranjas a \$25 el kilo. Si una persona compra cierta cantidad de kilos de alguna de las frutas, ¿cuánto tiene que pagar?

Ejercicio 6.9. A partir de un peso en kilogramos y una estatura en metros dadas, el índice de masa corporal (IMC) se calcula como

$$IMC = \frac{peso}{estatura^2} \ .$$

Indicar si una persona tiene "bajo peso", si el IMC es menor que 18.5, "peso normal", si el IMC está entre 18.5 y 25, "sobrepeso", si el IMC es mayor que 25 y es menor que 30, u "obesidad", si el IMC es mayor que 30.

Ejercicio 6.10. Dado un mes del año, indicar cuántos días tiene.

Ejercicio 6.11. ¿Cómo se llama la figura geométrica que tiene n lados? Considera que n va de 3 a 8.

[11] ¿Cuáles son los orígenes del año bisiesto? National Geographic. URL: https://www.nationalgeographic.es/historia/origen-ano-bisiesto. Fecha de última visita: 22/09/2023.

Ejercicio 6.12. ¿Cuál es el proceso histórico o personaje que aparece en un billete de la familia G emitido por el Banco de México[12]?

Ejercicio 6.13. La siguiente tabla contiene rangos de magnitud de sismos en la escala de Richter y sus descriptores:

Magnitud	Descriptor
Menos de 2.0	Micro
De 2.0 a menos de 4.0	Menor
De 4.0 a menos de 5.0	Ligero
De 5.0 a menos de 6.0	Moderado
De 6.0 a menos de 7.0	Fuerte
De 7.0 a menos de 8.0	Mayor
De 8.0 a menos de 10.0	Cataclismo
Más de 10.0	Legendario

Dada la magnitud de un sismo, indicar el descriptor de éste.

Ejercicio 6.14. Doña Lore tiene una cocina económica que vende comidas de tres tiempos. En el primer tiempo ofrece sopa de lentejas a $20, crema de elote a $25 y sopa de fideos a $15. En el segundo tiempo tiene arroz a $15, espagueti a $20 y ensalada a $10. En el tercer tiempo tiene pollo con papas a $35, bistec en morita a $45 y chile relleno a $30. ¿Cuánto tiene que pagar una persona que elige una opción de cada tiempo?

Ejercicio 6.15. Una frutería vende un kilo de manzana Golden Delicious a $46 y el de manzana Red Delicious a $50, un kilo de naranja Washington cuesta $19 y el de naranja Valencia cuesta $26, un kilo de pera d'Anjou cuesta $47 y el de pera Bosc cuesta $42. Si una persona compra cierta cantidad de kilos de una variedad de alguna de las frutas, ¿cuánto tiene que pagar?

Ejercicio 6.16. Dado un billete emitido por el Banco de México, ¿cuál es el proceso histórico o personaje que aparece en dicho billete si conocemos su denominación y familia? Considera que la familia de billetes puede ser F[13] o G.

[12] Diseños actuales. Banco de México. URL: https://www.banxico.org.mx/billetes-y-monedas/disenos-actuales-circulacion-.html. Fecha de última visita: 22/09/2023.

[13] Otros diseños. Banco de México. URL: https://www.banxico.org.mx/billetes-y-monedas/otros-disenos-circulacion-ban.html. Fecha de última visita: 22/09/2023.

Ejercicio 6.17. Dados un mes y un año, ¿cuántos días tiene el mes? Considera los años bisiestos. Recordatorio: un año bisiesto es múltiplo de 4 pero no de 100 o es múltiplo de 400.

Ejercicio 6.18. Dada una fecha, calcular la fecha del siguiente día. Por ejemplo, si la fecha es 2022-10-30, entonces la fecha del siguiente día es 2022-10-31. Si la fecha es 2022-10-31, entonces el día siguiente es 2022-11-01. Si la fecha es 2022-12-31 el día siguiente es 2023-01-01. Considera que la fecha tiene formato con tres valores numéricos: año, mes y día. Asegúrate de que tu algoritmo funcione correctamente para los años bisiestos.

Ejercicio 6.19. En la Ciudad de México, la Comisión Federal de Electricidad (CFE) cobra el servicio de energía eléctrica de uso doméstico de acuerdo con las unidades de KWh consumidas en el bimestre[14]:

- Consumo básico, hasta 150 KWh, a $0.945 por unidad.
- Consumo intermedio, mayor a 150 y hasta 280 KWh, a $1.153 por unidad.
- Consumo excedente, mayor a 280 y hasta 500 KWh, a $3.367 por unidad.
- Tarifa Doméstica de Alto Consumo (DAC), mayor a 500 KWh, a $6.741 por unidad.

En el caso del consumo intermedio, se cobra lo correspondiente al consumo básico (150 KWh), más las unidades que correspondan al consumo intermedio (lo que sobrepase a 150 KWh). En el caso del consumo excedente, se cobran las unidades correspondientes a los consumos básico (150 KWh) e intermedio (130 KWh), más las unidades que corresponden al consumo excedente (lo que sobrepase a 280 KWh). Finalmente, en el caso de la tarifa DAC, todo el consumo se cobra con la tarifa indicada. Se debe tomar en cuenta que la CFE considera un consumo mínimo mensual de 25 KWh. ¿Cuánto debe pagar una persona si se conoce su consumo de energía en KWh?

[14] Esquema tarifario vigente. Comisión Federal de Electricidad. URL: https://app.cfe.mx/Aplicaciones/CCFE/Tarifas/TarifasCRECasa/Casa.aspx. Fecha de última visita: 22/09/2023.

7 Estructura de control iterativa

La estructura de control iterativa (o de repetición), como su nombre lo indica, permite ejecutar más de una vez una instrucción o un conjunto de ellas. Esta estructura de control es especialmente útil para evitar repetir las instrucciones explícitamente cuando se deben ejecutar varias veces. Por ejemplo, si queremos preparar 15 tazas de café, se debe realizar 15 veces el mismo procedimiento por cada taza que preparemos. Por supuesto, podemos escribir las mismas instrucciones las 15 veces que preparemos el café o podemos escribir una sola vez ese conjunto de instrucciones en una estructura de control iterativa para que las repita 15 veces.

Existen tres modalidades de la estructura de control iterativa:

- Control previo.
- Control posterior.
- Número predeterminado de iteraciones.

En este capítulo nos enfocaremos en la estructura de control iterativa con **control previo**, ya que es una generalización de este tipo de estructuras.

7.1. Estructura de control iterativa con control previo

La estructura de control iterativa con control previo permite ejecutar de forma cíclica un bloque de instrucciones, siempre y cuando se satisfaga cierta *condición de control*, la cual está representada como una expresión lógica o relacional o como una combinación de ambas. El ciclo se repetirá mientras dicha expresión, que representa la condición de control, devuelva como resultado un valor **Verdadero**. Generalmente, la estructura de control iterativa con control previo contiene una instrucción que modifica el valor de alguno de los identificadores incluidos en la expresión de control, de tal forma que eventualmente la condición deje de evaluar **Verdadero**. Evidentemente, no incluir una instrucción de esta naturaleza provocará que el ciclo no termine y que se ejecuten las instrucciones de forma indefinida.

Para comprender mejor qué es y cómo funciona la estructura iterativa con control previo, observemos el siguiente ejemplo.

Ejemplo 7.1. Una persona adulta debe cruzar un río ancho y profundo con ningún puente a la vista. Se da cuenta de que hay dos niños jugando con un bote de remos en la orilla. El bote es tan pequeño que solo puede transportar, a lo más, a dos niños o a un adulto. ¿Cómo puede la persona cruzar el río y dejar a los niños en posesión del bote?

Analizando el problema, seguramente llegaremos a una solución parecida a la siguiente:

1. Cruzan el río los dos niños.
2. Cruza el río un niño.
3. Cruza el río la persona.
4. Cruza el río el otro niño.

De esta manera, además de que la persona cruzó el río, cumplimos con la restricción de dejar a los niños en posesión del bote.

Ahora supongamos que no sólo es una persona la que desea cruzar el río, sino cinco. ¿Cómo podemos resolver este problema? Simple: repetimos el bloque de instrucciones anterior cinco veces. ¿Y si son 25 personas? Entonces, deberemos repetirlo 25 veces. Es decir, el bloque de instrucciones se deberá ejecutar por cada persona que desee cruzar el río.

La forma de representar la estructura de control iterativa en un diagrama de flujo es similar al de una estructura selectiva, que es mediante un rombo y un símbolo unión (⊗). Sin embargo, la diferencia radica en que, después de ejecutar el bloque de instrucciones que se repetirá, el flujo del algoritmo regresará a un punto antes del rombo, precisamente para que se evalúe nuevamente la condición de control. El diagrama de flujo de la Figura 7.1 muestra un ejemplo de este tipo de estructura. Al igual que la estructura de selección, cuando el flujo del algoritmo alcanza el rombo, se tiene que evaluar una expresión, la cual está indicada dentro del mismo, y es la que llamamos *condición de control*. En este punto, el flujo del algoritmo tiene dos opciones para continuar de acuerdo con el resultado de la evaluación de la expresión, **Falso** (**F**) y **Verdadero** (**V**). En caso de que la evaluación resulte **Verdadero**, entonces el flujo del algoritmo ejecutará las instrucciones dentro de la estructura de control iterativa y el flujo regresará al símbolo de unión que está justo antes del rombo. En consecuencia, la expresión de control se evaluará nuevamente. En caso de que la evaluación de la expresión resulte **Falso**, el algoritmo deja de ejecutar las instrucciones dentro del ciclo y continua con el flujo como lo indica la flecha correspondiente.

La estructura de control iterativa se escribe en pseudocódigo como se muestra en el Algoritmo 7.1. El rombo del diagrama de flujo se traduce a pseudocódigo con la palabra **mientras**, enseguida se escribe la expresión de la condición de control y el enunciado termina con la palabra **hacer**. Después, se escriben todas las instrucciones que se ejecutarán dentro del ciclo

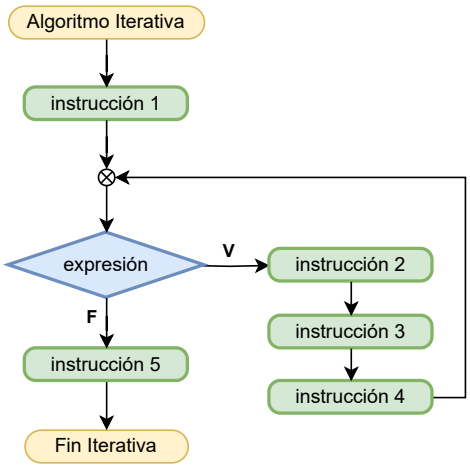


Figura 7.1: En un diagrama de flujo, la estructura de control iterativa se representa mediante un rombo, un símbolo de unión y una flecha que regresa antes del rombo.

Algoritmo 7.1: Iterativa

```
1 instrucción 1
2 mientras expresión evalúa Verdadero hacer
3 |   instrucción 2
4 |   instrucción 3
5 |   instrucción 4
6 fin mientras
7 instrucción 5
```

y la estructura termina con las palabras **fin mientras**. De esta forma, cuando un algoritmo contiene una estructura de control iterativa de este tipo, se debe evaluar la expresión de la condición de control y se ejecutarán las instrucciones dentro de la estructura mientras esta condición evalúe **Verdadero**. En el momento en que la condción de control evalúe **Falso**, el flujo "saltará" las instrucciones dentro de la estructura iterativa y continuará ejecutando la instrucción que aparezca después del **fin mientras**.

Es importante destacar que la expresión de la condición de control no cambia, pero los identificadores que están involucrados en este expresión si pueden cambiar de valor. Entonces, es común que una de las instrucciones que se repiten dentro del ciclo, esté relacionada con una asignación al menos a uno de estos identificadores.

Ejemplo 7.2. La división es una operación matemática que consiste en conocer cuántas veces cabe un número, llamado divisor, en otro, llamado dividendo, y cuyo resultado recibe el nombre de cociente. En el caso de la división entera, se puede tener un resto o residuo, resultado de que el divisor no cabe exactamente un número entero de veces en el dividendo.

La división entera se puede calcular utilizando restas sucesivas, es decir, contando el número de veces que se puede restar el divisor al dividendo y tomando el resultado de la última resta como el residuo. Por ejemplo, si dividimos 14 entre 3, podemos restar el 3 cuatro veces a 14, por lo que el cociente será 4 y el residuo 2. Estos resultados se obtienen de la siguiente manera:

resta 1	resta 2	resta 3	resta 4	
				⟵ cociente
$-\ 14$	$-\ 11$	$-\ 8$	$-\ 5$	
$\ \ 3$	$\ \ 3$	$\ \ 3$	$\ \ 3$	
11	8	5	2	⟵ residuo

Entonces, para diseñar este algoritmo, podemos identificar que los datos de entrada son, precisamente, el $dividendo$ y el $divisor$, de tipo entero, y que las instrucciones que se deben repetir en el ciclo son la resta del $dividendo$ menos el $divisor$ y llevar la cuenta del número de restas que se han hecho, utilizando un identificador auxiliar $cociente$, también de tipo entero. Esta resta se debe dejar de hacer cuando el $divisor$ ya no se pueda restar del $dividendo$, es decir, cuando $dividendo$ sea menor que $divisor$. En otras palabras, la resta se debe hacer mientras $dividendo \geq divisor$ sea **Verdadero**. Debido a que $cociente$ llevará la cuenta de las restas que se han hecho, se debe inicializar en cero antes de la estructura iterativa, precisamente porque no se han hecho restas. Este algoritmo está representado en el diagrama de flujo de la Figura 7.2 y en el pseudocódigo del Algoritmo 7.2.

Algoritmo 7.2: DivisionResta

```
1 leer dividendo, divisor
2 cociente ← 0
3 mientras dividendo ≥ divisor hacer
4 |   dividendo ← dividendo − divisor
5 |   cociente ← cociente + 1
6 fin mientras
7 residuo ← dividendo
8 escribir "El cociente es:" cociente
9 escribir "El residuo es:" residuo
```

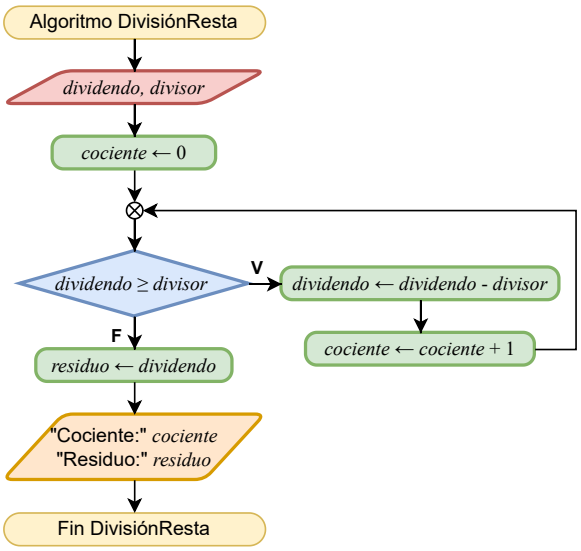


Figura 7.2: Diagrama de flujo para una división mediante restas sucesivas.

Como se dijo anteriormente, la expresión de la condición de control no cambia, pero si podría cambiar el valor asignado a alguno de los identificadores involucrados en la expresión. En el Ejemplo 7.2, el valor asignado a *dividendo* cambia en cada ciclo de la estructura de control iterativa. Dado que en cada iteración a *dividendo* se le resta *divisor*, el valor asignado a *dividendo* decrementará y, eventualmente, será menor que *divisor*. Consecuentemente, la condición de control evaluará **Falso** y la ejecución de la estructura de control iterativa terminará.

Después de haber presentado este ejemplo, es importante presentar una estrategia para diseñar una estructura de control iterativa, la cual consiste en determinar lo siguiente:

- Las instrucciones que se repetirán dentro de la estructura iterativa.
- La condición de paro.
- Los valores de inicialización de los identificadores que intervienen en la condición de paro.

Inicialización

Instrucción que asigna un valor por primera vez a un identificador.

Inicialización en la estructura iterativa

La inicialización es muy importante en la estructura de control iterativa, ya que podríamos tratar de evaluar la expresión de la condición de control sin que los identificadores que intervienen en ella no tengan un valor asignado.

Ejemplo 7.3. El máximo común divisor (MCD) de dos o más números enteros es el número mayor que los divide exactamente, es decir, que al dividirlos entre el MCD el residuo es cero. Un método sencillo para encontrar el MCD de dos números es elegir el menor de ambos y, a partir de él, buscar algún número menor que los divida exactamente.

Consideremos que queremos encontrar el MCD de los números a y b, que son los datos de entrada. Primero, tenemos que saber cuál de los dos números, a o b, es el menor. Esto lo podemos lograr mediante una expresión relacional, por ejemplo, $a \leq b$. Si el resultado de esta expresión es **Verdadero**, entonces el menor es a, si no, el menor es b. El número menor lo asignaremos provisionalmente a un identificador mcd. Esta asignación es, de hecho, la inicialización del identificador que aparecerá en la expresión de la condición de control de la estructura iterativa. A partir de este número, tenemos que verificar si a y b son divisibles entre mcd. Si sí son divisibles, encontramos el MCD de ambos números, por lo que estas condiciones deben ser parte de la condición de control de la estructura iterativa. En caso de que a y b no sean divisibles entre mcd, entonces el ciclo debe continuar.

De acuerdo con lo anterior, la expresión de la condición de control será a `MOD` $mcd \neq 0 \ \wedge \ b$ `MOD` $mcd \neq 0$. En caso de que la condición de control evaúe **Falso**, debemos decrementar en uno el valor de mcd, es decir, $mcd \leftarrow mcd - 1$, y verificar nuevamente. Este decremento se debe realizar en cada iteración de la estructura de control iterativa para que asigne un nuevo valor a mcd. Este algoritmo se muestra en el diagrama de flujo de la Figura 7.3 y en el pseudocódigo del Algoritmo 7.3.

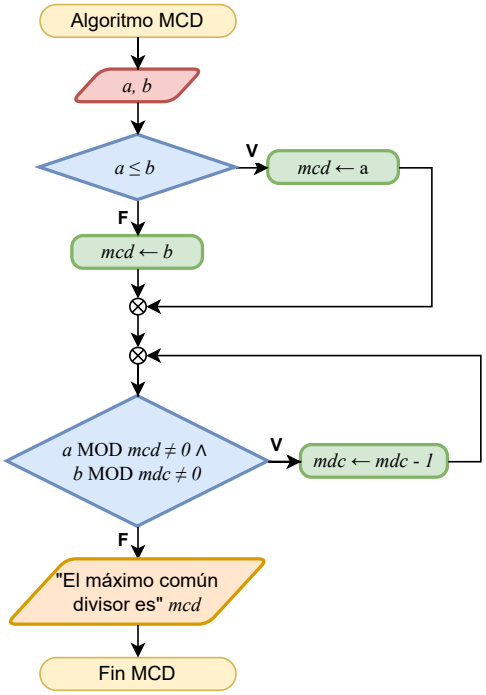


Figura 7.3: Diagrama de flujo para encontrar en máximo común divisor de dos números.

Algoritmo 7.3: MCD

```
1  leer a, b
2  si a ≤ b entonces
3  |  mcd ← a
4  si no
5  |  mcd ← b
6  fin si
7  mientras (a MOD mcd ≠ 0) ∧ (b MOD mcd ≠ 0) hacer
8  |  mcd ← mcd − 1
9  fin mientras
10 escribir "El máximo común divisor es" mcd
```

Identificadores auxiliares

Generalmente, la estructura de control iterativa considera identificadores auxiliares en la expresión de la condición de control. Estos identificadores permiten que un valor se pueda actualizar en cada iteración y, por lo tanto, que eventualmente la condición de control evalúe **Falso**.

Ejemplo 7.4. La calificación final de un curso es el promedio de cierto número de exámenes parciales. Para que se tenga derecho a una calificación final, el promedio de las calificaciones parciales debe ser aprobatorio, es decir, mayor o igual que 6. ¿Cuál es la calificación final de una persona si se conocen sus calificaciones parciales?

En este caso, para poder calcular el promedio de las calificaciones parciales, es preciso contar con estas como datos de entrada. Por esta razón, además, lo primero que necesitamos saber es el número de calificaciones parciales (num_calif, de tipo entero). Para calcular el promedio, debemos sumar las num_calif calificaciones parciales y después dividir la suma entre num_calif.

De esta forma, podemos usar una estructura de control iterativa que, en cada iteración, reciba una calificación parcial ($calificación$, de tipo real) y la sume a las anteriores, a saber, $suma \leftarrow suma + calificación$. Además, debemos llevar el registro de cuántas calificaciones se han recibido y sumado, por lo que usaremos un identificador auxiliar i, el cual se incrementará en cada iteración, esto es, $i \leftarrow i+1$. Así, se dejará de solicitar calificaciones cuando el contador i exceda el número de calificaciones parciales num_calif, en otras palabras, cuando ya no se cumpla que $i < num\_calif$.

Cuando se hayan recibido y sumado las num_calif calificaciones, la $suma$ se deberá dividir entre num_calif y se asignará el resultado a un identificador $promedio$, de tipo real. Finalmente, se evaluará si $promedio \geq 6$, de ser **Verdadero**, se devuelve la calificación calculada, de lo contrario, no se tiene derecho a calificación.

Este algoritmo está representado en el diagrama de flujo de la Figura 7.4 y en el pseudocódigo del Algoritmo 7.4. En este algoritmo, después de recibir el número de calificaciones parciales num_calif, se inicializan los identificadores $suma$ e i en cero, porque no se ha recibido ninguna suma hasta ese momento. De esta manera, la condición de control de la estructura iterativa verificará que no se hayan recibido el número de calificaciones parciales mediante la expresión $i < num\_calif$. Después, por cada $calificación$ parcial recibida, esta se sumará al identificador $suma$ y el resultado se asignará al mismo identificador $suma$. Además, se incrementará la cantidad de calificaciones parciales recibidas en uno. Eventualmente, se recibirán todas las calificaciones parciales, con lo que la estructura iterativa terminará y se podrá calcular el promedio.

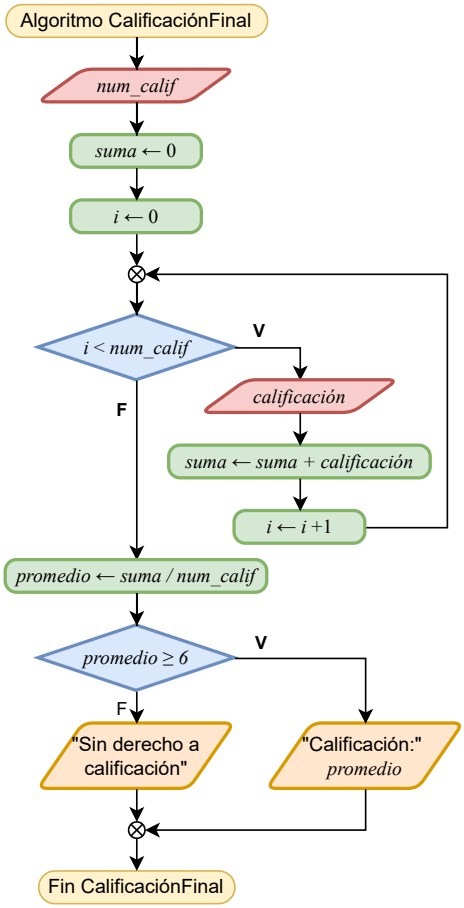


Figura 7.4: Diagramas de flujo para calcular la calificación final de una persona como el promedio de cierto número de exámenes parciales.

Notemos que en el Ejemplo 7.4, el identificador auxiliar i se inicializa en 0 y que la expresión de la condición de control verifica que $i < num\_calif$. Con esta combinación de valor de inicialización y condición de control garantizamos que sólo vamos a recibir num_calif calificaciones parciales como datos de entrada. Sin embargo, no es la única combinación de valor de inicialización y condición de control que puede garantizar esto. Por ejemplo, podríamos inicializar i en 1 y establecer la condición de control como $i \geq num\_calif$.

Algoritmo 7.4: CalificaciónFinal

```
1  leer num_calif
2  suma ← 0
3  i ← 0
4  mientras i < num_calif hacer
5  |  leer calificación
6  |  suma ← suma + calificación
7  |  i ← i + 1
8  fin mientras
9  promedio ← suma/num_calif
10 si promedio ≥ 6 entonces
11 |  escribir "Calificación:" promedio
12 si no
13 |  escribir "Sin derecho a calificación"
14 fin si
```

Ejemplo 7.5. Encontrar el máximo de los números ingresados por una persona.

Debemos recordar que el máximo de un conjunto de números es el mayor valor que tiene alguno de los números del conjunto. En el Capítulo anterior, en los Ejemplos 6.2 y 6.6 resolvimos el problema de encontrar el máximo de dos y tres números, respectivamente, y la solución fue hacer comparaciones por pares de números. En este caso, la estrategia será similar, comparando pares de números, y por cada par identificaremos el mayor valor, el cual será comparado con el siguiente número.

Comenzamos recibiendo del usuario la cantidad de números ($cant\_núm$, de tipo entero) de los cuales queremos conocer el máximo. Después, podemos inicializar un identificador $máximo$ en un número muy pequeño, por ejemplo, en $-\infty$, el cual nos servirá para asignar el valor máximo. Finalmente, también podemos inicializar otro identificador auxiliar i en 1, el cual nos ayudará a contar los números que hemos verificado. En cada iteración, el algoritmo recibirá un $número$ de la persona, el cual se comparará con el $máximo$. Si el $número$ que se acaba de recibir es mayor que $máximo$ ($número > máximo$ es **Verdadero**), entonces $máximo$ se actualizará con ese número ($máximo \leftarrow número$). Como último paso de la iteración, incrementamos el conteo de los números recibidos, es decir, $i \leftarrow i + 1$. Finalmente, la expresión de la condición de control deberá verificar que se han recibido todos los números de la persona. Cuando esto suceda ($i \leq cant\_núm$ es **Falso**), $máximo$ tendrá asignado el máximo valor. Este algoritmo está representado en el diagrama de flujo de la Figura 7.5 y en el pseudocódigo del Algoritmo 7.5.

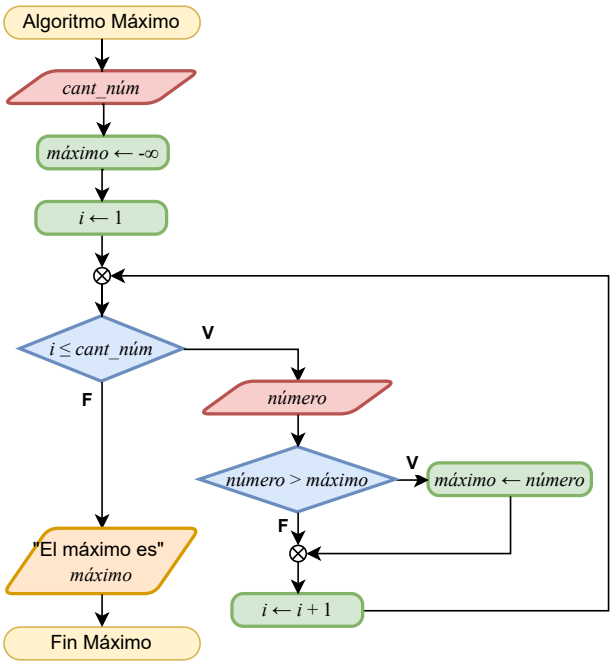


Figura 7.5: Diagramas de flujo para encontrar el máximo de cierta cantidad de números ingresados por una persona.

Algoritmo 7.5: Máximo

```
1  leer cant_núm
2  máximo ← −∞
3  i ← 1
4  mientras i ≤ cant_núm hacer
5  |  leer número
6  |  si número > máximo entonces
7  |  |  máximo ← número
8  |  fin si
9  |  i ← i + 1
10 fin mientras
11 escribir "El máximo es" máximo
```

Ejemplo 7.6. Dado un número entero positivo mayor que 1, decir si es un número primo o no lo es.

Para resolver este problema, primero debemos recordar que un número primo es un número natural mayor que 1 que solo es divisible entre él y entre 1. Esto quiere decir que, si dividimos un número entero x mayor que 1 entre todos los números desde 2 hasta $x-1$ y obtenemos un residuo diferente de cero en cada división, entonces x es un número primo. Sabemos que, de entre los operadores aritméticos, el operador módulo (`MOD`) nos sirve para conocer el residuo de una división. Por lo tanto, usaremos este operador para saber si el residuo de la división de x entre todos los números desde 2 hasta $x-1$ es igual o diferente de cero.

Como tenemos que dividir a x entre todos los números desde 2 hasta $x-1$, esta división la podemos realizar en cada repetición de una estructura de control iterativa. Utilizaremos el identificador auxiliar k para que tome todos los números desde 2 hasta $x-1$. Además, si el residuo de la división es cero (x `MOD` $k = 0$ es **Verdadero**), sabemos que x no es primo. De esta forma, debemos incluir una estructura selectiva dentro de la estructura iterativa que, en caso de que el residuo sea cero, asigne **Falso** a otra variable auxiliar es_primo y así obligaremos a terminar el ciclo, pues ya no tiene caso seguir verificando números. En caso de que x `MOD` $k = 0$ es **Falso**, se incrementará k en 1 ($k \leftarrow k+1$) y se ejecutará una nueva iteración. Entonces, el ciclo se detendrá cuando identifiquemos que x no es primo o cuando se hayan probado todos los números desde 2 hasta $x-1$. En otras palabras, la condición para que la estructura de control iterativa continúe es que es_primo sea **Verdadero** y que $k < x$. Finalmente, las inicializaciones que necesita el algoritmo son $k \leftarrow 2$ y $es\_primo \leftarrow$ **Verdadero**.

Este algoritmo está representado en el diagrama de flujo de la Figura 7.6 y en el pseudocódigo del Algoritmo 7.6.

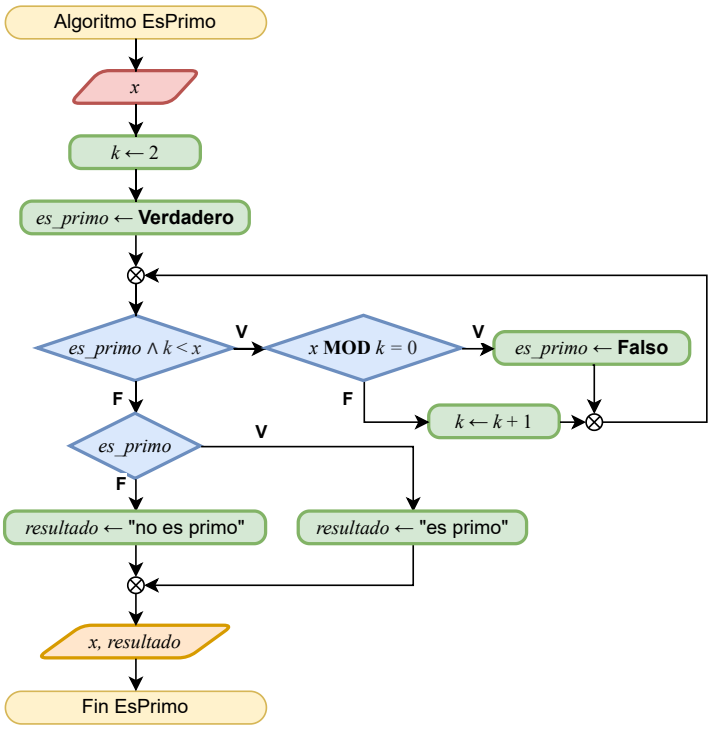


Figura 7.6: Diagramas de flujo para decidir si un número es primo o no.

Algoritmo 7.6: EsPrimo

```
1  leer x
2  k ← 2
3  es_primo ← Verdadero
4  mientras es_primo ∧ k < x hacer
5  |  si x MOD k = 0 entonces
6  |  |  es_primo ← Falso
7  |  si no
8  |  |  k ← k + 1
9  |  fin si
10 fin mientras
11 si es_primo entonces
12 |  resultado ← "es primo"
13 si no
14 |  resultado ← "no es primo"
15 fin si
```

7.2. Ejercicios

Ejercicio 7.1. Cierto número de personas adultas deben cruzar un río ancho y profundo con ningún puente a la vista. Se dan cuenta de que hay dos niños jugando con un bote de remos en la orilla. El bote es tan pequeño que solo puede transportar, a lo más, a dos niños o a un adulto. ¿Cómo pueden las personas cruzar el río y dejar a los niños en posesión del bote?

Ejercicio 7.2. Considera que hay n lobos, n cabras, n coles y n cazadores. El objetivo es colocarlos, uno por uno, en una fila de tal manera que ninguno esté en peligro. Es decir, que ningún cazador esté al lado de un lobo, ningún lobo esté al lado de una cabra y ninguna cabra esté al lado de una col. Además, no pueden estar dos o más de la misma clase juntos.

Ejercicio 7.3. ¿Cuántos árboles se necesitan para construir una valla de n metros sembrando los árboles cada 10 metros?

Ejercicio 7.4. Obtener el valor de un número x elevado a una potencia dada n. Recordemos que la potencia x^n se calcula multiplicando n veces x.

Ejercicio 7.5. Obtener el factorial de un número dado n. El factorial de un número n es el resultado de multiplicar todos los números desde 1 hasta n.

Ejercicio 7.6. Obtener los n primeros números de la serie de Fibonacci. Los números de Fibonacci están definidos por:

$$\begin{aligned} f_0 &= 0 \ , \\ f_1 &= 1 \ , \\ f_n &= f_{n-1} + f_{n-2} \ \text{, para } n \geq 2 \ . \end{aligned}$$

Ejercicio 7.7. Obtener la tabla de multiplicar del 1 al 10 de un número dado n.

Ejercicio 7.8. En matemáticas, se llama serie armónica a aquella que suma los inversos multiplicativos de los enteros positivos, denotándola con la siguiente serie infinita:

$$\sum_{k=1}^{\infty} \frac{1}{k} = 1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} + \cdots$$

Sumando los primeros n términos de la serie armónica se produce una suma parcial, conocida como el número armómico y está denotado como H_n:

$$H_n = \sum_{k=1}^{n} \frac{1}{k} \ .$$

Dado un número entero positivo n, calcular el número armónico H_n.

Ejercicio 7.9. El producto es una notación matemática que representa una multiplicación de una cantidad arbitraria (finita o infinita). Obtener el producto de dos elevado a las n primeras potencias positivas, es decir:

$$\prod_{i=1}^{n} 2^i = 2^1 \times 2^2 \times 2^3 \times 2^4 \times 2^5 \times 2^6 \times \ \cdots \ \times 2^{n-1} \times 2^n \ .$$

Ejercicio 7.10. Muestra la temperatura en grados Celsius y su correspondiente conversión a grados Fahrenheit. La fórmula para convertir grados Celsius (C) a grados Fahrenheit es:

$$F = \left(C \times \frac{9}{5} + 32 \right) \ .$$

La conversión debe incluir todas las temperaturas entre 0 y 100 grados Celsius queson múltiplos de 10.

Ejercicio 7.11. El acceso general a un acuario tiene un costo de $260.00. Las personas adultas mayores (60 años o más) pagan $210.00 y los niños menores de 3 años entran gratis. Un grupo de personas desea visitar el acuario, ¿cuánto tienen que pagar en total por ingresar?

Ejercicio 7.12. Un método sencillo de calcular el valor de π es mediante la multiplicación:

$$\pi \approx 4 \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \times \frac{6}{7} \times \frac{8}{7} \times \frac{8}{9} \times \frac{10}{9} \times \cdots$$

Dado un número entero positivo n mayor que 1, calcula el valor de π utilizando los primeros n términos de la multiplicación anterior. Por ejemplo, si $n \leftarrow 5$, la aproximación de π es:

$$\pi \approx 4 \times \frac{2}{3} \times \frac{4}{3} \times \frac{4}{5} \times \frac{6}{5} \ .$$

Ejercicio 7.13. Un palíndromo es una palabra o frase cuyas letras están dispuestas de tal manera que resulta la misma leída de izquierda a derecha que de derecha a izquierda. Por ejemplo, "Anita lava la tina" y "Yo dono rosas, oro no doy" son palíndromos. Verificar si la secuencia de n letras $s_1 s_2 \ldots s_n$ es un palíndromo.

Ejercicio 7.14. Los n vértices de un polígono tienen las coordenadas $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$. El vértice (x_n, y_n) es adyacente al vértice (x_1, y_1), el vértice (x_1, y_1) es adyacente al vértice (x_2, y_2), el vértice (x_2, y_2) es adyacente al vértice (x_3, y_3) y así sucesivamente hasta el vértice (x_{n-1}, y_{n-1}) que es adyacente al vértice (x_n, y_n). ¿Cuál es el perímetro de este polígono? Nota: Considera que puedes utilizar el símbolo $\sqrt{\ }$ para calcular la raíz cuadrada.

Ejercicio 7.15. El valor de π se puede aproximar mediante la siguiente suma infinita:

$$\pi \approx 3 + \frac{4}{2 \times 3 \times 4} - \frac{4}{4 \times 5 \times 6} + \frac{4}{6 \times 7 \times 8} - \frac{4}{8 \times 9 \times 10} + \frac{4}{10 \times 11 \times 12} - \frac{4}{12 \times 13 \times 14} + \cdots$$

La aproximación anterior la podemos expresar de la siguiente forma:

$$\pi \approx 3 + \sum_{i=1}^{\infty} (-1)^{i+1} \frac{4}{2i \times (2i+1) \times (2i+2)} \ .$$

Dado un número entero positivo n, calcula el valor de π utilizando los primeros n términos de la suma anterior. Es decir, calcular:

$$\pi \approx 3 + \sum_{i=1}^{n} (-1)^{i+1} \frac{4}{2i \times (2i+1) \times (2i+2)} \ .$$

Ejercicio 7.16. Hay n hot cakes, todos de diferentes tamaños, apilados uno encima del otro. Puedes deslizar una espátula debajo de uno de los hot cakes y voltear toda la pila que está sobre la espátula. El objetivo es ordenar los hot cakes según su tamaño, con el más grande en la parte inferior.

Ejercicio 7.17. Dada una secuencia de letras $X \leftarrow x_1\ x_2 \ldots x_m$ y una palabra $Y \leftarrow y_1\ y_2 \ldots y_n$, con $m > n$, decir si la palabra Y está contenida en X. Por ejemplo, si $X \leftarrow$ "a b c d e f g h i j k l m n o p q r s t u v w x y z" y $Y \leftarrow$ "i j k l m", entonces Y está contenida en X, pero si $Y \leftarrow$ "o q s u", entonces Y no está contenida en X.

Reflexiones

La UEA Taller de Algoritmos se incorporó a los Planes de Estudios de las Licenciaturas en Ingeniería en Computación y en Matemáticas Aplicadas en las adecuaciones aprobadas por el Consejo Divisional de Ciencias Naturales e Ingeniería en su Sesión CUA-DCNI-127-16, las cuales fueron presentadas ante el Colegio Académico en su Sesión 398 celebrada el 8 de junio de 2016. Esta UEA fue impartida por primera vez en el trimestre de Otoño de 2016. A casi siete años, nos parece pertinente acompañar este libro con un conjunto de reflexiones que reflejan nuestra experiencia impartiendo esta UEA.

Taller de Algoritmos surgió, como muchos otros cursos similares, a partir de observar a nuestra comunidad estudiantil durante el primer curso de programación a nivel licenciatura, Programación Estructurada, durante las muchas veces que impartimos esa UEA y algunas semejantes. Para quienes carecían de conocimientos previos sobre algoritmos, computadoras y programación, era una UEA difícil y, en muchas ocasiones, era causa de desmotivación en el alumnado. Una manera de subsanar esto y darle tiempo a las y a los principiantes de madurar los conceptos de programación, fue proponer la UEA Taller de Algoritmos. Esa propuesta es la que, después de diversas y fructíferas discusiones con colegas, fue aprobada y cuyo contenido se imparte actualmente.

Sin embargo, como todo proceso de diseño de un algoritmo, la experiencia nos ha permitido depurar y aprender. Al proponer Taller de Algoritmos, teníamos la idea de que sirviera como apoyo al alumnado y que posteriormente aplicaran esos conceptos en Programación Estructurada. Rápidamente nos dimos cuenta que este enfoque tenía la misma parvedad de otros libros [17, 8, 11, 6], cuya perspectiva se basa en enseñar a diseñar los algoritmos para que se programen y se ejecuten en una computadora. Por ello, en este tipo de materiales se integran, primero, conceptos sobre computación, informática, lenguajes de programación e incluso lenguaje binario. Esto evidentemente limitaba, por un lado, a la comunidad que podría beneficiarse de tomar la UEA y, por otro lado, se restringía a impartirse como una mera extensión de Programación Estructurada, lo que mantenía la misma problemática del primer curso de programación.

Al notar esto, repensamos sobre lo que queríamos que el alumnado se llevara de la UEA. Buscábamos que el alumnado desarrollara la habilidad de resolver problemas (*problem solving skill*), que es una habilidad completamente distinta e independiente a aprender la sintaxis de algún lenguaje de programación.

Así, encontramos tres términos relacionados con la habilidad de resolver problemas: pensamiento crítico, pensamiento algorítmico y pensamiento computacional. Por un lado, el pensamiento crítico es la capacidad de analizar y evaluar la consistencia de los razonamientos para distinguir y valuar la información importante. Por el otro, el pensamiento computacional se refiere a una manera de procesar la información de desarrollo cognitivo y creativo que requiere de pensamiento crítico y conceptos de programación y de computación para la resolución de problemas [1].

Como podemos observar en el Capítulo 4, la definición de pensamiento computacional y la definición de pensamiento algorítmico están relacionadas entre sí y en muchas ocasiones se utilizan indistintamente. Sin embargo, al analizar estas definiciones detenidamente, vemos que existen diferencias considerables. Desde nuestra experiencia, la principal diferencia es que el pensamiento algorítmico implica el descubrimiento o la creación de algoritmos para resolver problemas usando la lógica y la escritura como un procedimiento. A las personas que desarrollan el pensamiento algorítmico les da una idea de cómo tomar decisiones, de aprender de manera más sistemática y de sugerir nuevas formas de solucionar un problema. En otras palabras, conocer los algoritmos es distinto a pensar algorítmicamente, parecido a conocer una fórmula y a crear una nueva fórmula (sobre la diferencia entre pensamiento algorítmico, pensamiento computacional y codificación, se puede consultar [4]).

Pero, **¿por qué enseñar pensamiento algorítmico y no pensamiento computacional?** Una vez que nos dimos cuenta de que una de las bases del pensamiento computacional es el pensamiento algorítmico, saltó a la vista que esa es una *habilidad transversal* útil sin importar el área de conocimiento o la disciplina y que eso era justo lo que queríamos trasmitir. De aquí que el libro se intitule como la esencia de lo que hemos aprendido durante estos años: ***Pensamiento Algorítmico***.

Después de concluir el porqué, la pregunta que le siguió inmediatamente fue **¿cómo enseñar el pensamiento algorítmico?** En pocas palabras, con práctica, práctica, práctica y, después, más práctica. Tratar de solucionar tantos problemas como sea posible permite al alumnado la mejora continua de las habilidades de razonamiento general y de comunicación al tener que diseñar e interpretar procesos estructurados y sistemas de reglas. Así, en este libro incluimos problemas cuidadosamente elegidos que pueden, y algunos deben, solucionarse sin necesidad de conocimientos previos sobre el tema y sin estar relacionados con tecnología alguna. De esta manera, la comunidad principiante puede enfocarse en el diseño de algoritmos. Para cumplir con nuestra perspectiva de enseñar el diseño de algoritmos sin computadoras o *desconectados* (tomamos el término del francés *algorithmique débranchée* [14]), se eligió el diagrama de flujo y el pseudocódigo como las maneras para implementarlos, ya que son representaciones de alto nivel y orientadas a problemas que permiten enfocarse en la descripción de los algoritmos.

Una de las principales observaciones que nos han hecho durante estos años, y que hemos recibido más frecuentemente, es que si buscamos concentrarnos en el diseño de algoritmos, ¿por qué no usamos herramientas visuales como PSeInt (`https://pseint.sourceforge.net/`), Algogo

(https://www.algogo.xyz/), AlgoBox (https://www.algoboxpro.com), LARP (https://www.tice-education.fr) o incluso Scratch (https://scratch.mit.edu/)? ¿Por qué privilegiamos el uso de papel y lápiz? En primer lugar, porque queremos enfatizar que los algoritmos son un medio de comunicación, por lo que se debe desarrollar la habilidad de comunicarse, pero antes de pensar en comunicarse con una computadora queremos que se comuniquen y entiendan entre humanos. En segundo lugar, el uso de las herramientas visuales nos trae de vuelta a que, al momento de diseñar algoritmos, el alumnado maneje una representación en computadora y comience a depender de esas herramientas para determinar si sus instrucciones son correctas o no, cuestión que, como explicamos anteriormente, queremos evitar.

Actualmente, estamos tan acostumbrados a depender de tecnología y que sea esta la que nos resuelva en gran medida los problemas a los que nos enfrentamos, que tal pareciera que se nos ha olvidado cómo hacerlo por nuestra cuenta. Lo que proponemos entonces es tomar el conocimiento literalmente en nuestras manos, usando un papel y un lápiz, analizar el problema y comenzar a escribir la solución.

> "A menudo se ha dicho que una persona no entiende realmente algo hasta que lo enseña a alguien más. De hecho, una persona no entiende *realmente* algo hasta que se lo puede enseñar a una computadora, es decir, **expresarlo como un algoritmo**."
>
> - Donald Knuth [7]

Bibliografía

[1] Armendáriz Vega, Rosalina: *Rebo-binario*, 2020. https://www.coursera.org/learn/rebo-binario, Universidad Autónoma Metropolitana.

[2] Böhm, Corrado y Giuseppe Jacopini: *Flow diagrams, Turing machines and languages with only two formation rules*. Communications of the ACM, 9(5):366–371, 1966.

[3] Chapin, Ned: *Flowcharting with the ANSI standard: A tutorial*. ACM Computing Surveys, 2(2):119–146, 1970.

[4] Earp, Jo: *Teacher Q&A: Algorithmic thinking*. Teacher, 2018. https://www.teachermagazine.com/au_en/articles/teacher-qa-algorithmic-thinking, Fecha de última visita: 22/09/2023.

[5] Futschek, Gerald: *Algorithmic thinking: The key for understanding computer science*. En Mittermeir, Roland T. (editor): *Informatics Education - The Bridge between Using and Understanding Computers*, páginas 159–168. Springer, 2006.

[6] García, Matías: *Introducción a la informática: Ejercicios resueltos de algoritmos*, 2005. http://www.profmatiasgarcia.com.ar/uploads/tutoriales/Ej_resueltos_algoritmos.pdf, Fecha de última visita: 22/09/2023.

[7] Knuth, Donald E.: *Computer science and its relation to mathematics*. The American Mathematical Monthly, 81(4):323–343, 1974.

[8] Morales Gamboa, Rafael, Eduardo Morales Manzanares, Alberto Pacheco González, Marcela Quiroz Castellanos y Luis Enrique Sucar Succar (editores): *Pensamiento Computacional en México*. Academia Mexicana de Computación, A. C., 2021.

[9] Olsen, Florence: *Computer scientist says all students should learn to think 'algorithmically'*. The Chronicle of Higher Education, 2000. http://www.chronicle.com/article/Computer-Scientist-Says-All/105098, Fecha de última visita: 22/09/2023.

[10] Phillips, Charles A. y Alexander C. Grove: *Flowchart symbols and their usage in information processing*. Informe técnico ANSI X3.5-1970, American National Standards Institute. Subcommittee on Problem Definition and Analysis, New York, New York, 1970.

[11] Pinales Delgado, Francisco Javier y César Eduardo Velázquez Amador: *Problemario de algoritmos resueltos con diagramas de flujo y pseudocódigo*. Universidad Autónoma de Aguascalientes, 2014.

[12] Pino Ceballos, Juan Alejandro: *Concepciones y prácticas de los estudiantes de pedagogía media en matemáticas con respecto a la resolución de problemas y diseño e implementación de un curso para aprender a enseñar a resolver problemas*. Tesis de Doctorado, Universidad de Extremadura, 2012.

[13] Pólya, George y John Horton Conway: *How to solve it: A new aspect of mathematical method*. Princeton University Press, 1957.

[14] Primàbord, Le portail du numérique pour le premier degré: *Qu'est-ce que l'informatique débranchée? Définition et ressources*. Ministère de l'Education nationale et de la Jeunesse, 2017. https://primabord.eduscol.education.fr/qu-est-ce-que-l-informatique-debranchee, Fecha de última visita: 22/09/2023.

[15] Qin, Hong: *Teaching computational thinking through bioinformatics to biology students*. En *40th ACM Technical Symposium on Computer Science Education*, páginas 188–191. Association for Computing Machinery, 2009.

[16] Venit, Stewart y Elizabeth Drake: *Prelude to programming*. Pearson, 2021.

[17] Zapotecatl López, Jorge Luis: *Introducción al pensamiento computacional: conceptos básicos para todos*. Academia Mexicana de Computación, A. C., 2018.

[18] Zsakó, László y Péter Szlávi: *ICT Competences: Algorithmic Thinking*. Acta Didactica Napocensia, 5(2):49–58, 2012.

Pensamiento algorítmico, de Abel García Nájera, Karen Samara Miranda Campos y Saúl Zapotecas Martínez, es una obra que se puede encontrar en la página web del repositorio de la UAM Cuajimalpa Concéntric@.

La asistencia editorial estuvo a cargo de Denise Ocaranza.

Casa abierta al tiempo

UNIVERSIDAD AUTÓNOMA METROPOLITANA