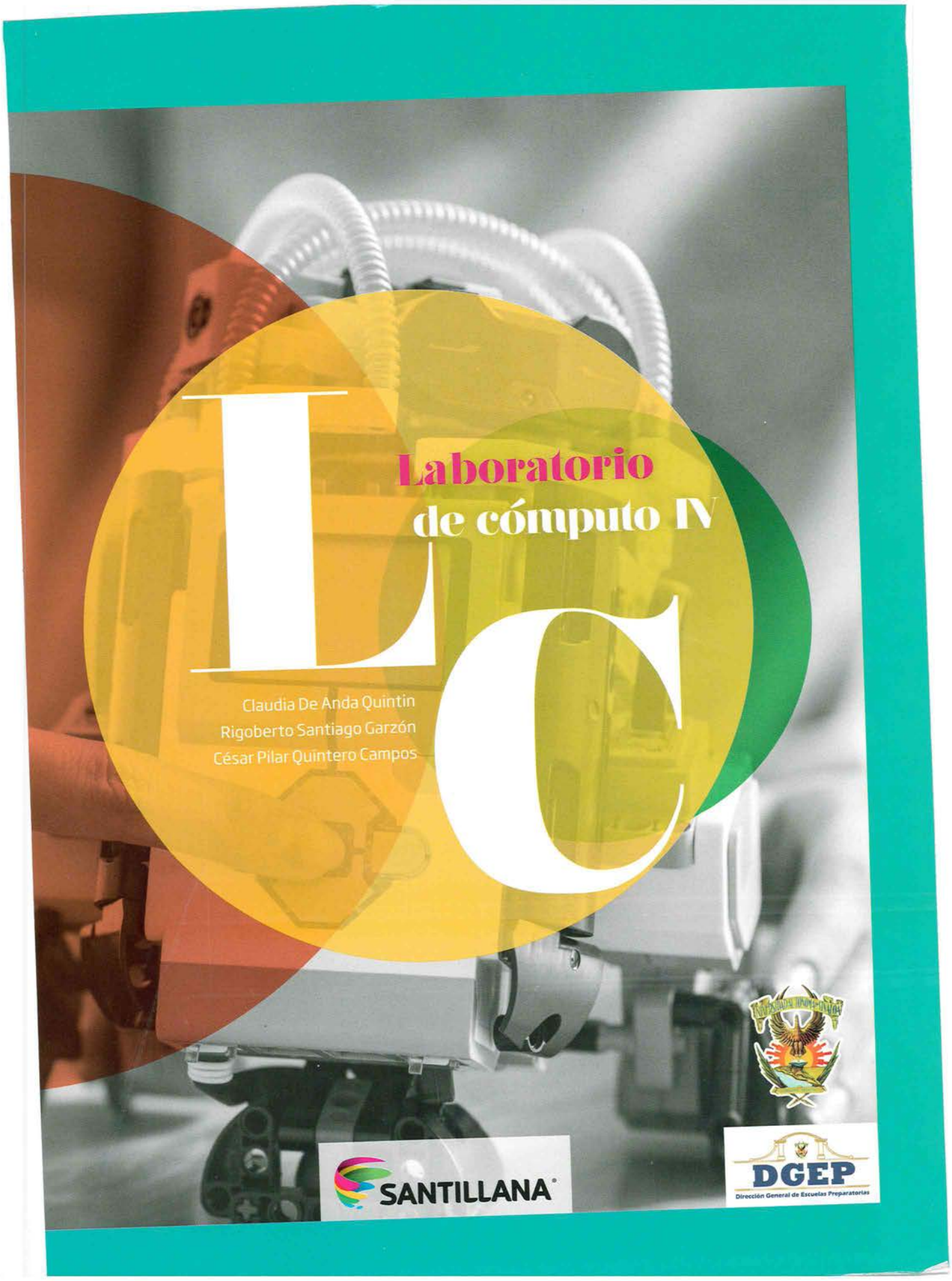


Laboratorio de cómputo IV

LC

Claudia De Anda Quintin
Rigoberto Santiago Garzón
César Pilar Quintero Campos

SANTILLANA

DGEP
Dirección General de Escuelas Preparatorias

Los autores queremos reconocer el apoyo que recibimos de los profesores de las diferentes Unidades Académicas del bachillerato de la UAS. ya que su colaboración y sus propuestas didácticas fueron de gran ayuda durante la realización de este libro. Gracias:

Edwin Ramón Romero Espíritu. DGEP

Yeimy Guadalupe López Camacho. DGEP

Juan Enrique Gutiérrez Moreno. DGEP

María del Carmen Melisa Quintero Félix. DGEP

José Luis Preciado Cueto. Preparatoria Cmdte. Víctor Manuel Tirado López

Sergio Luis Barraza Castillo. Preparatoria Carlos Marx

Nadya Rocío Galaviz Heredia. Preparatoria Los Mochis

Ernesto Efrén Castillón. Preparatoria CU de Los Mochis

Lluvia Selene Galaviz Heredia. Preparatoria Los Mochis

Guillermo Arturo Ríos Moreno. Preparatoria Guamúchil

Luis Fernando Sánchez López. Preparatoria Guamúchil

Laboratorio de cómputo IV

Claudia De Anda Quintin
Rigoberto Santiago Garzón
César Pilar Quintero Campos

SANTILLANA

Laboratorio de cómputo IV

El libro **Laboratorio de cómputo IV** fue elaborado en Editorial Santillana por el siguiente equipo:

Dirección General de Contenidos
Antonio Moreno Paniagua
Gerencia de Secundaria y Bachillerato
Iván Vásquez Rodríguez
Gerencia de Arte y Diseño
Humberto Ayala Santiago
Coordinación de Bachillerato
Adrián Romero Rodríguez
Coordinación de Iconografía
Nadira Nizametdinova Malekovna
Coordinación de Realización
Gabriela Armillas Bojorges

Edición
Corporativo Guenduvi S. C.
Revisión técnica
Karina Guadalupe Cueto Ruiz
Corrección de estilo
Daniela Barranco Ortiz
Edición de Realización
Haydée Jaramillo Barona
Edición de preprensa y control de calidad
Miguel Ángel Flores Medina
Diseño de portada e interiores
Jessica Gutiérrez López
y Beatriz E. Alatriste del Castillo
Gestión HUB
Alicia Prado Juárez
Iconografía
Iván Navarro Juárez
Digitalización de imágenes
José Perales Neria
y Ricardo Ríos Delgado
Fotografía de portada
Shutterstock

D. R. © 2016 por **EDITORIAL SANTILLANA, S. A. de C. V.** Avenida Río Mixcoac 274, Piso 4, colonia Acacias, C. P. 03240, delegación Benito Juárez, Ciudad de México.

ISBN: 978-607-01-3297-1
Primera edición: diciembre de 2016

Miembro de la Cámara Nacional de la Industria Editorial Mexicana. Reg. Núm. 802

Impreso en México/*Printed in Mexico*

Este libro se terminó de imprimir en diciembre de 2016 en los talleres de Impresora y Editora Xalco, S.A. de C.V. www.grupocorme.com
Tel. (55) 5784-6177

Presentación

Laboratorio de cómputo IV se desarrolló de acuerdo con el Programa de estudios 2015 del Bachillerato General por Competencias para esta asignatura, emitido por la Dirección General de Escuelas Preparatorias (DGEP) de la Universidad Autónoma de Sinaloa, cuya aplicación funciona a partir del mes de febrero de 2016. Este Programa de Estudios se basa en el **enfoque por competencias**, a las que considera como el conjunto de conocimientos, habilidades, actitudes y valores que debes poner en práctica para enfrentar los retos planteados por tu entorno social y familiar, en especial, para incorporarte a la educación superior o al mundo laboral.

En específico, el programa atiende las competencias genéricas y disciplinares básicas del campo de la comunicación de la RIEMS y de la propia UAS, organizadas bajo los principios educativos del modelo constructivista, donde tú, como estudiante, eres el centro del accionar educativo, presentando una alineación entre los propósitos curriculares, las competencias, los productos y los instrumentos de evaluación, pero sobre todo, en los ambientes de aprendizaje donde las interacciones entre alumnos y docentes son fundamentales.

Esta obra presenta un modelo didáctico que contribuirá a que desarrolles las competencias propuestas en el programa de estudio de la asignatura. De esta manera, te ofrece la oportunidad de construir diversos saberes y emplear los recursos tecnológicos a tu alcance como instrumentos de comunicación. Se encuentra organizado en dos unidades que te ayudarán a cumplir la competencia central de la materia: diseñar algoritmos para la resolución de problemas utilizando un lenguaje de programación estructurado.

En la primera unidad aprenderás a diseñar algoritmos, diagramas de flujo y pseudocódigos para resolver de manera óptima diversos problemas. En la segunda unidad utilizarás un lenguaje de programación estructurada para convertir pseudocódigos en pequeñas aplicaciones y resolver problemas.

Este libro, orientado al desarrollo de competencias, no descuida el rigor de los conceptos empleados en informática; sin embargo, hemos procurado exponerlos con un lenguaje claro, sencillo y directo, fácil de comprender. La estructura del libro y la manera de abordar los temas tienen como propósito que desarrolles un manejo cabal de las aplicaciones digitales mediante la práctica de habilidades, conocimientos y destrezas en la vida escolar y social.

Para cumplir con estos propósitos académicos no basta el conocimiento y la comprensión de los conceptos expuestos en este material, pues la asignatura no es solo una más que debas cursar para obtener un certificado, sino que también representa una práctica destinada a transformar y mejorar tu vida y el entorno social, económico y profesional en el que te desarrollas.

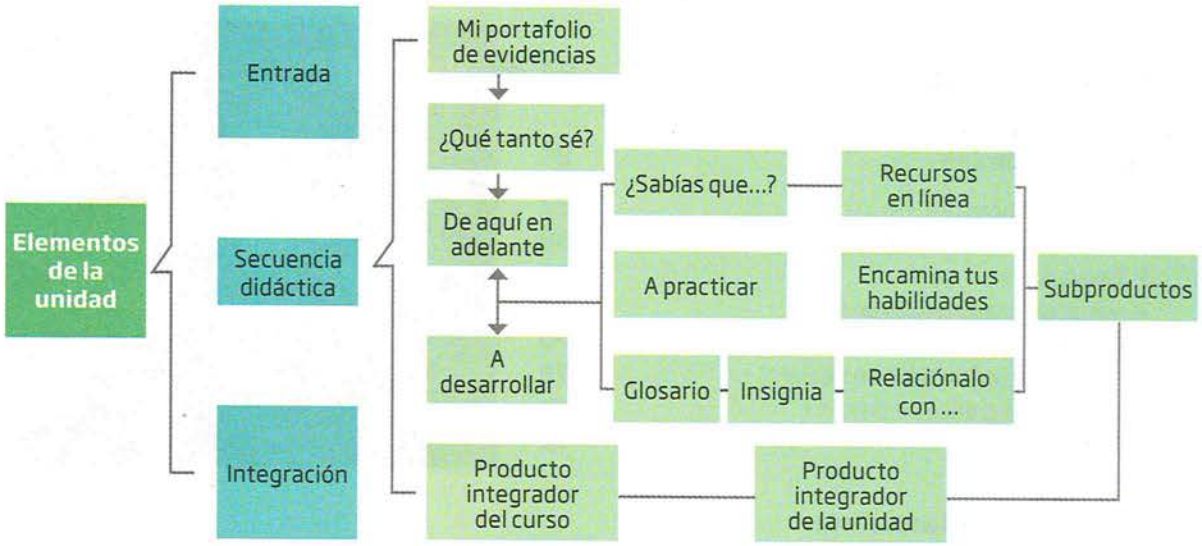
Contenido

Unidad 2

Shutterstock/Africa Studio

Tu libro

Laboratorio de cómputo IV está conformado por dos unidades, cuya organización tiene el propósito de ayudarte en la integración de los saberes necesarios para que alcances determinados niveles de desempeño. Cada unidad contiene secuencias didácticas que se estructuran de la siguiente manera:



Entrada de la unidad

En esta sección encontrarás tu meta (propósito de la unidad), lo que aprenderás (contenidos temáticos), cómo lo harás (saberes específicos), lo que lograrás (producto integrador de la unidad) y para qué te servirá en tu vida académica, cotidiana o laboral (competencias genéricas y disciplinares). También incluye una imagen relacionada con el contenido.

Mi portafolio de evidencias

Esta sección está compuesta por una lista de cotejo de las evidencias generadas en las secciones *Encamina tus habilidades*, los subproductos y productos integradores de la unidad; con esta lista de cotejo podrás dar seguimiento puntual y continuo a tu proceso de aprendizaje.

¿Qué tanto sé?

Contiene la evaluación diagnóstica inicial de cada unidad, la cual te ayudará a reconocer tu conocimiento previo de cada uno de los temas que estudiarás en las secuencias didácticas.

Secuencia didáctica

A desarrollar:

Expone los objetivos esperados de cada secuencia didáctica y enuncia las actividades, subproductos y atributos de las competencias genéricas que desarrollarás.

De aquí en adelante

Presenta una situación problemática y se formulan preguntas para que recuperes la información que ya conoces y establezcas su relación con los temas por estudiar, de modo que te ayuda a explorar tus conocimientos.

Desarrollo

Es el apartado principal de la secuencia didáctica. Contiene el discurso y las actividades necesarias para que desarrolles las competencias indicadas. La información y las sugerencias didácticas están organizadas en torno a los siguientes ejes de trabajo, que pueden darse de manera simultánea.

¿Qué necesito saber (conocimientos)...
¿Qué necesito saber hacer (habilidades)...
...para elaborar la evidencia de mi aprendizaje?

Además de estos elementos, cada secuencia didáctica incluye diferentes **secciones** y **cápsulas** que te permitirán desarrollar o ejercitar habilidades, descubrir tus actitudes y manifestarlas en los subproductos a partir de estrategias de aprendizaje; las cuales son:

A practicar. Propone actividades que contribuyen a la participación colaborativa, pero estas no se incluyen en el portafolio de evidencias. Pueden ser tareas o prácticas que se desarrollarán en el laboratorio de cómputo o como tarea en casa.

Encamina tus habilidades. Subproductos encaminados al producto integrador. Se considera para el portafolio de evidencias.

Ejemplo. Muestra la manera en que se debe resolver un problema invitando a encontrar su solución por medio del uso de las herramientas digitales y de programación.

Insignia. Incluye ejercicios, investigaciones o acciones específicas, con tiempo determinado para su elaboración, para que logres puntos extra (valor de rescate).

Glosario. Contiene definiciones de los vocablos técnicos más relevantes o poco usuales.

¿Sabías que...? Ofrece información para que profundices y amplíes tus saberes.

Recursos en línea. Las páginas de internet que se recomiendan en algunas de las secciones o cápsulas están referidas como **recurso** con un **número**. Para acceder a estos recursos hay que entrar a la página **www.bachilleratoenred.com.mx/enlaces/LCIVUAS2016** y dar clic en el número que corresponda.

Relaciónalo con... Describe los vínculos con otras asignaturas, con tu vida cotidiana, tu familia y tu comunidad.

Subproductos

Al finalizar cada secuencia didáctica tendrás la oportunidad de desarrollar tres subproductos, cada uno con un enfoque diferente (procedimental, declarativo y actitudinal-valoral), los cuales servirán de base para la realización del producto integrador de cada unidad.

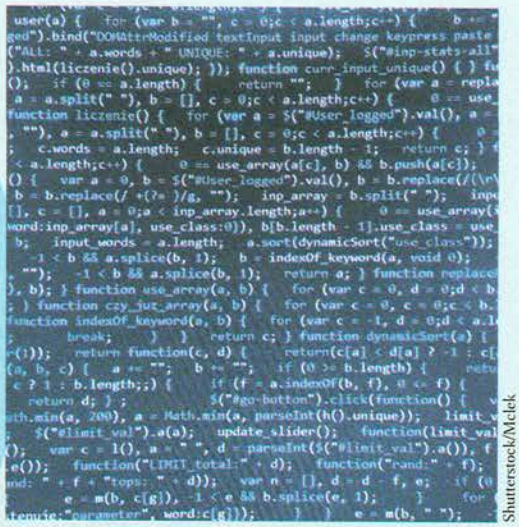
Producto integrador de la unidad

Propone una sola actividad con la que podrás integrar lo que aprendiste en las secuencias didácticas que integran cada unidad.

Producto integrador del curso

Es el último reto del libro, el cual describe una acción concreta para que demuestres todo lo aprendido en el curso.

Unidad 1
Fundamentos de programación



Shutterstock/Mclek

Propósito de la unidad

Tu meta será:

- Diseñar algoritmos, diagramas de flujo y pseudocódigos para solucionar de manera óptima diversos problemas.

Contenidos temáticos

¿Qué aprenderás?

1.1. Análisis y solución de problemas.
1.2. Diagramas de flujo.
1.3. Pseudocódigo y Scratch.

Competencias genéricas y disciplinares

En general, te servirá para:

4. Escuchar, interpretar y emitir mensajes pertinentes en distintos contextos mediante la utilización de medios, códigos y herramientas apropiados.
5. Desarrollar innovaciones y proponer soluciones a problemas a partir de métodos establecidos.
7. Aprender por iniciativa e interés propio a lo largo de la vida.
8. Participar y colaborar de manera efectiva en equipos diversos.

En particular, te servirá para:

C-12. Utilizar las tecnologías de la información y la comunicación para investigar, resolver problemas, producir materiales y transmitir información.

Saberes específicos

¿Cómo lo aprenderás?

- Ejercitando soluciones de problemas mediante algoritmos con base en su definición e identificación de sus componentes.
- Diagramando algoritmos al ordenar instrucciones de entrada, proceso y salida que conducen a la solución de un problema.
- Elaborando pseudocódigos con base en la definición de sus variables, constantes y datos e identificando las estructuras de control.
- Estructurando sentencias secuenciales de control selectivo y de control repetitivo.
- Diseñando algoritmos.
- Analizando la importancia del uso de algoritmos para resolver problemas de tu vida cotidiana.
- Asumiendo una actitud responsable ante el uso de la información que expresas mediante algoritmos.
- Demostrando creatividad en el diseño de algoritmos y diagramas de flujo.

¿Qué lograrás?

- El **producto integrador** de esta unidad consiste en diseñar, crear y compartir un algoritmo, diagrama de flujo y pseudocódigo en Scratch.

Mi portafolio de evidencias

Evidencias	Lograda	
	Sí	No
Conclusiones a partir de una lluvia de ideas (*De aquí en adelante*, página 10)		
Algoritmo (*Encamina tus habilidades*, página 19)		
Algoritmo (*Encamina tus habilidades*, páginas 29 y 30)		
Algoritmo (*Subproducto procedimental 1.1*, página 31)		
Respuestas (*Subproductos declarativo y actitudinal-valoral 1.1*, página 31)		
Conclusiones a partir de una lluvia de ideas (*De aquí en adelante*, página 32)		
Algoritmo y diagrama de flujo (*Encamina tus habilidades*, página 37)		
Algoritmo y diagrama de flujo (*Encamina tus habilidades*, página 42)		
Algoritmo y diagrama de flujo (*Subproducto procedimental 1.2*, página 44)		
Respuestas (*Subproductos declarativo y actitudinal-valoral 1.2*, páginas 42 y 43)		
Conclusiones a partir de una lluvia de ideas (*De aquí en adelante*, página 46)		
Algoritmo, diagrama de flujo y pseudocódigo (*Encamina tus habilidades*, página 56)		
Algoritmo, diagrama de flujo y pseudocódigo (*Subproducto procedimental 1.3*, página 61)		
Respuestas (*Subproductos declarativo y actitudinal-valoral 1.3*, página 61)		

¿Qué tanto sé?

Responde en tu cuaderno con base en tus conocimientos previos.

1. Describe al menos tres diferencias importantes entre software y hardware.

2. ¿Qué es un algoritmo y cuál es la función de una variable en este?

3. ¿Cuál es la diferencia entre un operador aritmético y un operador lógico?

4. ¿Cuál es la principal ventaja que tendría aprender a programar?

1.1. Análisis y solución de problemas

A desarrollar

Al terminar esta secuencia didáctica aprenderás a identificar las fases para solucionar problemas. Utilizarás los algoritmos para representar gráficamente la solución de diferentes problemas.

El **subproducto procedimental** de esta secuencia didáctica consistirá en elaborar, en parejas, un algoritmo que solucione un problema matemático.

Las actividades que realizarás en esta secuencia didáctica te servirán para:

4.2. Aplicar diversas estrategias comunicativas según quienes sean los interlocutores, el contexto en el que te encuentras y los objetivos que persigues.
4.5. Manejar las tecnologías de la información y la comunicación para obtener información y expresar ideas, de manera responsable y respetuosa.
5.1. Seguir instrucciones y procedimientos de manera reflexiva en la búsqueda y adquisición de nuevos conocimientos.
5.2. Ordenar información de acuerdo con categorías, jerarquías y relaciones.
8.1. Plantear problemas y ofrecer alternativas de solución al desarrollar proyectos en equipos de trabajo, y definir un curso de acción con pasos específicos.
8.3. Asumir una actitud constructiva al intervenir en equipos de trabajo, congruente con los conocimientos y las habilidades que posees.

De aquí en adelante

› **Pide a tu profesor que organice una lluvia de ideas en el grupo para responder las siguientes preguntas y escribe las conclusiones de cada una en el espacio correspondiente.**

- ¿Qué es un algoritmo y para qué sirve?

- Describe una situación en la que hayas usado un algoritmo en tu vida cotidiana.

Análisis de problemas

A diario enfrentas problemas y situaciones que debes solucionar de manera óptima bajo diversos contextos, es decir, según las circunstancias que rodean la situación. Un problema se entiende como aquella situación o cuestión, que involucra varios hechos, circunstancias, acciones y obstáculos que impiden alcanzar un objetivo, por lo que es indispensable obtener una solución.

Por otra parte, para solucionar problemas requieres desarrollar ciertas habilidades cognitivas, con el propósito de otorgar una respuesta correcta con base en una situación. En pocas palabras, para solucionar un problema es necesario entenderlo, y para ello se puede empezar por estudiar sus categorías. Los problemas se pueden clasificar en tres tipos, los estructurados, los semiestructurados y los no estructurados. A continuación se describe cada uno de estos.

- **Problema estructurado.** Plantea situaciones claras, bien definidas y delimitadas; en estos se puede establecer una solución de forma segura y precisa, la cual puede estar fundamentada en teorías, metodologías y tecnologías.
- **Problema semiestructurado.** Parte de la información disponible es clara y precisa, pero hay incógnitas y dudas en algunos aspectos de la situación. La labor de quienes tratan de resolver este tipo de problemas es aclarar incógnitas y dudas, con el fin de establecer de forma precisa los requerimientos del programa que se va a generar. Resolver problemas semiestructurados presenta dificultades que dependen del grado de las imprecisiones y de la incertidumbre que se tenga, así como de la complejidad de la solución. Los problemas estructurados y semiestructurados se pueden resolver utilizando una computadora, por lo que este tipo de problemas serán los que aprenderás a solucionar en este curso.
- **Problema no estructurado.** Su planteamiento es impreciso e incompleto; las relaciones entre sus elementos no son claras, lo que provoca ambigüedad y confusión en la solución requerida. Este tipo de problemas no se pueden resolver utilizando una computadora.

Recursos en línea

Entra a la página **bachilleratoenred.com.mx/enlaces/LCIVUAS2016** y abre el **enlace 1.1**, en el cual encontrarás más información acerca de la clasificación de los problemas.

A practicar 1.1.1

Analiza en equipo las siguientes situaciones y compartan las posibles soluciones de cada uno con el resto del grupo.

1. ¿Qué es más barato: invitar a un amigo a comer dos veces, o invitar a dos amigos una vez?
2. Un hombre se casa con la hermana de la que en vida fue su esposa, ¿qué parentesco adquiere su nueva compañera?
3. Un estudiante tiene tres montones grandes de plumas, otro tiene seis montones chicos y un tercero tiene cuatro montones medianos. Si deciden juntarlos todos, ¿cuántos montones habrá?
4. En una escuela es necesario cercar la parte frontal para brindar mayor seguridad a los estudiantes; si el terreno tiene diez metros de frente y se quiere colocar un poste cada dos metros, ¿cuántos postes se necesitan?

Shutterstock/Billion photos

¿Cuál ha sido el procedimiento que has seguido para resolver tus problemas hasta ahora?

Fases de la solución de problemas

Existen distintas metodologías para la solución de problemas. Todas coinciden en la importancia de plantearse preguntas durante el proceso que sirvan para identificar el problema y buscar alternativas de solución. En este libro de texto se considera que la siguiente metodología es la ideal para solucionar problemas como los que enfrentas a diario. Este procedimiento consta de tres pasos, que son:

1. Análisis profundo del problema.
2. Diseño y creación del algoritmo.
3. Implementación y verificación del algoritmo.

Análisis profundo del problema. Para esta fase necesitas leer un enunciado que describa el problema que se va a solucionar, partiendo de algunas preguntas, y así determinar los teoremas, las fórmulas, las variables y las constantes que se necesitan para dar solución óptima al problema. Las preguntas que pueden plantearse en esta fase son similares a las siguientes: ¿qué necesito resolver?, ¿qué debo saber para solucionarlo?, ¿cómo podría resolverse?, ¿cuáles son los datos que están involucrados en el problema?, ¿con qué paso se debe empezar?, ¿qué metodología puede aplicarse para resolver el problema?, entre otras.

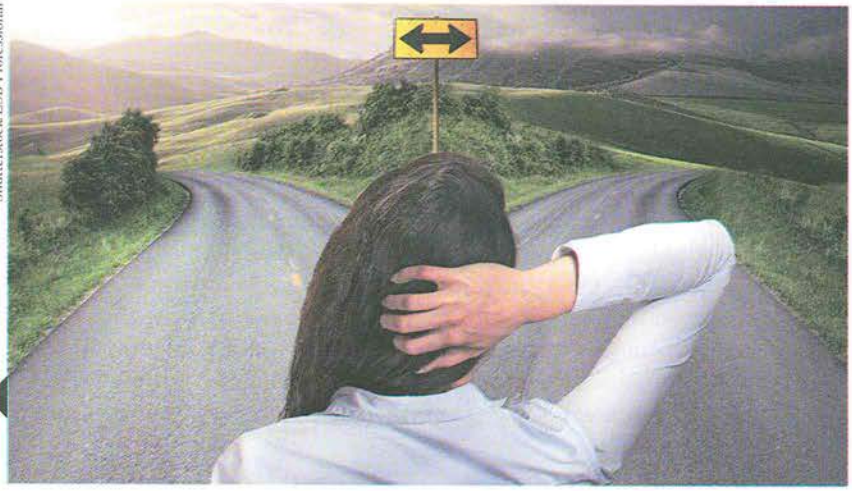
Al definir los teoremas o fórmulas que se van a usar se determina cuáles y cuántas variables y constantes se utilizarán, y qué procesos van a realizarse para dar solución al problema.

Diseño y creación del algoritmo. Con base en la etapa anterior se definirá la serie de pasos que se van a seguir para dar solución de forma óptima al problema planteado (algoritmo); obviamente, tomando en cuenta todo lo definido en la etapa anterior, como los teoremas, las fórmulas, las variables, las constantes, etcétera.

Implementación y verificación del algoritmo. Finalmente, en esta etapa se usará el algoritmo creado y diseñado en la etapa anterior, y se verificará si el resultado obtenido atiende de manera efectiva la solución del problema; de lo contrario, se debe regresar a la primera etapa para abordarlo de nuevo.

Recursos en línea

Entra al **enlace 1.2** y conocerás las estrategias que aplican otras disciplinas, como la psicología o la epistemología, para analizar y resolver los problemas.



Shutterstock/ESB Professional

© SANTILLANA

Un algoritmo está conformado por una serie de instrucciones a seguir que representan un modelo de solución para un problema.

Etapas del diseño y creación de algoritmos

Como lo aprendiste en el tema anterior, el diseño o creación de algoritmos es la segunda fase para la solución de problemas. El algoritmo debe considerar las fórmulas, los teoremas, las variables y demás elementos numéricos para llegar a la solución de uno o varios problemas.

Existen muchas definiciones de algoritmo, que pueden ser diferentes según la óptica de la materia que los aplica; sin embargo, todas las definiciones coincidenen que *un algoritmo es un conjunto de pasos, operaciones o acciones finitas, ordenados lógicamente, que permiten solucionar un problema.*

Con base en esta deficición toda tu vida has utilizado algoritmos para alcanzar objetivos, incluso sin darte cuenta, solo que hasta ahora no habías aplicado la formalidad con la que se deben plantear. Así, todos los algoritmos cuentan con tres etapas principales, que son:

1. **Entrada de información.** Cualquier algoritmo requiere de la información necesaria para procesarla y solucionar un problema, ya que no se puede partir de la nada, porque la nada no existe, ¿o acaso se podrían sumar dos números que no se conocen?
2. **Procesamiento de la información.** En esta etapa se aplican fórmulas, teoremas, procesos y demás metodologías que se determinaron en la fase de *Análisis profundo del problema*, utilizando obviamente la información que fue introducida en la etapa 1.
3. **Salida o muestra del resultado.** El propósito principal de los algoritmos es solucionar un problema, por lo que se requiere mostrar el resultado, el cual debe atender al problema planteado. Mostrar la solución del problema es el objetivo de esta fase.

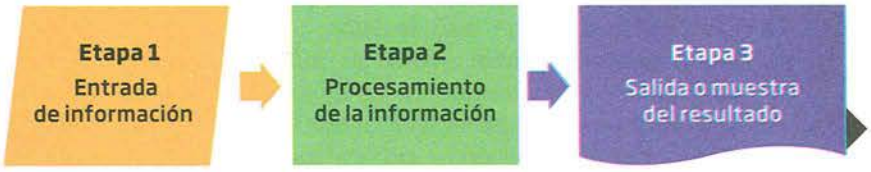


Diagrama de las etapas de un algoritmo.

Además de estas etapas, la creación de un algoritmo también debe cumplir con las siguientes características principales:

1. **Precisión:** la metodología debe estar definida de forma precisa, es decir, no debe existir la ambigüedad en el seguimiento de los pasos del algoritmo creado.
2. **Efectividad:** el algoritmo debe ser efectivo, es decir, las operaciones o procesos deben ser las más adecuadas para solucionar el problema.
3. **Determinismo:** al darle un conjunto de datos idénticos de entrada, el algoritmo debe llegar siempre al mismo resultado.
4. **Finitud:** un algoritmo debe terminar en un número finito de pasos, sin importar su complejidad.

En informática existen varias maneras de categorizar a los algoritmos, pero las más frecuentes consideran cuatro aspectos importantes: 1) quién los ejecutará, 2) qué signos se usarán, 3) cuál es su función y 4) cuál será la estrategia que usará para resolver el problema.

Relaciónalo con...

La programación de un robot se basa en algoritmos que ayudan al procesador a resolver problemas de manera inmediata; por ejemplo, si se enfrentan a un obstáculo las instrucciones en el algoritmo programado le ayudará a resolver hacia dónde debe girar para superarlo.

¿Sabías que...?

El diagrama de las fases de un algoritmo empata con el diagrama que explica la manera en que funciona una computadora, el cual fue creado por Von Neumann y se conoce como la *máquina de Von Neumann.*

Clasificación de los algoritmos

Los algoritmos se clasifican en **informales** y **computacionales**. Los primeros se diseñan para que sean ejecutados por el ser humano y no por una computadora. A diario ejecutas este tipo de algoritmos en diversas actividades, como vestirte, bañarte, preparar comida, ir a la escuela, etcétera. Los *algoritmos computacionales* se crean tomando en cuenta que una computadora los ejecutará, por lo que los pasos deben ser más precisos que los anteriores, y así aprovechar la velocidad del equipo de cómputo y obtener resultados más confiables.

Como se mencionó en la página anterior, los algoritmos también pueden clasificarse con base en el sistema de signos que utilizan, la función que realizan o la estrategia usada para obtener un resultado. Por lo anterior, se establece que los algoritmos se nombran como sigue:

1. **Según su sistema de signos**

    a) **Cuantitativos:** cuando se ocupan operaciones numéricas para resolver un problema; por ejemplo, al sumar los precios de dos productos adquiridos en el mercado.
    b) **Cualitativos:** cuando se utilizan instrucciones verbales en la resolución; por ejemplo, al atender las instrucciones de un instructor de manejo cuando muestra la manera de cambiar un neumático.

2. **Según su función**

    a) **De ordenamiento:** cuando se ordenan datos con base en determinadas normas o reglas.
    b) **De búsqueda:** cuando su tarea principal es encontrar un elemento en una lista determinada de datos.
    c) **De encaminamiento:** cuando se utilizan para buscar datos siguiendo una ruta determinada y mediante una serie de pasos enlazados, los cuales pueden ser *adaptativos* o *estáticos*. Los primeros permiten ajustarse según las circunstancias y los segundos funcionan de forma mecanizada y no permiten adaptación.

3. **Según la estrategia usada para llegar a un resultado**

    a) **Probabilísticos:** presentan soluciones que pueden ser erróneas o correctas, las cuales se denominan aproximaciones.
    b) **Cotidianos:** se utilizan en la vida diaria y no pueden ser ejecutados en una computadora.
    c) **Heurísticos:** se emplean cuando no se puede llegar a una solución de las maneras tradicionales.
    d) **En escalada:** para llegar a la solución de un problema con estos algoritmos se parte de una solución equivocada, la cual se modifica hasta llegar o estar cerca de la solución correcta.
    e) **Voraz:** el propósito único de este tipo de algoritmos es analizar cada paso como único y darle una solución óptima.
    f) **Deterministas:** estos algoritmos son lineales y predictivos, ya que se conoce exactamente cuáles son los datos o la información que contienen y cómo realizan los procesos para solucionar un problema.

Los algoritmos que se diseñarán en esta asignatura serán computacionales, y para este tipo de algoritmos existen algunas instrucciones o sentencias que debes aprender a utilizar. El cuadro de abajo muestra las principales acciones y sentencias que se utilizarán a lo largo del curso.

Acción	Sentencia que se utilizará
Establecer el valor para una variable	Asignar (variable = respuesta / variable = proceso)
Mostrar un mensaje	Decir: "mensaje"
Pedir algún dato o valor	Preguntar: "¿pregunta?"
Establecer un valor inicial para las variables	Programar (variable = 0 / variable = "")

Conceptos básicos para la creación de algoritmos

En la creación de los algoritmos también debes utilizar **variables** y **constantes**, que están estrechamente relacionadas con las fórmulas o teoremas que se emplearán, pues los algoritmos al ser usados por alguien más pueden darle solución a diversos problemas sin importar los datos que se introduzcan; siempre y cuando el problema sea del mismo tipo del que se basó el diseño del algoritmo.

Variables

El uso de variables no es nuevo para ti, ya que las has usado en asignaturas como Matemáticas, Física o Química. El concepto que conoces sobre variables hasta el momento es que estas son la expresión simbólica de un elemento cuyo valor es desconocido, esto quiere decir que una variable puede adquirir distintos valores. Las variables sin duda alguna son elementos fundamentales para diseñar y crear algoritmos, por tal motivo resulta de vital importancia que aprendas a utilizarlas en este apartado.

Relaciónalo con...

Has usado variables desde que estabas en la primaria; solo recuerda alguna fórmula que usaste para determinar el perímetro, área o volumen de una figura o cuerpo geométrico. Por ejemplo, el área del cuadrado: $A = L^2$.

En programación, una variable básicamente es un espacio de memoria en el que, de manera temporal, se almacenan datos de varios tipos, los cuales se utilizarán durante alguno de los procesos que se van a ejecutar durante la implementación del algoritmo.

Como podrás darte cuenta, el uso de variables en programación tiene la misma finalidad que en las asignaturas que se mencionaron antes; solo que al programar debes considerar que los recursos necesarios para resolver un problema los obtendrás de una computadora, y que las variables ocuparán un espacio temporal en la memoria RAM.

En programación, el espacio de memoria que ocupará una variable puede ser de longitud fija o variable. Las **variables de longitud fija** son aquellas cuyo tamaño o espacio no cambia a lo largo de la ejecución de los procesos, pues solo almacenan datos de cualquier tipo, y solo existen algunas excepciones, como las variables de tipo listas, arreglos o cadenas. Por el contrario, las **variables de longitud variable** modifican su tamaño, pues este tipo de variables se utilizan en las colecciones de datos o arreglos dinámicos.

Las variables usadas en programación deben cumplir ciertas normas o reglas para darles el nombre que las identifica; estas reglas son las siguientes:

1. Evitar el uso de caracteres especiales, acentos, espacios en blanco y símbolos. Si el nombre de la variable se compone de dos palabras se puede usar un guion bajo para unirlas; por ejemplo: numero_cuenta.
2. El nombre de la variable debe escribirse con letras minúsculas; por ejemplo: perimetro. Esto es por factibilidad a la hora de identificar las variables.
3. El nombre de la variable debe tener relación con el dato que almacenará, para ser identificada más rápido y reconocer el tipo de dato que se almacenará en ella. Por ejemplo, si se requiere almacenar dos números en dos variables distintas, las variables podrían llamarse numero1 y numero2.

Existen dos formas de clasificar a las variables informáticas, ya sea por su contenido o por su uso, como se describe a continuación:

Por su contenido:

- **Numéricas:** son aquellas que almacenan números positivos o negativos, incluidos los decimales.
- **Lógicas:** estas solo pueden tener dos significados, es decir, verdadero o falso; se conocen también como *booleanas* y son usadas para ofrecer el resultado de una comparación.
- **Alfanuméricas:** estas almacenan caracteres alfanuméricos, es decir, letras y números.

Shutterstock/nehopelon

Y tú, ¿eres constante o variable en tu humor?

Por uso:

- **De trabajo:** son las variables que recibirán la asignación del resultado de un proceso u operación.
- **Contadores:** son variables que llevan el control de las ocasiones en las que se realiza un proceso u operación mientras alguna condición se cumple. Son utilizadas específicamente dentro de los bucles o ciclos.
- **Acumuladores:** son usadas para llevar la suma acumulativa de una serie de valores que se han leído o calculado de manera progresiva.

Constantes

En el proceso de la programación, al igual que en las matemáticas, la constante es un valor que no puede ser modificado o alterado durante el algoritmo. Por tal motivo, a estas les corresponde un tamaño fijo en un área reservada en la memoria principal de la computadora, donde se almacenan valores fijos.

Por uso y costumbre, los nombres de las constantes se escriben con mayúsculas. Por ejemplo: PI = 3.1416. Al igual que las variables, las constantes pueden almacenar datos numéricos y alfanuméricos, los cuales no cambiarán.

Tipos de datos fundamentales

Reconocer los tipos de datos te permitirá representar diferente información en el momento de diseñar o crear un algoritmo, sobre todo cuando este sea programable en una computadora.

Existen tres grandes categorías de tipo de datos, las cuales ya has manejado en Excel, estas son numéricos, alfanuméricos y *booleanos*.

Datos numéricos. Los datos numéricos, como su nombre lo indica, son aquellos que comprenden números positivos y negativos; estos, a su vez, se dividen en dos categorías: enteros y decimales o de punto flotante.

Datos alfanuméricos. Se conforman por aquellos caracteres de letras y números que permiten representar información cuyo contenido es descriptivo, es decir, algunos los encontrarás como nombres de personas, direcciones, fechas, horas, etcétera. En ocasiones un dato numérico puede volverse alfanumérico, pero esto implicaría perder su propiedad matemática, es decir, no será posible realizar operaciones con él. El contenido de este tipo de datos, en general, estará encerrado entre comillas.

Booleanos. Estos datos solo pueden conformarse por dos tipos de valores: uno verdadero y otro falso, los cuales representarán el resultado de comparaciones realizadas durante diferentes procesos.

Como podrás darte cuenta, conocer el tipo de dato que vas a utilizar te ayudará a definir las características de las variables que emplearás en el algoritmo, pues se encuentran estrechamente relacionados.

Relaciónalo con...

En Microsoft Excel usaste y manipulaste datos que pertenecen alguna de las tres categorías de datos fundamentales, aunque en ese programa cumplen con distintas funciones a las de la programación; por ejemplo, las funciones BUSCARV, BUSCARH y PROMEDIO usan alguno de los tres tipos de datos.

Estructuras de control

Las estructuras de control en programación son aquellas que permiten controlar el orden de ejecución de las instrucciones del algoritmo empleado para la solución de un problema en específico. Las estructuras de control usan **comparaciones** como parámetro principal, y para poder especificar estas se usan operadores lógicos y los llamados de condición. El siguiente cuadro muestra los operadores lógicos y de condición más utilizados.

Operadores de comparación	
Operador	**Significado**
<	Menor que
>	Mayor que
< =	Menor o igual
> =	Mayor o igual
< > o !=	Diferente
==	Igual
Operadores lógicos	
Operador	**Significado**
.AND o && o Y	Y (unión)
OR o || u O	O (disyunción)
NOT o ! o NO	Negación

¿Sabías que...?

Los mensajes de los algoritmos computacionales deben escribirse entre comillas dobles y cumplir con todas las reglas ortográficas necesarias. Por ejemplo: "¡Hola, mundo!". Cuando los mensajes están compuestos por variables y caracteres se debe utilizar el símbolo + para unirlos o *concatenarlos*, así el mensaje será más claro.

Existen tres tipos de estructuras de control: secuenciales, selectivas y repetitivas. A continuación se describen las características de cada una de estas estructuras de control.

Estructuras de control secuencial

Este tipo de estructuras de control se ejecuta conforme aparece en el algoritmo, es decir, en orden secuencial, del paso 1 al 2, al 3 y así sucesivamente. Para comprender mejor este tipo de estructuras, piensa en la manera en que diseñaste y creaste un algoritmo para solucionar los problemas planteados en el tema anterior. Además, realiza las actividades que se describen a continuación con el apoyo de tu profesor. Cuando se quiere mostrar los valores incluidos en las variables es necesario concatenarlas usando el signo +.

Ejemplo 1.1.1

Problema: obtener el perímetro y el área de un cuadrado del cual no se conocen sus dimensiones.
Solución: diseñar un algoritmo con base en las fases de solución de problemas, usando estructuras de control secuencial.

Fase 1: se requiere conocer las fórmulas para calcular el perímetro y el área de un cuadrado. Con base en la definición de un cuadrado, la medida de sus cuatro lados es la misma. Por tanto:

- Fórmula del perímetro: P = 4*medida del lado.
- Fórmula del área: A = (medida del lado)^2.
- Variables: *perimetro*, *area* y *m_lado*.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

1. Inicio.
2. Programar todas las variables que se van a utilizar: *m_lado* = 0, *perimetro* = 0 y *area* = 0.
3. Decir: "Este algoritmo servirá para obtener el Perímetro y el Área de un cuadrado".
4. Preguntar: "¿Cuántos metros miden los lados del cuadrado?".
5. Asignar el valor de la respuesta a la variable *m_lado* (*m_lado* = respuesta).
6. Asignar el resultado al aplicar la fórmula para obtener el perímetro a la variable *perimetro* (*perimetro* = 4**m_lado*).
7. Asignar el resultado al aplicar la fórmula para obtener el área a la variable *area* (*area* = *m_lado***m_lado*).
8. Decir: "Para un cuadrado de" + *m_lado* + "metros por lado, su Perímetro mide" + *perimetro* + "metros lineales, y su Área" + *area* + "metros cuadrados.".
9. Fin.

Fase 3: para verificar el algoritmo anterior se debe asignar valores a las variables y realizar todos los procesos; así se obtendrá el resultado.

a) *m_lado* = 4.
b) *perimetro* = 4*4 (sustituyendo *m_lado* por 4).
c) *area* = 4*4 (sustituyendo *m_lado* por 4).
d) El resultado obtenido se leerá como sigue: "Para un cuadrado de 4 metros por lado, su Perímetro mide 16 metros lineales y su Área 16 metros cuadrados.".

A practicar 1.1.2

1. Con ayuda de tu profesor, crea el algoritmo adecuado para resolver el siguiente problema: calcular cuántas horas, minutos y segundos hay en una cantidad determinada de días.
2. Comienza por escribir una solución, como en el ejemplo 1.1.1.

3. Describe los pasos por seguir en cada una de las tres fases, tal como se mostró en el ejemplo 1.1.1. de la página anterior.

4. Usa las estructuras de control secuencial contemplando las tres fases de solución de problemas.
5. Escribe el algoritmo en un documento de Word y guárdalo porque lo utilizarás más adelante.

Insignia 1.1.1

Crea un algoritmo, usando estructuras de control secuencial, para resolver el siguiente problema basado en las tres fases de solución de problemas: calcular el área y el volumen de un cilindro. Guarda el algoritmo en un documento de Word y envíalo a tu profesor.

Encamina tus habilidades

1. De manera individual, crea un algoritmo para resolver el siguiente problema:

   En una gasolinera se registra el ingreso del líquido, el cual se almacena en galones, pero el precio de la gasolina se establece en litros y en dólares. Se debe obtener como resultado el precio de venta al público en pesos mexicanos por la gasolina cargada; además, debe agregarse la información del tipo de cambio y el costo por litro de gasolina.

2. Al crear el algoritmo, sigue las fases de solución de problemas y usa estructuras de control secuencial.
3. Guarda el algoritmo en un documento de Word y envíalo a tu profesor para su evaluación; pídele que realice una retrolimentación de tu trabajo porque usarás el mismo algoritmo más adelante en el curso.

Relaciónalo con...

Las estructuras de control ya las has usado antes, solo recuerda las funciones SI, SI-Y y SI-O de Excel.

Estructuras de control selectivo

Este tipo de estructuras se usa cuando la continuación del desarrollo del algoritmo debe tomar una decisión para continuar con un proceso o una ruta alternativa. Estas decisiones se basan en la valoración de una o más condiciones, las cuales señalan hacia dónde se modificará el sentido del algoritmo. Las estructuras de control selectivo se dividen en tres grandes tipos: simples, dobles y múltiples.

Las **estructuras de control selectivo simple** tienen como característica principal que se componen por una sola condición, es decir, el que se ejecute uno o más procesos va a depender si la condición evaluada se cumple; de lo contrario no se realiza ningún proceso o acción. Este tipo de estructuras expresan el algoritmo de la siguiente manera:

```
Si (condición(es)) entonces
   ..... Proceso(s)
Fin del Si
```

Las **estructuras de control selectivo doble**, al igual que las simples, están compuestas por una sola condición, pero en estas habrá dos caminos que tomar con relación a los procesos del algoritmo; uno será si la condición evaluada se cumple y el otro si no se cumple. En otras palabras, si se cumple o no se realizarán distintos procesos. Este tipo de estructuras se expresan así:

```
Si (condición(es)) entonces
   ..... Proceso(s)
De lo contrario
   ..... Proceso(S)
Fin del Si
```

Ejemplo 1.1.2

Problema: conocer si un número es positivo o negativo.
Solución: diseñar dos algoritmos, uno para solucionar el problema usando estructuras de control selectivo simple y otro mediante estructuras de control selectivo doble.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructuras de control selectivo simple

1. Inicio.
2. Programar las variables que se van a utilizar (*numero* = 0).
3. Decir: "Este algoritmo ayudará a conocer si un número es Positivo o Negativo.".
4. Preguntar: "¿Cuál número deseas saber si es Positivo o Negativo?".
5. Asignar la respuesta a la variable *numero* (*numero* = respuesta).
6. Si (*numero* > 0), entonces.
7. Decir: "El número" + *numero* + "es Positivo.".
8. Fin del *Si*.
9. Si (*numero* < 0), entonces.
10. Decir: "El número" + *numero* + "es Negativo.".
11. Fin del Si.
12. Fin.

Estructuras de control selectivo doble

1. Inicio.
2. Programar las variables que se van a utilizar (*numero* = 0).
3. Decir: "Este algoritmo ayudará a conocer si un número es Positivo o Negativo.".
4. Preguntar: "¿Cuál número deseas saber si es Positivo o Negativo?".
5. Asignar la respuesta a la variable *numero* (*numero* = respuesta).
6. Si (*numero* > 0), entonces.
   6.1. Decir: "El número" + *numero* + "es Positivo.".
7. De lo contrario
   7.1. Escribir: "El número" + *numero* + "es Negativo.".
8. Fin del Si.
9. Fin.

A practicar 1.1.3

1. Con ayuda de tu profesor, crea el algoritmo adecuado para resolver el siguiente problema: con base en la calificación de un alumno como dato de entrada, obtener como resultado si está "Aprobado", si la calificación mínima es de 6 y "Reprobado" si es lo contrario.
2. Comienza por escribir una solución en el siguiente espacio.

3. Describe los pasos por seguir en cada una de las tres fases, tal como se mostró en el ejemplo 1.1.1 de la página 18.

4. Usa una estructura de control simple o doble para resolver el problema planteado.
5. Escribe el algoritmo en un documento de Word y guárdalo porque lo utilizarás más adelante.

Por último, las **estructuras de control selectivo múltiples**, también conocidas como *anidadas*, se dividen para su estudio en dos tipos, las **estructuras de control selectivo simple anidado** y las selectivas de caso. Las primeras están formadas por estructuras de control selectivas simples anidadas con la instrucción **de lo contrario Si**; estas estructuras se pueden identificar en los algoritmos de la siguiente manera:

```
Si (condición(es)) entonces
   ..... Proceso(s)
De lo contrario Si (condición(es)) entonces
   ..... Proceso(s)
De lo contrario
   ..... Proceso(s)
Fin del Si
```

Las **estructuras de control selectivo de caso** se identifican en los algoritmos con la expresión **En caso de**, las cuales toman como parámetro una variable tipo opción, la cual también sirve de guía para el flujo del algoritmo con relación a los procesos por realizar, y siempre tendrá una opción por defecto. En un algoritmo se identifican de la siguiente manera:

```
En caso de (opción)
   Caso valor1:
        .... Proceso(s)
        Interrumpir
   Caso valor2:
        ..... Proceso(s)
        Interrumpir
   Caso defecto:
        ..... Proceso(s)
Fin del caso
```

Ejemplo 1.1.3

Problema: conocer si un número es positivo, negativo o igual a cero.
Solución: diseñar un algoritmo para solucionar el problema utilizando estructuras de control selectivo simple anidado.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructuras de control selectivo simple anidado

1. Inicio.
2. Programar las variables que se van a utilizar (*numero* = 0).
3. Decir: "Este algoritmo ayudará a conocer si un número es Positivo o Negativo.".
4. Preguntar: "¿Cuál número deseas saber si es Positivo o Negativo?".
5. Asignar la respuesta a la variable *numero* (*numero* = respuesta).
6. Si (*numero* > 0), entonces.
   6.1. Decir: "El número" + *numero* + "es Positivo.".
7. De lo contrario, Si (*numero* < 0), entonces.
   7.1. Decir: "El número" + *numero* + "es Negativo.".
8. De lo contrario.
   8.1. Decir: "El número es igual a Cero".
9. Fin del Si.
10. Fin.

Ejemplo 1.1.4

Problema: conocer el nombre del día de la semana con solo introducir el número de día que es.
Solución: diseñar un algoritmo para solucionar el problema utilizando estructuras de control selectivo de caso.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructuras de control selectivo de caso

1. Inicio.
2. Programar las variables que se van a utilizar (*dia* = 0).
3. Decir: "Con este algoritmo conocerás el nombre del día de la semana que es con solo introducir el número.".
4. Preguntar: "Dame un número entero del 1 al 7 para saber qué día es.".
5. Asignar la respuesta a la variable *dia* (*dia* = respuesta).
6. En caso de (*dia*).
   - 6.1. Caso 1:
     - 6.1.1. Decir: "El día" + *dia* + "de la semana es Lunes.".
     - 6.1.2. Interrumpir.
   - 6.2. Caso 2:
     - 6.2.1. Decir: "El día" + *dia* + "de la semana es Martes.".
     - 6.2.2. Interrumpir.
   - 6.3. Caso 3:
     - 6.3.1. Decir: "El día" + *dia* + "de la semana es Miércoles.".
     - 6.3.2. Interrumpir.
   - 6.4. Caso 4:
     - 6.4.1. Decir: "El día" + *dia* + "de la semana es Jueves.".
     - 6.4.2. Interrumpir.
   - 6.5. Caso 5:
     - 6.5.1. Decir: "El día" + día + "de la semana es Viernes.".
     - 6.5.2. Interrumpir.
   - 6.6. Caso 6:
     - 6.6.1. Decir: "El día" + *dia* + "de la semana es Sábado.".
     - 6.6.2. Interrumpir.
   - 6.7. Caso 7:
     - 6.7.1. Decir: "El día" + *dia* + "de la semana es Domingo.".
     - 6.7.2. Interrumpir.
   - 6.8. Caso defecto:
     - 6.7.1. Decir: "Opción no válida.".
7. Fin.

A practicar 1.1.4

1. Con todo el grupo, diseña y crea un algoritmo en Word usando la estructura de control selectivo (simple, anidado o de caso) necesaria para resolver el siguiente problema: calcular el pago del sueldo y horas extras de un empleado con base en los datos de la siguiente tabla:

Categoría	A	B	C	D
Sueldo	$83.55	$93.55	$103.55	$113.55
Hora extra	$30.50	$37.60	$48.90	$65.50

Estructuras de control repetitivo

Las estructuras de control repetitivo, también llamadas de *iteracción*, permiten ejecutar uno o varios procesos de manera repetida. En algunas referencias bibliográficas las podrás encontrar con el nombre de *ciclos* o *bucles* porque estas estructuras de control permiten que se repita una serie de procesos. Para diseñar y crear este tipo de estructuras es necesario considerar dos aspectos muy importantes: ¿cuántas veces se repite el ciclo?, y ¿cuál será el contenido o cuerpo del ciclo? En este tipo de estructuras de control siempre se usan las variables del tipo contador, las cuales juegan un papel muy importante en el funcionamiento de los ciclos o bucles porque las condicionantes con las que se trabajan los algoritmos responderán al contenido de los contadores; es decir, sin un contador los ciclos quedarán ciclados, valga la redundancia, pues nunca se podría salir de ellos; por tanto, se debe tener cuidado de siempre incluir contadores en los ciclos.

Shutterstock/Viktorija Reuta

¿Qué tipo de problemas se podrían resolver con las estructuras de control repetitivo?

El contador puede comportarse de dos maneras, ya sea en *aumento* o en *decremento*. El comportamiento de los contadores depende de los procesos que necesites ejecutar dentro de los ciclos. Los contadores se identifican con variables cuyo nombre empiezan con las letras *i, j, k, l, m, n*. Además de los contadores, en ocasiones usarás variables tipo bandera, conocidas también como *booleanas*, cuyo propósito es servir de interruptor dentro del ciclo. Estas variables deben estar inicializadas en 0 antes de entrar al ciclo; una vez dentro de este, según el proceso o condición, puede cambiar su valor a 1, ya que 0 es *false* (falso o apagado) y 1 es *true* (verdadero o encendido).

Existen tres principales estructuras de control repetitivas, que son: **mientras**, **para** y **haz mientras**; de las cuales solo se aplicarán las dos primeras. La estructura **Mientras** se conoce también como *while* en el ámbito de la programación, y se utiliza principalmente cuando no se conoce el número de repeticiones o vueltas que debe dar el bucle al realizar los procesos establecidos. Este número depende de las proposiciones o condiciones dentro del ciclo, es decir, los procesos que se encuentren en el cuerpo del ciclo se ejecutarán siempre y cuando su condición siga siendo verdadera; en el momento en el que esta no se cumpla el ciclo será interrumpido y el flujo del algoritmo continuará con la siguiente secuencia de pasos.

La estructura de este ciclo consta de dos partes, que son: el ciclo (conjunto de procesos o instrucciones que se ejecutarán de manera iterativa), y la condición (elemento que se evaluará para permanecer o salir de la ejecución del ciclo en sí). En los algoritmos, la estructura *Mientras* se identificará como se muestra a continuación:

```
Mientras (condición(es))
   ..... Proceso(s)
Fin del Mientras
```

La estructura de control repetitivo **Haz Mientras** se conoce también como *do while*. Su funcionamiento es muy similar al del ciclo *mientras*, solo que en esta los procesos dentro del cuerpo del ciclo se ejecutarán al menos una vez antes de que la condición sea evaluada, es decir, la estructura de este bucle permite una ejecución al menos antes de salirse por completo si la condición no se llega a cumplir.

Al igual que en la estructura *Mientras* (*while*), *Haz Mientras* consta de dos partes, que son el ciclo (conjunto de procesos o instrucciones que se ejecutarán de manera iterativa), y la condición (elemento que se evaluará para permanecer o salir de la ejecución del ciclo en sí). En los algoritmos, la estructura *Haz Mientras* se identifica con la estructura siguiente:

```
Haz
   .... Proceso(s)
Mientras (condición(es))
```

La estructura **Para** se llama también *for*. A diferencia de las estructuras mencionadas antes, esta se usa cuando ya se encuentra definido el número de veces que se ejecutará el ciclo, aunque en ocasiones se puede utilizar para hacer ciclos infinitos.

Para es la estructura de control repetitivo más usada entre los desarrolladores debido a su sencilla estructura. Su expresión está compuesta por cuatro partes: la tarea de inicialización (indica a partir de qué número se empezará a contar), la condición (elemento que se evaluará para permanecer o salir de la ejecución del ciclo en sí), la tarea final por vuelta (define el comportamiento de la variable contador dentro del ciclo) y, por último, el ciclo (conjunto de procesos o instrucciones que se ejecutarán de manera iterativa). En los algoritmos se encontrará expresada de la siguiente manera:

```
Para (tarea_de_inicialización; condición; tarea_final_por_vuelta)
   .... Proceso(s)
Fin del Para
```

Ejemplo 1.1.5

Problema: visualizar la tabla de multiplicar de cualquier número entero positivo con resultados que abarquen hasta ser multiplicado por 15.
Solución: diseñar un algoritmo para solucionar el problema utilizando la estructura de control repetitivo *Mientras*.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructura de control repetitivo *Mientras*

1. Inicio.
2. Programar las variables que se van a utilizar (*numero* = *0*, *i* = *0*, *resultado* = 0).
3. Decir: "Con este algoritmo conocerás la tabla de multiplicar comprendida del 1 al 15 de cualquier número entero positivo.".
4. Preguntar: "¿De qué número quieres conocer su tabla de multiplicar?".
5. Asignar la respuesta a la variable *numero* (*numero* = respuesta).
6. Decir: "Tabla de multiplicar del número:" + *numero*.
7. Asignar *i* como valor inicial (*i* = 1).
8. Mientras (i <= 15).
   - 8.1. Asignar el resultado del cálculo a la variable *resultado* (*resultado* = *numero***i*).
   - 8.2. Decir: *numero* + "x" + *i* + "=" + *resultado*.
   - 8.3. *i* = *i* + 1.
9. Fin del *Mientras*.
10. Fin.

Ejemplo 1.1.6

Problema: visualizar la tabla de multiplicar de cualquier número entero positivo con resultados que abarquen hasta ser multiplicado por 15.
Solución: diseñar un algoritmo para solucionar el problema utilizando la estructura de control repetitivo *Para*.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructura de control repetitivo *Para*

1. Inicio.
2. Programar las variables por utilizar (numero = 0, *i* = 0, *resultado* = 0).
3. Decir: "Con este algoritmo conocerás la tabla de multiplicar comprendida del 1 al 15 de cualquier número entero positivo.".
4. Preguntar: "¿De qué número quieres conocer su tabla de multiplicar?".
5. Asignar la respuesta a la variable *numero* (*numero* = respuesta).
6. Decir: "Tabla de multiplicar del número:" + *numero*.
7. Para (i = 1; i <= 15; i = i + 1).
   7.1. Asignar el resultado del cálculo a la variable *resultado* (*resultado* = *numero***i*).
   7.2. Decir: *numero* + "x" + *i* + "=" + *resultado*.
8. Fin del *Para*.
9. Fin.

Encamina tus habilidades

1. De manera individual, crea un algoritmo para resolver el siguiente problema:

   Introducir un número entero cualquiera y determinar si este número es **Primo** o **No primo**.

2. Al crear el algoritmo, sigue las fases de solución de problemas y usa estructuras de control secuencial, selectivo y repetitivo que creas necesarias para que se cumpla el proceso y se obtenga el resultado adecuado.
3. Guarda el algoritmo en un documento de Word y envíalo a tu profesor para su evaluación; pídele que realice una retrolimentación de tu trabajo porque usarás el mismo algoritmo más adelante en el curso.

Insignia 1.1.2

Investiga en internet qué son los arreglos y, en un documento en Microsoft Word, elabora una tabla comparativa de las variables simples y los arreglos, en la cual deberás resaltar sus diferencias, ventajas y desventajas de uso.

A practicar 1.1.5

Con ayuda de tu profesor, y con todo el grupo, diseñen y escriban en el pizarrón el algoritmo que resuelva el siguiente problema: introducir cierta cantidad de números y mostrarlos ordenados de manera ascendente. Para lograrlo, usen las estructuras de control secuencial, selectivo y repetitivo que consideren necesarias.

Para solucionar problemas similares al de la actividad anterior se necesita implementar una **estructura de datos**, es decir, la manera en que se organiza la información. En general, una estructura de datos es una colección de datos de un mismo tipo agrupados en un contenedor. Existen dos grandes tipos de estructuras de datos, que son las siguientes:

a) Estáticas (arreglos y estructuras). Presentan un tamaño fijo.
b) Dinámicas (listas, pilas, colas y árbol). Presentan un tamaño variable o dinámico.

Las estructuras de datos, sin importar su tipo, presentan varios métodos comunes, que pueden ser los siguientes:

- Insertar elemento.
- Eliminar elemento.
- Modificar o editar elemento.
- Ordenar los elementos.
- Buscar uno o varios elementos en particular.

Por cuestiones didácticas de este material, de los dos grandes tipos de estructuras de datos centrarás tu estudio en las estructuras estáticas, específicamente en los arreglos.

¿Qué son los arreglos?

Un arreglo es una colección finita de datos del mismo tipo, que sirve para manipular un número determinado de elementos en común. En los arreglos, cada elemento contenido en él tiene un lugar específico asignado, el cual se nombra de manera específica, conjugando el nombre del arreglo y el dato del espacio o casilla donde se ubica.

A manera de metáfora, un arreglo puede representar una caja con muchos casilleros, y cada casillero tiene un nombre específico; dentro de cada uno se encuentran elementos siempre del mismo tipo.

Para manipular los elementos contenidos en los arreglos siempre se recurre a los ciclos o bucles (sin importar cuál), y para avanzar dentro del arreglo se usa siempre una variable de tipo índice, que es un contador de posiciones en el arreglo. La posición inicial de cualquiera de los tipos de arreglos que existen está definida en el elemento 0.

¿Sabías que...?

Para evitar confusiones en los algoritmos que usan variables tipo contador para recorrer o insertar elementos en un arreglo, se les asignan las letras del alfabeto comprendidas de la i a la n.

Tipos de arreglos

En programación existen tres tipos de arreglos, que son los unidimensionales, los bidimensionales y los multidimensionales.

Los **arreglos unidimensionales** presentan una estructura natural para mostrar listas, las cuales se componen de elementos iguales; para acceder a estos elementos se necesita un índice. Los **arreglos bidimensionales** básicamente se usan para representar datos en forma de tablas con filas y columnas; para acceder a sus elementos se necesitan dos índices. Por último, los **arreglos multidimensionales** son similares a los bidimensionales, pero en estos se necesita *n* cantidad de índices para acceder a sus elementos.

En este material didáctico se describirán las características únicamente de los *arreglos unidimensionales* para que logres diseñar y crear nuevos algoritmos. Para manipular arreglos con cualquiera de sus métodos se pueden aplicar hasta cinco metodologías, las cuales se describen a continuación:

1. Declaración de arreglos con elementos previamente definidos.

```
nombre_del_arreglo[dimensión] = {elemento1, elemento2,
   elemento3, …, elemento_n}
```

2. Declaración de arreglos con tamaño definido, pero sin elementos.

```
nombre del arreglo[dimensión]
```

3. Declaración o inicialización de arreglos sin tamaño ni elementos definidos, es decir, los elementos se leerán de la siguiente manera:

```
nombre_del_arreglo[]
```

4. Inserción y modificación de elementos.

```
nombre_del_arreglo[posición] = elemento
```

O bien, con la siguiente estructura:

```
Para (contador=0; contador<dimensión; contador+1)
   nombre_del_arreglo[contador] = elemento
Fin del Para
```

5. Ordenamiento con el algoritmo denominado método burbuja:

```
Para (contador_1=0; contador_1<dimensión-1; contador_1+1)
   Para (contador_2=0; contador_2<dimensión-1; contador_2+1)
      Si (nombre_del_arreglo[contador_2] > nombre_del_
      arreglo[contador_2+1]) entonces
         band = nombre_del_arreglo[contador_2]
         nombre_del_arreglo[contador_2] = nombre_del_arreglo[contador_2+1]
         nombre_del_arreglo[contador_2+1] = band
      Fin del Si
   Fin del Para
Fin del Para
```

Ejemplo 1.1.7

Problema: calcular el exponente al cubo de cinco números enteros cualquiera.
Solución: diseñar un algoritmo para solucionar el problema utilizando las estructuras de control secuencial, selectivo y repetitivo, así como las estructuras de datos que consideres necesarias.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructura de datos

1. Inicio.
2. Programar las variables por utilizar (*numeros[5]* , *i* = 0, *cubo* = 0).
3. Decir: "Con este algoritmo conocerás el exponente al cubo de cinco números cualquiera.".
4. Para (i = 0; i < 5; i = i + 1).
   4.1. Preguntar: "Dame un número cualquiera:".
   4.2. Insertar respuesta como elemento de *numeros[]* en la posición *i* (*numeros[i]* = respuesta).
5. Fin del *Para*.
6. Para (i = 0; i < 5; i = i + 1).
   6.1. Asignar a la variable exponente al cubo el resultado de la operación *cubo = numeros[i]*numeros[i]*numeros[i])*.
   6.2. Decir: *numeros[i]* + "al cubo es:" + *cubo*.
7. Fin del *Para*.
8. Fin.

A practicar 1.1.6

1. Con ayuda de tu profesor, crea el algoritmo adecuado para resolver el siguiente problema: introducir cierta cantidad de números y mostrarlos ordenados de manera ascendente. Usa todo lo que has aprendido en esta secuencia didáctica para diseñar y crear el algoritmo adecuado
2. Comienza por escribir una solución en el siguiente espacio.

3. Describe los pasos que se van a seguir en cada una de las tres fases, tal como se mostró en el ejemplo 1.1.1 de la página 18.

Fase 1: describir las fórmulas por utilizar para solucionar el problema:

Fase 2: describir los pasos para crear el algoritmo:

Fase 3: describir los elementos para validar el algoritmo:

4. Escribe el algoritmo en un documento de Word y guárdalo porque lo utilizarás más adelante.

Subproductos

Subproducto procedimental

1. Reúnete con otro compañero y, con base en lo que aprendieron en esta secuencia didáctica, diseñen y elaboren un algoritmo que permita resolver el siguiente problema: conocer cuántos y cuáles números primos existen antes de un número entero positivo cualquiera.
2. Retomen las evidencias que obtuvieron al realizar las actividades de las secciones *Encamina tus habilidades* para diseñar las estructuras de control secuencial, selectivo y repetitivo, así como las de datos que crean necesarias para resolver el problema planteado.
3. Compartan su algoritmo con el profesor para que lo evalúe y reciban retroalimentación.
4. Apliquen las modificaciones necesarias y guarden el algoritmo en un documento de Word.

Subproducto declarativo

› Contesta lo siguiente con base en las indicaciones de tu profesor.

1. ¿Cuáles son las etapas de la resolución de problemas?
2. ¿Cuáles son las etapas del diseño y creación de algoritmos?
3. Explica por qué se deben usar variables en el diseño y creación de algoritmos.
4. Según el funcionamiento de los ciclos y bucles, explica cuál es su propósito y cuál te resultó más fácil de usar.
5. ¿Por qué y para qué es necesario el uso de arreglos para lo resolución de problemas?

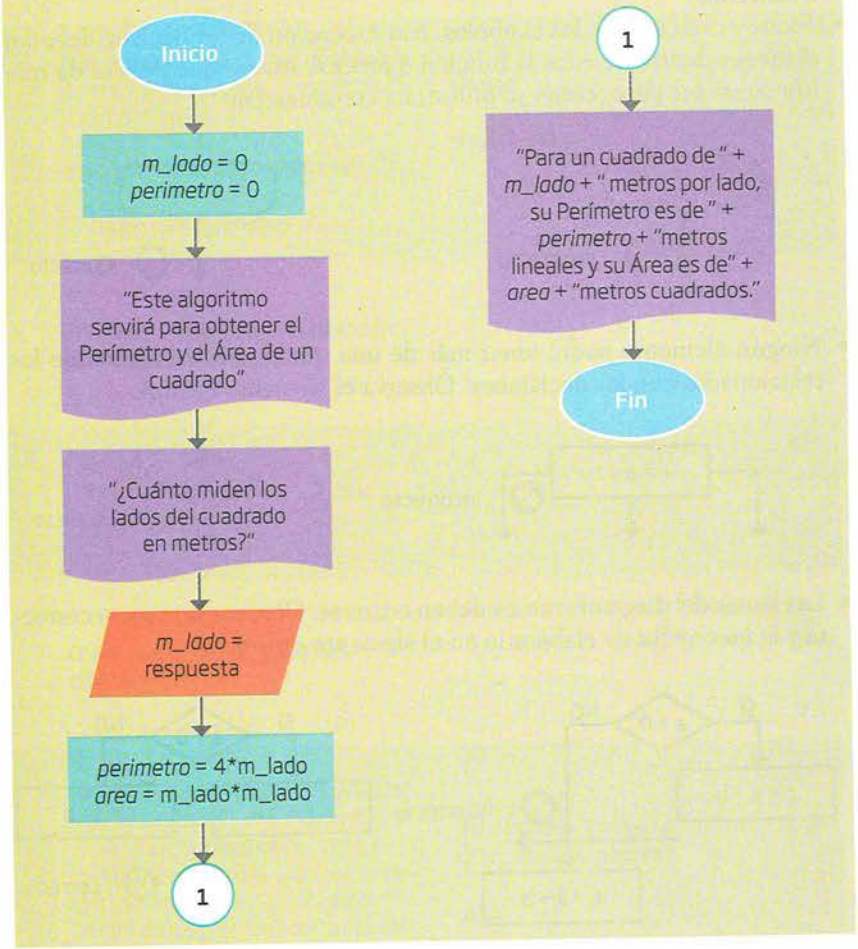
Subproducto actitudinal-valoral

› Con base en lo que has aprendido en esta secuencia didáctica, comenta con tu grupo lo siguiente: ¿ha cambiado tu forma de analizar los problemas que se te presentan en otras asignaturas? Justifica tu respuesta.

Ejemplo 1.2.1

Problema: obtener el perímetro y el área de un cuadrado del que se desconocen sus dimensiones.
Solución: retomar el algoritmo que elaboraste al resolver el problema del *Ejemplo 1.1.1*, en el cual usaste estructuras de control secuencial, y crear el diagrama de flujo con base en ese algoritmo.

Fase 2: crear el diagrama de flujo:



Insignia 1.2.1

Para obtener esta insignia deberás crear un algoritmo y diagrama de flujo usando estructuras de control secuencial, que resuelvan el siguiente problema: calcular el perímetro y el área de un terreno poligonal de más de cuatro lados.

A practicar 1.2.1

Retoma la evidencia que obtuviste al realizar la actividad de la sección *A Practicar 1.1.2* de la página 19 y, con ayuda de tu profesor, crea el diagrama de flujo con estructuras de control secuencial que resuelva el problema mencionado; no olvides estructurarlo con base en las fases para la solución de problemas. Guarda el diagrama de flujo en el mismo documento de Word donde está el algoritmo, recuerda que lo utilizarás de nuevo más adelante.

Encamina tus habilidades

1. De manera individual, crea el algoritmo y el diagrama de flujo que ayude a resolver el siguiente problema: un alumno necesita saber el promedio final de sus calificaciones obtenidas durante un semestre de una asignatura específica, en la cual fue evaluado en cuatro ocasiones distintas.
2. Al crear el algoritmo y el diagrama de flujo, sigue las fases de solución de problemas y usa la estructura de control secuencial para que se cumpla el proceso y se obtenga el resultado adecuado.
3. Guarda el algoritmo en un documento de Word y envíalo a tu profesor para su evaluación; pídele que realice una retrolimentación de tu trabajo y aplícala en cuanto la conozcas.

Ejemplo 1.2.2

Problema: saber, de tres números cualquiera, cuál de ellos es el mayor, cosiderando que pueden ser iguales.
Solución: crear un algoritmo y un diagrama de flujo, usando estructuras de control selectivo, ya sea simples o multiples.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructura de control selectivo

1. Inicio.
2. Programar las variables que se van a utilizar (*num1* = 0, *num2* = 0, *num3* = 0).
3. Decir: "A partir de tres números cualquiera podrás saber cuál es el mayor.".
4. Preguntar: "Dame el primer número.".
5. Asignar la respuesta a la variable *num1* (*num1* = respuesta).
6. Preguntar: "Dame el segundo número.".
7. Asignar la respuesta a la variable *num2* (*num2* = respuesta).
8. Preguntar: "Dame el tercer número.".
9. Asignar la respuesta a la variable *num3* (*num3* = respuesta).
10. Si (*num1* > *num2*), entonces:
    - 10.1. Si (*num1* > *num3*), entonces:
        - 10.1.1. Decir: "El número" + *num1* + "es el mayor de los tres capturados.".
    - 10.2. De lo contrario, Si (*num1* = = *num3*), entonces:
        - 10.2.1. Decir: "Los números" + *num1* + "y" + *num3* + "son iguales y son los mayores.".
    - 10.3. De lo contrario:
        - 10.3.1. Decir: "El número" + *num3* + "es el mayor de los tres capturados.".
    - 10.4. Fin del *Si*.
11. De lo contrario, Si (*num2* = = *num1*), entonces:
    - 11.1. Si (*num2* > *num3*), entonces:
        - 11.1.1. Decir: "Los números" + *num1* + "y" + *num2* + "son iguales y son los mayores.".

11.2. De lo contrario, Si (*num2* = = *num3*), entonces:
11.2.1. Escribir: "Todos los números son iguales".
11.3. De lo contrario:
11.3.1. Decir: "El número" + *num3* + "es el mayor de los tres capturados.".
11.4. Fin del *Si*.
12. De lo contrario, Si (*num2* > *num3*), entonces:
12.1. Decir: "El número" + *num2* + "es el mayor los tres capturados.".
13. De lo contrario, Si (*num2* = = *num3*), entonces:
13.1. Decir: "Los números" + *num2* + "y" + *num3* + "son iguales y son los mayores.".
14. De lo contrario:
14.1. Decir: "El número" + *num3* + "es el mayor de los tres capturados.".
15. Fin del *Si*.
16. Fin.

Fase 2: crear el diagrama de flujo:

Estructura de control selectivo

En la página 39 podrás observar un ejemplo del diagrama de flujo para resolver este problema.

Ejemplo 1.2.3

Problema: conocer el nombre del día de la semana con solo introducir el número de día que es.
Solución: retomar el algoritmo que elaboraste al resolver el problema del *Ejemplo 1.1.4*, de la página 23, en cual usaste estructuras de control selectivo, y crear el diagrama de flujo con base en ese algoritmo.

Fase 2: crear el diagrama de flujo:

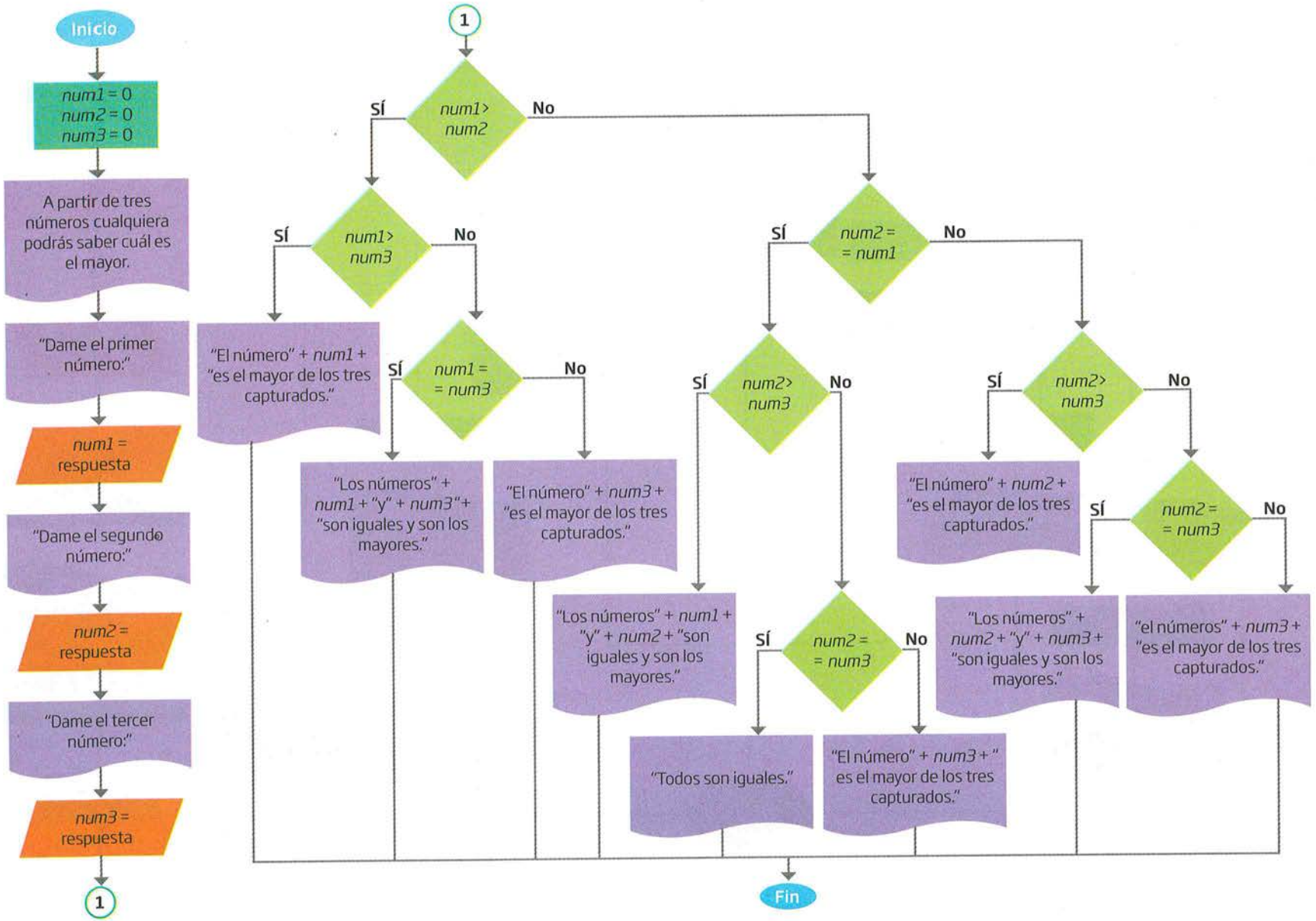
Estructura de control selectivo

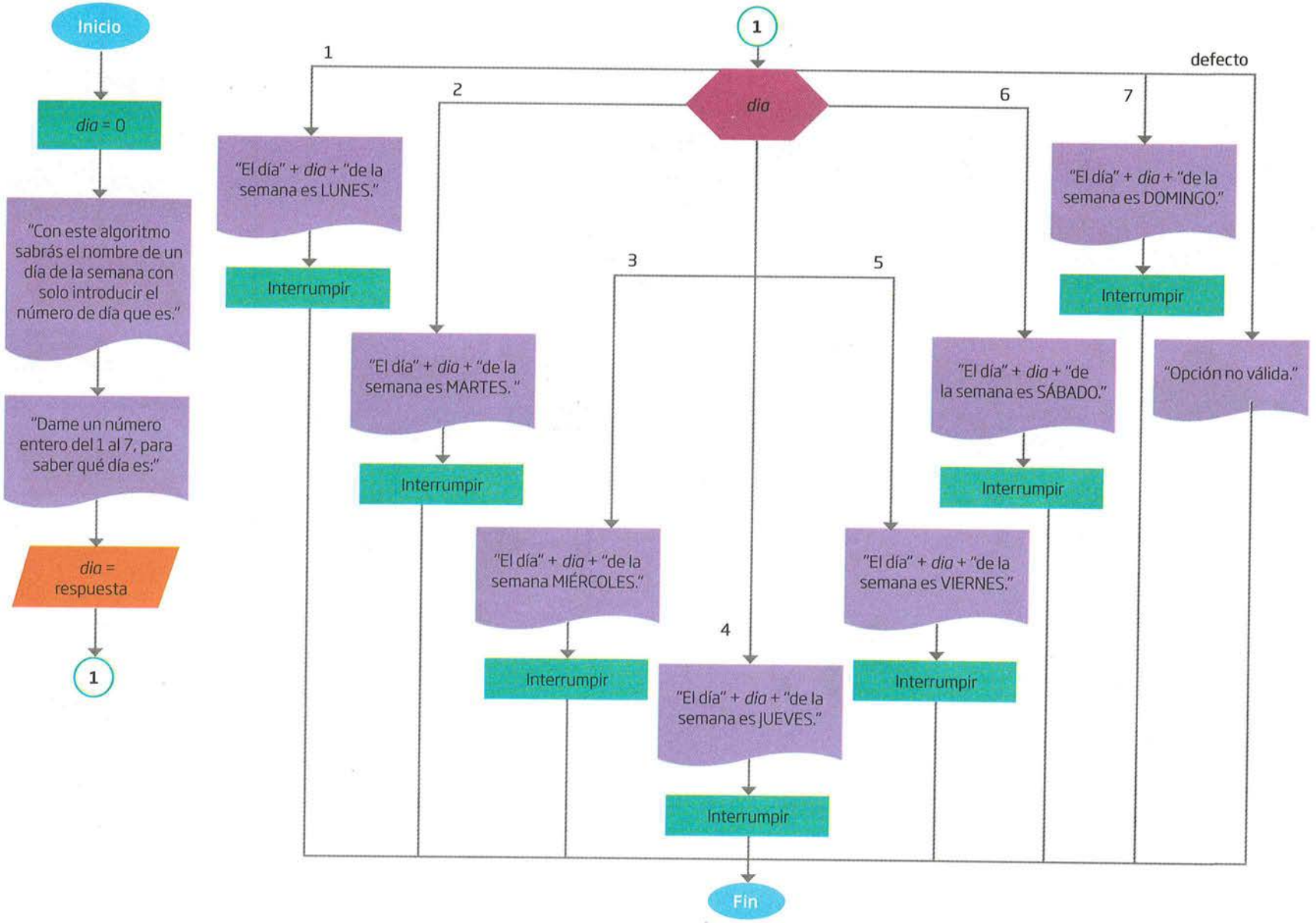
En la página 40 podrás observar un ejemplo del diagrama de flujo para resolver este problema.

A practicar 1.2.2

Retoma la evidencia que obtuviste al realizar la actividad de la sección *A Practicar 1.1.3* de la página 21 y, con ayuda de tu profesor, crea el diagrama de flujo con estructuras de control simple o doble que resuelva el problema mencionado; no olvides estructurarlo con base en las fases para la solución de problemas.

Guarda el diagrama de flujo en el mismo documento de Word donde está el algoritmo, recuerda que lo utilizarás de nuevo más adelante.


Inicio
num1 = 0
num2 = 0
num3 = 0
A partir de tres números cualquiera podrás saber cuál es el mayor.
"Dame el primer número:"
num1 = respuesta
"Dame el segundo número:"
num2 = respuesta
"Dame el tercer número:"
num3 = respuesta
1
1
num1 > num2
SÍ
No
num1 > num3
SÍ
No
"El número" + num1 + "es el mayor de los tres capturados."
num1 = = num3
SÍ
No
"Los números" + num1 + "y" + num3 + "son iguales y son los mayores."
"El número" + num3 + "es el mayor de los tres capturados."
num2 = = num1
SÍ
No
num2 > num3
SÍ
No
"Los números" + num1 + "y" + num2 + "son iguales y son los mayores."
num2 = = num3
SÍ
No
"Todos son iguales."
"El número" + num3 + " es el mayor de los tres capturados."
num2 > num3
SÍ
No
"El número" + num2 + "es el mayor de los tres capturados."
num2 = = num3
SÍ
No
"Los números" + num2 + "y" + num3 + "son iguales y son los mayores."
"el números" + num3 + "es el mayor de los tres capturados."
Fin


Inicio
dia = 0
"Con este algoritmo sabrás el nombre de un día de la semana con solo introducir el número de día que es."
"Dame un número entero del 1 al 7, para saber qué día es:"
dia = respuesta
1
1
dia
1
2
3
4
5
6
7
defecto
"El día" + dia + "de la semana es LUNES."
Interrumpir
"El día" + dia + "de la semana es MARTES."
Interrumpir
"El día" + dia + "de la semana MIÉRCOLES."
Interrumpir
"El día" + dia + "de la semana es JUEVES."
Interrumpir
"El día" + dia + "de la semana es VIERNES."
Interrumpir
"El día" + dia + "de la semana es SÁBADO."
Interrumpir
"El día" + dia + "de la semana es DOMINGO."
Interrumpir
"Opción no válida."
Fin

Ejemplo 1.2.4

Problema: después de recibir como datos cierta cantidad de números enteros, determinar cuál es el mayor y cuál es el menor de todos, pero mostrando la posición en el arreglo donde se encuentra cada uno de ellos y su valor.
Solución: diseñar y crear un algoritmo y un diagrama de flujo, utilizando estructuras de control secuencial, selectivo y repetitivo, así como estructuras de datos.

Fase 2: seguir el siguiente algoritmo para resolver el problema:

Estructuras de control repetitivo y de datos

1. Inicio.
2. Programar las variables que se van a utilizar (*numeros[]*, *mayor* = 0, *menor* = 0, *i* = 0, *n* = 0, *band* = 0, *pos_mayor* = 0, *pos_menor* = 0).
3. Decir: "Ahora podrás capturar varios números y almacenarlos en un arreglo, y al final saber cuál es el mayor y cuál el menor, además de conocer exactamente en qué posición del arreglo se encuentra cada uno.".
4. Preguntar: "¿Cuántos números deseas capturar?".
5. Asignar la respuesta a la variable *n* (*n* = respuesta).
6. Para (*i* = 1; *i* < = *n*; *i* = *i* + 1).
   6.1. Preguntar: "Número" + *i* + ":".
   6.2. Insertar respuesta como elemento del arreglo *números* en la posición *i* (*números[i]* = respuesta).
7. Fin del *Para*.
8. Para (*i* = 1; *i* < = *n*; *i* = *i* + 1).
   8.1. Si (*band* = = 0), entonces:
      8.1.1. Asignar a la variable *band* valor de 1 (*band* = 1).
      8.1.2. Asignar a la variable *mayor* el valor del elemento *i* del arreglo *números* (*mayor* = *numeros[i]*).
      8.1.3. Asignar a la variable *pos_mayor* la posición de *i* (*pos_mayor* = *i*).
   8.2. De lo contrario, (*mayor* < *numeros[i]*), entonces:
      8.2.1. Asignar a la variable *mayor* el valor del elemento *i* del arreglo *números* (*mayor* = *numeros[i]*).
      8.2.2. Asignar a la variable *pos_mayor* la posición de *i* (*pos_mayor* = *i*).
   8.3. Fin del *Si*.
9. Fin del *Para*.
10. Asignar a la variable *band* valor de 0 (*band* = 0).
11. Para (*i* = 1; *i* < = *n*; *i* = *i* + 1).
   11.1. Si (*band* = = 0), entonces:
      11.1.1. Asignar a la variable *band* valor de 1 (*band* = 1).
      11.1.2. Asignar a la variable *menor* el valor del elemento *i* del arreglo *números* (*menor* = *numeros[i]*).
      11.1.3. Asignar a la variable *pos_menor* la posición de *i* (*pos_menor* = *i*).
   11.2. De lo contrario, Si (*menor* > *numeros[i]*), entonces:
      11.2.1. Asignar a la variable *menor* el valor del elemento *i* del arreglo *números* (*menor* = *numeros[i]*).
      11.2.2. Asignar a la variable *pos_menor* la posición de *i* (*pos_menor* = *i*).

12. Fin del *Para*.
13. Decir: "De los números analizados, el mayor fue el:" + *mayor* + "y se encuentra en la posición" + *pos_mayor* + "del arreglo.".
14. Decir: "De los números analizados, el menor fue el:" + *menor* + "y se encuentra en la posición" + *pos_menor* + "del arreglo.".
15. Fin.

Fase 2: crear el diagrama de flujo:

Estructuras de control repetitivo y de datos

En la página 43 podrás observar un ejemplo del diagrama de flujo para resolver este problema.

A practicar 1.1.4

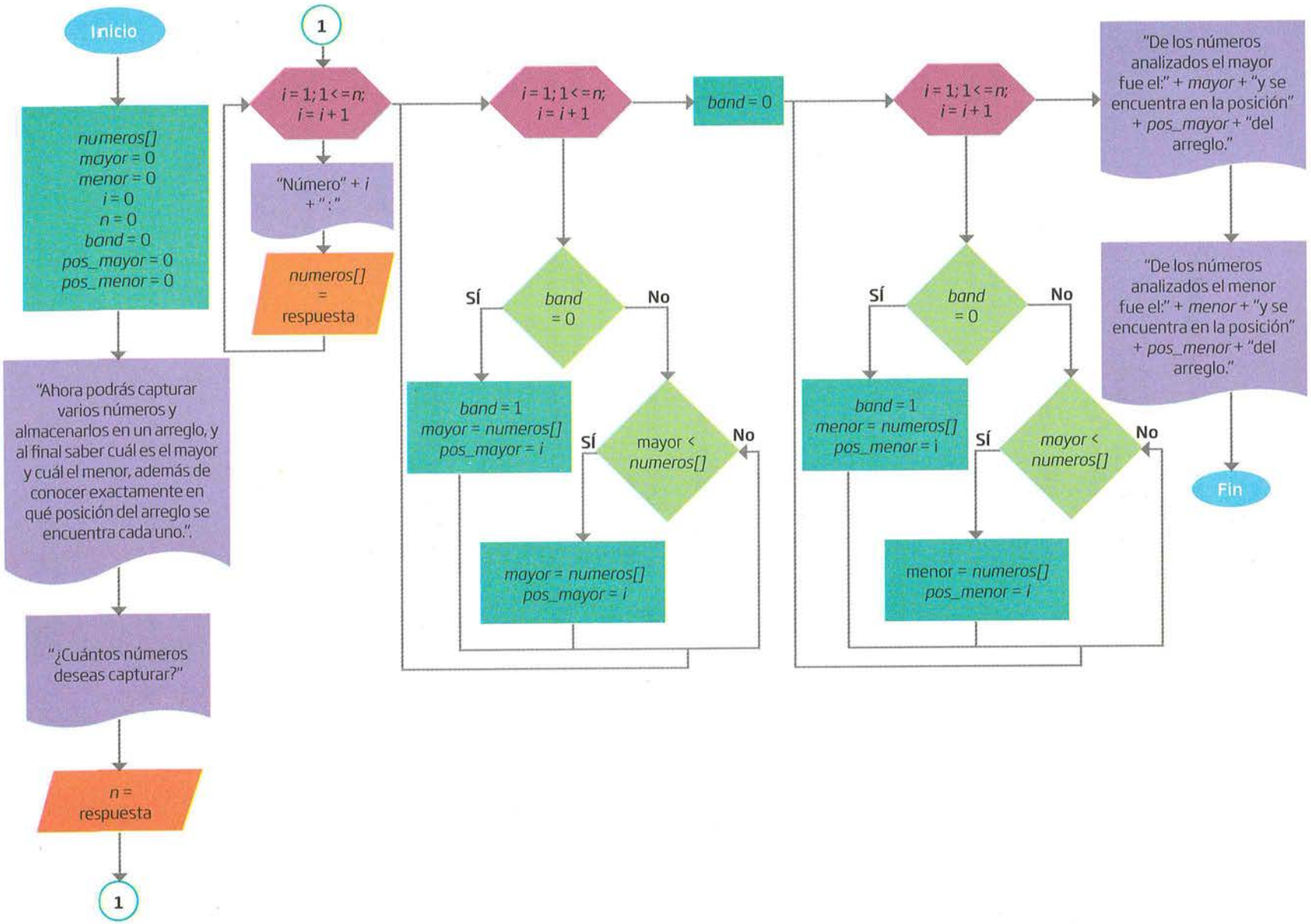
Retoma la evidencia que obtuviste al realizar la actividad de la sección *A Practicar 1.1.5* de la página 27 y, con ayuda de tu profesor, diseña y crea el diagrama de flujo que resuelva el problema mencionado; no olvides estructurarlo con base en las fases para la solución de problemas. Guarda el diagrama de flujo en el mismo documento de Word donde está el algoritmo, recuerda que lo utilizarás de nuevo más adelante.

Encamina tus habilidades

1. De manera individual, crea el algoritmo y el diagrama de flujo que ayude a resolver el siguiente problema: conocer de cierta cantidad de números los siguiente:
   a) Determinar su promedio y mostrarlo.
   b) Indicar cuántos números de los introducidos son mayores que el promedio y mostrar cuáles son.
   c) Indicar cuántos números de los introducidos son menores que el promedio y mostrar cuáles son.
2. Al crear el algoritmo y el diagrama de flujo, sigue las fases de solución de problemas y usa la estructura de control secuencial, selectivo o repetitivo que creas convenientes para que se cumpla el proceso y se obtenga el resultado adecuado.
3. Guarda el algoritmo en un documento de Word y envíalo a tu profesor para su evaluación; pídele que realice una retrolimentación de tu trabajo y aplícalas en cuanto las conozcas.

Insignia 1.2.2

Crea un algoritmo y diagrama de flujo para resolver el siguiente problema: el Servicio Meteorológico Nacional necesita conocer, después de registrar los promedios mensuales de lluvias durante el año pasado, las caídas en las regiones Norte y Sur; pero debe considerar a) el promedio anual de lluvia de cada una de las regiones, b) el mes y el registro con menor lluvia en la región del Norte y c) la región, el mes y el registro con mayor cantidad de lluvia en todo el país.


Inicio
numeros[]
mayor = 0
menor = 0
i = 0
n = 0
band = 0
pos_mayor = 0
pos_menor = 0
"Ahora podrás capturar varios números y almacenarlos en un arreglo, y al final saber cuál es el mayor y cuál el menor, además de conocer exactamente en qué posición del arreglo se encuentra cada uno.".
"¿Cuántos números deseas capturar?"
n = respuesta
1
1
i = 1; i <= n; i = i + 1
"Número" + i + ":"
numeros[] = respuesta
i = 1; i <= n; i = i + 1
band = 0
SÍ
band = 0
No
band = 1
mayor = numeros[]
pos_mayor = i
SÍ
mayor < numeros[]
No
mayor = numeros[]
pos_mayor = i
i = 1; i <= n; i = i + 1
"De los números analizados el mayor fue el:" + mayor + "y se encuentra en la posición" + pos_mayor + "del arreglo."
SÍ
band = 0
No
band = 1
menor = numeros[]
pos_menor = i
SÍ
mayor < numeros[]
No
menor = numeros[]
pos_menor = i
"De los números analizados el menor fue el:" + menor + "y se encuentra en la posición" + pos_menor + "del arreglo."
Fin

Subproductos

Subproducto procedimental

1. Reúnete con otro compañero y, con base en lo que aprendieron en esta secuencia didáctica, diseñen y elaboren un algoritmo que permita resolver el siguiente problema:

El Instituto Electoral Nacional (INE), durante el proceso electoral de 2012, en el que se eligió un nuevo presidente de México, participaron seis candidatos, de los cuales, y para efectos de este problema, cada uno se identificará con los números del 1 al 6; resolver los siguientes puntos:

a) La cantidad de votos que obtuvo cada uno de los candidatos.
b) ¿Cuál fue el candidato ganador, cuántos votos obtuvo y cuál fue el porcentaje con relación al total de la votación?

Los votos se registrarán de la siguiente manera:

1 5 2 1 6 3 4 4 0

Nota: el cero determina el final de los registros.

2. Retomen las evidencias que obtuvieron al realizar las actividades de las secciones *Encamina tus habilidades* para diseñar las estructuras de control secuencial, selectivo y repetitivo, así como las de datos que crean necesarias para resolver el problema planteado.
3. Compartan su algoritmo y diagrama de flujo con el profesor para que los evalúe y reciban retroalimentación.
4. Apliquen las modificaciones necesarias y guarden el algoritmo y el diagrama de flujo en un documento de Word.

Subproducto declarativo

› **Contesta lo siguiente con base en las indicaciones de tu profesor.**

1. ¿Qué ventajas tiene diagramar los algoritmos que diseñas?

2. ¿Qué otras aplicaciones puedes utilizar para crear los diagramas de flujo que diseñas para resolver los problemas de índole académico? Describe sus funciones y justifica tu elección.

3. ¿Cuáles son las reglas que debe cumplir un diagrama de flujo?

Subproducto actitudinal-valoral

- Cuando creas diagramas de flujo para representar algoritmos, ¿es más fácil o difícil entender hacia dónde se mueve la información en ellos?

- ¿Crees que se debe promover el uso de los diagramas de flujo como organizadores gráficos en otras asignaturas? Justifica tu respuesta.

1.3. Pseudocódigo en Scratch

A desarrollar

Al terminar esta secuencia didáctica utilizarás el software llamado Scratch para operar pseudocódigos, así como diseñar y crear algoritmos, incluso hasta la fase de comprobación.

El **subproducto procedimental** de esta secuencia didáctica consistirá en elaborar el algoritmo, el diagrama de flujo y el pseudocódigo en el programa Scratch para resolver un problema específico.

Las actividades que realizarás en esta secuencia didáctica te servirán para:

4.2. Aplicar diversas estrategias comunicativas según quienes sean los interlocutores, el contexto en el que te encuentras y los objetivos que persigues.
4.5. Manejar las tecnologías de la información y la comunicación para obtener información y expresar ideas, de manera responsable y respetuosa.
5.1. Seguir instrucciones y procedimientos de manera reflexiva en la búsqueda y adquisición de nuevos conocimientos.
5.2. Ordenar información de acuerdo con categorías, jerarquías y relaciones.
7.3. Articular saberes de diversos campos y establecer relaciones entre ellos y tu vida cotidiana.
8.1. Plantear problemas y ofrecer alternativas de solución al desarrollar proyectos en equipos de trabajo, y definir un curso de acción con pasos específicos.
8.2. Aportar puntos de vista con apertura y considerar los de otras personas de manera reflexiva.
8.3. Asumir una actitud constructiva al intervenir en equipos de trabajo, congruente con los conocimientos y las habilidades que posees.

De aquí en adelante

En la primera secuencia didáctica de este curso aprendiste que la tercera fase de la resolución de problemas implica comprobar el algoritmo creado; esta etapa podría resultar una tarea muy compleja o muy sencilla, según lo establezca el problema y el algoritmo creado para resolverlo.

› **Con base en tus conocimientos previos, pide a tu profesor que organice un debate con todo el grupo acerca de los programas o aplicaciones que conozcan tú y tus compañeros para comprobar algoritmos. Toma nota de las participaciones en tu cuaderno y escribe a continuación una conclusión:**

¿Qué es un pseudocódigo?

El proceso de desarrollo de programas informáticos contempla una fase que resulta muy importante, que es la creación del *pseudocódigo*, nombre con el que se conoce a la forma de escribir los pasos que va a ejecutar un programa, la cual debe cumplir una estructura similar a la del lenguaje de programación.

El propósito principal de diseñar y crear un pseudocódigo es darle la formalidad necesaria a los algoritmos y diagramas de flujo, para después codificarlos en el lenguaje de programación que se requieren para su ejecución. Para lograrlo, se deben seguir ciertas reglas, las cuales conocerás a continuación.

Shutterstock/ronstik

¿Qué será más difícil de escribir para un programador, un algoritmo o un pseudocódigo?

Reglas para la creación de un pseudocódigo

Comúnmente, los pseudocódigos se escriben a mano o en algún procesador de texto, ya que realmente terminan siendo casi igual a los algoritmos, solo que se les agregan algunos componentes que determinan su función, como pueden ser los siguientes:

a) Después de la palabra *Inicio* se debe colocar el carácter {.
b) Antes de la palabra *Fin* se escribe el símbolo }.
c) Se debe agregar el símbolo ; al terminar cada sentencia o proceso a realizar.
d) Se coloca el símbolo { al iniciar las estructuras de control y el símbolo } al finalizar, esta última sustituye la leyenda usada en el algoritmo que define el fin de la estructura.

Con base en estas diferencias, analiza el siguiente comparativo de escritura entre un algoritmo y un pseudocódigo:

Comparación de escrituras entre...	
algoritmo	**pseudocódigo**
Si (condición(es)) entonces	Si (condición(es)){
... Proceso(s)	... Proceso(s);
Fin del *Si*	}

Recursos en línea

Abre el **enlace 1.4** y conocerás más ejemplos acerca de cómo se escriben los pseudocódigos a partir de algoritmos.

En ocasiones, este procedimiento resulta complejo, sobre todo cuando se hace "a pie" como comúnmente se dice cuando se escribe a mano y paso por paso.

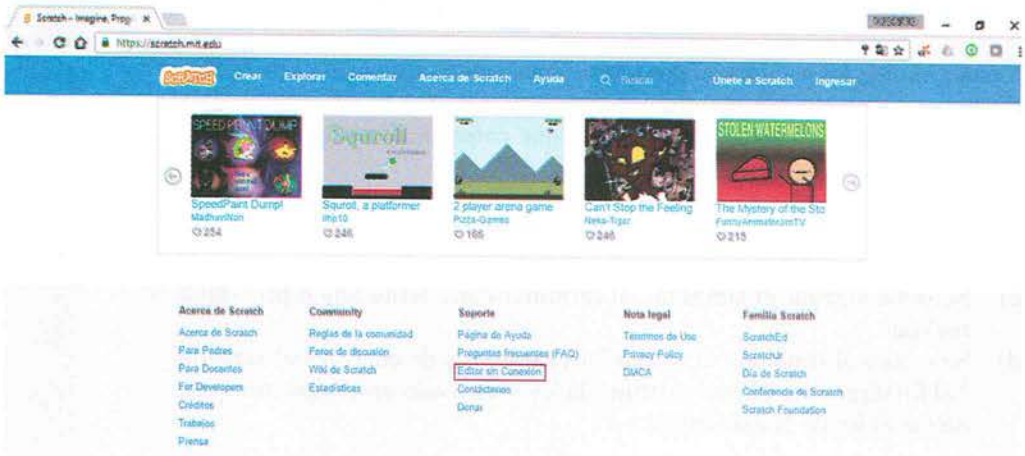
Por lo anterior, desde hace años se han diseñado diversas aplicaciones que permiten que este proceso sea más fluido y amigable para el usuario; una de las aplicaciones más usadas en el mundo se llama **Scratch**, y es la que aprenderás a usar en esta secuencia didáctica para elaborar pseudocódigos.

¿Qué es Scratch?

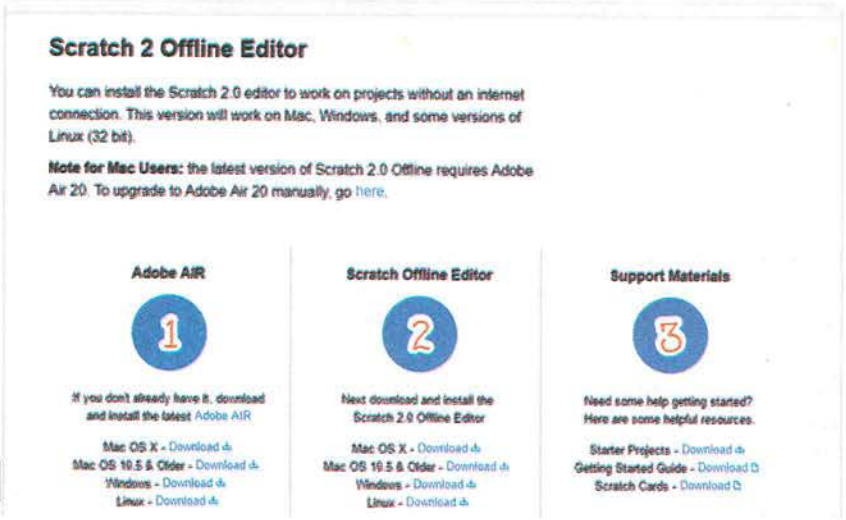
Scratch es una aplicación informática que te permitirá explorar y experimentar de forma gráfica los conceptos de la programación que has aprendido hasta ahora. La interfaz de este software es muy sencilla y amigable en su uso para el usuario, ya que permite comprobar incluso si los algoritmos diseñados realmente funcionan o si requieren alguna mejora, para después llevarlos a codificar a cualquier programa.

Descarga e instalación de Scratch

La aplicación Scratch se puede descargar de manera segura desde la siguiente URL: https://scratch.mit.edu/ que se ubica dentro del portal de Scratch, solo hay que desplazarse a la parte inferior de la página inicial y buscar la categoría *Soporte*, una vez en ese sitio se debe dar clic sobre el enlace llamado *Editor sin conexión*, como se muestra en la siguiente imagen:



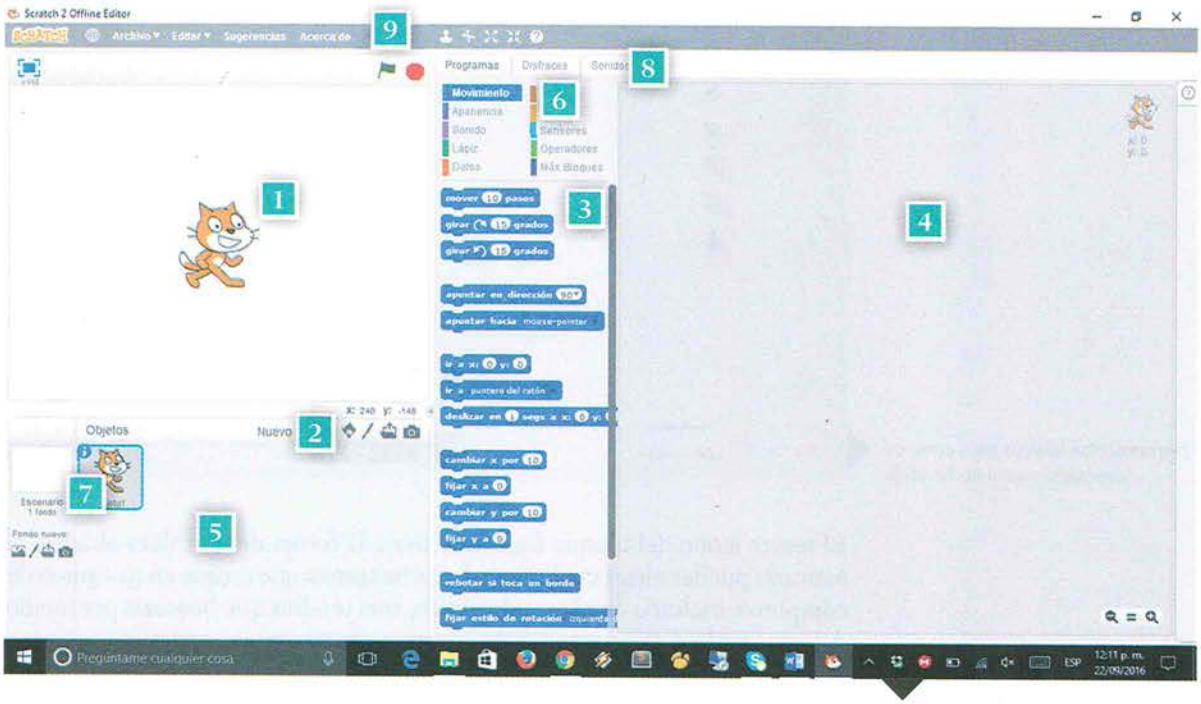
Después de haber seleccionado la opción *Editor sin conexión*, sigue los pasos que aparecerán de manera secuencial en las siguientes pantallas y, al final, se descargará el programa en tu computadora o dispositivo para que puedas iniciar el uso de Scratch, ya sea en línea o desconectado a la red.



Opciones de descarga de la aplicación Scratch.

Interfaz gráfica de Scratch

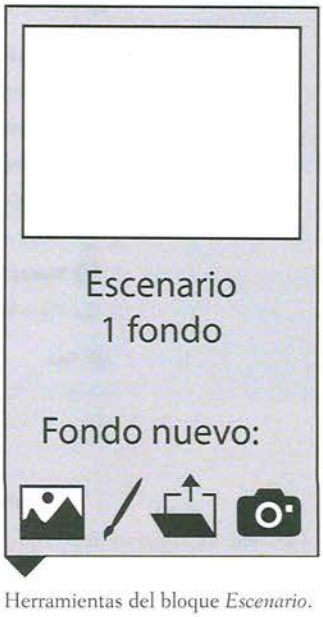
Los elementos que se enlistan a continuación son los componentes principales del entorno gráfico de Scratch, ubica cada uno de ellos en la imagen de abajo con base en la relación de su número de lista con el que indica la imagen.



Entorno gráfico de la aplicación Scratch.

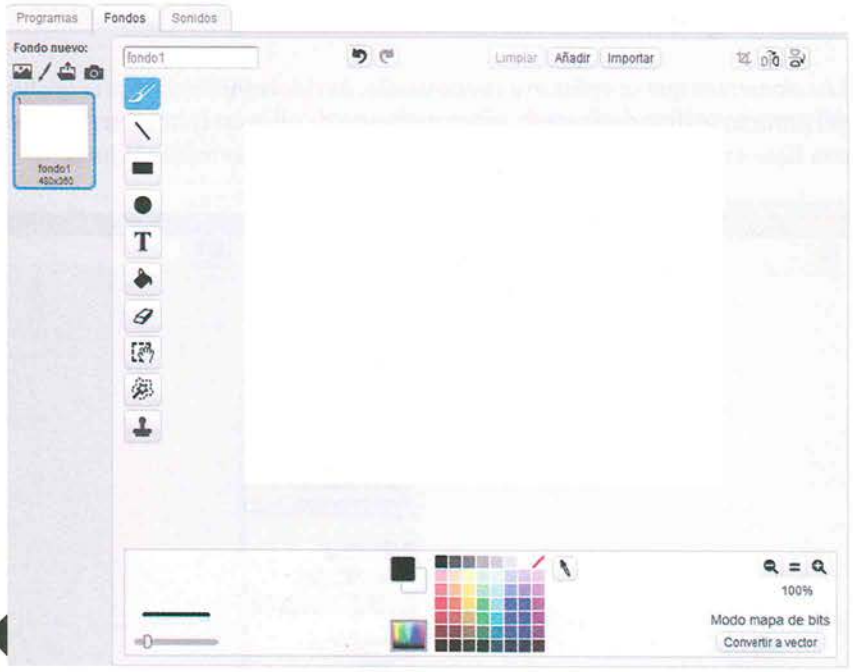
1. Escenario.
2. Objetos (*Sprites*).
3. Paleta de bloques.
4. Área de programas.
5. Lista de objetos.
6. Disfraces.
7. Información del objeto actual.
8. Pestaña *Sonidos*.
9. Barra de herramientas y menú.

El **Escenario** de Scratch es donde se ejecuta el pseudocódigo armado con ayuda de los bloques que puedes usar en Scratch. Siempre aparecerá con fondo blanco, aunque puede personalizarse usando las herramientas del bloque *Escenario* de la interfaz gráfica de la aplicación, el cual podrás ubicar en la parte inferior izquierda de la pantalla.



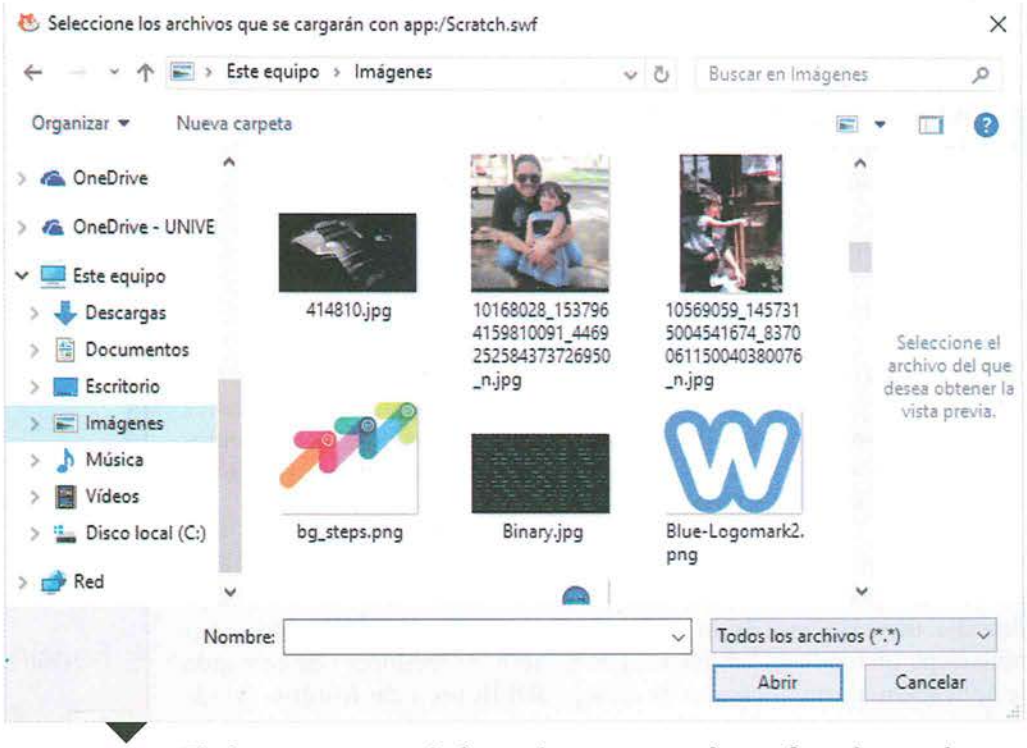
Herramientas del bloque *Escenario*.

Las herramientas principales del bloque *Escenario* son cuatro, y cada una se distingue por un icono diferente. El primer icono, vistos de izquierda a derecha, tiene la forma de un paisaje de montaña con el Sol al fondo, y permite elegir un fondo de la biblioteca de Scratch. Al presionar este comando la aplicación abrirá el cuadro de diálogo **Biblioteca de fondos**, donde podrás observar los fondos existentes para que elijas el que más te agrade. El segundo icono tiene la forma de un pincel, y sirve para dibujar un escenario nuevo; al activarlo, del lado derecho de la interfaz se mostrarán las herramientas para crear un escenario nuevo.



Herramientas básicas para crear un escenario nuevo en Scratch.

El tercer icono del bloque *Escenario* tiene la forma de un folder abierto, al activarlo puedes elegir cualquiera de las imágenes que tengas en tu equipo de cómputo e incluirla en el pseudocódigo, solo tendrás que buscarla por medio del cuadro de diálogo emergente, como se ilustra en la siguiente imagen:

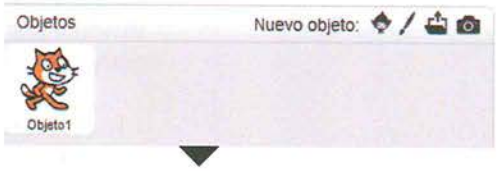


Opciones de imágenes disponibles en un equipo y que pueden incluirse en el pseudocódigo.

El último icono tiene la forma de una cámara fotográfica, al activarlo puedes usar la cámara web de la computadora para capturar una fotografía e incluirla en el pseudocódigo que estés creando.

El segundo elemento de Scratch que conocerás se llama **Objetos** (*Sprites*, en inglés), y este espacio te ayudará a agregar tantos objetos o avatares necesites, según el pseudocódigo que estés creando. Del mismo modo que el comando *Escenario*, el personaje creado por la aplicación (que es un gato) se puede perosnalizar; incluso, los personajes que puedas crear se podrán agregar como auxiliares de la herramienta.

Otra de las similitudes entre los comandos *Escenario* y *Objetos* es que este último también cuenta con cuatro iconos para personalizar los avatares o personajes que vayas creando. La imagen del lado derecho contiene estos cuatro iconos, que son denominados, de izquierda a derecha, como *Nuevo objeto*, *Dibujar objeto*, *Archivo de objetos* y *Captura de objeto*.



Todas las herramientas se activan al seleccionar cualquier elemento del mapa de sitio.

La función de *Nuevo objeto*, cuyo icono es una caricatura, permite elegir un nuevo personaje desde la biblioteca de objetos de Scratch, lo cual se logra por medio del cuadro de diálogo **Biblioteca de objetos**; solo deberás elegir el que más te guste para darle tu toque personal al pseudocódigo. La siguiente imagen muestra algunos ejemplos de lo que podrás en contrar en esa *Biblioteca de objetos*.

¿Sabías que...?

Para cambiar el personaje inicial de Scratch debes dar clic con el botón secundario del ratón sobre el personaje y elegir la opción ***Borrar*** del menú contextual. Aplica esta dinámica cada vez que necesites eliminar cualquier elemento del programa.

Después de elegir el personaje u objeto con el que te ayudarás para la ejecución del pseudocódigo, puedes editarlo y cambiar su apariencia activando la pestaña *Disfraces* desde el lado derecho de la interfaz.

El segundo icono de herramientas del *Objeto*, al igual que el de la función *Escenarios*, tiene la forma de un pincel y sirve para dibujar un objeto o personaje nuevo auxiliándote de las herramientas de dibujo de la pestaña *Disfraces*.

El tercer icono del bloque *Objetos* también tiene la forma de un folder abierto, con este comando podrás elegir alguna de las imágenes que tengas almacenada en tu equipo de cómputo, agregándola como objeto o personaje del pseudocódigo que estás creando.

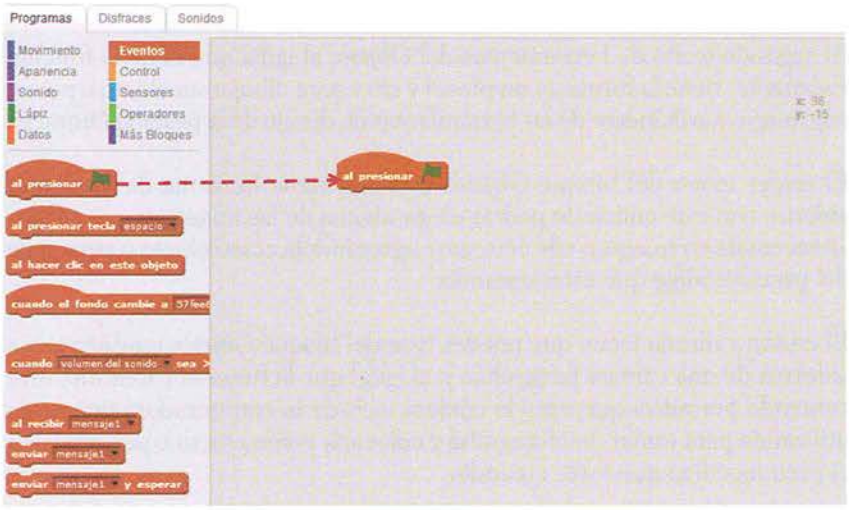
El cuarto y último icono que puedes usar del bloque *Objetos* también tiene la forma de una cámara fotográfica y, al igual que la función *Escenario*, este comando permitirá que uses la cámara web de la computadora que estés utilizando para tomar una fotografía y colocarla como objeto o personaje en el pseudocódigo que estés creando.

El tercer elemento a estudiar del entorno gráfico de Scratch, con base en lo que se definió en la imagen de la página 49, se denomina **Paleta de bloques**; para conocer más acerca de sus funciones es importante que se defina primero qué son los bloques en este programa.

Los **bloques** son los segmentos de Scratch que contienen las estructuras que podrás utilizar en el diseño y creación de pseudocódigos. Cada bloque está dividido por categorías, según la función que se le asigne. Las categorías se activan con la pestaña *Programas*, y están divididas de la siguiente manera:

1. **Movimiento:** opciones para darle movimiento a los objetos o avatares.
2. **Apariencia:** es uno de los bloques más importantes porque en él encontrarás las estructuras para escribir las frases que se mostrarán en el escenario.
3. **Sonido:** a lo largo de los bloques de esta categoría podrás insertar algún audio a los avatares u objetos.
4. **Lápiz:** hace cambios referentes al lápiz del escenario.
5. **Datos:** esta categoría es de vital importancia porque permite crear las variables necesarias en los pseudocódigos, de acuerdo con los algoritmos y diagramas de flujo diseñados previamente.
6. **Eventos:** los bloques de esta categoría sirven para indicar cuándo empezarán a ejecutarse ciertas instrucciones y otros eventos.
7. **Control:** en estos bloques se encuentran las estructuras de control selectivo y repetitivo para aplicarse en los pseudocódigos; además, este bloque indicará cuándo debe detenerse la ejecución del pseudocódigo.
8. **Sensores:** por medio de los bloques de esta categoría podrás realizar procesos como pedir algún dato, leerlo y asignarlo a una variable.
9. **Operadores:** contiene los bloques para realizar distintas operaciones de carácter lógico o matemático.
10. **Más bloques:** permite crear bloques con instrucciones personalizadas por el usuario. Esta categoría es utilizada con frecuencia en la Programación Orientada a Objetos (POO).

Al activar alguna categoría de la *Paleta de bloques* podrás observar todos los bloques relacionados con dicha categoría en la división inferior del contenedor, solo deberás seleccionar el bloque que deseas usar y arrastrarlo hasta el área llamada *Área de programas*, tal como se observa en la siguiente imagen:



El cuarto elemento por estudiar del programa Scratch se llama **Área de programas**, y es el comando que sirve para colocar los bloques de las estructuras que se necesiten al diseñar y crear un pseudocódigo.

El uso de los cinco elementos que restan, según lo que se definió en la página 49 acerca del entorno gráfico de Scratch, es tan amigable con el usuario que podrás identificar su potencialidad de manera autodidáctica, solo considera lo siguiente acerca de cada uno de estos elementos.

Lista de objetos. En esta área se muestran imágenes en miniatura que representan todos los objetos disponibles en el proyecto actual.

Pestaña Disfraces. Esta pestaña activa el área en la que puedes editar la apariencia de los avatares u objetos disponibles en el proyecto.

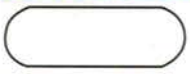
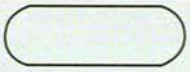
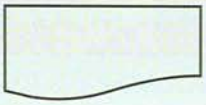
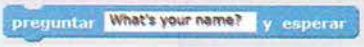
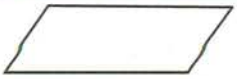
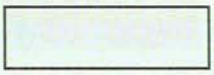
Información del objeto actual. Contiene el nombre del objeto, así como su posición, dirección, estado de giro, estado de su lápiz y condición de bloqueo.

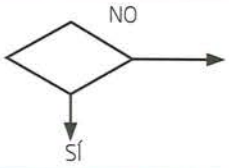
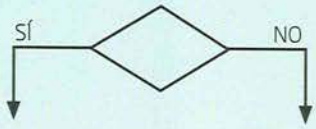
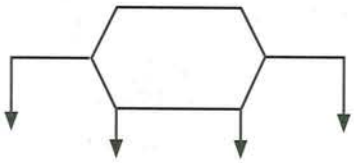
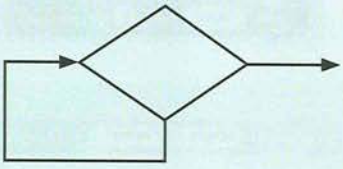
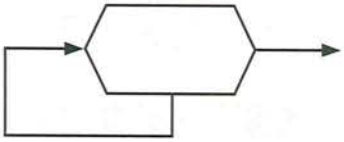
Pestaña Sonidos. Con esta pestaña podrás incluir sonidos a los pseudocódigos a partir de tres opciones: seleccionar sonidos de la librería, grabar nuevos sonidos o importar archivos de audio.

Barra de herramientas y menú. En esta barra encontrarás comandos para crear nuevos proyectos, guardar el proyecto actual, cambiar el idioma de Scratch, borrar y duplicar, entre otras funciones.

Bloques en Scratch

El uso de los bloques en Scratch requiere de experiencia en el manejo de la aplicación, pero con el siguiente cuadro podrás aprender sus funciones de manera ágil, pues muestra los bloques que debes usar para expresar correctamente los algoritmos y símbolos del diagrama de flujo que conoces.

Símbolo	Significado	Bloque
	Inicio del diagrama	
	Fin del diagrama	
	Escribir o expresar	
	Preguntar	
	Asignar respuesta a...	
	Procesos y operaciones	

Símbolo	Significado	Instrucción algorítmica
"mensaje" + varible + "mensaje"	Concatenar o unir (+) mensajes	
+ - * /	Operadores aritméticos	
Y O NO	Operadores lógicos	
MOD	Residuo	
	Si (condición(es)) entonces Fin del Si	
	Si (condición(es)) De lo Contrario Fin del Si Si (condición(es)) De lo contrario Si (condición(es)) De lo contrario Fin del Si	
	En caso de (opción) Fin de en caso de	
	Mientras (condición(es)) Fin del mientras	
	Para (tarea_de_inicialización; condición; tarea_final_por_vuelta) Fin del Para	

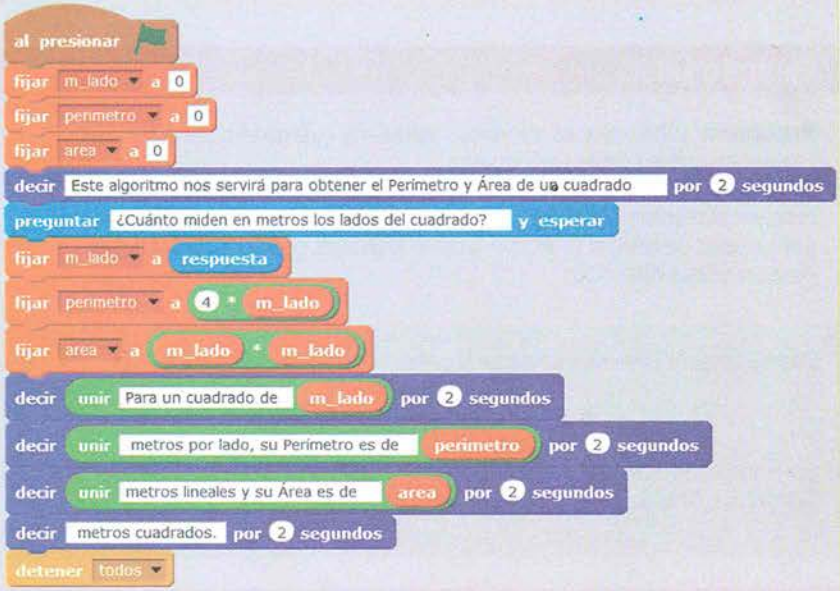
Ejemplo 1.3.1

Problema: obtener el perímetro y el área de un cuadrado del cual no se conocen sus dimensiones.
Solución: retomar el algoritmo que elaboraste al resolver problema del *Ejemplo 1.1.1*, así como el diagrama de flujo del *Ejemplo 1.2.1*, en los cuales se usaron estructuras de control secuencial, y observa cómo se crea el pseudocódigo en Scratch.

Fase 2: crear el pseudocódigo en Scratch:

Estructura de control secuencial

La siguiente imagen muestra los bloques que se utilizan en Scratch para resolver este problema:



A practicar 1.3.1

Retoma las evidencias que obtuviste al realizar las actividades de la sección *A Practicar 1.1.2* y *1.2.1* de las páginas 19 y 36, respectivamente, y con ayuda de tu profesor crea el pseudocódigo en Scratch con estructuras de control secuencial que resuelva el problema mencionado.

Guarda el pseudocódigo en un directorio junto con el documento de Word donde está el algoritmo y el diagrama de flujo, recuerda que lo utilizarás de nuevo más adelante.

Insignia

Crea el algoritmo, el diagrama de flujo y el pseudocódigo en Scratch, usando estructuras de control secuencial, que permita resolver el siguiente problema: después de saber el precio de un artículo comprado por un cliente, determina la cantidad que pagó el cliente y el cambio que recibió. Coloca el archivo de Word y el archivo de Scratch en un directorio comprimido y envíalo a tu profesor para su evaluación, no debes dejar de lado las fases de la solución de problemas.

¿Sabías que...?

En Scratch se puede usar más de un avatar. Por ejemplo, en caso de que el algoritmo creado deba mostrar las opciones del menú, en el momento de elaborar el pseudocódigo en Scratch se deben usar dos avatares para que el secundario sirva de auxiliar. Cada avatar debe armarse bajo una secuencia de acciones por separado.

Encamina tus habilidades

1. De manera individual, crea el algoritmo, el diagrama de flujo y el pseudocódigo en Scratch que ayude a resolver el siguiente problema: conocer el nombre de un oso polar que vive en un zoológico, su peso en libras y su longitud en pies, convertir su peso a kilogramos y su longitud a metros; mostrar como resultado el nombre del oso polar, su peso en kilogramos y su longitud en metros.
2. Al crear el algoritmo y el diagrama de flujo, sigue las fases de solución de problemas y usa las estructuras de control que creas necesarias para que se cumpla el proceso y se obtenga el resultado adecuado.
3. Coloca el archivo de Word que incluye el algoritmo y el diagrama de flujo, junto con el archivo de Scratch, que incluye el pseudocódigo, en un directorio comprimido y envíalo a tu profesor para su evaluación.

Ejemplo 1.3.2

Problema: saber, de tres números cualquiera, cuál de ellos es el mayor, cosiderando que pueden ser iguales.
Solución: retomar el algoritmo y el diagrama de flujo que elaboraste al resolver el problema del *Ejemplo 1.2.2*, de la página 37, en el cual usaste estructuras de control selectivo, y crear el pseudocódigo en Scratch con base en esas evidencias.

Fase 2: crear el pseudocódigo en Scratch:

Estructura de control selectivo

En la página 57 podrás observar un ejemplo del pseudocódigo creado en Scratch para resolver este problema.

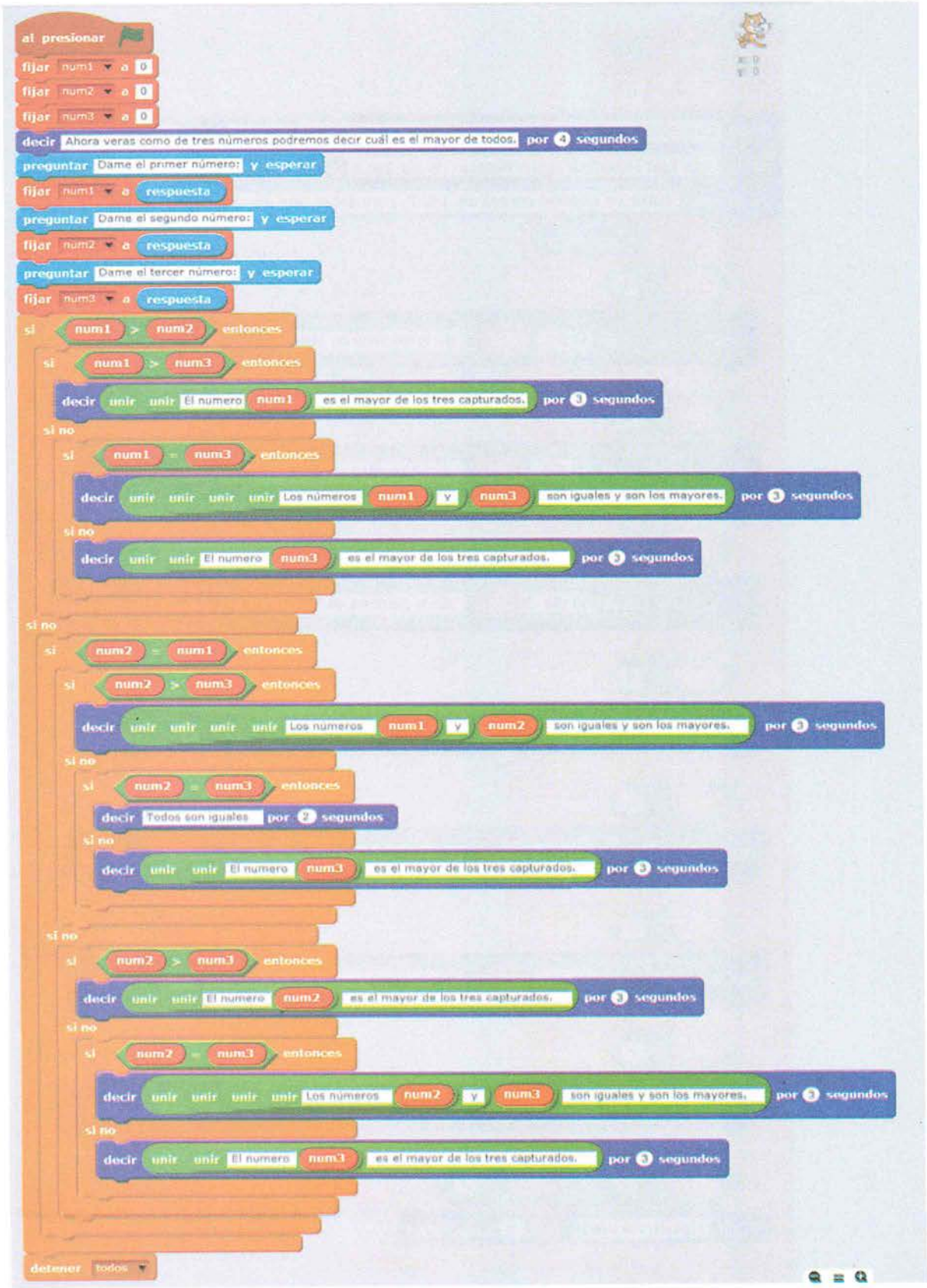
Ejemplo 1.3.3

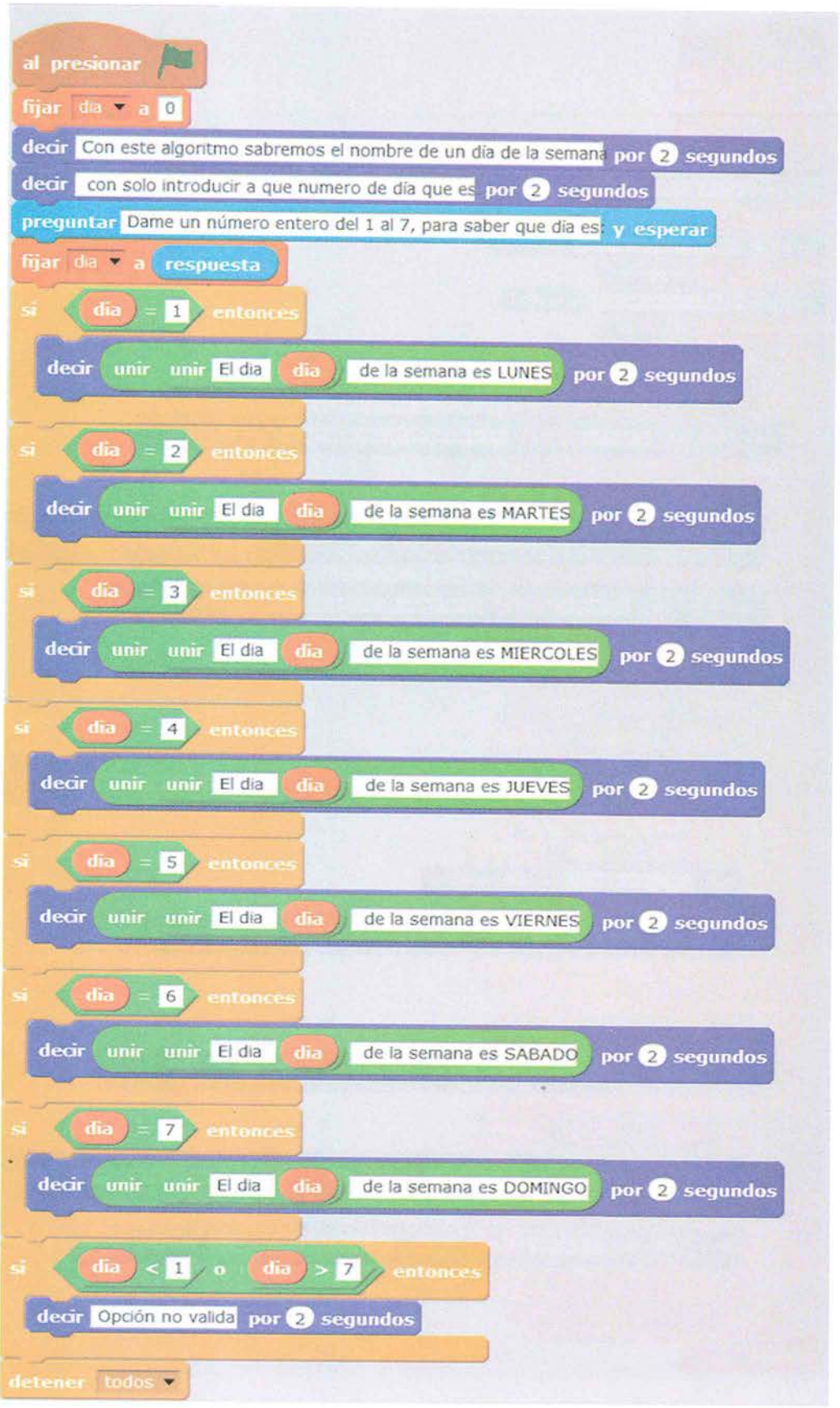
Problema: conocer el nombre del día de la semana con solo introducir el número de día que es.
Solución: retomar el algoritmo que elaboraste al resolver problema del *Ejemplo 1.1.4*, de la página 23, así como el diagrama de flujo del *Ejemplo 1.2.3*, de la página 38, en los cuales se usaron estructuras de control selectivo, y observa cómo se crea el pseudocódigo en Scratch.

Fase 2: crear el pseudocódigo en Scratch:

Estructura de control selectivo

En la página 58 podrás observar un ejemplo del pseudocódigo creado en Scratch para resolver este problema.


al presionar
fijar num1 a 0
fijar num2 a 0
fijar num3 a 0
decir Ahora veras como de tres números podremos decir cuál es el mayor de todos. por 4 segundos
preguntar Dame el primer número: y esperar
fijar num1 a respuesta
preguntar Dame el segundo número: y esperar
fijar num2 a respuesta
preguntar Dame el tercer número: y esperar
fijar num3 a respuesta
si num1 > num2 entonces
si num1 > num3 entonces
decir unir unir El numero num1 es el mayor de los tres capturados. por 3 segundos
si no
si num1 = num3 entonces
decir unir unir unir unir Los numeros num1 y num3 son iguales y son los mayores. por 3 segundos
si no
decir unir unir El numero num3 es el mayor de los tres capturados. por 3 segundos
si no
si num2 = num1 entonces
si num2 > num3 entonces
decir unir unir unir unir Los numeros num1 y num2 son iguales y son los mayores. por 3 segundos
si no
si num2 = num3 entonces
decir Todos son iguales por 2 segundos
si no
decir unir unir El numero num3 es el mayor de los tres capturados. por 3 segundos
si no
si num2 > num3 entonces
decir unir unir El numero num2 es el mayor de los tres capturados. por 3 segundos
si no
si num2 = num3 entonces
decir unir unir unir unir Los numeros num2 y num3 son iguales y son los mayores. por 3 segundos
si no
decir unir unir El numero num3 es el mayor de los tres capturados. por 3 segundos
detener todos


al presionar
fijar dia a 0
decir Con este algoritmo sabremos el nombre de un dia de la semana por 2 segundos
decir con solo introducir a que numero de dia que es por 2 segundos
preguntar Dame un número entero del 1 al 7, para saber que dia es: y esperar
fijar dia a respuesta
si dia = 1 entonces
decir unir unir El dia dia de la semana es LUNES por 2 segundos
si dia = 2 entonces
decir unir unir El dia dia de la semana es MARTES por 2 segundos
si dia = 3 entonces
decir unir unir El dia dia de la semana es MIERCOLES por 2 segundos
si dia = 4 entonces
decir unir unir El dia dia de la semana es JUEVES por 2 segundos
si dia = 5 entonces
decir unir unir El dia dia de la semana es VIERNES por 2 segundos
si dia = 6 entonces
decir unir unir El dia dia de la semana es SABADO por 2 segundos
si dia = 7 entonces
decir unir unir El dia dia de la semana es DOMINGO por 2 segundos
si dia < 1 o dia > 7 entonces
decir Opción no valida por 2 segundos
detener todos

A practicar 1.3.2

Retoma las evidencias que obtuviste al realizar las actividades de la sección *A Practicar 1.1.3* y *1.2.2* de las páginas 21 y 38, respectivamente, y con ayuda de tu profesor crea el pseudocódigo en Scratch con las estructuras de control que resuelvan el problema mencionado. Guarda el pseudocódigo en el mismo directorio de las evidencias.

Ejemplo 1.3.4

Problema: después de recibir como datos cierta cantidad de números enteros, determinar cuál es el mayor y cuál es el menor de todos, pero mostrando la posición en el arreglo donde se encuentra cada uno de ellos y su valor.
Solución: retomar el algoritmo y el diagrama de flujo que elaboraste al resolver el problema del *Ejemplo 1.2.4*, de las páginas 41 y 42, en el cual usaste estructuras de control secuencial, selectivo y repetitivo, así como estructuras de datos, y crear el pseudocódigo en Scratch con base en esas evidencias.

Fase 2: crear el pseudocódigo en Scratch:

Estructuras de control secuencial, selectivo, repetitivo y de datos

En la página 60 podrás observar un ejemplo del pseudocódigo creado en Scratch para resolver este problema.

A practicar 1.3.3

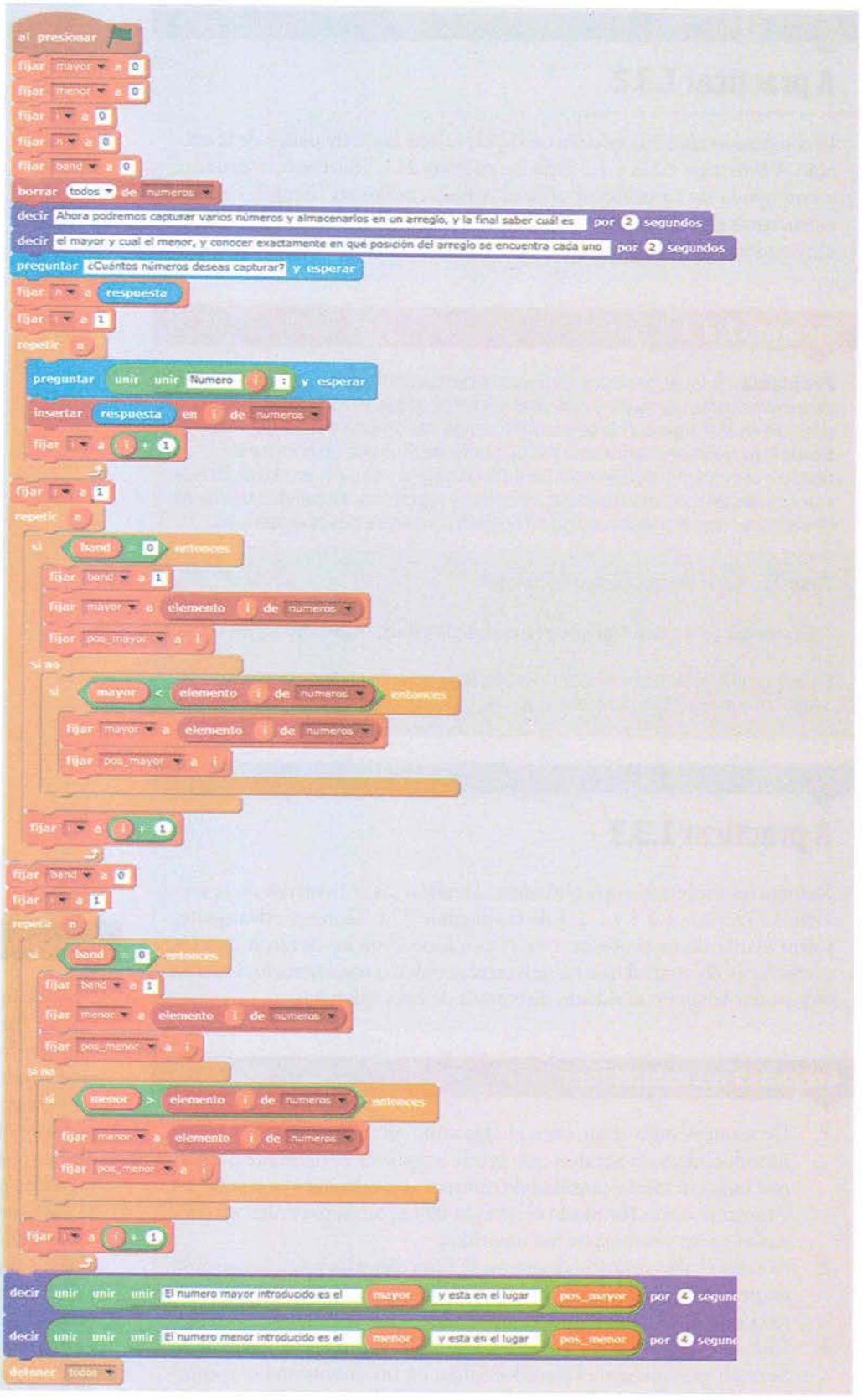
Retoma las evidencias que obtuviste al realizar las actividades de la sección *A Practicar 1.1.5* y *1.2.3* de las páginas 27 y 42, respectivamente, y con ayuda de tu profesor crea el pseudocódigo en Scratch con las estructuras de control que resuelvan el problema mencionado. Guarda el pseudocódigo en el mismo directorio de las evidencias.

Encamina tus habilidades

1. De manera individual, crea el algoritmo, el diagrama de flujo y el pseudocódigo en Scratch que ayude a resolver el siguiente problema: conocer cierta cantidad de números enteros (tal vez repetidos) y mostrar como resultado el arreglo de los números enteros ordenados, pero sin mostrar los repetidos.
2. Al crear el algoritmo y el diagrama de flujo, sigue las fases de solución de problemas y usa las estructuras de control que creas necesarias para que se cumpla el proceso y se obtenga el resultado adecuado.
3. Coloca el algoritmo y el diagrama de flujo, junto con el archivo de Scratch que incluye el pseudocódigo, en un directorio comprimido y envíalo a tu profesor para su evaluación.

Insignia

Crea el algoritmo, diagrama de flujo y pseudocódigo en Scratch, usando las estructuras de control que creas necesarias para resolver el siguiente problema: conocer cuántos y cuáles números del 1 al 100 son pares y cuántos y cuáles son impares. Coloca el archivo de Word y el archivo de Scratch en un directorio comprimido y envíalo a tu profesor para su evaluación, no debes dejar de lado las fases de la solución de problemas.


al presionar
fijar mayor a 0
fijar menor a 0
fijar i a 0
fijar n a 0
fijar band a 0
borrar todos de numeros
decir Ahora podremos capturar varios números y almacenarlos en un arreglo, y la final saber cuál es por 2 segundos
decir el mayor y cual el menor, y conocer exactamente en qué posición del arreglo se encuentra cada uno por 2 segundos
preguntar ¿Cuántos números deseas capturar? y esperar
fijar n a respuesta
fijar i a 1
repetir n
preguntar unir unir Numero i : y esperar
insertar respuesta en i de numeros
fijar i a i + 1
fijar i a 1
repetir n
si band = 0 entonces
fijar band a 1
fijar mayor a elemento i de numeros
fijar pos_mayor a i
si no
si mayor < elemento i de numeros entonces
fijar mayor a elemento i de numeros
fijar pos_mayor a i
fijar i a i + 1
fijar band a 0
fijar i a 1
repetir n
si band = 0 entonces
fijar band a 1
fijar menor a elemento i de numeros
fijar pos_menor a i
si no
si menor > elemento i de numeros entonces
fijar menor a elemento i de numeros
fijar pos_menor a i
fijar i a i + 1
decir unir unir unir El numero mayor introducido es el mayor y esta en el lugar pos_mayor por 4 segund
decir unir unir unir El numero menor introducido es el menor y esta en el lugar pos_menor por 4 segund
detener todos

Subproductos

Subproducto procedimental

1. Reúnete con otro compañero y, con base en lo que aprendieron en esta secuencia didáctica, diseñen y elaboren un algoritmo, diagrama de flujo y pseudocódigo en Scratch que permita resolver el siguiente problema: el usuario de una compañía de telefonía móvil necesita calcular el costo de las llamadas telefónicas internacionales que realizó durante el mes pasado, dicho costo depende de la zona geográfica del país donde se recibe la llamada y del número de minutos en que se habló. Mostrando como resultado cuántas llamadas se hicieron a cada zona, el pago total por zona y cuál es el cargo completo del recibo telefónico. En la siguiente tabla se presenta el costo del minuto por zona; cada zona tiene asignada una clave.

Tabla de zona y precios		
Clave	Zona	Precio
12	América del Norte	$2.20
15	América Central	$2.80
18	América del Sur	$4.75
19	Europa	$3.65
23	Asia	$6.25
25	África	$6.25
29	Oceanía	$5.15

Subproducto declarativo

› **Contesta lo siguiente con base en las indicaciones de tu profesor.**

1. ¿Cómo se llama el contenedor blanco donde puedes ver la ejecución del pseudocódigo?
2. ¿Cuál es el propósito principal de Scratch?
3. ¿En qué categoría encuentras los bloques para definir las variables y los arreglos?
4. ¿En qué categorías puedes encontrar los bloques de las estructuras de control?

Subproducto actitudinal-valoral

› **Con base en lo que has aprendido en esta secuencia didáctica, comenta con tu grupo lo siguiente: ¿consideras que es fácil aprender a programar?, ¿te sientes capaz ahora de elaborar tus algoritmos, diagramas de flujo y pseudocódigo con un lenguaje de programación estructurada? Justifica tus respuestas.**

Producto integrador de la unidad 1

Síntesis

1. Reúnete con los compañeros con quienes trabajarás el producto integrador del curso para esta asignatura.

2. Retomen las evidencias que obtuvieron en la sección *Subproducto procedimental* de cada una de las tres secuencias didácticas para esta primera unidad, que son las siguientes:
   - Algoritmo para resolver un problema específico (página 31)
   - Algoritmo y diagrama de flujo para resolver un problema específico (páginas 44 y 45)
   - Algoritmo, diagrama de flujo y pseudocódigo creado en Scratch para resolver un problema específico (página 61)

3. Diseñen y elaboren un algoritmo, un diagrama de flujo y su respectivo pseudocódigo en Scratch que incorpore todos y cada uno de los ejercicios mencionados en los puntos anteriores, de tal forma que mostrarán un menú para poder elegir qué ejercicio resolver. Por ejemplo:

   Menú:
   - Opción 1: "Números Primos"
   - Opción 2: "Votaciones"
   - Opción 3: "Llamadas"

4. Una vez funcionando todo, tendrán que grabar la ejecución de su Scratch, subirla a su canal de YouTube creado en el semestre anterior y compartir el enlace en un archivo .txt para después colocarlo en la carpeta contenedora del producto integrador.

5. Como punto final deberán comprimir la capeta contenedora de los dos archivos: documento en Word con algoritmo y diagrama de flujo, y archivo Scratch; y enviarla a su profesor para su revisión.

Autoevaluación

Demuestra el nivel de tu conocimiento mediante la siguiente guía de evaluación, utilizando la escala del 1 al 5, siendo 1 el valor más bajo y 5 el más alto.

Mis conocimientos en:	Corresponden a un nivel de:				
	1	2	3	4	5
La importancia de las fases de la solución de problemas en la vida académica y cotidiana.					
La identificación de los datos necesarios para la solución de problemas programables.					
El uso de algoritmos estructurados para darle solución óptima a problemas diversos.					
El uso de organizadores gráficos, como los diagramas de flujo, en la vida académica y cotidiana.					
La importancia del trabajo en equipo para compartir conocimientos y habilidades de manera responsable.					
El manejo de las tecnologías de la información y la comunicación para obtener y expresar información de manera responsable y respetuosa.					

Coevaluación

Realiza la siguiente coevalución con el compañero de equipo con quien realizaste los subproductos procedimentales. Sé honesto y justo con tus respuestas. Justifica cada una de las opciones.

Nombre del compañero	
Participó activamente, medianamente o pobremente en la realización de las actividades	
Aportó buenas ideas, señaló sus puntos de vista y realizó el trabajo que le correspondió	
Fue respetuoso y accedió a las decisiones de la mayoría de los integrantes	
Podría mejorar si...	

Unidad 2
Codificación de pseudocódigo

Shutterstock/Africa Studio

Propósito de la unidad

Tu meta será:

- Utilizar un lenguaje de programación estructurado para convertir pseudocódigos en pequeñas aplicaciones y resolver problemas.

Contenidos temáticos

¿Qué aprenderás?

2.1. Introducción al lenguaje de programación C++.
2.2. Estructuras en el lenguaje de programación C++.

Competencias genéricas y disciplinares

En general, te servirá para:

4. Escuchar, interpretar y emitir mensajes pertinentes en distintos contextos mediante la utilización de medios, códigos y herramientas apropiados.
5. Desarrollar innovaciones y proponer soluciones a problemas a partir de métodos establecidos.
7. Aprender por iniciativa e interés propio a lo largo de la vida.
8. Participar y colaborar de manera efectiva en equipos diversos.

En particular, te servirá para:

C-12. Utilizar las tecnologías de la información y la comunicación para investigar, resolver problemas, producir materiales y transmitir información.

Saberes específicos

¿Cómo lo aprenderás?

- Identificando las características de un lenguaje de programación.
- Interpretando las reglas sintácticas y semánticas.
- Reconociendo el tipo de datos primitivos y operadores de un lenguaje de programación.
- Definiendo, declarando y programando variables.
- Programando sentencias de control de flujo secuencial, selectivo y repetitivo.
- Expresando algoritmos en lenguaje de programación.
- Reflexionando acerca de la importancia del uso de algoritmos para resolver problemas de la vida cotidiana.
- Asumiendo una actitud responsable ante la información que expresas mediante lenguajes de programación.
- Demostrando creatividad en el diseño de aplicaciones.
- Colaborando en equipos diversos.

¿Qué lograrás?

Como **producto integrador de esta unidad** elaborarás, en equipo y de forma colaborativa, la codificación en el lenguaje de programación C++ del producto integrador de la unidad 1.

Mi portafolio de evidencias

Evidencias	Lograda	
	Sí	No
Conclusiones a partir de una lluvia de ideas (*De aquí en adelante*, página 66)		
Tabla de datos completa (*Encamina tus habilidades*, página 69)		
Programa ejecutable en Dev C++ (*Subproducto procedimental 2.1*, página 72)		
Respuestas (*Subproductos declarativo y actitudinal-valoral 2.1*, página 73)		
Características de las estructuras de control (*De aquí en adelante*, página 74)		
Código en C++ (*Encamina tus habilidades*, página 77)		
Código en C++ (*Encamina tus habilidades*, página 83)		
Algoritmo, diagrama de flujo, pseudocódigo y código en lenguaje C++ (*Subproducto procedimental 2.2*, página 84)		
Respuestas (*Subproductos declarativo y actitudinal-valoral 2.2*, página 84)		

¿Qué tanto sé?

Responde con base en tus conocimientos previos.

1. ¿Qué significa para ti la programación en informática?

2. Describe lo que entiendes por "lenguaje de programación".

3. Escribe lo que has leído o escuchado sobre C++.

4. ¿A qué se refiere el término *codificación de pseudocódigo*?

2.1. Introducción al lenguaje de programación C++

A desarrollar

Al terminar esta secuencia didáctica conocerás las características del lenguaje de programación C++ a partir de las reglas sintácticas y semánticas. Utilizarás los datos primitivos y operadores de este lenguaje de programación, así como las variables y constantes que pueden aplicarse. Además, aprenderás a descargar e instalar el editor Dev C++ y conocerás su entorno gráfico.

Como **subproducto procedimental** elaborarás una aplicación en C++ que emule una hoja de presentación de un proyecto académico en el editor Dev C++. Las actividades que realizarás en esta secuencia didáctica te servirán para:

4.2. Aplicar diversas estrategias comunicativas según quienes sean los interlocutores, el contexto en el que te encuentras y los objetivos que persigues.
5.1. Seguir instrucciones y procedimientos de manera reflexiva en la búsqueda y adquisición de nuevos conocimientos.
5.2. Ordenar información de acuerdo con categorías, jerarquías y relaciones.
7.3. Articular saberes de diversos campos y establecer relaciones entre ellos y tu vida cotidiana.
8.1. Plantear problemas y ofrecer alternativas de solución al desarrollar proyectos en equipos de trabajo, y definir un curso de acción con pasos específicos.
8.3. Asumir una actitud constructiva al intervenir en equipos de trabajo, congruente con los conocimientos y las habilidades que posees.

De aquí en adelante

› **Pide a tu profesor que organice una lluvia de ideas en el grupo para escribir la definición de cada uno de los siguientes conceptos de lenguaje.**

- Lenguaje de programación: _____
- Lenguaje tipo máquina: _____
- Lenguaje ensamblador: _____
- Lenguaje de alto nivel: _____

Características de C++

El lenguaje de programación puede definirse como el conjunto de instrucciones capaz de manipular los componentes de una computadora, tanto el hardware como el software.

Los lenguajes de programación se clasifican en dos grandes tipos, según su nivel de abstracción o propósito. Los primeros se subdividen en tres niveles, bajo, medio y alto; por su parte, el lenguaje de programación, según su propósito, se subdivide en los niveles general, específico, de sistema y *script*. A continuación se describe cada categoría.

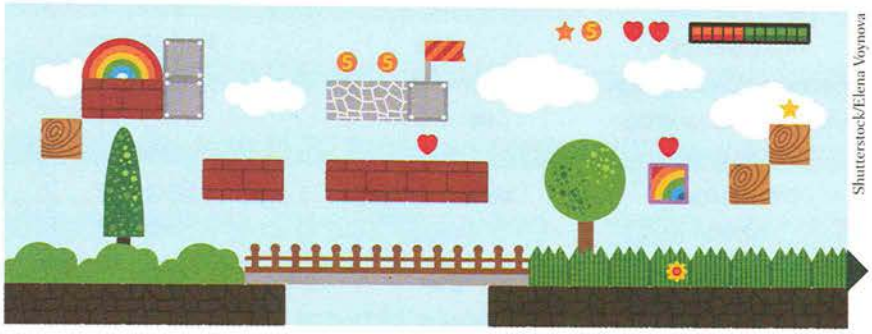
1. **Nivel de abstracción.**
   a) **Nivel bajo:** se codifican con base en las características del procesador.
   b) **Nivel medio:** mantienen algunas características y cualidades de los de nivel bajo, pero con un nivel de abstracción mayor.
   c) **Nivel alto:** son lenguajes de programación similares al lenguaje humano, usando diversos tipos de datos, conceptos, símbolos, etc.

2. **Propósito.**
   a) **General:** pueden realizar cualquier tipo de procesos.
   b) **Específico:** solo pueden procesar información con relación a su propósito específico.
   c) **Programación de sistemas:** son utilizados para crear sistemas operativos o *drivers*, por ejemplo.
   d) ***Script*:** se utilizan para codificar diversas tareas de control y auxiliares dentro de otros lenguajes.

C++ es un derivado del lenguaje de programación de alto nivel C. Un lenguaje de este tipo le permite al programador escribir la codificación de programas con palabras reservadas en inglés, como *if*, *while* o *for*.

Otra de las características de C++ es que, al ser un lenguaje de alto nivel, su programación necesita instrucciones en lenguaje ensamblador para algunas sentencias, el cual es otro tipo de lenguaje de bajo nivel que trabaja con **microcontroladores**.

Los programas diseñados y creados en lenguaje ensamblador se conocen como **código fuente**, pero estos programas se ejecutan en lenguaje máquina, que es el que realmente puede descifrar una computadora, pues es específico para su arquitectura computacional.



Shutterstock/Elena Voynova

Los códigos de nivel alto, como el C++, pueden tener grandes alcances en la programación.

¿Sabías que...?

C++ es un lenguaje de programación creado por Bjarne Stroustrup en los laboratorios de AT&T en 1983. Stroustrup tomó como base el lenguaje de programación más popular de aquella época: el lenguaje C.

Glosario

microcontrolador. Circuito integrado que contiene una unidad central de procesamiento (CPU), unidades de memoria (RAM y ROM), puertos de entrada y salida, así como componentes periféricos.

Sintaxis del lenguaje C++

Todos los lenguajes de programación cuentan con su propia sintaxis, la cual comprende las instrucciones específicas del lenguaje, conocidas como palabras reservadas. La estructura general de un programa en C++ es la siguiente:

```
#include <iostream>
using namespace std;

int main(int argc, char** argv) {

        return 0;
}
```

¿Sabías que...?

La sentencia cin>>variable no puede utilizarse cuando se lee una cadena de caracteres porque reconoce el espacio en blanco o TAB como el fin de la cadena. Para leer este tipo de datos debes usar la sentencia getline(cinvariable). Además, a todos los programas o aplicaciones en C++ se le agrega una línea más de código antes del return 0, que diga system("PAUSE");. Esta línea sirve para que la ventana del cmd o símbolo del sistema no se cierre sola al terminar la ejecución de los procesos.

El análisis de esta estructura es el siguiente:

1. *#include* se utiliza para incluir o llamar ficheros de cabecera ubicados en directorios base de las librerías de C++, poniendo el nombre del fichero entre los símbolos <>. Existen archivos de cabecera estándar muy utilizados, como los siguientes:
   a) *stdlinb.h*: indispensable para usar estructuras de datos dinámicos.
   b) *string*: lo necesitarás cuando uses datos tipo cadenas de carácter.
   c) *math.h*: contiene funciones matemáticas y conversiones avanzadas.
   d) *iostream*: contiene el uso de funciones de entrada y salida (*cout* y *cin*).
   e) *type.h*: funciones de clasificación de caracteres.
2. *using namespace std* indica que las instrucciones o miembros de *namespace* se usarán durante la ejecución del programa de forma frecuente.
3. *int main(int argc, char** argv)* es una línea que especifica el inicio del cuerpo principal del programa, donde *argc* y *argv* son los argumentos que se usarán en todo el programa.
4. *return 0* indica que el programa principal, al final de todos los procesos no regresa nada, es decir, es vacío o *void*.
5. Los signos { } especifican el inicio y el fin de una estructura, respectivamente.
6. El símbolo ; tal vez parezca insignificante, pero es una parte importante del programa que debe ir al final de cada línea de instrucciones específicas que no sean estructuras.

El cuadro siguiente muestra algunas de las palabras reservadas en el lenguaje de programación C++ que usarás a lo largo de esta segunda unidad.

Instrucción en C++	Acción	Instrucción en C++	Acción
`cin>>variable;`	Leer variable	`String`	Cadena de caracteres
`cout<<"mensaje";`	Salida a pantalla	`if`	Si
`<<`	Une mensajes	`else`	De lo contrario
`//`	Comenta una línea de código	`while`	Mientras
`/* */`	Comenta un bloque código	`do while`	Haz mientras
`endl;`	Fin de línea	`for`	Para
`\n`	*Enter* de línea	`switch case`	En caso de
`int`	Entero	`break`	Terminar
`double`	Decimal	`pow`	Elevar a una potencia un número
`char`	Carácter	`sqrt`	Raíz cuadrada

Operadores

C++ cuenta con un gran número de operadores, los cuales están divididos en varias categorías, como son aritméticos, relacionales o lógicos, por asignación, acceso de datos y tamaño, manipulación de bits y varios. Por cuestiones didácticas, en este material solo se usan las primeras tres categorías, es decir, operadores aritméticos, por asignación y relacionales o lógicos. A continuación se muestra otro cuadro con los signos que se pueden usar en cada una de estas categorías.

Operadores aritméticos			
Operador	**Nombre**	**Propósito**	**Ejemplo**
+	Suma	Suma *a* y *b*	c = a + b;
-	Resta	Resta *a* y *b*	c = a - b;
*	Multiplicación	Multiplica *a* y *b*	c = a * b;
/	División	Divide *a* entre *b*	c = a / b;
%	*Mod*	Residuo de *a* entre *b*	c = a%b;
++	Incremento	Incremento de *a*	a++;
--	Decremento	Decremento de *a*	a--;
Operadores de asignación			
+=	Suma corta	Aumentar	a += b;
-=	Resta corta	Disminuir	a -=b;
*=	Multiplicación corta	Multiplicar	a *= b;
/=	División corta	Dividir	a /=b;
Operadores lógicos o relacionales			
&&	AND	Inclusión	a && b
||	OR	Disyunción	a || b
!	NOT	Negación	!a
<	Menor que	Comparar	a<b
<=	Menor o igual que	Comparar	a<=b
>	Mayor que	Comparar	a>b
>=	Mayor o igual que	Comparar	a>=b
==	Igual que	Comparar	a==b
!=	Diferente de	Comparar	a!=b

Encamina tus habilidades

1. Visita la página de **bachilleratoenred.com.mx/enlaces/LCIVUAS2016** y abre el **enlace 2.1** para este libro, el cual contiene una tabla de datos.
2. Con base en esta tabla, completa la cuarta columna colocando la instrucción del lenguaje C++ que corresponda según su acción.
3. Guarda la tabla en un archivo y envíasela a tu profesor para su evaluación y espera su retroalimentación.
4. Edita la tabla con base en las indicaciones de tu profesor y guarda el archivo porque lo usarás más adelante.

Variables y constantes en C++

Al igual que en los algoritmos, en los diagramas de flujo y en los pseudocódigos, la declaración de variables y constantes en el lenguaje C++ debe cumplir con las reglas planteadas en la secuencia didáctica inicial del curso, con la adición de que en este lenguaje se debe cumplir también con la sintaxis de definición que se establece.

A continuación se presenta la manera de declarar las **variables** en este lenguaje de programación, lo cual también dependerá del tipo de datos que se manejarán; recuerda que esto es algo muy importante.

a) Variables de tipo numérico entero: *int x = 0;*
b) Variables de tipo numérico real: *float x = 1.5;*
c) Variables de tipo numérico real más grande que *float: double x = 1.46235;*
d) Variables de tipo carácter: *char x = 'a';*
e) Variables de tipo cadena de caracteres: *string x = "Hola";*

La definición de las **constantes** en C++, al igual que las variables, debe especificar su tipo, pero precedido de la palabra *const*, asignándole además su valor, como se indica a continuación:

a) Constantes de tipo numérico entero: *const int x = 1;*
b) Constantes de tipo numérico real: *double x = 1;*

Editor de texto Dev C++

Glosario

compilador. Programa informático que se encarga de traducir el código fuente de una aplicación que esté en desarrollo, es decir, convierte un programa hecho en lenguaje de programación de nivel alto a un lenguaje de máquina, de tal forma que el equipo de cómputo pueda procesarlo en el programa que se ejecuta.

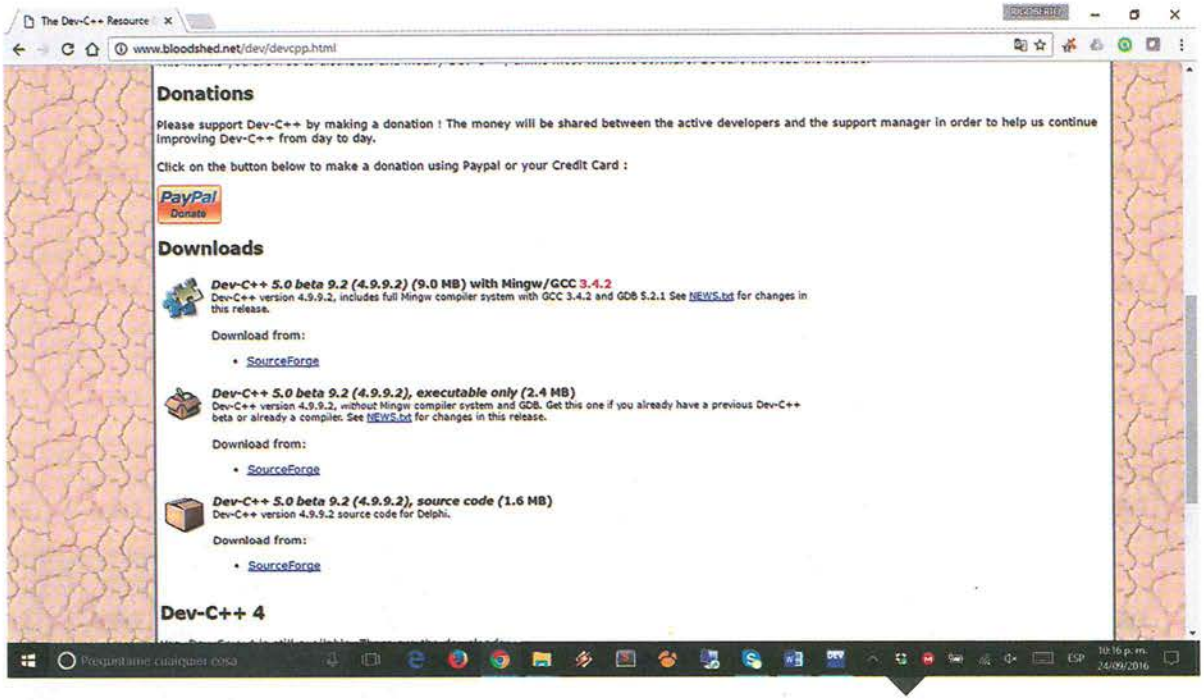
Dev C++ es un editor con Entorno Integrado de Desarrollo (IDE, por sus siglas en inglés), usado principalmente para el lenguaje de programación C++. Este editor es de libre distribución y sus características principales son las siguientes:

1. Soporta los **compiladores** básicos en GCC, por ejemplo, Mingw.
2. Cuenta con un depurador integrado basado en GDB (GNU DeBugger).
3. Mantiene una lista con las clases utilizadas durante la edición de un programa.
4. Mantiene una lista de las funciones definidas en la implementación del programa.
5. Tiene un manejador de proyectos.
6. Soporta la actualización del software y bibliotecas por medio de internet.

Cada vez que realices una secuencia de código en Dev C++ deberás crear un directorio para guardar el proyecto completo, ya que este programa genera los archivos que se requieren para la ejecución de forma automática, los cuales deben estar dentro de un mismo directorio para no ocasionar problemas de ejecución.

Descarga e instalación de Dev C++

El editor Dev C++ lo podrás descargar desde la siguiente dirección URL: http://www.bloodshed.net/dev/devcpp.html. En esta misma página podrás desplazarte hacia abajo y buscar la opción de la versión más actual que contenga Mingw/GCC, descárgalo y sigue las instrucciones para instalarlo.



Opciones de descarga del programa Dev C++.

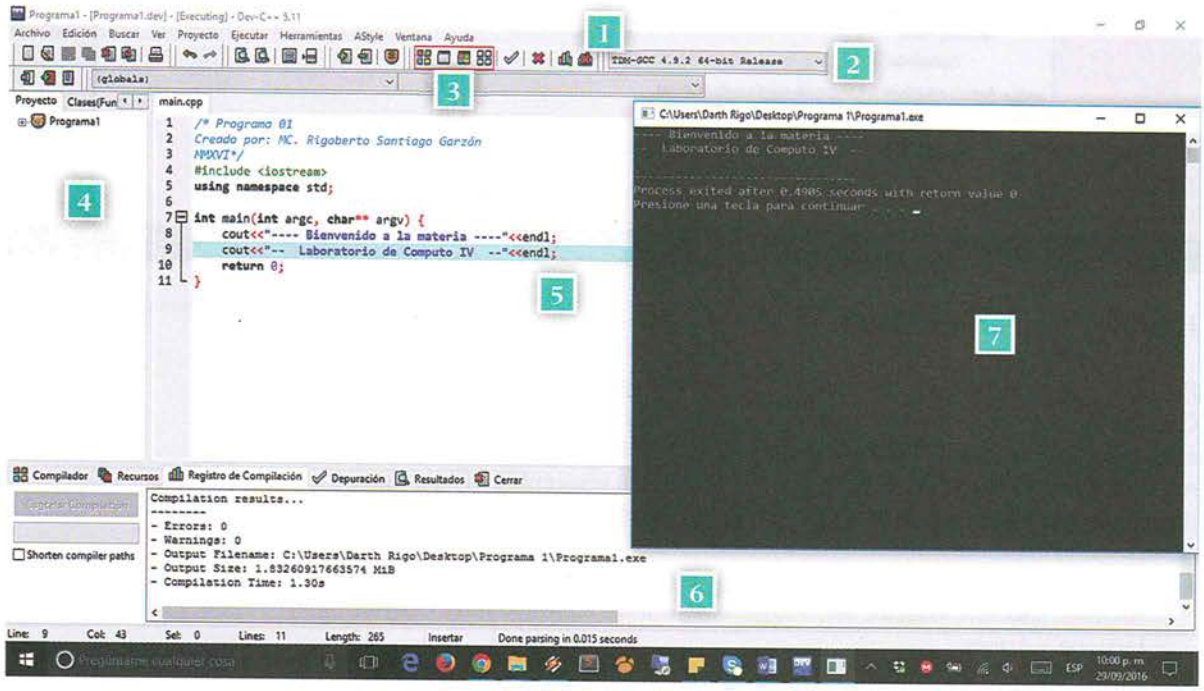
Entorno gráfico de Dev C++

Para comenzar a codificar pseudocódigos en el lenguaje de programación C++ debes conocer primero el entorno gráfico del editor Dev C++, cuyos componentes principales se describen a continuación; relaciona los números de las descripciones con los que se ubican en la imagen de la siguiente página, la cual ilustra el entorno gráfico de este programa.

1. **Barra de Menú.** Contiene los menús principales del editor, que son *Archivo, Edición, Buscar, Ver, Proyecto, Ejecutar, Herramientas, AStyle, Ventana* y *Ayuda*.
2. **Barra de herramientas.** Al igual que en otros *softwares* esta barra presenta las opciones más usadas por los usuarios, como pueden ser *Nuevo, Abrir proyecto, Guardar, Guardar todo, Cerrar* y *Cerrar todo*, entre otras.
3. **Botones de ejecución.** Estos botones contienen las opciones necesarias para realizar las siguientes acciones con el código:
   a) **Compilar (F9).** Esta acción traduce el código escrito en el lenguaje C++ a un código ejecutable por la máquina. En este proceso, el compilador busca que no existan errores en la sintaxis del código; en caso de encontrar alguno se mostrará en la casilla *Registro de Compilación (6)*, informando cuáles son los errores y en qué línea se encuentran. Los errores se clasifican solo en dos tipos, si son *de compilación* o *de ejecución*.
   b) **Ejecutar (F10).** Este botón se encargará de ejecutar la última compilación del código, generando un archivo *.exe(ejecutable)* en el directorio del proyecto; al ejecutarse en este editor el código abrirá de forma automática una ventana del *cmd* o *símbolo del sistema (6)*, en la cual podrás ver la ejecución del programa.
   c) **Compilar y Ejecutar (F11).** Realiza con un solo clic las dos acciones anteriores, es decir, compila el código y lo ejecuta, siempre y cuando no tenga errores.

d) **Reconstruir todo (F12).** Vuelve a compilar el código y hace el archivo *.exe* con los cambios realizados, pero sin ejecutarlo.

4. **Navegador de Proyectos/Clases.** Área que muestra el o los proyectos abiertos para poder acceder a ellos de manera directa.
5. **Área de codificación.** En esta zona se deben teclear todas las instrucciones en el lenguaje C++, siguiendo las reglas de sintaxis y de estándar de codificación de programación estructurada.
6. **Registro de codificación.** Muestra los resultados de la compilación o reconstrucción de la codificación del programa en C++.
7. **Ventana Promt (cmd o símbolo del sistema).** Esta será la ventana de ejecución de los programas; en ella podrás leer, procesar y mostrar información, según la codificación.



Subproductos

Subproducto procedimental

1. De manera individual, elabora tu primer programa en C++, cuya secuencia de código solo usará algunas de las sentencias o palabras claves iniciales del lenguaje C++. El programa solo deberá mostrar mensajes en la ventana del CMD, imitando una hoja de presentación de un trabajo escolar.
2. Para lograr la estructura definida en el punto anterior, considera los siguientes datos para la presentación de un trabajo escolar: nombres de la universidad, del colegio, de la asignatura, del estudiante y del profesor, así como el número de la unidad y del grupo.
3. Guarda el proyecto en un directorio siguiendo las instrucciones de tu profesor y envíaselo para su evaluación.

Subproducto declarativo

› **Relaciona las siguientes columnas con base en lo que has aprendido en esta secuencia didáctica.**

1. ¿Quién es el creador del lenguaje C++?	( ) iostream
2. ¿En qué año se creó el lenguaje C++?	( ) Enteros
3. ¿Cuál es el carácter utilizado para hacer comentarios en un bloque de código?	( ) ;
4. ¿Cuál es el carácter utilizado para separar instrucciones?	
5. ¿Cuál es el fichero de cabecera básico que se debe importar para cualquier programa en lenguaje C++?	( ) Compilación
6. ¿Cuál es el carácter utilizado para hacer comentarios en una línea de código?	( ) 1983
7. ¿Cuál es el carácter que simboliza el inicio y el fin de una estructura?	( ) Reales grandes
8. ¿Cuál es el nombre del proceso al que se somete el código de los programas, donde se encuentran posibles errores en su codificación o ejecución?	( ) Bjarne Stroustrup
9. ¿Qué tipos de datos se declaran colocando la palabra *int* en C++?	( ){ }
10. ¿Qué tipos de datos se declaran colocando la palabra *double* en C++?	( ) /* */
	( ) //

Subproducto actitudinal-valoral

› **Describe una situación en la que puedas aplicar el conocimiento que has adquirido en esta secuencia didáctica.**

2.2. Estructuras en el lenguaje de programación C++

A desarrollar

Al término de esta secuencia didáctica aprenderás a programar en lenguaje C++ con base en las sentencias de control de flujo secuencial, selectivo y repetitivo.

El **subproducto procedimental** de esta secuencia didáctica consiste en la transcripción de la solución de un problema en el lenguaje de programación C++.

Las actividades que realizarás en esta secuencia didáctica te servirán para:

4.2. Aplicar diversas estrategias comunicativas según quienes sean los interlocutores, el contexto en el que te encuentras y los objetivos que persigues.
5.1. Seguir instrucciones y procedimientos de manera reflexiva en la búsqueda y adquisición de nuevos conocimientos.
5.2. Ordenar información de acuerdo con categorías, jerarquías y relaciones.
5.6. Utilizar las tecnologías de la información y la comunicación para procesar e interpretar información.
7.3. Articular saberes de diversos campos y establecer relaciones entre ellos y tu vida cotidiana.
8.1. Plantear problemas y ofrecer alternativas de solución al desarrollar proyectos en equipos de trabajo, y definir un curso de acción con pasos específicos.
8.2. Aportar puntos de vista con apertura y considerar los de otras personas de manera reflexiva.
8.3. Asumir una actitud constructiva al intervenir en equipos de trabajo, congruente con los conocimientos y las habilidades que posees.

De aquí en adelante

› **Describe las características de las siguientes estructuras de control.**

- Estructuras de control secuencial: _____
- Estructuras de control selectivo: _____
- Estructuras de control repetitivo: _____

Estructuras de control en el lenguaje C++

Las estructuras que se manejan en el lenguaje de programación C++ son las mismas que aprendiste al inicio del curso. Como recordarás, el principal objetivo de los algoritmos es que sean programados o codificados en un lenguaje de programación; por tanto, en esta secuencia didáctica aprenderás a elaborar los algoritmos que resuelven problemas concretos en el lenguaje C++ de programación de nivel alto.

Estructuras de control secuencial en el lenguaje C++

Los códigos en el lenguaje C++, al igual que los algoritmos que has elaborado, están compuestos por una secuencia de acciones o instrucciones que se deben ejecutar en orden, lo que se conoce como estructura. Estas estructuras deben escribirse en el código separándolas por el carácter punto y coma (;). Por ejemplo, si necesitas agrupar en un bloque una serie de acciones, este bloque deberá colocarse entre el símbolo de la llave de apertura ({) que marcará el inicio y la llave de cierre (}), la cual marcará el final.

¿Sabías que...?

Cuando se codifica un algoritmo en el lenguaje C++ se deben seguir ciertas normas llamadas *Estándares de codificación*, esto significa que debe cumplirse un estilo homogéneo de codificación que permita que cualquier programador logre entender el código. Este estándar de codificación posee características específicas como las sangrías, los comentarios de código o la declaración de variables, entre muchas otras.

Ejemplo 2.2.1

Problema: obtener el perímetro y el área de un cuadrado del que se desconocen sus dimensiones.
Solución: retomar el algoritmo que elaboraste al resolver el problema del *Ejemplo 1.1.1* y el diagrama de flujo del *Ejemplo 1.2.1* así como el pseudocódigo creado en Scratch elaborado en el *Ejemplo 1.3.1*, en los cuales usaste estructuras de control secuencial para resolver el problema. Con base en estas evidencias, generar el código en lenguaje C++.

Fase 2: crear el código en lenguaje C++:

Estructuras de control secuencial

```
/* Programa 01
Creado por: MC. Rigoberto Santiago Garzón
MMXVI*/
#include <iostream>
#include <math.h>
using namespace std;

int main(int argc, char** argv) {
    double m_lado  = 0;
    double perimetro = 0;
    double area    = 0;
    cout<<"Este algoritmo servirá para obtener el
    Perímetro y el Área de un cuadrado"<<endl;
    cout<<"¿Cuánto miden en metros los lados del
    cuadrado?";
    cin>>m_lado;
    perimetro     =     m_lado*4;
    area          =     pow(m_lado,2);
    cout<<"Para un cuadrado de "<<m_lado<<" metros
    por lado, su Perímetro es de "<<perimetro<<"
    metros lineales y su Área es de "<<area<<" metros
    cuadrados.";
    system("PAUSE");
    return 0;
}
```

A practicar 2.2.1

Retoma las evidencias que obtuviste al realizar las actividades de la sección *A Practicar 1.1.2, 1.2.1* y 1.3.1 de las páginas 19, 36 y 55, respectivamente, y con ayuda de tu profesor codifica la solución del problema mencionado en el lenguaje C++.

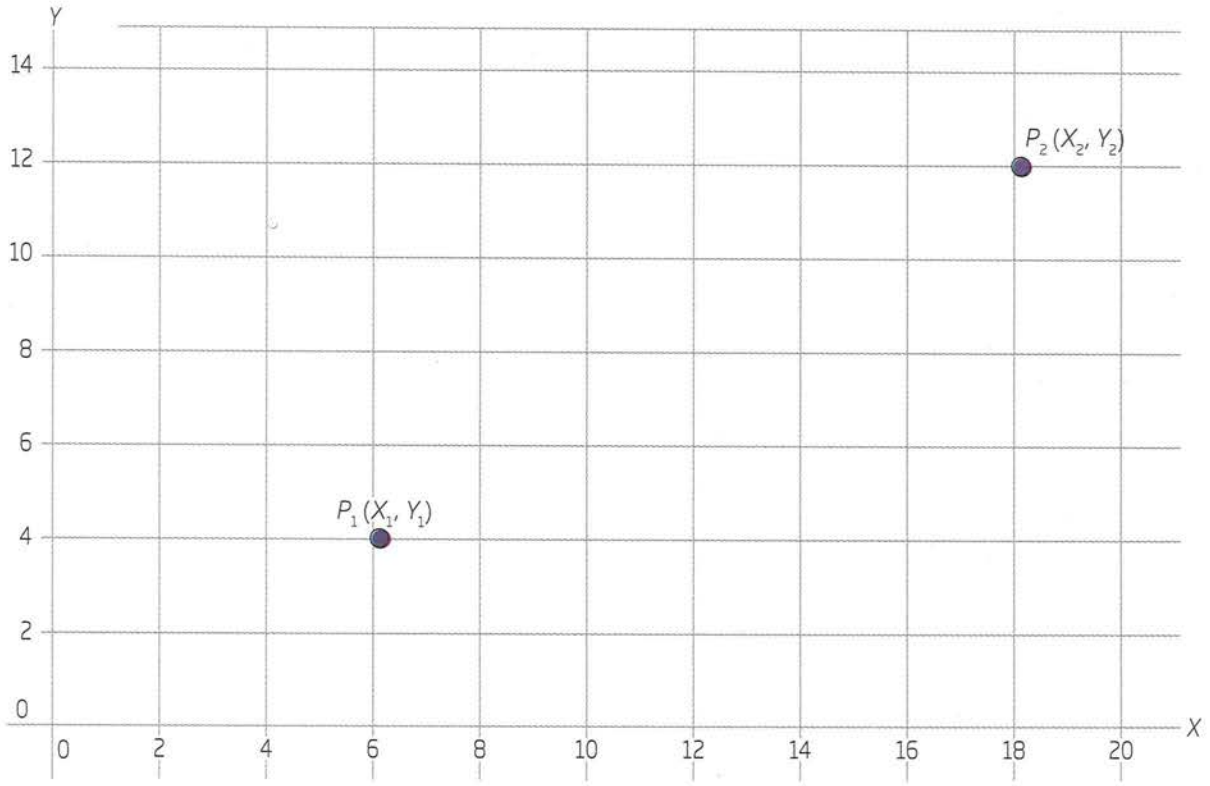
Guarda el directorio del proyecto junto con el documento de Word donde está el algoritmo y el diagrama de flujo, así como el archivo de Scratch que contiene el pseudocódigo.

Insignia

Hasta este momento ya has aprendido cómo elaborar algoritmos, diagramas de flujo, pseudocódigos en Scratch y codificación en C++.

Por tanto, para obtener esta insignia debes realizar los cuatro procesos mencionados antes para solucionar el siguiente problema:

Calcular la distancia entre dos puntos con base en sus coordenadas, las cuales se ilustran en la siguiente gráfica:



Encamina tus habilidades

1. Reúne las evidencias que obtuviste al realizar la actividad propuesta en la página 56, que son el algoritmo, el diagrama de flujo y el pseudocódigo que sirven para solucionar el siguiente problema: conocer el nombre de un oso polar que vive en un zoológico, su peso en libras y su longitud en pies, convertir su peso a kilogramos y su longitud a metros; mostrar como resultado el nombre del oso polar, su peso en kilogramos y su longitud en metros.
2. De manera individual, codifica la solución en el lenguaje C++.
3. Coloca el directorio del proyecto de C++ en el mismo directorio donde están el archivo de Word y el archivo de Scratch, el cual deberás comprimirlo y enviarlo a tu profesor para su evaluación.

Estructuras de control selectivo en el lenguaje C++

Como recordarás, este tipo de estructuras son capaces de organizar una serie de instrucciones, acciones o procesos para que se realicen con base en el resultado de una condición. Existen tres grandes tipos: **simples**, **dobles** y **múltiples**, la sintaxis de cada una de ellas se muestra a continuación:

***Simples:**

```
If (condición(es)){
        ..... Proceso(s);
}
```

***Dobles:**

```
If(condición(es){
        ..... Proceso(s);
}else{
        ..... Proceso(s);
}
```

***Múltiples**

- Selectivas simples anidadas

```
if (condición(es)){
        ..... Proceso(s);
}else if (condición(es)){
        ..... Proceso(s);
}else{
        ..... Proceso(s)
}
```

- Selectivas de caso

```
switch (opción){
        case valor1:
                        .... Proceso(s);
                        break;
        case valor2:
                        ..... Proceso(s);
                        break;
        default:
                        ..... Proceso(s);
}
```

Fase 2: crear el código en el lenguaje C++:
Estructuras de control secuencial, selectivo, repetitivo y de datos

```
/* Programa 04
Creado por: MC. Rigoberto Santiago Garzón
MMXVI*/
#include <iostream>
using namespace std;

int main(int argc, char** argv) {
   int* numeros    =      NULL;
   int mayor       =      0;
   int menor       =      0;
   int i           =      0;
   int n           =      0;
   int band        =      0;
   int pos_mayor   =      0;
   int pos_menor   =      0;
   cout<<"Ahora podremos capturar varios numeros y
   almacenarlos en un arreglo, y la final saber cual es
   el mayor y cual el menor, y conocer exactamente en que
   posicion del arreglo se encuentra cada uno."<<endl;
   cout<<"¿Cuántos numeros deseas capturar?";
   cin>>n;
   numeros         =      new int[n];
   for(i=0;i<n;i++){
           cout<<"Numero "<<i+1<<": ";
           cin>>numeros[i];
   }
   for(i=0;i<n;i++){
           if(band==0){
                   band          =      1;
                   mayor         =      numeros[i];
                   pos_mayor     =      i;
           }else if(mayor<numeros[i]){
                   mayor         =      numeros[i];
                   pos_mayor     =      i;
           }
   }
   band    =      0;
   for(i=0;i<n;i++){
           if(band==0){
                   band          =      1;
                   menor         =      numeros[i];
                   pos_menor     =      i;
           }else if(menor>numeros[i]){
                   menor         =      numeros[i];
                   pos_menor     =      i;
           }
   }
   cout<<"De los numeros que introdujiste el mayor fue
   el: "<<mayor<<" y se encuentra en la posicion "<<pos_
   mayor+1<<" del arreglo."<<endl;
   cout<<"De los numeros que introdujiste el menor fue
   el: "<<menor<<" y se encuentra en la posicion "<<pos_
   menor+1<<" del arreglo."<<endl;
   system("PAUSE");
   return 0;
}
```

A practicar 2.2.3

Retoma las evidencias que obtuviste al realizar las actividades de la sección *A Practicar 1.1.5, 1.2.3* y 1.3.3 de las páginas 27, 42 y 59, respectivamente, y con ayuda de tu profesor codifica la solución del problema mencionado con el lenguaje C++.

Guarda el directorio del proyecto junto con el documento de Word donde está el algoritmo y el diagrama de flujo, así como el archivo de Scratch que contiene el pseudocódigo.

Encamina tus habilidades

1. Reúne las evidencias que obtuviste al realizar la actividad propuesta en la sección *Encamina tus habilidades* de la página 59, que son el algoritmo, el diagrama de flujo y el pseudocódigo que sirven para solucionar el siguiente problema: conocer cierta cantidad de números enteros (tal vez repetidos) y mostrar como resultado el arreglo de los números enteros ordenados, pero sin mostrar los repetidos.
2. De manera individual, codifica la solución en el lenguaje C++.
3. Coloca el directorio del proyecto de C++ en el mismo directorio donde están el archivo de Word y el archivo de Scratch, el cual deberás comprimirlo, y envíalo a tu profesor para su evaluación.

Insignia

Para conseguir la última insignia del curso deberás aplicar todo lo que aprendiste para diseñar y crear el algoritmo, el diagrama de flujo, el pseudocódigo en Scratch y el proyecto codificado en el lenguaje C++, usando estructuras de control y de datos, para resolver el siguiente problema: capturar dos vectores para que sus elementos se multipliquen y obtener en vector resultante lo siguiente:

$$\begin{bmatrix} A_1 \\ A_2 \\ A_3 \\ \cdot \\ \cdot \\ \cdot \\ A_n \end{bmatrix} \cdot \begin{bmatrix} B_1 \\ B_2 \\ B_3 \\ \cdot \\ \cdot \\ \cdot \\ B_n \end{bmatrix} = \begin{bmatrix} C_1 \\ C_2 \\ C_3 \\ \cdot \\ \cdot \\ \cdot \\ C_n \end{bmatrix}$$

Subproductos

Subproducto procedimental

1. Reúnete con el compañero con el que resolviste el subproducto de la secuencia didáctica 1.3, que consistía en resolver el siguiente problema:

   El usuario de una compañía de telefonía móvil necesita calcular el costo de las llamadas telefónicas internacionales que realizó durante el mes pasado, dicho costo depende de la zona geográfica del país donde se recibe la llamada y del número de minutos en que se habló. Mostrando como resultado cuántas llamadas se hicieron a cada zona, el pago total por zona y cuál es el cargo completo del recibo telefónico. En la siguiente tabla se presenta el costo del minuto por zona; cada zona tiene asignada una clave.

Tabla de zona y precios		
Clave	**Zona**	**Precio**
12	América del Norte	$2.20
15	América Central	$2.80
18	América del Sur	$4.75
19	Europa	$3.65
23	Asia	$6.25
25	África	$6.25
29	Oceanía	$5.15

2. Retomen las evidencias que obtuvieron al resolver este problema, que son el algoritmo, el diagrama de flujo y el pseudocódigo en Scratch, y codifiquen la solución en el lenguaje C++ usando las estructuras de control secuencial, selectivo y repetitivo, así como las estructuras de datos que consideren necesarias.
3. Compartan su código C++ con el profesor para que los evalúe y reciban retroalimentación.
4. Apliquen las modificaciones necesarias y guarden el código siguiendo las indicaciones de su profesor.

Subproducto actitudinal-valoral

› **Contesta lo siguiente:**

1. Con base en lo que has aprendido acerca de la programación ¿qué alcances visualizas en el uso de tu equipo de cómputo?

2. ¿Qué otras situaciones en tu vida académica, personal o profesional se te ocurren que puedes solucionar con lo que has aprendido en el curso?

3. Explica la manera en que ahora percibes la comunicación entre el software y el hardware de una computadora, así como la interacción entre el operador y el computador.

Subproducto declarativo

1. Para incluir un fichero de cabecera en C++ se utiliza la sentencia... ( ) Alto nivel
2. La función *sqrt* se refiera a... ( ) Compilación y ejecución
3. La sentencia en C++ *const float PI = 3.14* hace referencia a... ( ) int a = 0;
4. Un código en el lenguaje C++ debe pasar por los procesos... ( ) #include
5. IOSTREAM, CONIO y STDLIB, son... ( ) Calcular raíz cuadrada
6. La sentencia *Decir* y *Preguntar* en un pseudocódigo equivale en el lenguaje C++ a... ( ) Declaración de una constante
7. Sentencia que se usa para leer información en C++. ( ) cout<<
8. Ejemplo de declaración de una variable en C++. ( ) cin>>
9. Los lenguajes de programación se clasifican por su... ( ) Ficheros de cabecera
10. C++ es un lenguaje de programación de... ( ) Nivel de abstracción y propósito

Producto integrador de la unidad 2

Síntesis

1. Reúnete con los compañeros con quienes trabajarás el producto integrador del curso para esta asignatura.

2. Retomen el producto integrador de la unidad 1 (Fundamentos de programación) y las evidencias que obtuvieron en la sección *Subproducto procedimental* de cada una de las cinco secuencias didácticas de este curso, que son las siguientes:
   - Algoritmo para resolver un problema específico (página 31).
   - Algoritmo y diagrama de flujo para resolver un problema específico (páginas 44 y 45).
   - Algoritmo, diagrama de flujo y pseudocódigo creado en Scratch para resolver un problema específico (página 61).
   - Portada de un trabajo escolar diseñado en Dev C++ (página 72).
   - Código C++ para resolver un problema específico (página 84)

3. Codifiquen en el lenguaje de programación C++. Para lograrlo, deben usar las estructuras de control secuencial, selectivo y repetitivo, así como las estructuras de datos.

4. Una vez terminado, comprueben que el código funcione.

5. Coloquen el directorio del proyecto en el mismo directorio donde se encuentran los archivos del *Producto Integrador* antes mencionado, compriman los archivos del directorio y envíenlo a su profesor para su revisión.

Autoevaluación

Demuestra el nivel de tu conocimiento mediante la siguiente guía de evaluación, utilizando la escala del 1 al 5, en la que 1 es el valor más bajo y 5 el más alto.

Mis conocimientos en:	Corresponden a un nivel de:				
	1	2	3	4	5
Identificar cómo el software manipula el hardware del equipo de cómputo.					
Identificar cómo funcionan los lenguajes de programación.					
Elaborar programas en el lenguaje de programación C++.					
La importancia del trabajo en equipo para compartir conocimientos y habilidades de, manera responsable.					
El manejo de las tecnologías de la información y la comunicación para obtener y expresar información de manera responsable y respetuosa.					

Coevaluación

Realiza la siguiente coevalución con tus compañeros de equipo con los que realizaste el Producto Integrador de la unidad 2. Sé honesto y justo con tus respuestas. Justifica cada una de las opciones.

Nombre del compañero	
Participó activamente, medianamente o pobremente en la realización de las actividades	
Aportó buenas ideas, señaló sus puntos de vista y realizó el trabajo que le correspondió	
Fue respetuoso y accedió a las decisiones de la mayoría de los integrantes	
Podría mejorar si...	

Producto integrador del curso

Síntesis

1. Reúnete con los compañeros con quienes has trabajado en el curso de manera colaborativa.

2. Retoma el Producto Integrador de las unidades 1 y 2. Apliquen los siguientes cambios a cada uno de sus elementos, que son: el algoritmo, el diagrama de flujo, el pseudocódigo y su codificaión en el lenguaje C++.
   - Agreguen una opción más al menú, la cual denominarán *"Juego del Ahorcado"*. Las estructuras de control y de datos que diseñen para esta opción deberán resolver el siguiente problema: simular el juego del ahorcado usando solo una palabra, sin importar su longitud y sin caracteres especiales, con un máximo de cinco errores, es decir, si se piden cinco letras y no están en el texto original, debe de terminarse el juego y el jugador habrá perdido.
   - Después de agregar la opción anterior, agregarán una más con el texto: "Salir", logrando que el algoritmo siga en función si no se selecciona esa opción, es decir, están en un ciclo infinito que se romperá cuando se elija la opción "Salir" (ver Anexo).

3. Una vez que hayan probado que funciona todo lo anterior de manera adecuada, lo codificarán en el editor Dev C++, con el lenguaje C++.

4. Para entregar este Producto Integrador del Curso deberán colocar el directorio del proyecto en C++, junto con los archivos del algoritmo, el diagrama de flujo y el pseudocódigo en Scratch; después deben comprimirlo y enviarlo a su profesor para su evaluación.

Autoevaluación

Demuestra el nivel de tu conocimiento mediante la siguiente guía de evaluación, utilizando la escala del 1 al 5, en la que 1 es el valor más bajo y 5 el más alto.

Mis conocimientos en:	Corresponden a un nivel de:				
	1	2	3	4	5
Aplicar las fases de solución de problemas.					
Distinguir los elementos principales que enmarcan los problemas, como las variables y las constantes.					
Aplicar diversas estructuras de control y de datos para solucionar de manera óptima diversos problemas.					
Identificar el funcionamiento del software.					
Identificar el funcionamiento del hardware.					
Aplicar la programación estructurada para la codificación de programas que solucionen problemas programables.					
Reconocer la importancia del trabajo en equipo para compartir conocimientos y habilidades de manera responsable.					

Coevaluación

Realiza la siguiente coevalución con tus compañeros de equipo con los que realizaste el Producto Integrador del curso. Sé honesto y justo con tus respuestas. Justifica cada una de las opciones.

Nombre del compañero	
Participó activamente, medianamente o pobremente en la realización de las actividades	
Aportó buenas ideas, señaló sus puntos de vista y realizó el trabajo que le correspondió	
Fue respetuoso y accedió a las decisiones de la mayoría de los integrantes	
Podría mejorar si...	

Anexo

Como sabes, los diversos software que has utilizado hasta ahora en el ámbito académico siempre se mantienen en ejecución hasta que le indicas que quieras salir de él, ya sea presionando un botón de salida o algunas teclas específicas. Este anexo está dirigido a que analices el diseño de dos algoritmos cuya función específica es **salir** de los programas en que han sido creados.

Si analizas detenidamente, la intención de los algoritmos es ejecutar los programas o aplicaciones por medio de ciclos o bucles infinitos que solo se romperán cuando el usuario lo indique.

Ahora, si te remontas a la primera secuencia didáctica de este libro, específicamente en el apartado de las estructuras de control repetitivo, recordarás que existen tres tipos, que son: *Para (for)*, *Mientras (while)* y *Haz mientras (do while)*. De estas estructuras de control repetitivo la única que no funciona para realizar la acción es *Para* o *for*, pero las otras dos (*Mientras* y *Haz mientras*) sí permiten crear una estructura para realizar la ejecución de los programas de manera cíclica hasta aplicar la orden de salir.

Entre los ciclos *Mientras* y *Haz mientras*, como recordarás, solo existe una diferencia, que el segundo asegura que se ejecuten al menos una vez los procesos que están contenidos en su interior; por tanto, este ciclo (*Haz mientras*) es el ideal para usarlo y lograr que las aplicaciones se ejecuten de forma continua, sin embargo, no significa que no se pueda lograr con el *Mientras* también.

En las siguientes páginas, leerás dos algoritmos diseñados para calcular el perímetro y el área de un cuadrado en repetidas ocasiones, hasta que el usuario decida salir, cada uno ofrece la opción numérica o con un carácter.

Fase 2: crear el algoritmo en el bucle *Haz mientras*:

Opción numérica

1. Inicio.
2. Programar las variables a utilizar (opcion = 0, m_lado = 0, perimetro = 0, area = 0).
3. Haz.
   3.1. Decir: "Este algoritmo servirá para obtener el Perímetro y el Área de un cuadrado".
   3.2. Preguntar: "¿Cuánto miden los lados del cuadrado expresado en metros?".
   3.3. Asignar el valor de la respuesta a la variable m_lado (m_lado = respuesta).
   3.4. Asignar el resultado de aplicar la fórmula para obtener el perímetro a la variable *perimetro* (perimetro = 4*m_lado).
   3.5. Asignar el resultado de aplicar la fórmula para obtener el área a la variable *area* (area = m_lado*m_lado).
   3.6. Decir: "Para un cuadrado de" + m_lado + "metros por lado, su Perímetro es de" + *perimetro* + "metros lineales y su Área es de" + *area* + "metros cuadrados.".
   3.7. Preguntar: "¿Quieres calcular otro cuadrado con distintas medidas (1=Sí, 0=No)?".
   3.8. Asignar el valor de la respuesta a la variable *opcion* (opcion = respuesta).
4. *Mientras* (opción = 1).
5. Fin.

Fase 2: crear el algoritmo en el bucle *Haz mientras:*

Opción con un carácter

1. Programar las variables a utilizar (opcion = '', m_lado = 0, perimetro = 0, area = 0).
2. Haz.
   2.1. Decir: "Este algoritmo servirá para obtener el Perímetro y el Área de un cuadrado".
   2.2. Preguntar: "¿Cuánto miden los lados del cuadrado expresado en metros?".
   2.3. Asignar el valor de la respuesta a la variable m_lado (m_lado = respuesta).
   2.4. Asignar el resultado de aplicar la fórmula para obtener el perímetro a la variable *perimetro* (perimetro = 4*m_lado).
   2.5. Asignar el resultado de aplicar la fórmula para obtener el área a la variable *area* (area = m_lado*m_lado).
   2.6. Decir: "Para un cuadrado de" + m_lado + "metros por lado, su Perímetro es de" + *perimetro* + " metros lineales y su Área es de" + *area* + "metros cuadrados.".
   2.7. Preguntar: "¿Quieres calcular otro cuadrado con distintas medidas [S/N]?".
   2.8. Asignar el valor de la respuesta a la variable *opcion* (opcion = respuesta).
3. *Mientras* (opcion = 's' O opcion = 'S').
4. Fin.

Fase 2: crear el algoritmo en el bucle *Mientras:*

Opción numérica

1. Inicio.
2. Programar las variables a utilizar (opcion = 0, m_lado = 0, perimetro = 0, area = 0).
3. Asignar el valor de 1 a la variable *opcion* (opcion = 1).
4. Mientras (opción = 1).
   - 4.1. Decir: "Este algoritmo servirá para obtener el Perímetro y el Área de un c uadrado".
   - 4.2. Preguntar: "¿Cuánto miden los lados del cuadrado expresado en metros?".
   - 4.3. Asignar el valor de la respuesta a la variable *m_lado* (m_lado = respuesta).
   - 4.4. Asignar el resultado de aplicar la fórmula para obtener el perímetro a la variable *perimetro* (perimetro = 4*m_lado).
   - 4.5. Asignar el resultado de aplicar la fórmula para obtener el área a la variable *area* (area = m_lado*m_lado).
   - 4.6. Decir: "Para un cuadrado de" + m_lado + "metros por lado, su Perímetro es de" + *perimetro* + "metros lineales y su Área es de" + *area* + "metros cuadrados.".
   - 4.7. Preguntar: "¿Quieres calcular otro cuadrado con distintas medidas (1=Sí, 0=No)?".
   - 4.8. Asignar el valor de la respuesta a la variable *opcion* (opcion = respuesta).
5. Fin del *Mientras*.
6. Fin.

Fase 2: crear el algoritmo en el bucle *Mientras:*

Opción con un carácter

1. Inicio.
2. Programar las variables a utilizar (opcion = '', m_lado = 0, perimetro = 0, area = 0).
3. Asignar el valor de 's' a la variable *opcion* (opcion = 's').
4. Mientras (opcion = 's' O opcion = 'S').
   - 4.1. Decir: "Este algoritmo servirá para obtener el Perímetro y el Área de un cuadrado".
   - 4.2. Preguntar: "¿Cuánto miden los lados del cuadrado expresado en metros?".
   - 4.3. Asignar el valor de la respuesta a la variable *m_lado* (m_lado = respuesta).
   - 4.4. Asignar el resultado de aplicar la fórmula para obtener el perímetro a la variable *perimetro* (perimetro = 4*m_lado).
   - 4.5. Asignar el resultado de aplicar la fórmula para obtener el área a la variable *area* (area = m_lado*m_lado).
   - 4.6. Decir: "Para un cuadrado de" + m_lado + "metros por lado, su Perímetro es de" + *perimetro* + "metros lineales y su Área es de" + *area* + " metros cuadrados."
   - 4.7. Preguntar: "¿Quieres calcular otro cuadrado con distintas medidas [S/N]?".
   - 4.8. Asignar el valor de la respuesta a la variable *opcion* (opcion = respuesta).
5. Fin del *Mientras*.
6. Fin.

Como podrás darte cuenta, en el primer algoritmo de cada bucle se asigna un valor numérico de 1 a la respuesta Sí y de 0 al No. En el segundo algoritmo de cada bucle, en cambio, la respuesta Sí se asigna a los caracteres 's' o 'S' y a la respuesta No los caracteres 'n' o 'N'.

Al usar el ciclo *Haz mientras* se asegura que por lo menos una vez entren a ejecución los procesos del programa y al terminar la ejecución se le pregunta al usuario si quiere volver a ejecutar de nuevo el programa o no. En el caso del ciclo *Mientras*, necesitas agregar un paso más en el algoritmo para darle un valor verdadero a la variable que almacena la opción para que la condición del ciclo se cumpla la primera vez, por esta razón se dice que la estructura de control repetitivo *Haz Mientras* es la idónea para hacer que los programas se ejecuten hasta que el usuario indique lo contrario, porque siempre debes tener claro que un paso más en el algoritmo representa una línea de código más y, por ende, un paso más a ejecutar por la computadora cuando el algoritmo se programe.

Cabe mencionar que en lenguajes de programación cuya ejecución del código se da en consola, como lo es C++, es necesario realizar este tipo de acciones, pero en los lenguajes que ejecutan sus aplicaciones en modo grafico lo que se hace es solo colocar una opción, ya sea en un menú o en un botón para salir. Sin embargo, no hay necesidad de colocar la ejecución de los procesos dentro de un ciclo, ya que el mismo lenguaje tiene la instrucción de manera interna, para que la ejecución del programa siga hasta que se presiona la opción para salir.

Bibliografía

- Cairó Battistutti, O. (2005). *Metodología de la programación* (3a ed.). Alfaomega. Ciudad de México, México.
- Deitel, Paul y Harvey Deitel (2014). *Cómo programar en C++*. Pearson educación. Ciudad de México, México.
- Miranda Chávez, Edna Martha y Fuenlabrada Velázquez, Sergio (2015). *Manejo de técnicas de programación*. Pearson Educación. Ciudad de México, México.
- Marcelo Villalobos, Ricardo (2008). *Fundamentos de programación C+. Más de 100 algoritmos codificados*. Macro E.I.R.L editorial. Perú.
- Puntambekar, A. A. (2009). *Data, Structures & Algorithms*. Technical Publications Pune. India.

ISBN 978-607-01-3297-1

Laboratorio de cómputo IV acerca al estudiante de nivel medio superior al tratamiento digital de la información por medio de las computadoras en su vida académica y personal. Los contenidos teóricos, los ejercicios y las actividades se diseñaron de acuerdo con el Programa de Estudios 2015 del Bachillerato General por Competencias, emitido por la Dirección General de Escuelas Preparatorias (DGEP) de la Universidad Autónoma de Sinaloa. Los lineamientos de este programa se basan en la Reforma Integral de la Educación Media Superior (RIEMS), que a su vez se sustenta en el **enfoque educativo por competencias**.

El libro tiene dos unidades que, por su contenido, se agrupan en los siguientes ámbitos del conocimiento: fundamentos de programación y codificación de pseudocódigo. Cada unidad se divide en secuencias didácticas, en las cuales se exponen los contenidos y los procedimientos con un vocabulario sencillo y adecuado, aunque no carente del rigor que el estudio de la asignatura requiere. Para que resulten claras y precisas, las explicaciones se acompañan de ejemplos e imágenes que hacen más amable el encuentro con la teoría.

El mundo actual exige a nuestros estudiantes estar cada vez más preparados para enfrentar sus retos. Este libro puede ser una herramienta decisiva para que tengan acceso al nivel de competencia que les corresponde.

Bachilleratoenred.com.mx