BLOQUE 5 Introducción a la programación estructurada

¿Qué actividades realizas en un día normal? Desde que te levantas hasta que llegas a la escuela realizas una serie de acciones: ¿siempre son las mismas?, ¿las haces en forma consciente? Por lo general, las acciones cotidianas se efectúan de manera automática.

1. Te levantas

2. Te aseas

3. Te vistes

4. Desayunas

5. Te lavas los dientes

6. Tomas tu mochila

7. Te trasladas a la escuela

Programar es la acción de crear procedimientos que nos lleven a efectuar una tarea o a resolver un problema en forma sencilla y sistemática. El procedimiento lógico que facilita la solución se conoce como **algoritmo**.

Las áreas donde participa activamente el personal informático son:

- **Usuario**. Realiza cualquier actividad académica o profesional utilizando computadoras y aplicaciones de propósito específico, como hojas o libros de cálculo, procesadores de palabras o **programas** de diseño.
- **Desarrollador**. Es la persona que crea los programas o aplicaciones con las cuales trabajas en la computadora.
- **Técnico**. Es el ingeniero o técnico de mantenimiento que se encarga de reparar y mantener en funcionamiento los equipos y las redes de computadoras.

Ya conoces muchas de las tareas que realizan los usuarios y qué programas utilizan, ahora el reto es mayor y más interesante. Entrar al mundo de los desarrolladores te abre un horizonte de mayor realización personal, ya que despierta tu imaginación y desarrolla tus habilidades mentales, que serán de gran utilidad durante tu vida.

5.1 Lenguajes de programación

Para resolver problemas con la computadora mediante la programación, es necesario aprender un lenguaje de programación capaz de traducir las órdenes del usuario al lenguaje que entiende la máquina. Un lenguaje de programación es similar a uno humano; utiliza un conjunto de símbolos, instrucciones (generalmente en inglés) y enunciados que están sujetos a una serie de reglas, se compone de:

- **Léxico**: conjunto de símbolos conocido como vocabulario.
- **Sintaxis**: Reglas para construir el lenguaje.
- **Semántica**: Conjunto de significados de un lenguaje.

Generaciones de los lenguajes de programación

Igual que las computadoras, los lenguajes de programación se han clasificado por generaciones que coinciden de alguna manera, ya que se han desarrollado en forma paralela.

Primera generación

Se remontan a las primeras computadoras integradas por bulbos y relevadores. Los programas se realizaban con base en instrucciones en *lenguaje máquina*, de *bajo nivel*, que utilizaban sólo los símbolos binarios **0** y **1** (código binario).

Segunda generación

Se desarrollaron los lenguajes ensambladores, también de bajo nivel, que traducían al lenguaje máquina las órdenes simples como sumar, restar o almacenar.

Tercera generación

En 1957 se dio un gran paso en la programación de computadoras. **John Backus**, programador de IBM inventa el primer **lenguaje de alto nivel** llamado FORTRAN por las siglas en inglés, ***FOR****mula* ***TRAN****slator*.

Después aparecieron lenguajes como APL, COBOL, ALGOL, LISP, Basic, Pascal, C, C++, C#, Visual Basic, LOGO, Java y muchos otros, algunos de ellos conocidos y utilizados en la actualidad.

Algunos de estos lenguajes producen un **programa objeto** que se ejecuta directamente en la computadora; otros crean sólo un **programa fuente** y requieren del lenguaje de programación para traducir las instrucciones, una por una, al momento de ejecutar el programa. Para traducir las órdenes al lenguaje máquina, utilizan intérpretes y compiladores.

Por la forma en que realizan la traducción del programa fuente a programa objeto o ejecutable, se dividen en:

- **Intérpretes**: leen una orden, la traducen a lenguaje máquina y la ejecutan. Si en el proceso detectan un error de sintaxis, envían un mensaje a la pantalla para que el programador corrija la orden.
- **Compiladores**: primero traducen todo el programa fuente a lenguaje máquina y, una vez compilado, lo ejecutan.

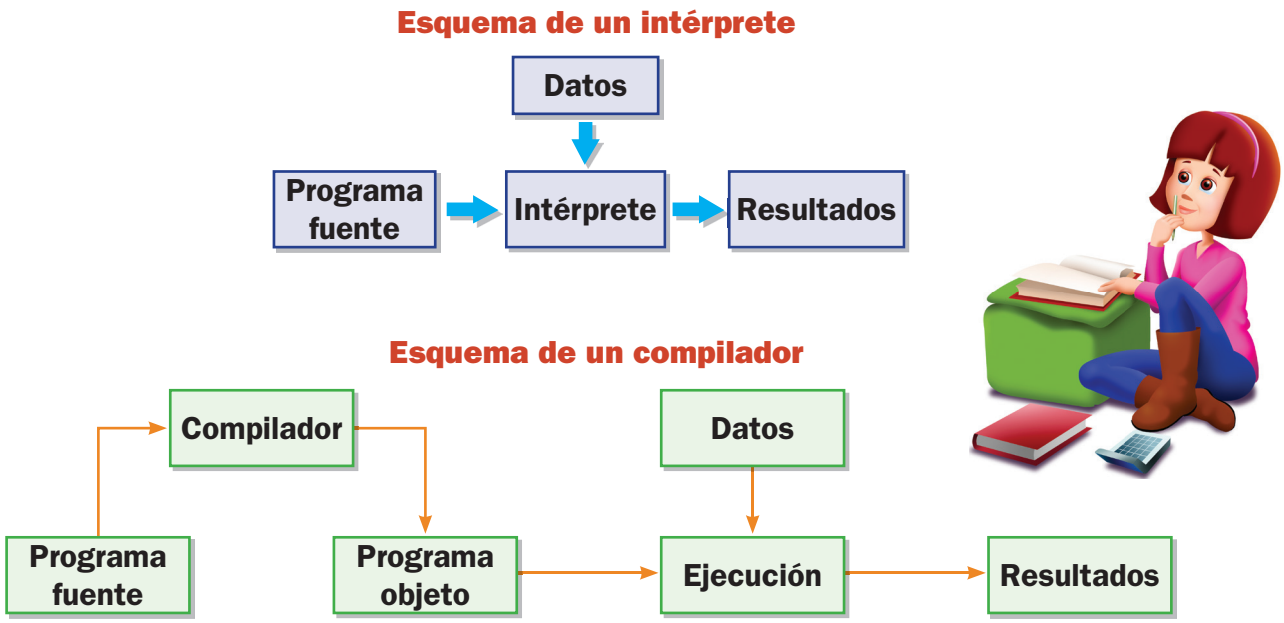


Figura 5.1 Esquemas de los procesos que realizan los intérpretes y los compiladores.

Cuarta generación

Los lenguajes evolucionan hacia las interfaces gráficas, que permiten generar el código relacionando objetos e instrucciones. Se conocen como lenguajes orientados a objetos y gestión de bases de datos como Visual Basic, C++, SQL, Java y otros.

Algoritmo. Secuencia de pasos que permiten resolver un problema.

Programas. Conjuntos de declaraciones e instrucciones que se ejecutan una a una para procesar una serie de datos y obtener un resultado.

Lenguajes de alto nivel. Utilizan órdenes o instrucciones en un lenguaje entendible para los humanos (generalmente en inglés), las cuales deben ser traducidas al lenguaje máquina, que entiende la computadora.

Programa objeto. Programa ejecutable traducido a lenguaje máquina. Generalmente tienen las extensiones **.exe** o **.com**.

Programa fuente. Programa escrito en un lenguaje de alto nivel. Para ejecutarlo requiere de traducir cada instrucción. Su extensión coincide con el nombre del programa; por ejemplo, **.pas** para los de Pascal, **.bas** los de Basic, etcétera.

Quinta generación

Son lenguajes que se emplean para desarrollar aplicaciones de robótica e inteligencia artificial como LISP y PROLOG.

Lee y deduce

Hay de lenguajes a lenguajes

Por el grado de madurez que has alcanzado, sabes que un lenguaje es la expresión de nuestro pensamiento (si no la única, sí la más importante y generalizada), porque a través de él, podemos comunicar lo que queremos, con la certidumbre de que otras personas lo comprenden.

Todos los contenidos en el lenguaje tradicional tienen tres dimensiones de referencia al mundo:

- **Cognitiva-expositiva**: trasmite una situación registrada en cifras.
- **Afectiva-emocional**: expresa algún sentimiento en relación con tales cifras.
- **Evaluativo-valorativa**: califica los significados concretos de una expresión lingüística.

Relaciona esta información de los lenguajes humanos con los lenguajes informáticos, escribe y explica tres características que compartan ambos mundos:

1. _____

2. _____

3. _____

¿Consideras que cualquiera de los dos lenguajes y sus diversas manifestaciones puedan ser individuales?, ¿sí o no? _____

Especifica por qué _____

_____

De la reflexión anterior, ¿qué característica universal definirías para cualquier lenguaje, ya sea humano o de computadora?

_____

_____

_____

Actividad 21. Programación

Actividad 5.1

1. Escribe en siete pasos el algoritmo para crear un párrafo de texto en Word. Considera todos los pasos a seguir, como encender la computadora, etcétera.

a) Encender la computadora

b) _____

c) _____

d) _____

e) _____

f) _____

g) _____

2. Ahora describe el procedimiento para guardar el archivo de Word, cerrar el programa y apagar la computadora, en seis pasos.
   - **a)** _____
   - **b)** _____
   - **c)** _____
   - **d)** _____
   - **e)** _____
   - **f)** _____

¡Lo que estás haciendo es programación!

Actividad 5.2
Actividad grupal

1. Organicen equipos de cuatro alumnos en el salón y realicen lo siguiente:
   - **a)** Cada equipo debe investigar sobre el desarrollo de uno de los siguientes lenguajes de programación: FORTRAN, Basic, Pascal, Visual Basic, Visual C++, Pyhton, Perl y Java.
   - **b)** En un documento de Word, cada grupo debe escribir uno o dos párrafos con lo investigado. El texto debe incluir un título en WordArt y un formato agradable y claro.
   - **c)** Cada grupo debe guardar el documento con el nombre del lenguaje investigado; por ejemplo, *Basic.docx* o *Pascal.docx*, etcétera.
   - **d)** En una presentación de PowerPoint, incluyan cada una de las investigaciones en una diapositiva con los siguientes datos:

Nombre del lenguaje
Año de creación
Ilustración de la persona o grupo que lo desarrolló
Lugar en donde lo crearon
Breve descripción
Si es posible, una línea de programación en el lenguaje descrito

   - **e)** Guarden el documento con el nombre *Lenguajes de programación.pptx*.
   - **f)** Realicen la presentación ante todo el grupo.

5.2 Programación estructurada

La programación estructurada se basa en el método científico y utiliza estructuras de control. Para resolver problemas complejos, se dividen en pequeños módulos y se van resolviendo por partes. La solución de problemas utilizando computadoras involucra también ciertos operadores aritméticos y de relación, diagramas de flujo y un lenguaje natural denominado **seudocódigo**.

Operadores aritméticos

La mayoría de los programas y lenguajes de programación utilizan una notación especial para ejecutar las operaciones aritméticas, que se desarrollan de acuerdo con sus jerarquías. Si tienes dudas respecto a la jerarquía de los operadores, repasa la sección correspondiente en el bloque relativo a Excel.

Seudocódigo. Descripción, en tu idioma, de la solución de un problema paso a paso.

Operador	Operación	Ejemplo	Explicación
( )	Agrupar	(6+3) da 9	Agrupa +6 y +3
^	Potencia	6^3 da 216	6 elevado a la potencia 3
*	Multiplicación	6*3 da 18	6 multiplicado por 3
/	División	6/3 da 2	6 dividido entre 2
+	Suma	6+3 da 9	6 más 3
–	Resta	6-3 da 3	6 menos 3
=	Asignación	A = 3	Se asigna a la letra A el valor de 3

Ejemplo:
Para resolver la **ecuación** x = (6+4)*2+3^2–32/8 se procede de la siguiente manera:

Jerarquía	Operación			Resultado
1ª	Desagrupa	(6+4) = 10	➔	= 10*2+3^2–32/8
2ª	Eleva a la potencia	3^2 = 9	➔	= 10*2+9–32/8
3ª	Multiplica:	10*2 = 20	➔	= 20+9–32/8
3ª	Divide:	32/8 = 4	➔	= 20+9–4
4ª	Suma:	20+9 = 29	➔	= 29–4
4ª	Resta	29–4 = 25	➔	= 25
5ª	Asigna	x = 25	➔	Se asigna a la letra **x** el valor 25

Existe otro tipo de operadores que se utilizan para relacionar valores, llamados operadores de relación.

Símbolo	Se lee
Si A = B	Si A es igual que B
Si A < B	Si A es menor que B
Si A > B	Si A es mayor que B
Si A <= B	Si A es menor o igual que B
Si A >= B	Si A es mayor o igual que B
Si A <> B	Si A es diferente que B

Constantes y variables

Constantes

Los datos o valores constantes que se utilizan en el curso o ejecución de un programa se denominan **constantes**, porque no cambian su valor. En este caso se encuentran constantes como π, que siempre tiene el valor **3.14159265358979323846264338327 95**, abreviado **3.1416** por conveniencia, o el número **3**, que siempre será **3**, mientras no se asigne a una variable.

Variables

Los datos para ser procesados se almacenan en una localidad en la memoria de la computadora. Estos datos reciben el nombre de variables y pueden cambiar su valor durante la ejecución de un programa. Existen variables para todo tipo de datos; por el momento, únicamente las dividiremos en dos:

1. **Variables de texto**. Son aquellas que se emplean para almacenar cadenas de caracteres; no se utilizan en operaciones aritméticas, por ejemplo, nombres de personas o números de teléfono. Nombres válidos para estas variables pueden ser NOMBRE, TEL, DIRECCIÓN, etcétera.
2. **Variables numéricas**. Almacenan números reales, enteros o decimales, positivos o negativos. Los términos que integran la fórmula que soluciona el problema se sustituyen por el nombre de las variables donde se almacenan los datos correspondientes. Por ejemplo, los valores de una fórmula como ÁREA = BASE x ALTURA; las variables pueden ser B y AL, que al variar dan diferentes valores de ÁREA.

Nota: No se pueden utilizar como variables los comandos propios de un lenguaje de programación o del sistema, como PRINT, CLS, GO, ADD, LIST, VER y otros. Los nombres de variables no pueden comenzar con un número, como 3COM, 2PASOS, 5NOMBRE, etcétera.

Algoritmos

Como se mencionó, algoritmo es un conjunto de pasos que deben seguirse para solucionar un problema. Todo algoritmo debe ser:

- **Preciso**. Debe ser concreto, no tener pasos de más y llevar a una solución clara.
- **Congruente**. Al probarse varias veces, los resultados deben ser los mismos.
- **Finito**. Al seguir los pasos deben conducir a la solución.

Los algoritmos se representan mediante:

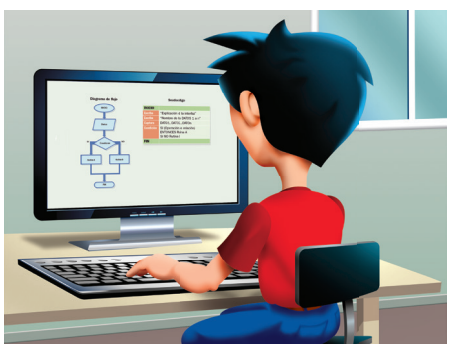
Diagramas de flujo. Representaciones gráficas de la secuencia de pasos para resolver un problema o ejecutar una determinada tarea, mediante símbolos.

Seudocódigo. Descripción en tu idioma, paso a paso, de la solución de un problema o la ejecución de una tarea. Cada orden se expresa en una línea.

Diagramas de flujo

Desde los primeros lenguajes de programación se utilizaron los diagramas de flujo, que ayudan a visualizar de manera clara los procedimientos que se deben seguir para resolver un problema de no muy grandes dimensiones.

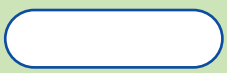
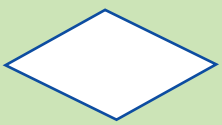
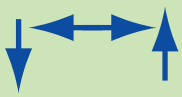
Quienes continúen con una carrera informática y se dediquen a elaborar complejos programas para resolver no uno, sino muchos problemas, verán que los diagramas de flujo ayudan, pero no sirven de mucho. ¡Imagina el diagrama de flujo para hacer un programa como Microsoft Excel!, no cabría en miles de hojas.



Nota: La mejor manera de solucionar los problemas utilizando computadoras, es llevar a cabo una serie de pasos lógicos. La computadora, como elemento físico (chips, tarjetas, cables y circuitos electrónicos) no puede llevar a cabo ninguna tarea, si no cuenta con instrucciones bien definidas.

Ecuación. Igualdad. Expresión algebraica polinomial que oculta una o varias incógnitas.

Tabla 5.1 Símbolos utilizados en los diagramas de flujo.

Símbolo	Explicación
	Se utiliza siempre para dar **inicio** en un diagrama de flujo. También se puede utilizar como inicio el símbolo de **fin** del diagrama.
	Representa la **entrada** de datos, que generalmente se hace mediante el teclado.
	Representa un **proceso**. Dentro del rectángulo se escriben las fórmulas que se van a utilizar en la solución del problema.
	Se utiliza para la **salida de datos**. Los resultados se imprimen o se muestran en la pantalla.
	Indica el **final** de un diagrama de flujo. También se puede usar para el fin, el símbolo de inicio del diagrama.
	Símbolo de **selección**. Se plantea una condición; si se cumple el programa fluye por la derecha, si no, sigue su flujo por la izquierda. También se utiliza para realizar operaciones repetitivas, hasta que se cumpla una condición dada.
	Indican las **direcciones de los flujos** del diagrama.

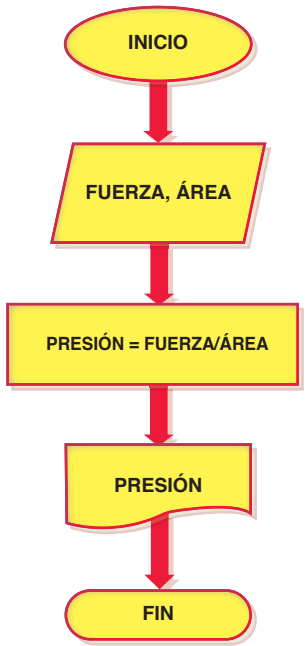
Seudocódigo

El seudocódigo o seudolenguaje se integra por una serie de instrucciones en lenguaje natural como inglés, alemán o español, además de expresiones y operadores que representan cada uno de los pasos para resolver un problema (o muchos). Quienes sigan una carrera profesional como programadores verán la utilidad de usarlo, porque transformar un algoritmo en pseudocódigo, a un lenguaje de programación, es muy sencillo.

Cuando elaboras programas, es conveniente no traducir las estructuras IF-THEN-ELSE, FOR o DOWHILE, que verás más adelante; sin embargo, al escribir pseudocódigo te lo puedes permitir. Para utilizar la programación en la solución de tus tareas escolares de física, matemáticas y de cualquier otra materia, con el seudocódigo en español bastará.

Enseguida, observa el algoritmo para la solución de un problema de cálculo de la presión que ejerce una fuerza sobre una superficie, realizado con un diagrama de flujo y el seudocódigo correspondiente.

Diagrama de flujo



Seudocódigo	
INICIO	
Escribe	"Calcula la presión que ejerce una fuerza sobre un área"
Escribe	"Da la fuerza en newtons"
Captura	FUERZA
Escribe	"Da el área en metros"
Captura	ÁREA
Fórmula	PRESIÓN = FUERZA / ÁREA
Escribe	"La presión en pascales es:"
Imprime	PRESIÓN
FIN	

Lee y deduce

Orígenes de las lenguas

Tú sabes que el idioma español es una lengua romance derivada del latín; también conoces que los árabes fueron maestros de la aritmética y el álgebra, además tuvieron gran influencia sobre España debido a los siglos de dominación que ejercieran sobre ella; es por esto que muchas palabras relativas a la matemática provienen de esa lengua. Seguramente habrás escuchado hablar de **Mohammed ben Musa** (780-850 d.C.), llamado el "padre del álgebra", tenía el seudónimo de **Al Jwarizmi**, término que con el tiempo degeneraría en "algorismo" para convertirse finalmente en "algoritmo".

Si algoritmo son los pasos para solucionar un problema, ¿desde cuándo crees que fueron utilizados?

Describe el algoritmo de alguna tarea que desarrollas cotidianamente:

Finalmente, ¿consideras que la formulación de algoritmos puede facilitarte la realización de alguna tarea? Explica por qué.

Actividad 22. Operadores aritméticos

Actividad 5.3

1. Resuelve las siguientes operaciones considerando la jerarquía de los operadores aritméticos:
   - **a)** 2+(8–3)*2 = _____
   - **b)** (10+4)/7–2*4 = _____
   - **c)** 12–8*2 = _____
   - **d)** 3^2–18/2 = _____
   - **e)** (16–2^3)*4–64/2 = _____

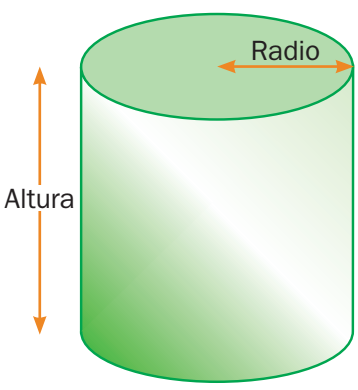
Actividad 5.4

1. Escribe un modelo que calcule el área de cualquier círculo, dado su radio, conforme a lo siguiente:
   - **a)** ¿Qué datos se necesitan conocer?
   - **b)** ¿Qué variables se deben definir?
   - **c)** ¿Cuál es la fórmula algebraica?
   - **d)** Escribe la expresión utilizando nombres de variables.
   - **e)** Dibuja un círculo y escribe a un lado la fórmula del área.

   Modelo:

Actividad 5.5

1. Elabora el algoritmo que calcule el volumen de cualquier cilindro, dados su radio y altura.



Fórmulas:

Área de la base = $\pi \times \text{Radio}^2$

Volumen = Área de la base × Altura

2. Haz una presentación de PowerPoint que incluya lo siguiente:
   - **a)** Diapositiva 1. Portada relativa al tema.
   - **b)** Diapositiva 2. Análisis del problema (copia la figura y las fórmulas).
   - **c)** Diapositiva 3. Elabora la solución en seudocódigo.
   - **d)** Diapositiva 4. Elabora la solución mediante un diagrama de flujo.
3. Guarda el archivo con el nombre *Volumen de un cilindro.pptx*.

Estructuras de programación

Llamadas también estructuras de control, representan la manera de dirigir el flujo de acciones encaminadas a la solución de problemas, de manera ordenada; es decir, bien estructurada. Las estructuras de programación tienen la función de facilitar la comprensión de un problema y evitar los errores en la programación.

Estructuras secuenciales

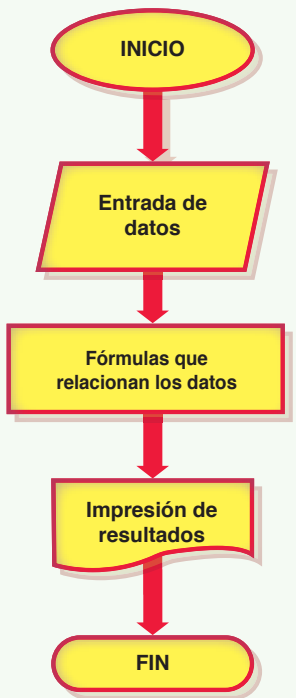
Los programas, cuando están correctamente estructurados, son necesariamente secuenciales. El flujo de procesamiento de los datos se lleva a cabo de manera ordenada y disciplinada. Se van procesando los datos en forma secuencial, paso por paso.

Se aplican en la solución de problemas sencillos con la siguiente secuencia:

1. Captura de datos
2. Proceso de datos
3. Impresión de resultados

Las instrucciones se ejecutan linealmente en el orden en que fueron escritas, como se muestra en el siguiente diagrama de flujo y seudocódigo.

Diagrama de flujo



Seudocódigo	
INICIO	
Escribe	"Explicación de la interfaz"
Escribe	"Nombre de los DATOS 1 a n"
Captura	DATO1, DATO2...DATOn
Fórmula	FÓRMULA
Escribe	"La respuesta es:"
Imprime	RESULTADO
FIN	

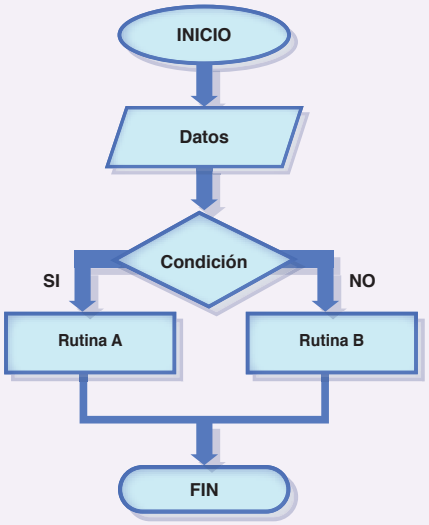
Estructuras de decisión o selección

A veces la estructura secuencial no es suficiente para llevar a cabo el procesamiento desde el principio hasta la obtención de resultados, por lo que es necesario tomar decisiones para saber hacia dónde deberá continuar el proceso del programa.

Se utilizan en la solución de problemas donde se establece una condición. El programa evalúa la condición mediante los operadores lógicos y determina: si es verdadera realiza la rutina A, si no, ejecuta la rutina B.

IF Condición verdadera, THEN Rutina A, ELSE Rutina B

Diagrama de flujo



Seudocódigo	
INICIO	
Escribe	"Explicación de la interfaz"
Escribe	"Nombre de los DATOS 1 a n"
Captura	DATO1, DATO2...DATOn
Condición	SI (Operación de relación) ENTONCES Rutina A SI NO Rutina B
FIN	

Actividad 23. Estructuras secuenciales y de decisión

Actividad 5.6

1. Elabora un diagrama de flujo secuencial que describa el algoritmo que desarrollaste en el apartado **1** de la **Actividad 5.1** (crear un párrafo de texto en Word).
   - **a)** Inicia una sesión de Excel.
   - **b)** En la **Hoja1** dibuja el diagrama de flujo utilizando la barra de herramientas **Dibujo**.
   - **c)** Incluye el título *Cómo crear un párrafo de texto en Word*, con la herramienta WordArt.
   - **d)** Guarda el archivo con el nombre *Párrafo de texto en Word.xlsx*.

Actividad 5.7

1. Elabora en seudocódigo el algoritmo secuencial que desarrollaste en el apartado **2** de la **Actividad 5.1** (guardar el archivo de Word).
   - **a)** Inicia una sesión de Word.
   - **b)** Dibuja el diagrama de flujo utilizando las formas de la cinta de opciones **Insertar**.
   - **c)** Incluye el título *Cómo guardar un archivo de Word*, con la herramienta WordArt.
   - **d)** Guarda el archivo con el nombre *Guardar archivo de Word.docx*.

Actividad 5.8

Elabora un diagrama de flujo que describa gráficamente el siguiente algoritmo, utilizando una estructura de decisión.

1. José consulta todos los días su horario de clases a las 10:00 A.M. para saber si entra a clase de matemáticas o al laboratorio de física.
2. Los lunes, miércoles y viernes tiene matemáticas y los martes y jueves va al laboratorio de física.
3. En la primera diapositiva de una presentación de PowerPoint dibuja el diagrama.
4. Incluye el título *¿A qué clase entro?*, con la herramienta WordArt.
5. Guarda el archivo con el nombre *Decisión _01.pptx*.

Estructura de repetición o iteración

Otras veces se hace necesario repetir en múltiples ocasiones una rutina o secuencia de instrucciones, hasta cumplir con una condición establecida. La condición arroja sólo dos valores como resultado: Verdadero (*True*) o Falso (*False*). Dependiendo de ellos, el flujo del programa se desvía hacia una u otra ramificación o camino.

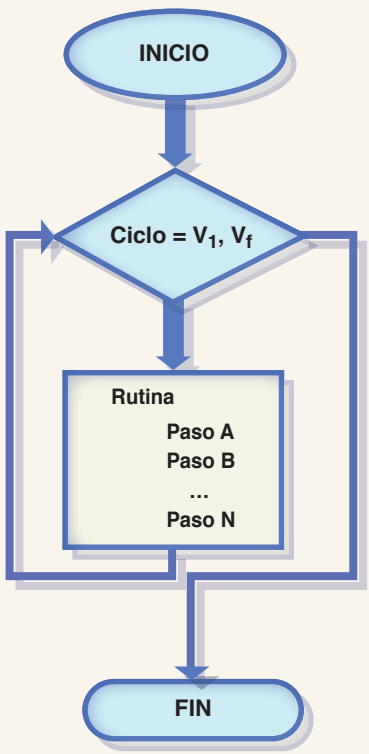
Esta estructura se emplea en la solución de problemas que realizan una misma tarea varias veces, con la salvedad de que en cada **ciclo** utiliza datos diferentes. Se requiere una variable que controle el ciclo. La estructura es:

VC = V_i hasta V_f
.
.
Rutina
.
.
Termina el ciclo

Hay dos tipos de estructuras de repetición y son las siguientes:

Estructuras de Ciclos predefinidos. Cuando se conoce previamente el número de repeticiones que debe realizar la rutina.

Diagrama de flujo

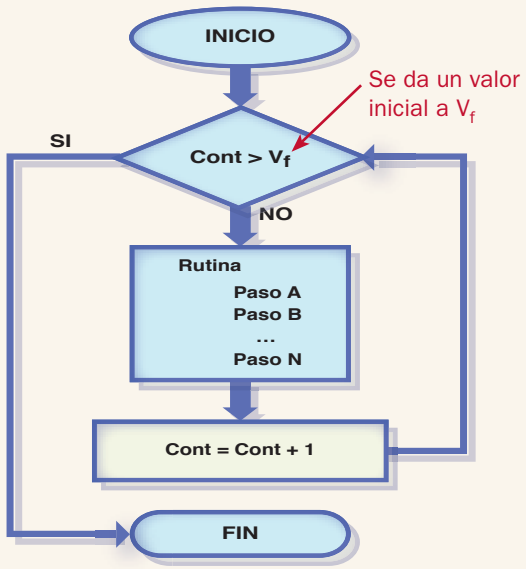


Seudocódigo	
INICIO	
Escribe	"Explicación de la interfaz"
Condición	A = V_i, V_f
Rutina	Paso A Paso B ... Paso N
FIN	

Ciclo. Repetición controlada de un bloque de instrucciones. Es conveniente prever el término de la iteración mediante una condición o comparación de valores para evitar ciclos infinitos.

Estructuras de Ciclos condicionados. Cuando el número de repeticiones depende de una condición que se evalúa en cada ciclo.

Diagrama de flujo



Seudocódigo	
INICIO	
Escribe	"Explicación de la interfaz"
Condición	Cont > V_f
Rutina	Paso A Paso B ... Paso N
Contador	Cont = Cont + 1
	Regresa a evaluar la condición
FIN	

Actividad 24. Estructuras de repetición o iteración

Actividad 5.9

Elabora un diagrama de flujo y su representación en seudocódigo, que describan el siguiente algoritmo utilizando una estructura de repetición.

1. Dados los números 1 al 30, sumar todos aquellos que sean pares e imprimir el resultado.
2. En la primera diapositiva de una presentación de PowerPoint dibuja el diagrama y escribe el programa en seudocódigo.
3. Incluye el título *Suma de números pares* con la herramienta WordArt.
4. Guarda el archivo con el nombre *Suma pares.pptx*.
5. Contesta lo siguiente: ¿de qué tipo de estructura de repetición se trata?

Actividad 5.10

Elabora un diagrama de flujo y su representación en seudocódigo, que describan el siguiente algoritmo utilizando una estructura de repetición.

1. Arroja 20 veces una moneda al aire y ve registrando (imprimiendo) cuántas veces cae la moneda en *cara* y cuántas en *reverso* o *cruz*.
2. Detén el ciclo en el evento número 20.
3. Imprime el número de veces para *cara* y el número para *reverso* y finaliza el programa.
4. Realiza el diagrama y el seudocódigo en un documento de Word con el título *Cara o Cruz*.
5. Guarda el archivo con el nombre *Cara o cruz.docx*.
6. Contesta lo siguiente: ¿de qué tipo de estructura de repetición se trata?

5.3 Solución de problemas con computadora

Existen muchas aplicaciones de desarrollo con las cuales podrías crear tus propios programas y resolver problemas utilizando la computadora. Dependiendo de tu campo de acción, puedes aprender a programar en el lenguaje de programación apropiado.

Metodología para la solución de problemas

Los pasos sistemáticos que debe tener en cuenta el programador para resolver problemas en la computadora son:

- Identificar el problema que se va a resolver
- Analizar el problema
- Diseñar la solución *del problema*
- Trasladar la solución a un lenguaje de programación
- Verificar la solución
- Entregar la interfaz de solución

Nunca olvides éstos sencillos pasos que te serán muy útiles siempre, aunque también deberás poner toda tu inventiva y crear el programa escrito en un lenguaje informático, que dé solución al problema planteado de una manera sencilla y fácil de entender por cualquier usuario de computadoras.

Identificar el problema por resolver

En esta etapa se toma conciencia del problema, al que se le puede asignar un nombre como identificador; se analizan las limitaciones que se tienen y se define la herramienta informática que se va a utilizar para implantar la solución.

Ejemplo:
Tu profesora o profesor de Ciencias te pide como tarea escolar para realizar en casa, un ensayo sobre la estructura de la materia. La identificación de tu problema o tarea a resolver se puede desglosar en los siguientes pasos:

1. ¿Qué problema tengo?
   Respuesta: Buscar información para hacer un ensayo sobre un tema de ciencias.
2. ¿Cómo identifico el problema?
   Respuesta: El nombre puede ser *Tarea 1 de ciencias* o *Estructura de la materia*.
3. ¿Cuáles son mis limitaciones?
   Respuesta: No tengo en casa una enciclopedia, ni una enciclopedia virtual en la computadora.
   Solución: Puedo buscar información sobre el tema en Internet o acudir a una biblioteca.
4. ¿Qué herramienta informática puedo utilizar para resolver el problema?
   Respuesta: Word, Publisher o PowerPoint.

Analizar el problema

Se hace un estudio profundo del problema, se plantean varias posibles soluciones y se opta por la mejor. Algunas veces, para realizar el análisis se parte del final hacia el principio; es decir, primero se identifica el objetivo y después, se determinan los elementos que se requieren para alcanzarlo:

1. Define los datos que debe arrojar la solución del problema.
2. Se identifican todas las variables que participan en el problema.
3. Se crean los modelos que relacionan las variables y constantes del problema (los modelos pueden ser matemáticos).

Ejemplo:
Rutina que realiza la suma de dos números.

1. **Nombre**: *Suma dos números*.
2. **Análisis**:

¿Qué se busca?	Conocer el resultado de la suma de dos números.
¿Qué datos se necesitan?	Número 1 Número 2
¿Qué fórmula algebraica utilizar?	Suma = Número 1 + Número 2
¿Qué variables se deben definir?	NUM1 para guardar el Número 1 NUM2 para guardar el número 2 SUM para guardar el resultado de la suma
¿Cuál es la fórmula informática?	SUM=NUM1+NUM2

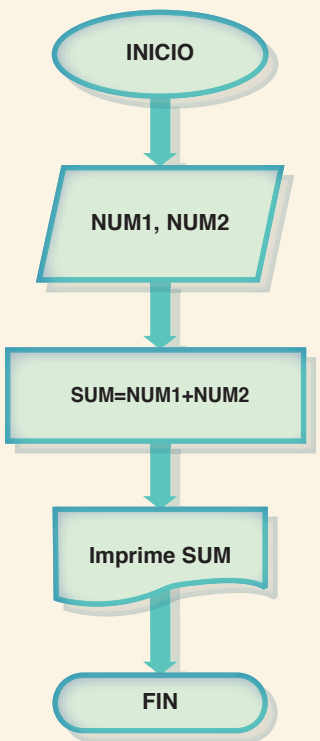
Diseñar la solución del problema

Una vez realizado el análisis del problema, el siguiente paso consiste en diseñar el algoritmo que le dé solución al problema.

Recuerda que los algoritmos se pueden representar con un diagrama de flujo y en seudocódigo.

Ejemplo:
Diagrama de flujo y programa en seudocódigo que resuelven el problema de sumar dos números.

Diagrama de flujo

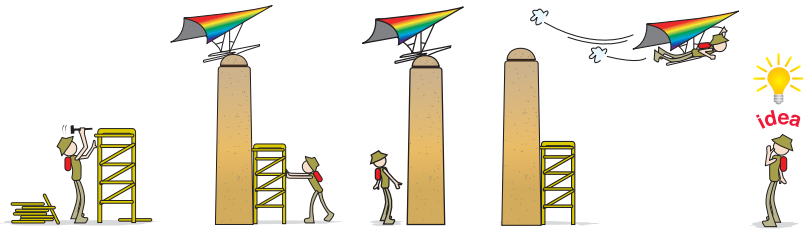


Seudocódigo	
INICIO	
Escribe	"Conocer el resultado de la suma de dos números"
Escribe	"Números a sumar"
Captura	NUM1, NUM2
Fórmula	SUM=NUM1+NUM2
Escribe	"El resultado de la suma es:"
Imprime	SUM

Actividad 5.11

Cómo enfrentar un reto.

1. Analiza los dibujos y relaciona cada viñeta con las palabras de la derecha.



( ) ( ) ( ) ( ) ( )

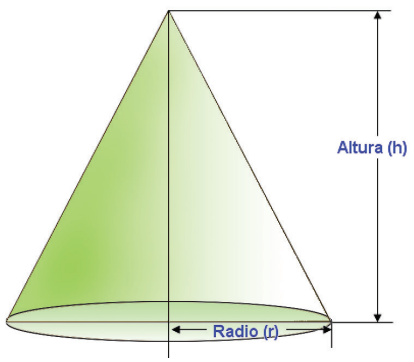
1. Identifica el problema (reto)
2. Analiza el problema
3. Diseña la solución
4. Implementa la solución
5. Logra tu objetivo

Actividad 5.12

1. ¿Qué nombre le darías al problema de resolver el modelo que calcula el volumen de cualquier cono circular recto?

2. ¿Qué herramienta informática, que conozcas, utilizarías para resolver el problema?

Volumen de un cono circular recto



Ecuación:

$$V = \frac{\pi}{3} r^2 h$$

3. Inicia una sesión de Word y escribe los datos de la tabla:

Encabezado	Tus datos personales
Título (WordArt)	Nombre del algoritmo
Cuerpo del texto (Listado con viñetas)	Desarrollo del algoritmo Dibujo y ecuación Diagrama de flujo
Conclusiones	Escribe tus conclusiones

4. Guarda el documento con un nombre apropiado.

Actividad 5.13

1. Realiza el análisis para diseñar el modelo que calcule la velocidad de un automóvil que recorre un trayecto, en un lapso de tiempo.

¿Qué se busca?	
¿Qué datos se necesitan?	
¿Qué fórmula algebraica utilizar?	
¿Qué variables se deben definir?	
¿Cuál es la fórmula informática?	

Actividad 5.14

1. Realiza el análisis para diseñar el modelo que pregunte el nombre de un alumno y calcule su calificación promedio, conociendo los datos de calificaciones de las materias Español, Matemáticas, Ciencias, Tecnología e Historia.

   a) ¿Qué nombre le pondrías al procedimiento?

   _____

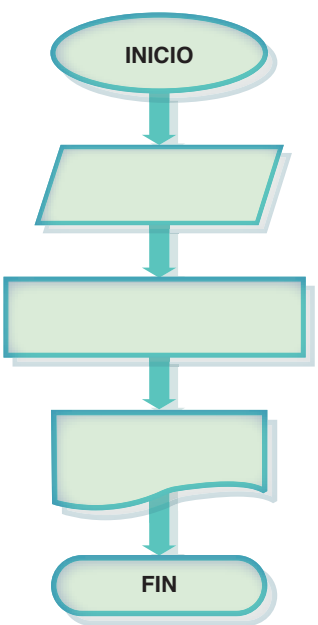
   b) Llena los datos de la tabla

¿Qué se busca?	
¿Qué datos se necesitan?	
¿Qué fórmula algebraica utilizar?	
¿Qué variables se deben definir?	
¿Cuál es la fórmula informática?	

Actividad 5.15

1. Diseña el algoritmo para resolver la Actividad 5.14, dibujando un diagrama de flujo y escribiendo el programa en seudocódigo.

Diagrama de flujo



Seudocódigo	
INICIO	
FIN	

Trasladar la solución a un lenguaje de programación

Se conoce como etapa de programación porque consiste en traducir el algoritmo a un lenguaje informático, respetando su estructura y su sintaxis.

Los programas que se generan con un lenguaje de programación deben tener las siguientes características:

- Ser un conjunto de instrucciones que entiende la computadora, escritas con un lenguaje de programación.
- Deben ser claros y estar documentados para que los entienda cualquier persona.
- Ejecutarse en una computadora.
- Realizan tareas específicas.

Verificar la solución

Es la prueba exhaustiva del programa desarrollado, para eliminar todos los errores que pudiera tener. Los resultados de las pruebas se comparan con soluciones conocidas del problema resuelto.

Ejemplo:
Verifica el algoritmo que calcula el área de cualquier triángulo, probando con datos enteros y positivos:

Prueba	La base mide	La altura mide	Fórmula	Operación	Resultado
1	5	4	= (BASE * ALTURA) / 2	(5 * 4) / 2 =	10
2	2	6	= (BASE * ALTURA) / 2	(2 * 6) / 2 =	6
3	8	8	= (BASE * ALTURA) / 2	(8 * 8) / 2 =	32

Entregar la interfaz de solución

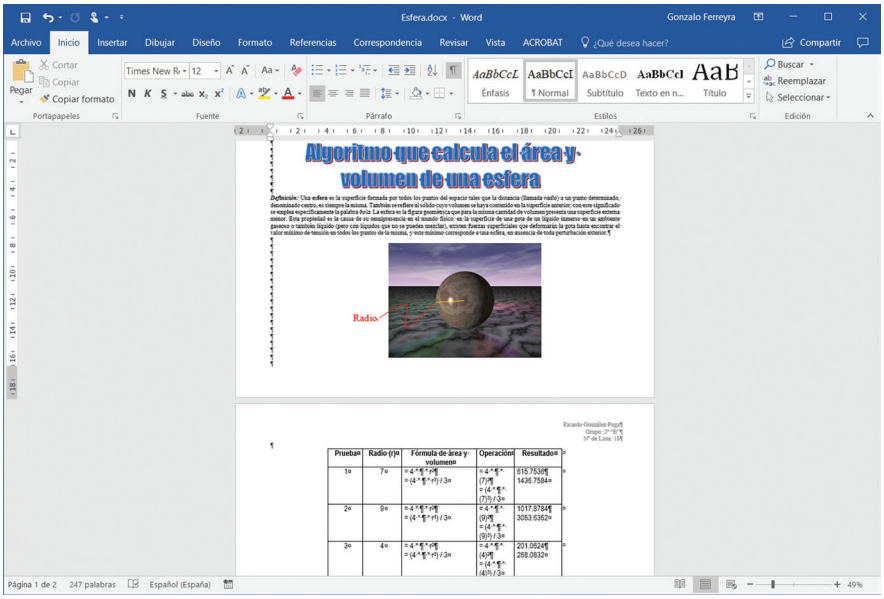
Los programas son realizados por desarrolladores, pero la finalidad es que los utilice cualquier usuario de computadoras para sus funciones específicas. Por eso, una vez que has concluido todo el procedimiento, hay que entregar la interfaz a otros usuarios para ellos sean quienes hagan las pruebas finales.

Actividad 25. Solución de problemas con computadora

Actividad 5.16

Verifica la solución del algoritmo que calcula el área y el volumen de cualquier esfera.

1. Inicia una sesión de Word.
   - En el encabezado escribe tus datos generales.
2. Escribe el título *Algoritmo que calcula el área y volumen de una esfera*.
3. Escribe la definición de esfera.
4. Dibuja una esfera indicando su radio.
   - Inserta una tabla de 4 filas y 5 columnas.
   - Coloca el encabezado de la prueba en la fila 1.
   - Realiza tres pruebas del algoritmo.
5. Da una presentación agradable al documento.
6. Guarda el documento con el nombre *Esfera.docx*.
   - Debes haber obtenido un documento como el de la figura.



Actividad 5.17

1. Realiza un algoritmo que calcule el promedio de calificaciones de un alumno y lo compare con 9.5.
2. Si el promedio es igual o mayor que 9.5, el alumno es apto para participar en las *Olimpiadas Escolares del Conocimiento*.
   a) ¿Qué nombre le pondrías al procedimiento?

   _____
3. Llena los datos de la tabla para el algoritmo propuesto.

¿Qué se busca?	
¿Qué datos se necesitan?	
¿Qué fórmula algebraica utilizar?	
¿Qué variables se deben definir?	
¿Cuál es la fórmula informática?	

4. Por último, dibuja en Word un diagrama de flujo y escribe el programa en seudocódigo.
5. Guarda el archivo con el nombre *Olimpiadas Escolares del Conocimiento.docx*.

Actividad 5.18

Elabora el análisis y diseña el algoritmo que calcule e imprima el cuadrado y el cubo de todos los números enteros que se encuentren dentro de un intervalo definido por el usuario.

1. Inicia una sesión de PowerPoint.
2. En la diapositiva 1 diseña una portada.
3. En la diapositiva 2 realiza el análisis del problema.
4. En la diapositiva 3 escribe el seudocódigo.
5. En la diapositiva 4 dibuja el diagrama de flujo.
6. Da un formato agradable a la presentación.
7. Guarda el archivo con el nombre *Potencia cuadrada y cúbica.pptx*.
8. Presenta ante el grupo la solución que diste al problema.

Actividad 5.19

Realiza el análisis y diseña el algoritmo que calcule la solución de cualquier ecuación de la forma: $Ax + B = C$.

1. Llena los datos de la tabla para el algoritmo anterior.

¿Qué se busca?	
¿Qué datos se necesitan?	
¿Qué fórmula algebraica utilizar?	
¿Qué variables se deben definir?	
¿Cuál es la fórmula informática?	

2. Inicia una sesión de PowerPoint.
3. En la diapositiva 1 diseña una portada.
4. En la diapositiva 2: realiza el análisis.
5. En la diapositiva 3: escribe el seudocódigo.
6. En la diapositiva 4: dibuja el diagrama de flujo.
7. Da un formato agradable a la presentación.
8. Guarda el archivo con el nombre *Solución de ecuaciones.pptx*.
9. Presenta ante el grupo la solución que diste al problema.

Proyecto 5

Programación de robots

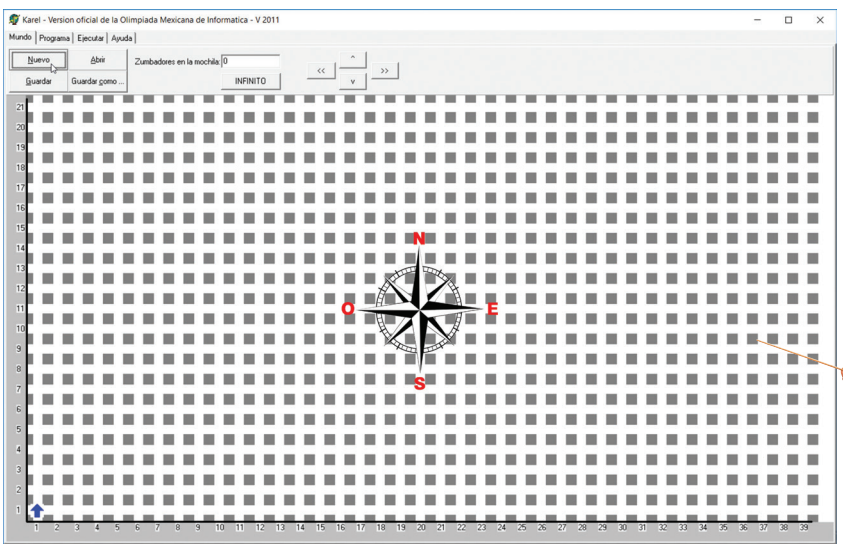
Después de haber estudiado este bloque, uno de los más importantes para tu formación escolar, porque la *programación* aporta competencias, técnicas, habilidades y hábitos que te servirán para toda la vida, programa tu primer robot. Aprender a resolver los problemas que se nos presentan a lo largo de la existencia utilizando un orden establecido, el *algoritmo* adecuado y la *lógica*, nos lleva al desarrollo pleno en todos los ámbitos: en el hogar, en la escuela y en el lugar de trabajo, cuando sea el momento.

El propósito de este proyecto es que comiences a aplicar los conocimientos adquiridos sobre programación, y qué mejor, que solucionar los problemas de un robot llamado **Karel**, que debe encontrar el camino para recoger los *zumbadores* que le escondas.

En el siguiente curso, además de crear robots electrónicos y mecánicos, deberás programarlos mediante otras aplicaciones específicas para eso; pero todo a su tiempo.

1. Karel es un lenguaje de programación desarrollado por el profesor **Richard E. Pattis** del *Departamento de Ciencias de la Computación* de la *Universidad de Carnegie Mellon* en Estados Unidos, en 1981; Karel es el lenguaje que se utiliza en la *Olimpiada Mexicana de Informática* (OMI) a nivel nacional, donde participan jóvenes de secundaria y bachillerato.

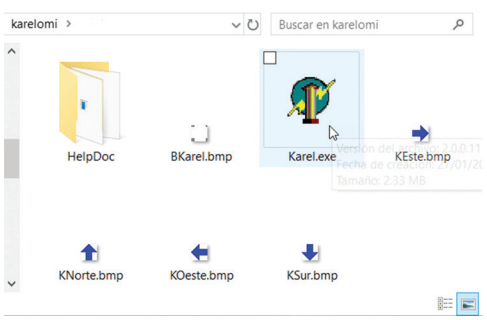
   a) Este sencillo lenguaje de programación te ayudará a aplicar la lógica y tu creatividad, de manera agradable y adictiva ya que, mediante sencillos comandos permite programar al robot **Karel**, que vive en su propio mundo formado por calles y avenidas.



El **Universo de Karel** es una rejilla de vías horizontales (calles) y verticales (avenidas) de 100 x 100, en donde tú creas los mundos por donde circula el robot. El robot se mueve en los sentidos *Norte*, *Sur*, *Este* y *Oeste*.

b) En el material que se descarga de la página web https://www.informaticaactiva3-0.com, busca el archivo *karelomi.zip* y descomprímelo. Abre la carpeta descomprimida y pulsa dos veces sobre el ícono del programa para ejecutarlo.

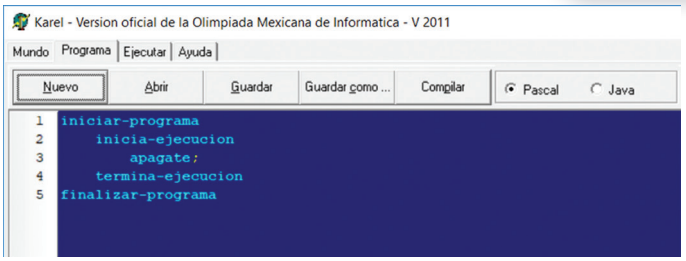
c) Observa que en la carpeta hay otros archivos que debes dejar ahí para que el programa funcione; también hay una carpeta llamada **HelpDoc**, que puedes abrir para ver un tutorial en HTML sobre el uso de Karel. Para abrirlo pulsa dos veces sobre el archivo *Karel.htm*.



d) Karel es una interfaz o "ambiente" de programación que utiliza una reducida cantidad de comandos y las sintaxis de los lenguajes *Pascal* (en español) y *Java* (en inglés). Karel reconoce los siguientes comandos básicos:

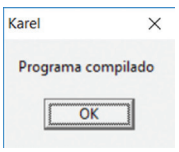
```
avanza
gira-izquierda
coge-zumbador
deja-zumbador
apágate
```

La interfaz del programa es muy amigable, contiene las fichas **Mundo**, **Programa**, **Ejecutar** y **Ayuda**, en las cuales debes pulsar para ver la ventana correspondiente. Pulsa la ficha **Programa**, selecciona la casilla de verificación **Pascal** y presiona el botón **Nuevo**, automáticamente aparece la estructura básica del lenguaje Pascal.

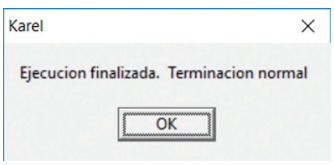


2. Después de la instrucción `inicia-ejecucion`, pulsa las teclas ↵ y luego ⇥; escribe la siguiente secuencia de instrucciones: siete veces `avanza;` ↵, tres veces `gira-izquierda;` ↵, cinco veces `avanza;` ↵, tres veces `gira-izquierda;` ↵, siete veces `avanza;` ↵, tres veces `gira-izquierda;` ↵, y por último cinco veces avanza;. No olvides poner el signo punto y coma (;) al final de cada instrucción. Recuerda que puedes utilizar el atajo **Copiar** y **Pegar** con las teclas Ctrl +C y Ctrl+V.

   a) Al finalizar, pulsa el botón **Compilar**; si escribiste todo correctamente, deberás recibir el mensaje de **Programa compilado**, si no, debes corregir los errores y volver a compilar. Pulsa en el botón **OK**.



   b) Pulsa la ficha **Ejecutar** y luego el botón **Correr**, el robot comienza a moverse hasta el final del programa, si no encontró tropiezos en su camino, debes recibir un mensaje de ejecución finalizada, pulsa en el botón **OK**; en las líneas explica lo que sucedió:



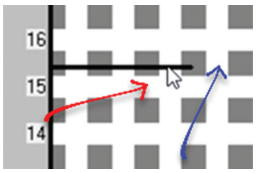
Para volver a ejecutar el programa pulsa en el botón **Inicializar** y nuevamente en **Correr**. Como prueba, ve a la ventana **Programa**, elimina dos instrucciones gira-izquierda; (de las primeras tres) y compila el programa, ejecuta el programa y describe lo que pasó:

Nota: Si deseas que Karel comience a caminar hacia la derecha, debes incluir al inicio tres instrucciones `gira-izquierda;` para que quede apuntando hacia la derecha, ya que no existe la instrucción `gira-derecha`. Luego, ya puedes continuar con instrucciones `avanza;`. Cada instrucción `avanza;` hace caminar a Karel sólo un espacio por lo que, si deseas que camine 20 calles, debes insertar 20 veces la instrucción. El robot comienza el recorrido desde la posición (**1**, **1**) ejecutando las instrucciones de arriba hacia abajo, paso a paso.

Cada programa necesita un **mundo**; si no lo creas, el robot se desplaza por todo el **Universo**. Los mundos sirven para delimitar el espacio de trabajo, insertar paredes, recoger y dejar zumbadores, y hasta para determinar el punto de partida y la dirección inicial de Karel. Guarda el programa creado con el botón **Guardar como**; en el cuadro de diálogo **Guardar como** selecciona la carpeta para tus programas, escribe un nombre como *Vuelta a casa.txt* y pulsa el botón **Guardar**.

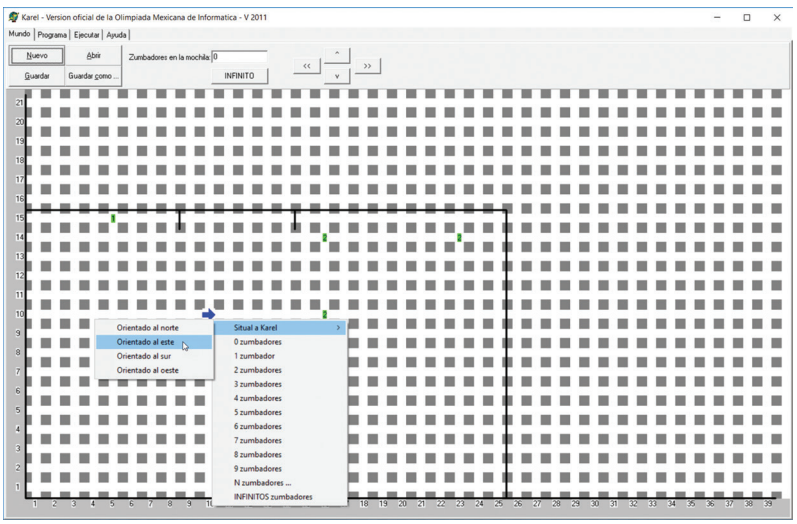
Para crear mundos de Karel, puedes limitar el área (aunque no es una condición necesaria), poner paredes y regar zumbadores, para que el robot los recoja en su camino. Hazlo de la siguiente manera:

1. Pulsa en la ficha **Mundo**.
   - Para poner muros o paredes horizontales o verticales, pulsa con el botón izquierdo entre dos "manzanas" (los cuadritos grises).
   - Para continuar el muro horizontal, pulsa donde indica la flecha **azul** en la figura; si quisieras un muro vertical, deberías pulsar donde indica la flecha **roja**. Si ubicas una pared donde no la querías, no hay problema; vuelve a pulsar entre los cuadritos y desaparece.



2. En este ejercicio se ha limitado el mundo a la calle **15** y la avenida **25**, y se han puesto dos muros para bloquear la calle **15**.
   - Pulsa el botón derecho del ratón en el crucero (**10**, **10**), para fijar la posición de inicio de Karel, selecciona **Situar a Karel** (aunque el comando tiene un error ortográfico), y luego, **Orientado al este**.
   - Pon dos zumbadores en la posición (**10**, **16**), dos en (**14**, **16**), otros dos en (**14**, **23**) y uno más en (**15**, **5**), para que Karel los recoja a su paso. Pulsa con el botón derecho del ratón en la intersección, y en el menú contextual selecciona **1 zumbador**, o **2 zumbadores**.
3. Al escribir el código del programa, cambia entre las vistas **Mundo** y **Programas** para ir describiendo los movimientos que debe hacer el robot para lograr su objetivo, o traza la ruta en una hoja de papel y escribe luego el código.

Al terminar el código, compila el programa antes de ejecutarlo. Observa que Karel recoge los zumbadores y en el campo **Zumbadores en la mochila** se va anotando el número de zumbadores recogidos. Guarda el programa y el mundo con nombres apropiados como *Recoger 7 Zumbadores.MDO* y *Recoger 7 Zumbadores.TXT*.



El código completo del programa es el siguiente:

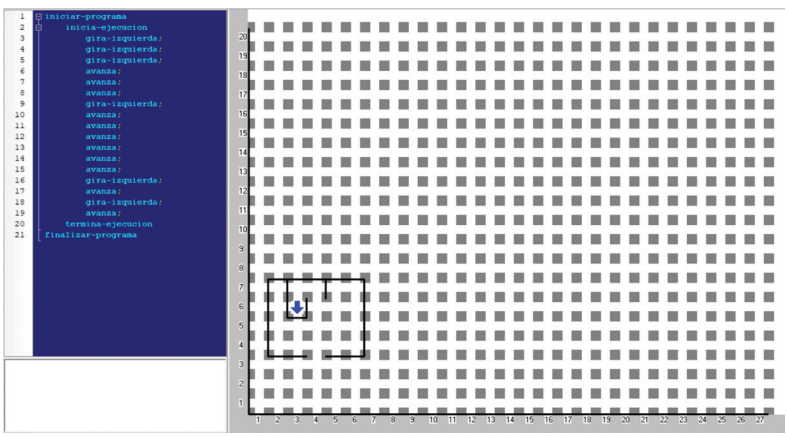
```
iniciar-programa
inicia-ejecucion
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
coge-zumbador;
coge-zumbador;
gira-izquierda;
avanza;
avanza;
avanza;
avanza;

coge-zumbador;
coge-zumbador;
gira-izquierda;
gira-izquierda;
gira-izquierda;
avanza;
avanza;
avanza;
avanza;
avanza;
coge-zumbador;
coge-zumbador;
gira-izquierda;

avanza;
gira-izquierda;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
gira-izquierda;
avanza;
avanza;
gira-izquierda;
gira-izquierda;
gira-izquierda;

avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
avanza;
gira-izquierda;
gira-izquierda;
gira-izquierda;
avanza;
gira-izquierda;
avanza;
avanza;
avanza;
```

```
coge-zumbador;
gira-izquierda;
avanza;
avanza;
avanza;
avanza;
avanza;
gira-izquierda;
avanza;
avanza;
avanza;
avanza;
avanza;
apagate;
termina-ejecucion
finalizar-programa
```

Tienes que escribir **76** líneas de código para lograr que Karel realice su recorrido y junte **7** zumbadores en la mochila, para terminar en el lugar (**10**, **10**), que es de donde partió. Hay maneras de optimizar el código para reducir el número de instrucciones, pero por el momento, ingéniatelas para copiar y pegar instrucciones y no escribir todo a mano.

Lo más interesante de este tipo de programación es que puedes "inventar" tus propios escenarios, crear tus propias historias e imponer los retos que se te ocurran:

3. Karel, aunque es un robot, también tiene necesidades fisiológicas. Debe ir al taller a cargar sus baterías, y a lubricar sus articulaciones. El sitio de inicio natural es la posición (**1**, **1**), mirando hacia el norte; desde ahí debe avanzar hacia la puerta del taller, entrar, y llegar hasta el departamento de servicio. Observa en el código, que para ir a la derecha Karel debe girar tres veces a la izquierda.



a) Pulsa en la ficha **Mundo** y construye el mundo de Karel poniendo paredes para formar el taller.

b) Dentro del taller, diseña el departamento de servicio, como se muestra en la figura. Debe tener la puerta abierta para que entre y salga el robot.

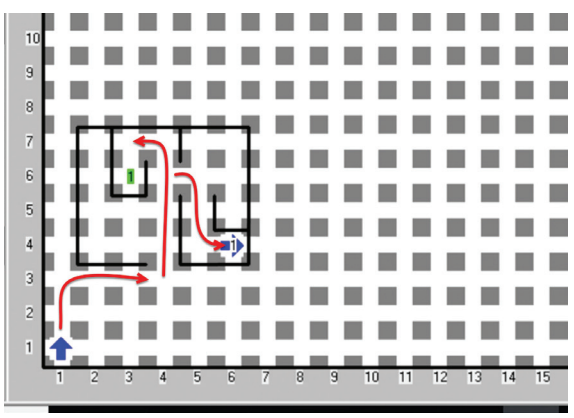
c) Pulsa en la ficha **Programas** y escribe el código iniciando con las tres instrucciones para girar a la izquierda. Luego escribe instrucciones para dirigir a Karel hasta el departamento de servicio. Ten cuidado de que el robot no choque con las paredes.

d) Compila el programa y pulsa en la ficha **Ejecutar**. Al ejecutar el programa, puedes modificar el valor del **Retardo de ejecución**. Si disminuyes el valor, haces más rápido el recorrido. Si todo salió bien, obtienes el mensaje de **Terminación normal**.

Programar, además de despertar el interés por la resolución de situaciones que parecen complicadas, es una actividad muy divertida. Además, es muy satisfactorio ver funcionando a la computadora de la manera en que tú lo planeaste. Así, puedes comprender mejor que la computadora no es una máquina inteligente, es una herramienta que sirve para que chicos inteligentes como tú la aprovechen para aprender más y mejor.

Crea el mundo y el programa para resolver lo siguiente:

4. La mamá de Karel le pide que vaya a la recámara de su hermano, recoja el control de la TV y lo lleve a la sala de televisión.

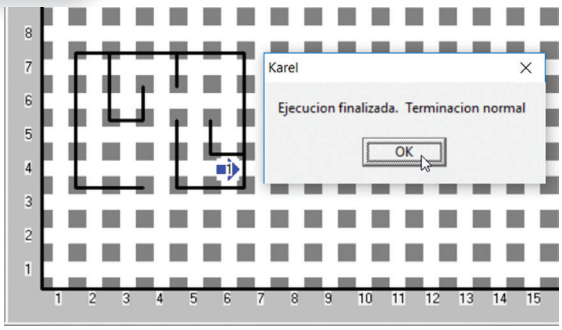


a) La recámara del hermano de Karel se encuentra en la ubicación (**6**, **3**) y la sala en (**4**, **6**).

b) El robot debe caminar desde su ubicación inicial (**1**, **1**), tomar el zumbador (control de TV) y dejarlo en la sala.

c) Escribe el código con las menores líneas que puedas.

d) Además, su mamá le pide que vaya rápidamente. Ejecuta el programa y configura el **Retardo de ejecución** para que Karel corra por el control.

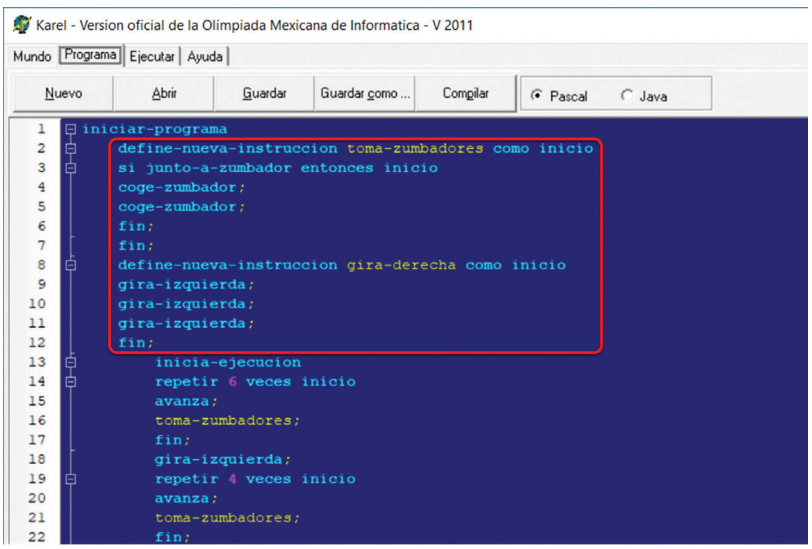


¿Misión cumplida! Gracias Karel, dice su mamá.

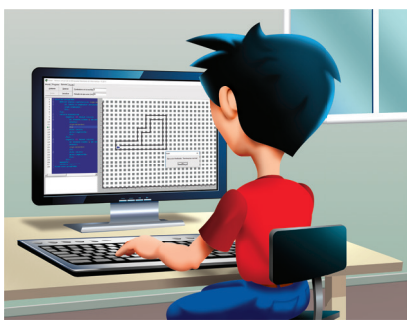
e) Guarda el mundo y el programa con los nombres *Control TV*.MDO y *Control TV.TXT*.

Para optimizar el código, define nuevas instrucciones después de la sentencia `iniciar-programa`; la única condición es que estén referidas a las cinco que reconoce el programa.

Con estas nuevas instrucciones (enmarcadas en la figura), se optimizó el código del programa *Recoger 7 Zumbadores.TXT*, de **76** a **57** líneas. Luego debes escribir el código, después de la sentencia `inicia-ejecución`, utilizando las nuevas instrucciones definidas:

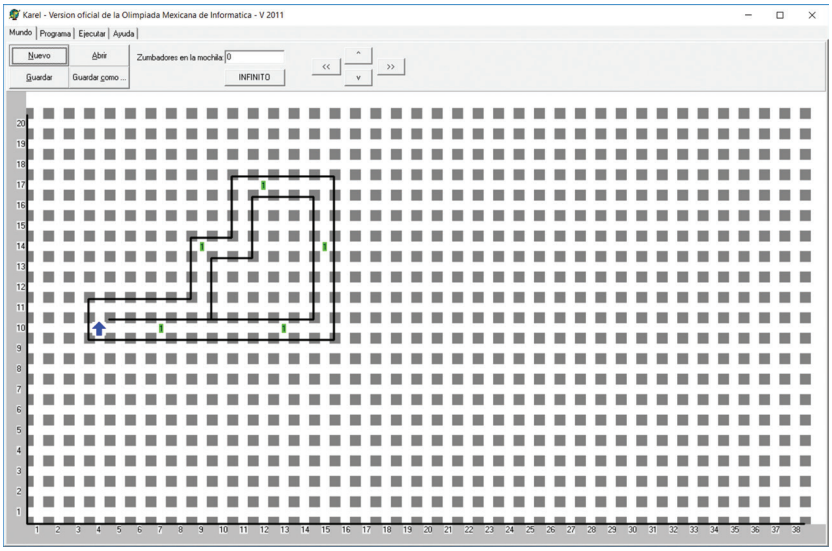



```
inicia-ejecucion
repetir 6 veces inicio
avanza;
toma-zumbadores;
fin;
gira-izquierda;
repetir 4 veces inicio
avanza;
toma-zumbadores;
fin;
gira-derecha;
repetir 7 veces inicio
avanza;
toma-zumbadores;
fin;
gira-izquierda;
avanza;
gira-izquierda;
repetir 8 veces inicio
avanza;
fin;
gira-izquierda;
avanza;
gira-derecha;
repetir 7 veces inicio
avanza;
fin;
gira-derecha;
avanza;
gira-izquierda;
repetir 3 veces inicio
avanza;
fin;
coge-zumbador;
gira-izquierda;
repetir 5 veces inicio
avanza;
fin;
gira-izquierda;
repetir 5 veces inicio
avanza;
fin;
apagate;
termina-ejecucion
finalizar-programa
```



5. Envía a Karel a recorrer los pasillos de un castillo, y haz que recoja todas las monedas que encuentre. Este código te puede llevar una gran cantidad de líneas, pero si optimizas algunos pasos, lo haces más sencillo.

a) Diseña los pasillos del castillo y "siembra" algunos zumbadores en varias partes, para que Karel los encuentre a su paso, como se muestra en la figura siguiente.



b) Define, por ejemplo, una nueva instrucción `gira-derecha`, que en un solo paso realice los tres giros a la izquierda, como se muestra en la figura del código.

c) También puedes utilizar sentencias de control como las que aprendiste a hacer en la programación estructurada, como `si junto-a-zumbador entonces inicio:`

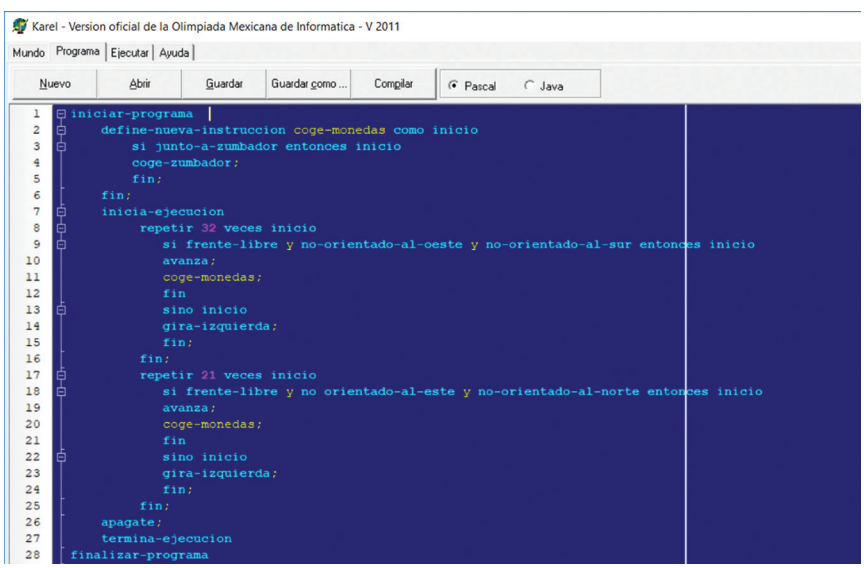
`Si/Entonces` `repite/Nº de veces`

`Si/Entonces/Sino` `Mientras/Hacer`

d) Guarda el mundo y el programa con los nombres *El recolector.MDO* y *El recolector.TXT*.

6. Para conocer el uso de las sentencias, pulsa en la ficha **Ayuda**; en la ventana que aparece pulsa el vínculo *TUTORIAL DE KAREL* y selecciona el tema *Sentencias de Control de Karel*, que no son otra cosa que estructuras de control.

a) Cuando entiendas el uso de las sentencias de control, podrás optimizar el código de cualquier programa, como el siguiente código que, en pocas líneas, recolecta las monedas de los pasillos del castillo.




```
iniciar-programa
    define-nueva-instruccion coge-monedas como inicio
        si junto-a-zumbador entonces inicio
        coge-zumbador;
        fin;
    fin;
    inicia-ejecucion
         repetir 32 veces inicio
            si frente-libre y no-orientado-al-oeste y no-orientado-al-sur entonces inicio
            avanza;
            coge-monedas;
            fin
            sino inicio
            gira-izquierda;
            fin;
         fin;
         repetir 21 veces inicio
            si frente-libre y no orientado-al-este y no-orientado-al-norte entonces inicio
            avanza;
            coge-monedas;
            fin
            sino inicio
            gira-izquierda;
            fin;
         fin;
    apagate;
    termina-ejecucion
finalizar-programa
```

No olvides compilar los programas cada vez que abres uno o realizas modificaciones, antes de ejecutarlos. Sigue trabajando duro y podrás programar tus próximos robots.

5.4 Reafirmación del aprendizaje

1. Define con tus propias palabras lo que es programar.

2. Describe brevemente lo que es un algoritmo.

3. ¿Cuáles son las tres principales reglas a las que se sujetan los lenguajes de programación?

4. ¿Cuántas generaciones de lenguajes de programación se conocen hasta hoy?

5. Realiza las siguientes operaciones, tomando en cuenta la jerarquía de los operadores aritméticos.

   7+(4*3)-6= _____

   (6+3)*4/2= _____

   8/2*12-6^2= _____

   (8–4/2)*(5^2*2) _____

6. Di que función realizan los siguientes operadores de relación:

Operador	Función
=	
>=	
< >	
<	
<=	

7. En un documento de Word elabora un algoritmo con un diagrama de flujo y seudocódigo con los pasos lógicos que seguirías para cambiar un neumático a un automóvil. Guarda el documento con el nombre *Cambiar neumático.docx*.

8. En una hoja cuadriculada verifica la siguiente expresión: $y = 2x + 4$. Realiza las pruebas para los valores –5, 0 y 5 de x y escribe los resultados de y.

9. Analiza la expresión $y = 2x - 8$ y crea un algoritmo que determine para qué valores enteros de x dentro del intervalo $0 \geq x \geq 10$, y tiene un valor positivo.

10. Crea un algoritmo que permita convertir grados Fahrenheit a Celsius. Crea una presentación de PowerPoint, en una diapositiva dibuja el diagrama de flujo y en otra escribe el programa en seudocódigo. Guarda el archivo con el nombre *Conversión de grados.pptx*.