INFORMÁTICA

noveno grado

INFORMÁTICA

noveno grado

M. Sc. César Labañino Rizzo †
M. Sc. Armando Abella Agüero
Lic. Susana Oceguera Martínez

EDITORIAL
PUEBLO Y EDUCACIÓN

Este material forma parte del conjunto de trabajos dirigidos al Tercer Perfeccionamiento Continuo del Sistema Nacional de la Educación General. En su elaboración participaron maestros, metodólogos y especialistas a partir de concepciones teóricas y metodológicas precedentes, adecuadas y enriquecidas en correspondencia con el fin y los objetivos propios de cada nivel educativo, de las exigencias de la sociedad cubana actual y sus perspectivas.

Ha sido revisado por la subcomisión responsable de la asignatura perteneciente a la Comisión Nacional Permanente para la revisión de planes, programas y textos de estudio del Instituto Central de Ciencias Pedagógicas del Ministerio de Educación.

Edición y corrección:
• Lic. Mavis Valdés Pompa

Diseño, cubierta, ilustración y emplane:
• Instituto Superior de Diseño (ISDi):
Aitana Acosta Lechuga • Naomi Casellas González • Danay Cruz Bello • Carolina de Cordova Villegas • Leonardo De León Ramos • Laura Domínguez Machín • Adriana Flórez González • Gabriela Marrero Hernández • Mailen Mulet Segura • Dayanis Placeres Díaz • Liz Rashell Roque Martínez • Alejandra Vázquez Martínez • María Paula Lista Jorge • M. Sc. Maité Fundora Iglesias • Dr. C. Ernesto Fernández Sánchez

ISBN 978-959-13-5166-1 (Versión impresa)
ISBN 978-959-13-5180-7 (Versión digital)

EDITORIAL PUEBLO Y EDUCACIÓN
Av. 3.ª A, No. 4601, entre 46 y 60,
Playa, La Habana, Cuba. CP 11300.
epueblo@epe.gemined.cu

ÍNDICE

Prólogo

Estimado educando, ponemos en tus manos este libro de texto de Informática con el objetivo de que poseas un medio de enseñanza que aborde los contenidos del noveno grado en una asignatura que como sabes, es de suma importancia, tanto para el desarrollo de nuestro país, como para el tuyo personal como futuro ciudadano.

El libro contiene tres capítulos: Conceptos básicos de programación, Elementos de lógica de programación y Nociones de ciberseguridad.

El capítulo 1 Conceptos básicos de programación, trabaja las invariantes conceptuales de estas temáticas preparando al educando para el estudio posterior de cualquier lenguaje de programación. El capítulo 2 entra en las formas estándares de representación de algoritmos, lo cual se logra mediante ejercicios y ejemplos concebidos con una dosificación ascendente, para finalmente pasar al proceso de implementación desde un lenguaje de programación concreto. El capítulo 3 abordará de manera accesible y educativa los principios fundamentales de la ciberseguridad, con el objetivo de brindarles las herramientas necesarias para navegar de manera segura en internet y empoderarlos para que se conviertan en usuarios responsables y conscientes en el mundo digital en constante evolución.

Esperamos que hagas un uso óptimo de este medio de enseñanza y te sirva de estímulo para que des pasos firmes en este apasionante mundo de la Informática.

Los autores

CAPÍTULO 1

Conceptos básicos de programación

Para aprender a programar es necesario tener claro ciertos conceptos generales y específicos de todo lo que envuelve el proceso de la programación. En este capítulo se trata de dar una visión general de la programación y a la vez exponer los conceptos clave para la resolución de problemas por medio de cualquier dispositivo informático.

1.1 Algoritmos

¿Has programado alguna vez? La pregunta te podrá parecer un tanto extraña, sin embargo, es importante que sepas que sí, que tú programas y de manera permanente. Imagínate situaciones como estas:

Ejemplo 1: todas las mañanas de todos los días de la semana, excepto sábados y domingos: me levanto, me cepillo los dientes, desayuno y me voy para la escuela.

Ejemplo 2: te levantas un domingo con la idea de ir a la playa y piensas: *si me asomo a la ventana y está lloviendo, entonces me quedo en casa a estudiar; si no, tomaré el autobús y me encontraré con mis amigos para irnos a la playa.*

Ejemplo 3: para ir a la escuela tengo que cruzar una calle, al llegar al borde de la acera miro y digo: *mientras vengan carros, me detengo, de lo contrario cruzo la calle.*

Como ves, situaciones como las aquí enunciadas podrías enumerar cientos, día tras día tenemos que tomar decisiones como las mencionadas en los ejemplos 1, 2 y 3. Estos ejemplos están muy cerca de lo que se denomina ***algoritmo*** y estos constituyen el concepto básico de la programación de computadoras o sistemas informáticos.

En la vida cotidiana se emplean algoritmos para resolver problemas. Algunos ejemplos pueden ser:

- Las indicaciones que aparecen en manuales de aparatos o dispositivos y que usamos generalmente al comprarlos.
- Algunos ejemplos en Matemática son el algoritmo de multiplicación, para calcular el producto, el algoritmo de la división para calcular el cociente de dos números, el algoritmo de Euclides para obtener el máximo común divisor de dos enteros positivos, el método de Gauss para resolver un sistema lineal de ecuaciones o el procedimiento que empleamos para saber si un número es primo.
- Balanceo de una fórmula química.
- Cálculo de la trayectoria de un proyectil con movimiento uniformemente variado.
- Determinación de la acentuación gráfica de palabras agudas, llanas y esdrújulas.
- Identificación del período histórico a que pertenece un hecho a partir de la fecha en que ocurre.

Definición

Algoritmo: es una secuencia de pasos precisos que conducen a la solución de un problema.

Los algoritmos deben cumplir con las siguientes reglas o propiedades:

1. Deben ser finitos, o sea, tener una cantidad bien definida de pasos.
2. Los pasos deben estar ordenados.
3. Los pasos deben ser claros e igualmente comprensibles para cualquier persona (libres de ambigüedad).

Reflexiona

Analicemos el siguiente procedimiento y determinemos si es un algoritmo o no: mientras 5 = 2 + 3, gire y mire hacia la izquierda.

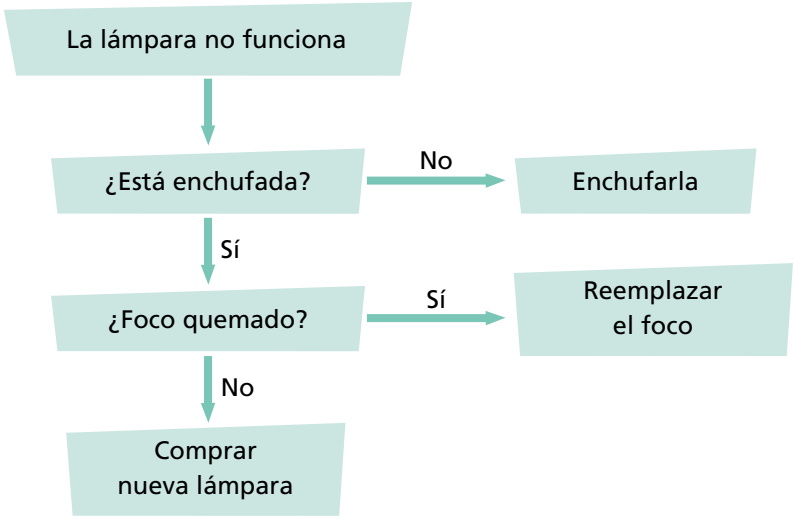
El análisis nos lleva a la siguiente conclusión: el procedimiento enunciado no es un algoritmo. En primer lugar, si nos preguntamos ¿cuántas veces se ejecutan las acciones que se enuncian?, la respuesta sería que infinitas

veces, fíjate que dice que lo que se plantea se debe realizar tantas veces como 5 sea igual a 2 + 3 y como sabemos esto será así siempre, o sea, que nunca se detendría el procedimiento, es decir, un procedimiento sin fin o infinito y esto rompe con la regla número uno enunciada. Por otro lado, existe en el procedimiento una acción que dice "gira", pero no dice hacia dónde, por lo que unas personas lo harían de una forma y otras de otra; esto es lo que se denomina una acción ambigua, no precisa, no clara y entonces estaríamos violando la regla número tres.

1.2 Formas de expresión de algoritmos

Los algoritmos generalmente pueden ser expresados de diferente manera:

1. Mediante el lenguaje natural (cuando se describe mediante palabras que usamos comúnmente, al comunicarnos en español, inglés, ruso o cualquier otro idioma).
2. Mediante diagramas de flujo (muy parecidos a los organigramas que estudiaste en octavo grado) (fig. 1.1).
3. Mediante un sistema escogido del lenguaje natural, mezclado con algunos símbolos matemáticos que se denomina ***pseudocódigo*** (código falso).
4. Mediante el empleo de lenguajes de programación.



Los diagramas de flujo sirven para representar algoritmos de manera gráfica.

Fig.1.1 Diagrama de flujo

En séptimo grado, al estudiar el concepto de ***software***, entendimos que su concepto se dividía en tres categorías:

1. *Software* básico (sistema operativo, manipuladores, BIOS, etcétera).
2. Sistemas de aplicaciones (procesadores de texto, graficadores, hojas de cálculo, sistemas de gestión de bases de datos, video juegos, *software* educativo, etcétera).
3. Lenguajes de programación.

En séptimo y octavo grados estudiaste ***software*** del tipo uno y dos, corresponde entonces en noveno grado estudiar los ***softwares*** de tipo tres, o sea, los lenguajes de programación.

Definición

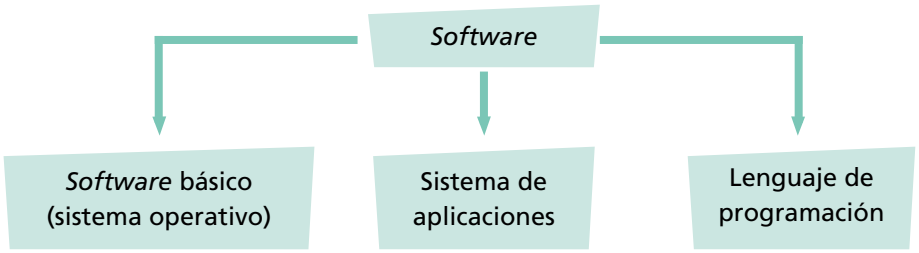
Programa informático: un programa informático, o simplemente programa, es un algoritmo que se escribe de manera tal que resulte comprensible para un sistema informático (una computadora, un teléfono inteligente, una tableta, etcétera).

1.3 Lenguajes de programación

En el mundo actual, dominado por la tecnología, el aprendizaje de lenguajes de programación se ha convertido en una habilidad esencial, no solo para los profesionales de la informática, sino para cualquier persona que desee comprender y participar activamente en la sociedad digital. Para los educandos de noveno grado, el estudio de los lenguajes de programación representa una oportunidad única para desarrollar el pensamiento lógico, la creatividad y la capacidad de resolver problemas de manera estructurada. Este epígrafe explora los conceptos básicos de los lenguajes de programación, su evolución histórica y su importancia en la formación de los jóvenes, preparándolos para enfrentar los desafíos del futuro con herramientas que les permitan no solo consumir tecnología, sino también crearla. A través de ejemplos prácticos y enfoques didácticos, se busca fomentar el interés y la comprensión de los educandos, sentando las bases para un aprendizaje continuo en el fascinante mundo de la programación.

Los programas informáticos se escriben mediante lo que se denominan lenguajes de programación.

Esquema 1.1 Lenguajes de programación



Tipos de lenguajes de programación

Lenguaje de máquina o lenguaje binario

Es el lenguaje de programación que comprende de manera directa los sistemas informáticos. Está compuesto por un alfabeto de dos símbolos 0 y 1 (cero y uno), formando cadenas binarias que conforman las instrucciones que la Unidad Central de Procesamiento (CPU) realiza.

Como es de suponer, resulta extremadamente difícil programar en este lenguaje propio solo de máquinas, como su nombre indica, por tal motivo han surgido otros lenguajes más fáciles de comprender para el hombre.

Lenguaje de bajo nivel o lenguaje ensamblador

Fue el primer intento de facilitar la escritura de programas acercando un tanto la manera de escribir las instrucciones con abreviaturas del lenguaje natural como son: ADD, MOV, XCHG, LD, etc. Algo característico del lenguaje ensamblador es que es particular para cada microprocesador, o sea, que es dependiente del dispositivo con que se trabaje. De cualquier manera, lo que se escribe en ensamblador deberá ser traducido al lenguaje de máquina.

Lenguaje de alto nivel

Son los más parecidos al lenguaje natural, en especial, al idioma inglés, y son independientes del dispositivo, o sea, que sirven para programar cualquier sistema informático solo en dependencia del sistema operativo con que se trabaje y algunos son multiplataforma, es decir, que funcionan con cualquier sistema operativo.

Como se ha explicado, en realidad el único lenguaje que verdaderamente comprenden los sistemas informáticos es el lenguaje binario, ese que solamente

está formado por 0 y 1, por lo tanto, en cualquiera de los casos se culmina realizando una traducción hacia 0 y 1 (cero y uno).

Tipos de lenguajes según la forma de traducción que emplean

En el mundo de la programación, los lenguajes de programación no solo se diferencian por su sintaxis o su nivel de abstracción, sino también por la forma en que son traducidos a un lenguaje que la computadora puede entender y ejecutar. Este proceso de traducción es fundamental, ya que determina cómo se ejecutan los programas y cómo interactúan con el *hardware.* En este epígrafe exploraremos los diferentes tipos de lenguajes de programación según la forma en que realizan esta traducción, ya sea a través de compilación, interpretación o una combinación de ambos. Comprender estas diferencias no solo amplía el conocimiento técnico, sino que también les permite elegir el lenguaje más adecuado para cada tipo de proyecto, optimizando el rendimiento y la eficiencia de sus programas. Mediante ejemplos y explicaciones claras, se busca desmitificar este proceso y mostrar su relevancia en el desarrollo de *software.*

A continuación damos a conocer dos tipos de lenguajes:

- Intérpretes
- Compiladores

Como hemos dicho, el único lenguaje que comprenden los sistemas informáticos de manera directa es el llamado lenguaje de máquina. Esto significa que cualquier otra variante tiene que ser traducida a este lenguaje.

Pues bien, en dependencia de la manera en que se realiza la traducción los lenguajes pueden ser intérpretes o compiladores.

Definición

Intérpretes: los lenguajes intérpretes realizan la traducción de manera simultánea, o sea, línea a línea del documento que constituye el programa. Traducen una línea e inmediatamente la ejecutan.

La principal ventaja de estos lenguajes radica en la facilidad con que se detecta un error de escritura o error sintáctico, ya que la línea que contiene un error, al intentar ejecutarse, pone de manifiesto el error inmediatamente y de esta forma se facilita la localización del error.

Definición

Compiladores: por el contrario de los lenguajes intérpretes, los compiladores traducen todas las líneas del programa antes de ejecutarlas, produciendo un documento que se denomina código Objeto, que es el que queda listo para ser ejecutado por la máquina. La ejecución de un programa compilado resulta siempre ser más rápido que desde un programa interpretado, sin embargo, la depuración o localización de errores sintácticos en la programación resulta ser más compleja.

De la historia

Fig. 1.2 Ada Augusta Byron

ADA: es un lenguaje de programación orientado a objetos diseñado bajo encargo del Departamento de Defensa de los Estados Unidos. El nombre se eligió en conmemoración de lady Ada Augusta Byron (1815-1852) Condesa de Lovelace, hija del poeta Lord George Byron, a quien se considera la primera programadora de la historia, por su colaboración y relación con Charles Babbage, creador de la máquina analítica.

Desde el punto de vista histórico los lenguajes de programación se pueden dividir en:

- Lenguajes de primera generación (1GL): (antes de 1950). Se corresponden con los ya mencionados lenguajes de máquina.
- Lenguajes de segunda generación (2GL): (1950-1955). Se introducen los lenguajes ensambladores.
- Lenguajes de tercera generación (3GL): (1956-1965). Poco a poco los intérpretes admitieron palabras más complejas y comienzan a parecerse a los lenguajes naturales como el inglés, el español, etc. En los años 70 se crean PASCAL, ADA y PROLOG. Mención especial debe atribuirse al lenguaje C, desarrollado por Dennis Ritchie, creador también del sistema operativo UNIX, que dio origen a LINUX y Android.
- Lenguajes de cuarta generación (4GL): (a partir de 1980). Se vinculan con la llamada Programación Orientada a Objeto (POO). Se eleva cada vez más la cercanía con el lenguaje natural y se desarrollan entornos de desarrollo llamados IDE con interfaces gráficas de usuario (GUI).

Representantes de esta generación son: Java, C++, Delphi, Visual Basic, Eiffel, SmalTalk, etcétera.

- En esta categoría aparecen una familia de lenguajes denominada ***Hyper Talk,*** diseñados en el mundo de las Macintosh y que son impresionantemente cercanos al lenguaje natural al límite de poseer una gramática similar a los lenguajes naturales con fenómenos de sinonimia, pronombres personales, preposiciones, etc. Ejemplos de esta familia son: ***OpenScript, Transcript, Livecode***, entre otros.
- Lenguajes de quinta generación: son lenguajes especializados en técnicas de inteligencia artificial.

De la historia

Fig. 1.3 Al-Juarismi

Al-Juarismi (fig.1.3) fue un matemático, astrónomo y geógrafo persa musulmán, que vivió aproximadamente entre 780 y 850 al que se le atribuye ser el creador de las palabras ***álgebra*** y ***algoritmo.*** Es considerado como el padre del álgebra y como el introductor de nuestro sistema de numeración denominado arábigo.

Comprueba lo aprendido

1. ¿Qué se entiende por algoritmo?

2. Identifique procedimientos relacionados con la vida cotidiana o procedente de otras asignaturas que puedan ser considerados algoritmos.

3. Mencione las características que debe cumplir cualquier algoritmo.

4. Escriba con sus palabras algoritmos para:
   a) Cruzar una avenida.
   b) Saber si un número dado es par.
   c) Freír un huevo.
   d) Ir de la casa a la escuela.
   e) Pasar una aplicación de un teléfono a otro utilizando Zapya.

5. Identifique y comente cuatro formas de expresar algoritmos.

6. ¿Por qué el lenguaje natural no es un recurso idóneo para expresar algoritmos?

7. ¿Qué se entiende por programa informático?

8. ¿Qué lugar ocupa el concepto de lenguaje de programación en el concepto de ***software***?

9. ¿Qué tipos de lenguajes de programación se discuten en este libro?

10. ¿Cuál es el único lenguaje que comprende un sistema informático de manera directa?

11. ¿Cómo se clasifican los lenguajes de programación según la forma en que producen su traducción al lenguaje de máquina?

12. ¿Por qué se crearon los lenguajes de alto nivel?

13. Investigue datos biográficos sobre la primera mujer programadora de la historia.

14. ¿De qué lugar es el matemático al que se le atribuye el concepto de algoritmo? ¿Qué país es este hoy día?

15. Elabore una línea de tiempo que recoja la evolución histórica de los lenguajes de programación.

CAPÍTULO 2

Elementos de lógica de programación

El estudio de la programación está conectado directamente al área de las ciencias exactas, presente en diversos momentos. Por otro lado, el inglés, hoy fundamental en cualquier área del saber, también será importante, ya que al programar encontrarás una mayor variedad de cursos y materiales en este idioma. Además, la mayoría de los lenguajes de programación utilizan el inglés como base.

Lo que te puede generar más confusión al inicio de este camino es escoger el lenguaje de programación indicado. Sin embargo, déjame decirte que el lenguaje en sí no importa mucho al inicio, puesto que la lógica es la misma para todos, por esta razón no te preocupes comenzar aprendiendo la lógica de programación, es el mejor camino.

2.1 ¿Qué es la lógica de programación?

Aprender una lengua extranjera implica cierta especificidad. Las bases fonéticas que poseen diferentes lenguas suelen ser diferentes; también las gramáticas de cada lengua son diferentes. De hecho, hay quien puede tener mejores resultados estudiando italiano que estudiando chino o francés.

Si extrapolamos esta situación al aprendizaje de un lenguaje de programación podríamos tener la misma situación, o sea, que se nos haga más fácil aprender un lenguaje como Visual Basic que aprender Java. Sin embargo, existe una solución para este dilema y está relacionada con una disciplina que se denomina lógica de programación.

La lógica de programación es una disciplina que contiene conceptos y fundamentos para aprender a programar cualquier lenguaje de programación. Trata los conceptos y procedimientos invariantes a todos los lenguajes de programación, por lo que aprender lógica de programación

es como prepararse de manera sólida para poder abordar cualquier lenguaje de programación; los de hoy y los de mañana.

Como valor añadido a este razonamiento hay que tener en cuenta que la informática es una de las ciencias que cambia con mayor velocidad. De tal manera, prepararse para un lenguaje específico, aun cuando se estime que es el más robusto y poderoso, constituye un riesgo alto, ya que todo está sujeto a cambios.

Comencemos entonces el estudio de esta disciplina, que como hemos dicho nos preparará para abordar no solo el lenguaje que utilizaremos en esta unidad, sino también para el resto de nuestra vida.

Definición

¿Qué es la informática?
La informática, o también llamada computación, es la ciencia que estudia el tratamiento automatizado de la información.

Definición

¿Qué es información?
La información es un conjunto organizado de datos procesados, que constituyen un mensaje que cambia el estado de conocimiento de la persona o sistema que recibe dicho mensaje.

Como vemos, el concepto de información se sustenta en el concepto de dato y plantea que para que un dato refleje información es necesario que estos estén organizados y procesados, de tal manera que se conviertan en un mensaje que modifique el estado de conocimiento de quien recibe el mensaje.

De aquí se desprende que dato que no tenga significado no refleja información y que para que estos reflejen información es necesario procesarlos (esquema 2.1).

Esquema 2.1 Obtención de la información

2.2 Datos

En programación, ***dato*** es la expresión general que describe las características de las cosas sobre las cuales aplicamos un algoritmo. Podríamos decir que es la mínima expresión susceptible de convertirse en información, de la misma manera que la ***célula*** es la unidad estructural y funcional de los seres vivos.

Cuando describimos algo lo hacemos mediante el manejo de datos, también llamados atributos.

Si decimos que Juan es alto, delgado, de sexo masculino, que obtuvo la calificación de excelente en el último trabajo de control y que es inteligente, estamos describiendo a Juan mediante sus datos o atributos.

Veamos un ejemplo: ¿cuántos datos se desprenden de la expresión?

La estatura de Juan es 1,73.

Un razonamiento sutil nos daría como respuesta tres datos: el nombre de la persona, la estatura de la persona y el sexo de la persona.

Nombre	Estatura	Sexo
Juan	1,73	M

A un conjunto de datos existentes alrededor de una misma entidad se le denomina ***registro*** o ***artículo*** y cada uno de los datos independientes de un registro se denomina ***campo.***

Procesando estos datos, atributos o campos podemos llegar a un conocimiento más profundo acerca de Juan y esto nos puede llevar a una certera toma de decisión con respecto a Juan.

2.3 Tipos de datos

Recuerda que el único alfabeto que comprende los sistemas informáticos son los códigos binarios, o sea, 0 (ceros) y 1 (unos), de ahí que increíblemente y por suerte hasta el momento, a pesar de los avances de la tecnología, hoy día los sistemas informáticos solo pueden procesar

cuatro tipos de datos y el resto de atributos de las cosas se reducen a estos cuatro tipos:

1. Datos de números enteros.
2. Datos denominados coma flotante o números reales.
3. Datos de cadenas o texto.
4. Datos lógicos o ***Boolean*** (verdadero o falso).

Datos de cadenas o texto

Al igual que los números, los caracteres, letras u otros se codifican mediante números. A cada letra del alfabeto se le asigna un determinado valor o código y evidentemente existen diferentes formas de codificar los caracteres, así por ejemplo existe la codificación ASCII. En el código ASCII la letra *a* se representa con el número 97 y la letra *A* con el código 65.

¿?¿Sabías qué...?

El código ASCII inicialmente no contemplaba a la letra **Ñ** y esto exigió la necesidad de crear el ASCII extendido que sí contempla la **ñ** y la **Ñ** entre otros símbolos latinos necesarios.

Ni el ASCII, ni el ASCII extendido incluían signos propios de idiomas como el francés o el alemán. Por esta razón surgió el código ANSI. Sin embargo, tanto el ASCII como el ANSI carecían de caracteres para idiomas como el árabe, el hebreo, el chino, el japonés o el ruso. Debido a esta limitación, por tal motivo en octubre de 1991 se creó en el estándar UNICODE, también conocido como UTF, con el objetivo de dar cobertura a estos y otros idiomas.

2.4 Variables y constantes

Generalmente los datos se almacenan en la memoria de los sistemas informáticos y estas constituyen unas localizaciones denominadas *variables* y este concepto guarda cierta relación con su concepto homólogo en matemática, pero no debe entenderse igual de manera literal como veremos más adelante.

Definición

Variable (en informática)

Es un espacio de memoria reservado en la computadora que almacena un valor o dato. Este valor puede cambiar durante la ejecución del programa, de ahí su nombre.

Una representación de la vida práctica que puede hacernos entender el concepto de ***variable*** es un *gavetero,* en las gavetas podemos guardar cosas. Unas veces podemos guardar una cosa y otras veces guardamos otra cosa, de ahí el nombre de variable. Sin embargo, hay ocasiones en que por comodidad decidimos que una vez guardado algo, no permitiremos que cambie el contenido, entonces decimos que estamos en presencia de una ***constante.***

Definición

Constante (en informática)

Es un valor que no cambia durante la ejecución del programa. Las constantes se utilizan para almacenar datos que deben permanecer sin alteraciones.

2.4.1 Identificadores de variables y constantes

Las mencionadas localizaciones de memoria denominadas variables o constantes son accedidas mediante nombres o identificadores. Las reglas para poner identificadores a una variable o constante es la siguiente:

- El primer carácter del nombre debe ser una letra o el carácter de subrayado, no puede ser un número.
- Los restantes caracteres del nombre pueden ser letras, números, excepto símbolos como #, (, ?, ...,), etcétera.
- No deben contener espacios.
- La longitud del identificador depende del lenguaje de programación con que se trabaje.

Ejemplo:
Identificadores de variables y constantes
Contador-correcto
4 Calificación-incorrecto (comienza con número)
Mi contador-incorrecto (posee espacio)

+ Abc-incorrecto (posee un símbolo inadecuado)
Var-correcto

Un ejemplo de *constante* es el número *Pi*, que es igual a *3,1416…*

Generalmente en la vida práctica una "gaveta" en la que guardamos zapatos tiene forma y posiciones diferentes que las gavetas en las que guardamos camisas; otro ejemplo muy evidente es que el estuche en que guardas un violín es muy diferente al estuche en que guardas una trompeta.

En la mayoría de los lenguajes de programación, antes de almacenar un dato, es necesario declarar el tipo de dato que se va a guardar. Esto permite guardarlo, que el sistema prepare o acondicione el espacio de memoria que se reservará para dicho dato. Estos tipos de lenguajes se denominan ***lenguajes fuertemente tipados***. Como puedes ver, aunque este mecanismo es sensato para ahorrar espacio y utilizarlo de manera eficiente (por ejemplo, no usar un cajón grande para guardar solo un pañuelo), puede resultar tedioso y pedante. Por esta razón, la mayoría de los lenguajes de programación modernos son débilmente tipados.

Los *lenguajes* ***débilmente tipados*** poseen la "inteligencia" de inferir el tipo de dato que van a almacenar a partir de la operación que los involucra, es algo así como deducir que si lo que se va a guardar sirve para ponérselo en los pies antes que los zapatos, entonces debe ser un par de medias y se reserva espacio solo para un par de medias.

Veamos un ejemplo más concreto vinculado con el símbolo dos.

Si decimos que Juan cumple dos años, el símbolo dos representa una cantidad y por ende es de tipo numérico, en particular un entero, pero si decimos que Juan vive en la calle dos entre San Bartolomé y Máximo Gómez, entonces el símbolo dos no representa una cantidad, sino el nombre de una calle. O sea, que un mismo símbolo puede representar una cosa u otra, en dependencia del contexto. Lo importante es que el intérprete o el compilador del programa se puedan percatar de estas sutilezas y a partir del contexto, inferir el tipo de dato.

2.5 Expresiones

Las operaciones que se realicen sobre estas variables y/o constantes se denominan *expresiones* y están definidas por una serie de operadores, entre los cuales se encuentran:

1. Operadores aritméticos

Operador	Operación	Ejemplo	Resultado
^	Potencia	4 ^ 4	16
*	Multiplicación	5,25 * 3	15,75
/	División	100/4	25
+	Suma	2 + 4	6
-	Resta	4 - 2	2
Mod	Módulo de la división entera	4 mod 3	1
Div	División entera	8 div 3	2
Sqrt	Raíz cuadrada	Sqrt (16)	4
ABS	Valor absoluto	Abs (-4)	4
Random	Número aleatorio	Random (10)	Un número al azar entre 1 y 10

La jerarquía de los operadores es similar a lo que aprendiste en Matemática: primero se opera lo que esté entre paréntesis y luego la potencia, y la multiplicación, la división, la división entera y el módulo según aparezcan, y finalmente la suma y la resta.

2. Operadores relacionales: son los operadores que sirven para comparar.

Operador	Operación	Ejemplo	Resultado
=	Igual que	"Lalo"="Lola"	Falso
<>	Desigual a	"Lalo"<>"Lola"	Verdadero
<	Menor que	4 < 8	Verdadero
>	Mayor que	4 > 8	Falso
<=	Menor o igual	5 <= 12	Verdadero
>=	Mayor o igual	5 >= 12	Falso

3. Operadores lógicos: son operadores que permiten conformar condicionales a partir de expresiones simples y pueden ser de tres tipos:

- Conjunción (y)
- Disyunción (o)
- Negación (No).

Definición

Conjunción (y)

Para que una conjunción de condiciones sea verdadera, todas las condiciones tienen que ser verdaderas.

5 = 2 + 3 y 4 < 8 y 4 > 1 será verdadera, ya que cada condición independiente es verdadera.

"Lola" = "Lola" y "Pepe" = "José" será falso, ya que informáticamente hablando "Pepe" no es igual a "José".

Definición

Disyunción (o)

Para que una disyunción de condiciones sea verdadera, bastará con que alguna al menos sea verdadera.

5 = 2 + 3 o 4 < 3 o 4 > 1 será verdadera, ya que la primera condición independiente es verdadera.

"Lola" = "Lalo" y "Pepe" = "José" será falso, ya que ninguna condición independiente es verdadera.

Definición

Negación (no)

Para que una negación sea verdadera, la condición que se niega tiene que ser falsa y viceversa.

No (5 = 3) es verdadera, ya que 5 = 3 es falsa.

No (4 < 8), es falsa, ya que es verdadero que 4 < 8.

Veamos los siguientes ejemplos:
A = 5, B = 16
A ^ 2 < B * 2
2 < 32 entonces el resultado es verdadero.

X = 6, B = 27
(X * 5 + B ^ 1/3) < X ^ 3/B
(30 + 3) < 216/27
33 < 8 entonces el resultado es falso.

4. Operadores alfanuméricos
5. Operador de concatenación
   Cad1 UNIR Cad2=Cad1Cad2

Ejemplo
María *UNIR* Elena=MaríaElena
María *UNIRUNIR* Elena=María Elena

Uso de paréntesis

Los paréntesis permiten alterar el orden lógico de las operaciones, por ejemplo: A / (2 + 3)

Según el orden de operaciones previsto se realiza siempre la división antes que la suma, sin embargo, al estar la suma entre paréntesis debe realizarse la suma antes y luego la división.

2.6 Contadores y acumuladores

Como se planteó en el epígrafe 2.4, una variable es una localización de memoria en la que se almacena un dato, y comparábamos las variables con gavetas de un escaparate. Pero entonces surge la pregunta ¿qué existe en esa localización antes de que en ella se deposite un dato? Como mismo ocurriría con una gaveta, diríamos que podría estar vacía o tener otro dato que fuese colocado anteriormente. Esta es la causa por la cual al trabajar con variables es necesario una operación que se denomina ***inicialización***.

Definición

Inicialización de una variable: inicializar una variable es asignarle de manera explícita un valor inicial al espacio de memoria que la variable ocupará.

Esta operación es imprescindible si la variable está sujeta a cambios como consecuencia de operar con su contenido.

Analicemos qué podrían significar las siguientes expresiones:

A = 2 (a un espacio de memoria, al que se le ha llamado ***A***, se le asigna o deposita el valor 2).

Var = "A" (a un espacio de memoria, al que se le ha llamado ***Var,*** se le asigna o deposita el valor de cadena ***A***).

X = X + 1 (como debes estar pensando esta expresión desde el punto de vista matemático es absurda, ya que, bajo el supuesto de que ***X*** es un número entero, la expresión plantea que el número ***X*** es igual a su sucesor ***X + 1,*** y como sabemos esto nos llevaría al absurdo de que ***0 = 1***, algo así como que el todo es igual a la nada).

Sin embargo, el hecho de que la Informática tenga tanta relación con la Matemática hizo que un mismo signo = se aplicara en ambas ciencias de manera distinta. En Matemática el signo = significa ***igualdad,*** mientras que en Informática significa ***asignación.*** Igualdad pretende decir que lo que está a la izquierda es igual a lo que está a la derecha y asignación significa que lo que está a la derecha deberá reemplazar el valor que está a la izquierda, por tal motivo la expresión informática ***X = X + 1*** quiere decir que donde estaba ***X*** se cambie por el valor que estaba, o sea, el valor de ***X,*** adicionándole 1. ¿Interesante verdad?

Tan delicado es el asunto, que algunos lenguajes de programación han abolido el signo de igualdad como equivalente del signo de asignación.

Por lo antes dicho NO USAREMOS el signo = para expresar asignación, sino que lo haremos usando una pequeña saeta o flecha. Entonces: al escribir correctamente la expresión ***X = X + 1,*** escribiremos ***X: = X + 1.***

Definición

Contador: un contador en informática se entiende como una expresión del tipo: ***C:=C+1.***

Definición

Acumulador: un acumulador en informática se entiende como una expresión del tipo: ***S:=S+A.***

2.7 Descripción de algoritmos

Construcción de algoritmos

La construcción de algoritmos se va a corresponder con las siguientes ideas:

- Secuenciación: es la idea que plantea que las acciones de los algoritmos se irán realizando de arriba hacia abajo, comenzando por la primera acción hasta la última.
- Las acciones pueden ser diferentes, pero siempre serán de lectura, procesamiento y salida de información.
- Es conveniente comenzar un algoritmo con un identificador de lo que hace, así como finalizarlo con la palabra fin.

Recuerda que...

Los algoritmos pueden ser representados de cuatro maneras diferentes:

1. Mediante el lenguaje natural (español).
2. Mediante diagramas de flujo.
3. Mediante pseudocódigo.
4. Mediante un lenguaje de programación o implementación.

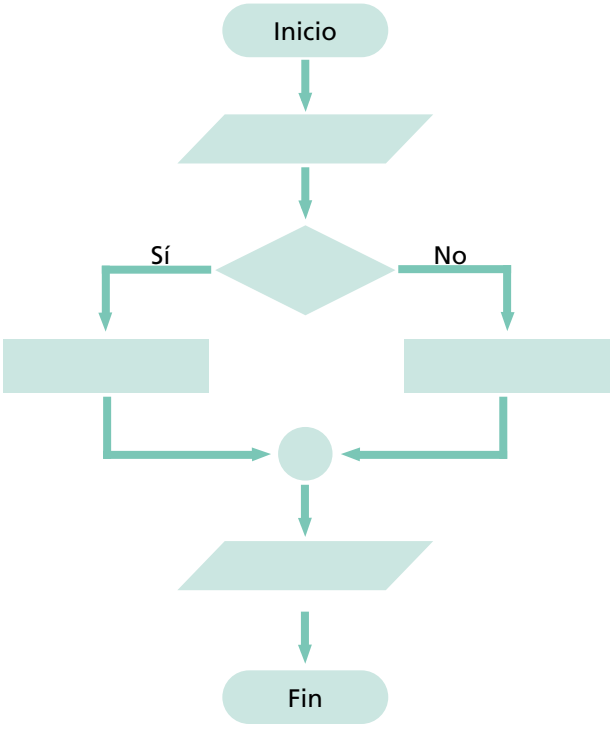
2.7.1 Expresión de algoritmos mediante diagramas de flujo

Antes de iniciar en el análisis y la construcción de algoritmos mediante diagramas de flujo es importante señalar que estos no son solo importantes en informática, sino en todos los procesos que llevan consigo una secuencia lógica. Recuerda este tipo de esquema estudiado en octavo grado mediante los organizadores gráficos.

Los diagramas de flujo son (esquema 2.2):

- Un tipo de expresión gráfica.
- Son sencillos y de fácil comprensión.
- Representan gráficamente el flujo de los datos en la solución de un problema.

Esquema 2.2 Diagrama de flujo



Simbología de diagramas de flujo

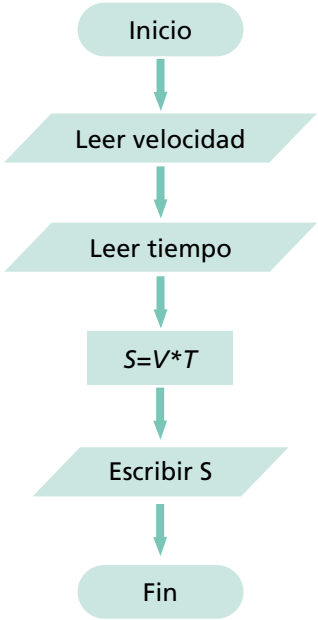
Los diagramas de flujo son herramientas visuales fundamentales en la representación de procesos, permitiendo describir de manera clara y ordenada las etapas, decisiones y acciones involucradas en un sistema. La simbología utilizada en estos diagramas juega un papel crucial, ya que cada figura geométrica tiene un significado específico que facilita la comprensión y el análisis de los flujos de trabajo. En este apartado exploraremos los símbolos más comunes y su aplicación, proporcionando una base sólida para la interpretación y creación de diagramas de flujo (esquema 2.3).

Esquema 2.3 Simbología de diagramas de flujo

Tres ejemplos clásicos con diagramas de flujo

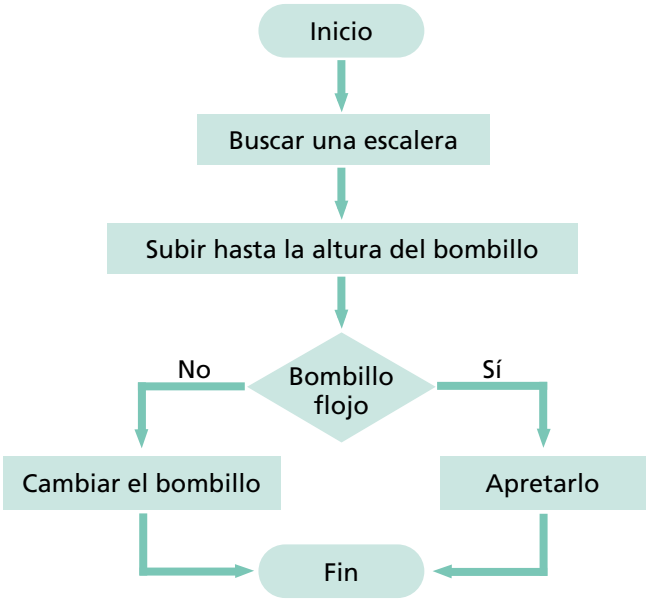
Ejemplo 1: describir un algoritmo que permita calcular el espacio recorrido por un cuerpo que se mueve con un Movimiento Rectilíneo Uniforme (MRU) si se conocen la velocidad, el tiempo transcurrido en el desplazamiento y la fuerza de rozamiento (esquema 2.4).

Esquema 2.4 Cálculo de recorrido con MRU



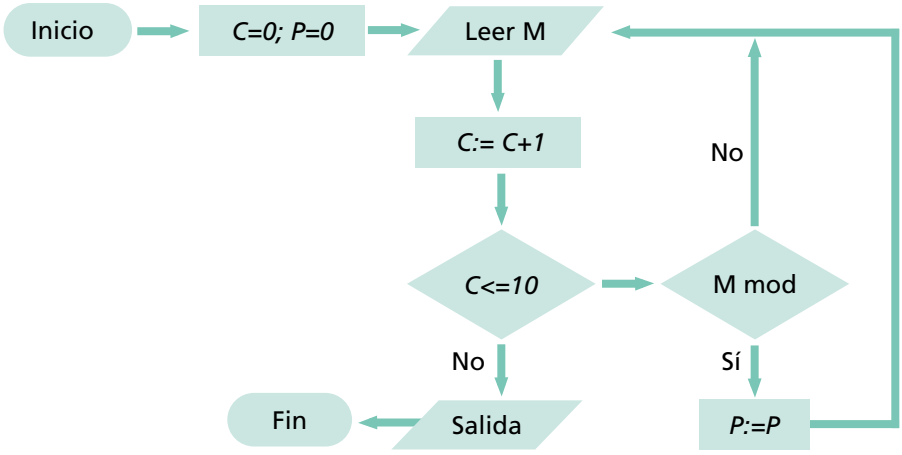
Ejemplo 2: al interactuar con el interruptor de la luz del comedor se constata que la luz no enciende. Elabore un diagrama de flujo para resolver este problema (esquema 2.5).

Esquema 2.5 Interactuar con el interruptor de luz



Ejemplo 3: dado un conjunto de diez números enteros, determinar cuántos son pares (esquema 2.6).

Esquema 2.6 Determinación de números pares



Posterior a la elaboración de los tres ejemplos es interesante realizar el siguiente análisis:

1. Los tres diagramas de flujo poseen inicio y fin.
2. En el ejemplo uno las acciones ocurrirán invariablemente siempre de la misma manera, o sea, en la misma secuencia.
3. En los ejemplos dos y tres está presente la figura que representa condicionales, o sea, que las acciones no siempre ocurrirán en la misma secuencia. Los datos una vez correrán por la rama de los Sí y otras veces podría correr por la rama de los No.
4. Para lograr un resultado en los ejemplos uno y dos las acciones ocurrirán una sola vez.
5. Para lograr un resultado en el ejemplo tres las acciones ocurrirán repetidamente varias veces (en particular ***n*** veces).

Los ejemplos realizados son representantes genuinos de una de las más importantes ideas de las ciencias de la computación y es que al resolver un problema con un sistema informático el flujo de11 los datos tendrá características ***secuenciales*** (algoritmo secuencial), características de flujo condicionado (algoritmos alternativos) o características repetitivas (algoritmos cíclicos o iterativos).

Saber más

Teorema de la programación estructurada

Establece que toda función computable puede ser implementada en un lenguaje de programación que combine solo tres estructuras lógicas o estructuras de control de flujo. Esas tres formas son:

- Secuencial: ejecución de una instrucción tras otra.
- Alternativa: ejecución de una de dos instrucciones (o conjuntos), según el valor de una variable booleana.
- Cíclica: ejecución de una instrucción (o conjunto) mientras una expresión booleana sea "verdadera". Esta estructura lógica también se conoce como **ciclo** o **bucle**.

2.7.2 Expresión de algoritmos mediante pseudocódigo

Los lenguajes naturales como el español o el inglés son extremadamente ricos, a tal extremo que gracias a estas características tanto la prosa

como el verso devienen del talento de oradores y escritores, cada uno empleando recursos lingüísticos resultado de la imaginación, la creatividad y el talento, en esto influye indiscutiblemente la versatilidad y el carácter polisemántico de los lenguajes naturales.

Al leer una obra literaria, por ejemplo, imaginamos lo escrito por el autor, pero desde una posición personalizada, o sea, acorde con nuestras características individuales que evidentemente dependerán de múltiples factores (cultura, idiosincrasia, contexto, etc.). Por lo antes dicho, la descripción de un algoritmo mediante el lenguaje natural puede verse afectado por los elementos antes mencionados, transfiriéndole a los algoritmos ciertos elementos de ambigüedad. Para evitar este problema se ha creado el ***pseudocódigo.***

Definición

Pseudocódigo: es la formalización o elección de un subconjunto de palabras del lenguaje natural mezclado con algunos símbolos para expresar algoritmos.

El pseudocódigo no llega a ser un lenguaje de programación y, por tanto, no es entendible por dispositivos informáticos, pero contiene muchas características comunes a todos los lenguajes de programación, de ahí su valor didáctico y por ende, su importancia para aprender a programar sistemas informáticos.

Tres ejemplos clásicos con pseudocódigo

Ejemplo 1: describir un algoritmo que permita calcular el espacio recorrido por un cuerpo que se mueve con MRU si se conocen la velocidad y el tiempo transcurrido en el desplazamiento.

Definición

Estructuras de control secuencial: las estructuras secuenciales son aquellas en que las acciones se suceden una detrás de la otra, sin que ninguna condición altere el sentido de ejecución de las acciones.

Inicio
Leer V
Leer t
*S:=V*t*
Escribir S
Fin

En Scratch esto sería (fig. 2.1):

Fig. 2.1 Algoritmo para calcular espacio recorrido

Ejemplo 2: al interactuar con el interruptor de la luz del comedor se constata que la luz no enciende. Elabore un algoritmo en pseudocódigo para resolver este problema.

Definición

Estructuras de control alternativas: las estructuras alternativas son aquellas en que las acciones se suceden una vez, pero en dependencia del cumplimiento o no de ciertas condiciones.

Inicio
Buscar escalera
Subir hasta la altura del bombillo
Si (bombillo flojo) entonces
Apretarlo
sino
Cambiarlo
Fin si
Fin

En Scracht esto sería (fig. 2.2):

Fig. 2.2 Estructuras de control alternativas

Ejemplo 3: dado un conjunto de diez números enteros, determinar cuántos son pares.

Definición

Estructuras de control cíclicas: las estructuras cíclicas o también llamadas iterativas son aquellas en que las acciones se repiten más de una vez, un número determinado de veces o hasta tanto se cumpla una condición.

Inicio
C:=0
P:=0
Mientras C<=10
Leer un número M
C:=C+1
Si (M mod 2)=0 entonces
P:=P+1
Fin si
Fin Mientras
Escribe "Cantidad de números pares": UNIRP
Fin

En Scratch esto sería (fig. 2.3):

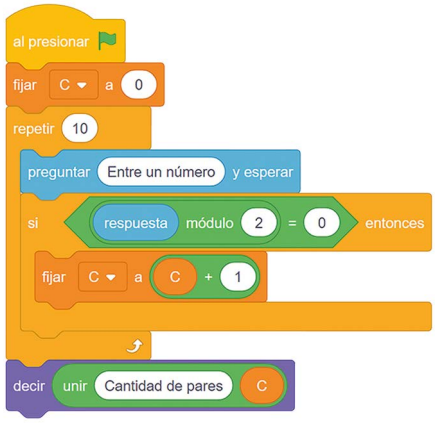


Fig. 2.3 Estructuras de control cíclicas

Comprueba lo aprendido

1. ¿Qué es lógica de programación?
2. ¿Qué diferencia existe entre dato e información?
3. ¿A qué se llama campo?
4. ¿A qué se llama registro?
5. ¿Qué tipos de datos maneja un sistema informático?
6. ¿Qué es en informática una variable?
7. ¿Qué es en informática una constante?

8. ¿Qué reglas deben tenerse en cuenta para nombrar variables?

9. Dadas las variables:

$a = 10$	$a = a + 3$	$a = a + c$
$b = 20$	$b = b + 4 - a$	$b = 4$
$c = 5$	$c = a + b + c$	$c = c + 3 - b + 2$

a) ¿Qué valores quedan almacenados en las variables *a*, *b* y *c*?

10. Dadas las variables:

$a = 5$	$a = a + 10$	$a = a + 1$
$b = 18$	$b = b + 5 - c$	$b = b + c$
$c = 15$	$c = c + 4 + b$	$c = b + c$
$d = 25$	$d = d + b + a$	$d = b + b$

a) ¿Qué valores quedan almacenados en las variables *a*, *b*, *c* y *d*?

11. ¿Qué es un lenguaje fuertemente y débilmente tipado?

12. Complete el cuadro de los operadores aritméticos.

Operador	Operación	Ejemplos	Resultados
^	Potencia		16
	Multiplicación	5,25 * 3	
/	División	100/4	
+		2 + 4	
-	Resta		2
Mod	Módulo de la división entera	5 Mod 2	
Div		8 div 3	2
Sqrt	Raíz cuadrada		4
ABS		Abs (-4)	4
Random	Número aleatorio	Random (10)	

13. ¿Cómo podemos alterar el orden natural de las operaciones aritméticas?

14. Complete el cuadro de los operadores relacionales.

Operador	Operación	Ejemplos	Resultados
	Igual que		Falso
<>		"Lalo" <> "Lola"	
<		4 < 8	
	Mayor que	4 > 8	
<=	Menor o igual		Verdadero
>=		5 >= 12	Falso

15. ¿A qué se llama conjunción? Ponga ejemplos.

16. ¿A qué se llama disyunción? Ponga ejemplos.

17. ¿Cuándo una negación es falsa? Ponga ejemplos.

18. Elabore diagramas de flujo y seudocódigos para los siguientes algoritmos:
a) Leer dos números, sumarlos y mostrar el resultado.
b) Encontrar la media aritmética de cuatro números que se leen.
c) Convertir un valor de distancia que se lee en centímetros a kilómetros.
d) Convertir un valor de temperatura de grados centígrados a Fahrenheit.
e) Leer dos números y simular una calculadora (+, -, / y *).

2.8 Algoritmos secuenciales

La estructura secuencial es aquella en la que una acción sigue a otra en secuencia. Las tareas se suceden de tal modo que la salida de una es la entrada de la siguiente y así sucesivamente hasta el fin del proceso. La desventaja de este tipo de algoritmos es su inflexibilidad, no permite un retorno a una secuencia de repetición, que en caso de necesitarse se deben escribir las acciones tantas veces como sea necesario.

Por ejemplo:

1. Se necesita obtener el promedio de las notas en asignaturas como Matemática, Lengua Española e Historia de un educando de noveno grado.

Algoritmo: promedio de tres notas
Inicio
Leer Mat
Leer Esp
Leer Hist
Prom:=(Mat+Esp+Hist)/3
Escribir "Promedio": UNIRProm
Fin

En Scratch esto sería (fig. 2.4):

Fig. 2.4 Promedio de tres notas

2. En una Escuela Secundaria Básica se efectúa un concurso de ortografía y en la clave se plantea que por cada error de acentuación otográfica se quitarán dos puntos, por errores de grafemas cuatro puntos y por errores de uso de mayúsculas cinco puntos. Una vez calificados los exámenes se conocen la cantidad de errores de cada tipo que fueron cometidos por cada educando. Elabore un algoritmo en pseudocódigo que permita calcular la cantidad de puntos que pierde cada educando.

Algoritmo: puntos ortográficos
Inicio
Leer Cant_acentos
Leer Cant_Grafemas
Leer Cant _May
*Puntos_Acent:=:Cant_Acentos*2*
*Puntos_Grafemas:=:Cant_Grafemass*4*
*Puntos_May:=:Cant_May*5*
Total_Puntos:=:Puntos_Acent+Puntos_Grafemas+Puntos_May
Escribir Total_Puntos
Fin

En Scratch esto sería (fig. 2.5):

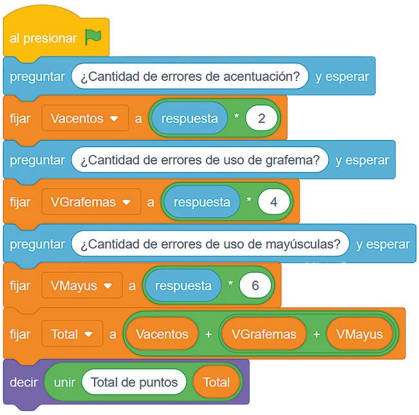


Fig. 2.5 Puntos ortográficos

Nota: analiza si el siguiente algoritmo es correcto o no y a qué conclusión puedes llegar.

Algoritmo: puntos ortográficos
Inicio
Leer Cant_acentos
Leer Cant:Grafemas
Leer Cant _May
*Cant_Acentos:=:Cant_Acentos*2*
*Cant_Grafemas:=:Cant_Grafemas*4*
*Cant_May:=:Cant_May*5*
Total_Puntos:=:Cant_Acentos+Cant_Grafemas+Cant_May
Escribir Total_Puntos
Fin

En efecto, esta segunda variante es también válida e, inclusive, más eficiente, ya que hace un mejor manejo de la memoria. Ten en cuenta lo que significa la expresión: ***Cant_Acentos:=:Cant_Acentos*2.***

Como habíamos dicho, en Matemática esto resultaría absurdo, ya que la interpretación matemática de esta expresión sería que un número es igual a él mismo multiplicado por dos y esto es válido solo para un número, el cero; sin embargo, en informática la expresión significa que se toma el contenido del espacio de memoria denominado ***Cant_Acentos***, ese contenido se multiplica por dos y el resultado se vuelve

a colocar en el mismo espacio de memoria denominado ***Cant_Acentos.*** De igual forma.se procede con las otras dos variables. Este algoritmo resulta más eficiente que el anterior, ya que hace un uso más racional de la memoria.

3. Como sabes, el área de un triángulo se calcula multiplicando la base del triángulo por su altura y dividiendo entre dos el resultado de la multiplicación. Elabora un algoritmo en pseudocódigo que calcule el área de un triángulo si se conoce su base y su altura.

Algoritmo: área de un triángulo
Inicio
Leer Base
Leer Altura
*Area:=Base*Altura/2*
Escribir Area
Fin

Nota: recuerda que se planteó que los identificadores de las variables no deben llevar tildes.

Evidentemente, este problema resulta muy sencillo si se conoce la base y la altura de un triángulo, pero el conocimiento de la altura de un triángulo no siempre es un dato sencillo de conseguir, solo en casos especiales como el triángulo rectángulo que la longitud de la base puede coincidir con la de la altura. En general, el conocimiento de la base puede involucrar el empleo de teoremas adicionales como el de Pitágoras o involucrar el uso de funciones trigonométricas, por tal motivo resulta conveniente obtener una solución más general mediante el modelo matemático siguiente:

$$A = \sqrt{(s\ (s - a)\ (s - b)\ (s - c))}$$

Donde ***a, b*** y ***c*** son los lados del triángulo y ***s*** es el semiperímetro.

Elabore un algoritmo en pseudocódigo usando este último modelo matemático.

Algoritmo: área de un triángulo_2
Inicio
Leer Lado_a

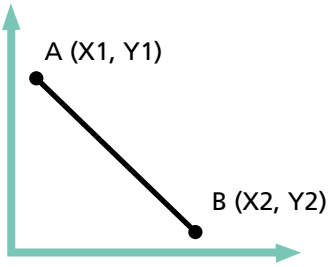
Leer Lado_b
Leer Lado_c
S:=(Lado_a+Lado_b+Lado_c)/2-Semiperímetro
Area:=sqrt(S(S-a)*(S-b)*(S-c))*
Escribir Área
Fin

En Scratch esto sería (fig. 2.6):

Fig. 2.6 Área de un triángulo

Nota: este algoritmo resulta más general que el anterior, ya que siempre podríamos medir las longitudes de los lados de cualquier triángulo.

4. Se tiene un segmento AB ubicado en el primer cuadrante de un sistema de ejes cartesiano. Elabore un algoritmo en pseudocódigo que calcule la distancia entre los puntos A y B. El modelo matemático que usaremos seña el Teorema de Pitágoras. Recuerda que: ***Hipotenusa²= Cateto_1²+ Cateto_2²***



Algoritmo: distancia entre dos puntos
Inicio
Leer x1
Leer x2
Leer y1
Leer y2
Cateto_1:=x2-x1
Cateto_2:=yx2-y1
Dist:=sqrt(Cateto_1^2+Cateto_2^2)
Escribir Dist
Fin

En Scratch esto sería (fig. 2.7):

Fig. 2.7 Teorema de Pitágoras

5. Elabore un algoritmo en pseudocódigo que lea dos valores consecutivos de la lectura de un reloj contador y devuelva el importe que se debe pagar por consumo energético.

Nota: se considera una familia ahorradora para la que el Kw/h cuesta 0,09 pesos.

Algoritmo: reloj-contador
Inicio
Leer Consumo_1
Leer Consumo_2
Diferencia:=Consumo_1-Consumo_2
*A_Pagar=Diferencia*0,09*
Escribir "Debe pagar":UNIR A_Pagar
Fin

En Scratch esto sería (fig. 2.8):

Fig. 2.8 Reloj contador

Algoritmos alternativos

1. Elabore un algoritmo en pseudocódigo que dada la entrada de dos números, determinar cuál es el mayor.

Inicio
Leer A
Leer B
Si A>B entonces
Escribir A
Sino
Escribir B
Fin si
Fin

En Scratch esto sería (fig. 2.9):

Fig. 2.9 Comparar números

2. Generalicemos el algoritmo anterior para determinar el mayor entre tres números. Reduzcamos el problema de tres al problema de dos.

Inicio
Leer A
Leer B
Leer C
Si A>B entonces
Temp:=A
Sino
Temp:=B
Fin si
Si Temp>C entonces
Mayor:=Temp
Sino
Mayor:=C
Fin si
Escribe Mayor
Fin

En Scratch esto sería (fig. 2.10):

Fig. 2.10 Comparar números

3. La historia de Cuba puede dividirse en etapas: Precolombina (antes de 1492), Colonial (1492-1858), Guerra de los Diez Años (1868-1878), Tregua Fecunda (1878-1895), Guerra Necesaria (1895-1898), Intervención norteamericana (1898-1902), Neocolonia (1902-1958) y Revolución en el poder (1959-). Existe una base de datos que contiene efemérides, o sea, un compendio de hechos históricos con la fecha en que ocurrieron. La fecha se estructura de la siguiente forma: día X del mes Y del año Z. Elabore un algoritmo en pseudocódigo que al introducir una efeméride identifique la época histórica con la que se corresponde el hecho.

```
Algoritmo Efemérides
Inicio
  Leer Efem
  Vano:= Efem
    Si Vano<1492 entonces
       Escribe "Precolombina"
    Sino
      Si Vano>=1492 y Vano<1868 entonces
        Escribe "Colonial-Antes de la Guerra de los Diez Años"
      Sino
        Si Vano>=1868 y Vano<=1878 entonces
          Escribe "Guerra de los Diez Años"
        Sino
          Si Vano>=1879 y Vano<1895 entonces
             Escribe "El período interguerras"
          Sino
            Si Vano>=1895 y Vano<=1898 entonces
               Escribe "Guerra Necesaria"
            Sino
              Si Vano>=1899 y Vano<1902 entonces
Escribe "Ocupación norteamericana"
              Sino
                Si Vano>=1902 y Vano<1959 entonces
  Escribe "Neocolonia"
                Sino
                    Si Vano>=1959 entonces
```

Escribe "Revolución en el poder"
Fin si
Fin si
Fin si
Fin si
Fin si
Fin si
Fin

En Scratch esto sería (fig. 2.11):

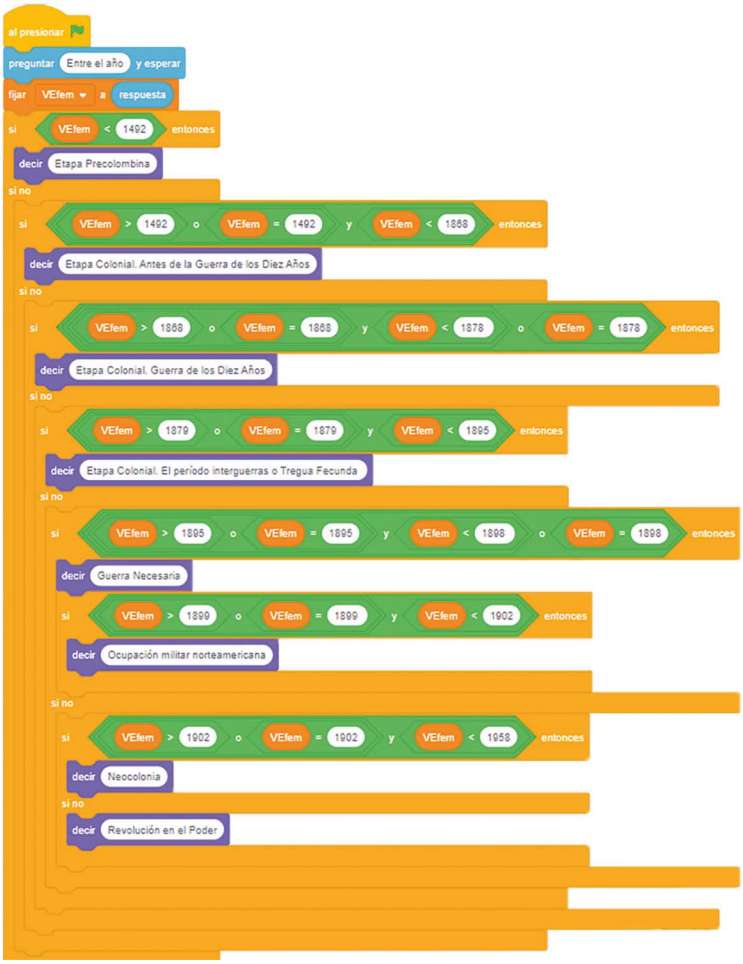


Fig. 2.11 Algoritmo Efemérides

- Note que cada sentencia ***Si*** tiene su correspondiente ***Fin Si***
- Observe el papel que juega la indentación en la escritura del pseudocódigo.
- Consecuentemente con el Teorema de la programación estructurada se ha demostrado que hasta el momento bastan estructuras secuenciales, alternativas y cíclicas para describir un algoritmo. La estructura que se muestra se denomina ***"si anidados".***
- Elabore un algoritmo en pseudocódigo que reciba un número del uno al diez y devuelva su equivalente en romano.
- Elabore un algoritmo en pseudocódigo que transfiera la evaluación de un examen calificado en base 100 a las calificaciones E, MB, B, R, M.
- Los 11 dígitos del carnet de identidad tienen un significado, como sabes, los seis primeros indican año, mes y día de nacimiento y se dice que el penúltimo dígito tiene una significación especial. Si es par, la persona es de sexo masculino y si es impar la persona es de sexo femenino. Elabora un algoritmo en pseudocódigo que al entrar el CI de una persona diga si es de sexo masculino o femenino.

Algoritmo Determina sexo
Inicio
Leer CI
Penúltimo_dígito:=última letra de CI-1
Si Penúltimo_dígito mod 2=0 entonces
Escribe "Sexo masculino"
Else
Escribe "Sexo femenino"
Fin si
Fin

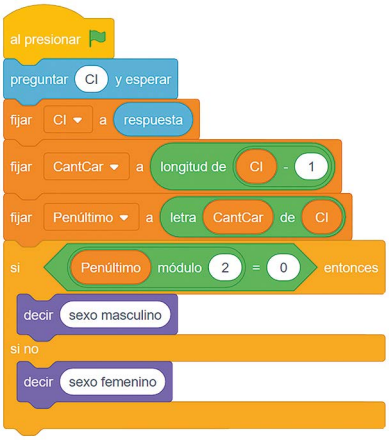


Fig. 2.12 Determina el sexo

En Scratch esto sería (fig. 2.12):

4. La diabetes es un padecimiento crónico que puede ser de dos tipos: tipo I, generalmente congénito y tipo II adquirido debido a malos hábitos alimentarios, falta de ejercicio físico, etc. Cuando el páncreas no segrega la suficiente insulina, la glucosa permanece en la sangre, no alimentando de esta forma a las células. Un indicador de diabetes es el nivel de glucosa que se encuentra en la sangre, posterior al proceso

de digestión, pero estos valores pueden variar en dependencia de las calorías consumidas en un día específico. Por tal motivo un indicio de presencia de diabetes se logra a partir de una indagación estadística comparando el promedio de las mediciones de varios días con el valor siete, considerado el umbral de máxima tolerancia de glucosa en sangre. Elabore un algoritmo en pseudocódigo que a partir de las lecturas con un glucómetro brinde indicios de presencia de la enfermedad.

Algoritmo Diabetes
Inicio
Leer Valor_gluc_L
Leer Valor_gluc_M
Leer Valor_gluc_Mi
Leer Valor_gluc_J
Leer Valor_gluc_V
Leer Valor_gluc_S
Leer Valor_gluc_D
Prom_sem:= (Valor_gluc_L+Valor_gluc_M
+Valor_gluc_Mi+Valor_gluc_J
+Valor_gluc_V+Valor_gluc_S+Valor_gluc_D)/7
Si Prom_sem>7 entonces
Escribe "Hay sospecha de diabetes"
Sino
Escribe "No hay sospecha de diabetes"
Fin si
Fin

Fig. 2.13 Algoritmo diabetes

En Scratch esto sería (fig. 2.13):

5. Elabore un algoritmo en pseudocódigo que sea capaz de clasificar un triángulo a partir de sus lados.

Algoritmo Tipo triángulo
Inicio
Leer L1
Leer L2

Leer L3
En caso Expresión
Caso L1<>L2 y L1<>L3 y L2<>L3
Escribe "Escaleno"
Caso (L1=L2) y (L2=L3)
Escribe "equilátero"
Ninguno de las anteriores
Escribe "Isósceles"
Fin Caso
Fin

6. Como debes saber, una ecuación de segundo grado posee dos raíces reales diferentes, o una sola raíz o ninguna, en dependencia del valor del discriminante. Elabore un algoritmo que a partir de los coeficientes ***a, b, c*** de una ecuación de segundo grado, determine cuántas raíces reales tendrá y de existir las calcule.

Algoritmo Ecuación de segundo grado
Inicio
Leer a
Leer b
Leer c
*Delta:=b^2-4*a*c*
En caso Expresión
Caso Delta>0
Escribe "Dos raíces reales"
*X1:=(-b+sqrt(Delta))/2*a*
*X2:=(-b-sqrt(Delta))/2*a*
Escribe "Raíz"=UNIRX1UNIRUNIR"Raíz"=UNIRX2
Caso Delta=0
Escribe "Una raíz doble"
*X1:=-b/2*a*
Escribe "Raíz"=UNIRX1
Ninguno de las anteriores
Escribe "No hay raíces reales"
Fin Caso
Fin

Algoritmos cíclicos

Los algoritmos cíclicos o iterativos son procesos repetitivos y suelen ser de dos tipos:

1. Cuando se sabe la cantidad de veces que se van a repetir las acciones.
2. Cuando la cantidad de repeticiones está condicionada por el cumplimiento de una condición.

La sintaxis en pseudocódigo de cada caso es como sigue:

1. Se conoce a priori la cantidad de veces que se deben repetir las acciones.
   PARA Contador = VALOR_1 A VALOR_2
   ACCIONES
   FIN PARA
2. No se conoce a priori la cantidad de veces que se deben reperir las acciones, por el contrario, se detendrá el algoritmo cuando se cumpla una condición.
   MIENTRAS condición
   ACCIONES
   FIN MIENTRAS

Elabore un algoritmo en pseudocódigo que defina los números pares existentes del 1 al 10.

Nota: como se observa en el ejemplo, se conoce la cantidad exacta de números que serán chequeados.

Algoritmo Números pares
Incio
Para i=1 a 10
Leer N
Si N mod 2=0 entonces
Escribir "Es par"
Sino
Escribir "Es impar"
Fin si
Fin Para
Fin

Elabore un algoritmo en pseudocódigo que determine la nota promedio de un grupo de ***n*** educandos.

Nota: como se observa en el ejemplo, no se conoce la cantidad exacta de números que serán chequeados, pero se parte del criterio de que es posible saberlo.

Algoritmo Nota promedio
Inicio
S:=0 inicializando el acumulador
Leer "Cantidad de educandos": UNIRCant
Para i=1 a Cant
Leer Nota
S:=S+Nota
Fin Para
Escribir S/Cant
Fin

Elabore un algoritmo en pseudocódigo que lea lados de triángulos y calcule su perímetro cada vez, hasta que el perímetro calculado sea mayor o igual a 20 unidades.

Nota: como se observa en el ejemplo, no se conoce la cantidad exacta de veces que leeremos los lados del triángulo y no es posible saberlo, por tal motivo no podemos usar la estructura de tipo PARA-FIN PARA; tendremos entonces que usar la estructura de tipo MIENTRAS-FIN MIENTRAS.

Algoritmo Perímetro
Inicio
Lee a
Lee b
Lee c
P:=a+b+c
Escribir P
Mientras P<20
Lee a

Lee b
Lee c
P:=a+b+c
Escribir P
Fin Mientras
Fin

Elabore un algoritmo en pseudocódigo que se use en la entrada de un cine y compute 3 posibles causas que promuevan el interés de ver la película. Se introducirá ***D-*** si el interés es deseo de ver un drama, ***P-*** si el interés es el país de origen de la película y ***M-*** por la banda sonora de la película. Se desea calcular los porcientos en que se manifiestan cada uno de los intereses.

Nota: por las características del problema está claro que no se sabe a priori la cantidad de personas que asistirán al cine, por tal motivo será necesario determinar un valor de entrada para detener el ciclo y mostrar los resultados.

Algoritmo Preferencias
Inicio
CD:=0
CP:=0
CM:=0
Preferencia:="X"
Mientras Preferencia<>"F"
Lee Preferencia
En Caso Expresión
Caso D
CD:=CD+1
Caso F
CF:=CF+1
Caso M
CM:=CM+1
Fin Caso
Fin Mientras
Escribe CD

Escribe CF
Escribe CM
Fin

Elabore un algoritmo en pseudocódigo que entre una cantidad indeterminada de números naturales devuelva la suma de todos los que sean pares y de todos los que sean impares.

Algoritmo Pares e impares
Inicio
SPar:=0
SImp:=0
N:=”X”
Mientras N<>”F”
Lee N
Si n mod 2=0 entonces
Spar:=Spat+N
Sino
Simp:=Simp+N
Fin si
Fin Mientras
Escribe SPar
Escribe Simp
Fin

CAPÍTULO 3

Nociones de ciberseguridad

En la era digital actual, la ciberseguridad se ha convertido en un tema fundamental para todos. Con el creciente uso de la tecnología en la educación y el entretenimiento, es crucial que comprendan los conceptos básicos de la seguridad en línea para protegerse a sí mismos y a su información personal en el mundo virtual.

Este capítulo tiene como objetivo brindarles las herramientas necesarias para navegar de manera segura en internet. Mediante explicaciones claras y ejemplos relevantes, aprenderán sobre la importancia de proteger sus contraseñas, identificar posibles amenazas en línea y mantener la privacidad de sus datos mientras disfrutan de las ventajas de la tecnología.

Al presentar de manera accesible y educativa los principios fundamentales de la ciberseguridad, este capítulo busca empoderar a los usuarios para que se conviertan en actores responsables y conscientes en el mundo digital, el cual está en constante evolución.

3.1 ¿Qué es la ciberseguridad? Su importancia

En un mundo cada vez más interconectado, la ciberseguridad se ha convertido en un pilar fundamental para proteger la información, los sistemas y las infraestructuras digitales. Este concepto no solo abarca la defensa contra amenazas cibernéticas, sino también la promoción de prácticas seguras que garantizan la privacidad, integridad y disponibilidad de los datos. Comprender su importancia es esencial para navegar de manera segura y responsable en el entorno digital, donde los riesgos evolucionan tan rápido como la tecnología.

Definición

La ciberseguridad es un conjunto de prácticas, tecnologías y procesos diseñados para proteger los sistemas informáticos, redes, dispositivos y datos contra ataques cibernéticos.

Consiste en salvaguardar la integridad, confidencialidad y disponibilidad de la información digital, así como proteger los activos tecnológicos de organizaciones y usuarios individuales de amenazas como ***malware, hackers, phishing,*** robo de datos y otros ataques cibernéticos.

Recuerda que...

El término ***malware*** proviene de la abreviatura del término inglés ***malicious software,*** también llamado ***badware*** o ***software*** malicioso, que ha sido diseñado para infectar y dañar un equipo. Existen muchos tipos, como los gusanos, los troyanos y el ***spyware.***

La ciberseguridad se enfoca en prevenir, detectar y responder a posibles vulnerabilidades y amenazas en el entorno digital, para garantizar la seguridad y privacidad de la información en línea, en un mundo cada vez más interconectado y dependiente de la tecnología.

Es fundamental para garantizar tu seguridad, privacidad y conocimiento en un entorno digital en constante cambio y evolución. Ayudarlos a comprender y practicar la ciberseguridad desde una edad temprana es esencial para su bienestar y desarrollo en la era digital actual. En la actualidad es de suma importancia por varias razones claves:

Protección de la información personal: al utilizar activamente dispositivos digitales y plataformas en línea para estudiar, comunicarse y entretenerse. Es crucial que comprendan cómo proteger su información personal, contraseñas y datos sensibles para evitar posibles robos de identidad o fraudes en línea.

Conciencia de las amenazas en línea: al educarlos sobre ciberseguridad, se les capacita para identificar y evitar posibles amenazas cibernéticas como virus, ***malware, phishing y ciberbullying.*** Esta conciencia les permite tomar decisiones informadas y seguras al interactuar en el entorno digital.

Desarrollo de habilidades digitales seguras: aprender sobre ciberseguridad les brinda las habilidades necesarias para navegar de manera segura

en internet, proteger sus dispositivos y datos, comprender la importancia de la privacidad en línea. Estas habilidades son esenciales en un mundo digitalizado y conectado.

Preparación para el futuro: a medida que la tecnología sigue evolucionando, la ciberseguridad se vuelve cada vez más relevante en todos los aspectos de la vida. Al introducir a los educandos en conceptos básicos de ciberseguridad desde una edad temprana, se los prepara para enfrentar los desafíos digitales futuros y les brinda una base sólida para protegerse en línea a lo largo de sus vidas.

Cuando se trata de ciberseguridad, un aspecto que se utiliza con frecuencia es el concepto de incidencias y se refiere a eventos o situaciones inesperadas que afectan la seguridad de los sistemas informáticos o la privacidad de los datos.

Definición

"Se considera un incidente de ciberseguridad cualquier evento que se produzca de forma accidental o intencional, que afecte o ponga en peligro las tecnologías de la información y la comunicación o los procesos que con ellas se realizan" (Decreto 360/2019).

Las incidencias se pueden presentar por diferentes razones:

- Ataques maliciosos: incluyen virus, ***ransomware, phishing*** y ataques DDoS.
- Errores humanos: acciones no intencionadas que causan problemas de seguridad.
- Fallas ***técnicas:*** problemas con ***hardware***, ***software*** o configuración.
- Incidentes de privacidad: fugas de datos personales o incumplimiento de regulaciones.

En nuestro país existe un marco regulatorio o legal, materializado en las siguientes resoluciones:

- Resolución No. 64/2002: creación de la Oficina de Seguridad para las Redes Informáticas (OSRI).
- Resolución No. 121/2017: establece las medidas básicas para configurar los servidores de correo electrónico que se deben implementar en las redes de datos del país debidamente autorizadas.
- Decreto Ley No. 370/2018: sobre la Informatización de la sociedad en Cuba.
- Decreto No. 360/2019: sobre la Seguridad de las Tecnologías de la Información y Comunicación y la Defensa del Ciberespacio Nacional.

- Resolución No. 128/2019: Reglamento de Seguridad de las Tecnologías de la Información y las Comunicaciones.

Además, existe un recurso de apoyo denominado Línea Única para gestionar incidentes de ciberseguridad. Esta iniciativa se enmarca en la actualización del marco jurídico relacionado con las telecomunicaciones y la ciberseguridad, en este caso la Resolución No. 105, que define el "Modelo de Actuación Nacional para la respuesta a incidentes de ciberseguridad".

La Línea Única permite a las personas notificar o denunciar afectaciones que puedan ser tipificadas como incidentes de ciberseguridad.

Para consultas o notificación de un incidente de ciberseguridad, los interesados se pueden comunicar con la Oficina de Seguridad para las Redes Informáticas (OSRI), por medio de los siguientes canales de comunicación:

- Sitio web: www.osri.gob.cu en el acápite incidentes.
- Correo electrónico: reportes@osri.gob.cu.
- Número único de atención a la población: 18810.

3.2 Tendencias en ciberseguridad: nuevas amenazas y tecnologías emergentes

Algunas tendencias destacadas en ciberseguridad incluyen el aumento de ataques de ***ransomware***, la importancia de la inteligencia artificial y el aprendizaje automático en la detección de amenazas, la creciente preocupación por la privacidad de los datos y el aumento de la ciberseguridad en la nube.

El aumento de los ataques de ***ransomware*** es una de las tendencias más preocupantes en ciberseguridad. Los ataques de ***ransomware*** involucran a los ciberdelincuentes que cifran los datos de las víctimas y exigen un rescate a cambio de restaurar el acceso a la información. Estos ataques han evolucionado en sofisticación y frecuencia, afectando a empresas, organizaciones e incluso, a usuarios individuales.

En respuesta a esta creciente amenaza, la importancia de la inteligencia artificial (IA) y el aprendizaje automático en la detección de amenazas ha aumentado significativamente. Las soluciones de ciberseguridad basadas en IA y aprendizaje automático pueden analizar grandes cantidades de datos en tiempo real, identificar patrones anómalos y predecir posibles ataques antes de que ocurran. Esto permite una detección más rápida y precisa de las amenazas cibernéticas, fortaleciendo las defensas contra el ***ransomware*** y otros tipos de ataques.

La integración de la inteligencia artificial y el aprendizaje automático en las estrategias de ciberseguridad es fundamental para mejorar la capacidad de las organizaciones y los usuarios para protegerse de las amenazas en constante evolución en el panorama cibernético actual. Esta combinación de tecnologías avanzadas ayuda a fortalecer las defensas y a anticiparse a posibles ataques, contribuyendo a una mayor seguridad en línea.

Los ataques de ***phishing***, que implican la suplantación de identidad para engañar a las personas y obtener información confidencial, son una amenaza común en ciberseguridad. Para identificar y prevenir los ataques de ***phishing*** es importante:

Verificar la fuente: si recibes un correo electrónico, mensaje o enlace sospechoso, comprueba la dirección de correo electrónico del remitente o la URL del sitio web. Los ciberdelincuentes a menudo utilizan direcciones falsas o similares a las legítimas para engañar a las víctimas.

Sospechar solicitudes urgentes o inusuales: ten cuidado con los mensajes que solicitan información personal, contraseñas o pagos urgentes. Los atacantes suelen crear un sentido de urgencia para presionar a las personas a actuar sin pensar.

Atender errores gramaticales y ortográficos: los correos electrónicos de ***phishing*** a menudo contienen errores gramaticales o de ortografía. Presta atención a estos detalles, ya que pueden ser señales de un intento de ***phishing.***

No hacer clic en enlaces sospechosos: evita hacer clic en enlaces en correos electrónicos o mensajes sospechosos. En su lugar, visita el sitio web oficial directamente escribiendo la URL en el navegador.

Utilizar software de seguridad: mantén tu ***software*** antivirus y antimalware actualizado para detectar y bloquear posibles amenazas de phishing.

Capacitar y concientizar: educa a los usuarios sobre los riesgos del ***phishing*** y la importancia de la seguridad cibernética. Realiza simulacros de ***phishing*** para entrenar a las personas en la identificación de intentos de suplantación de identidad.

Al estar alerta, verificar la autenticidad de las fuentes y seguir buenas prácticas de seguridad en línea, puedes reducir significativamente el riesgo de caer en ataques de ***phishing*** y proteger tu información personaly confidencial.

3.3 Contraseñas y autenticación: mejores prácticas para la gestión de contraseñas y autenticación multifactor

Gestionar contraseñas de forma segura y utilizar la autenticación multifactor son prácticas fundamentales para fortalecer la seguridad en línea (fig. 3.1). Aquí tienes algunas mejores prácticas para la gestión de contraseñas y la autenticación multifactor:

Gestión de contraseñas

- Crea contraseñas únicas y complejas que incluyan letras mayúsculas, minúsculas, números y caracteres especiales. Evita usar información personal fácilmente accesible.
- Utiliza contraseñas diferentes para cada cuenta en línea para evitar la propagación de vulnerabilidades en caso de una violación de datos.
- Utiliza administradores de contraseñas confiables para almacenar y gestionar tus contraseñas de forma segura. Estos programas encriptan tus contraseñas y facilitan el acceso seguro a tus cuentas.
- Cambia tus contraseñas regularmente, especialmente, en cuentas sensibles como el correo electrónico o la banca en línea.

Fig. 3.1 Dispositivo con contraseña segura

Autenticación multifactor (MFA)

- Utiliza la autenticación multifactor siempre que esté disponible. Esta capa adicional de seguridad requiere una segunda forma de verificación más allá de la contraseña, como un código de verificación enviado a tu teléfono (fig. 3.2).
- Utiliza diferentes métodos de verificación, como aplicaciones de autenticación, mensajes de texto, ***tokens*** físicos o biometría, para aumentar la seguridad de tu cuenta.
- Aprovecha las opciones de configuración personalizada de MFA para adaptar la autenticación a tus necesidades de seguridad, como establecer recordatorios para iniciar sesión o limitar los dispositivos autorizados.

Fig. 3.2 Autenticación multifactor

Al seguir estas mejores prácticas de gestión de contraseñas y autenticación multifactor, puedes fortalecer la seguridad de tus cuentas en línea y proteger tu información personal y confidencial de posibles amenazas cibernéticas. ¡Recuerda que la seguridad en línea es un esfuerzo continuo que requiere vigilancia y precaución!

3.4 Buenas prácticas para el uso seguro de internet

Para un uso seguro y responsable de internet, es esencial seguir algunas buenas prácticas que ayudarán a proteger tu privacidad, seguridad y datos en línea.

- Asegúrate de tener instalado un ***software*** antivirus actualizado en tus dispositivos para protegerte contra ***malware*** y otras amenazas cibernéticas.
- Crea contraseñas únicas y complejas para cada cuenta en línea y cámbialas periódicamente. Considera utilizar un administrador de contraseñas para gestionar tus credenciales de forma segura.
- Mantén actualizados tus sistemas operativos, navegadores y aplicaciones para corregir posibles vulnerabilidades de seguridad.
- Sé cauteloso al compartir información personal en línea, como tu dirección, número de teléfono, detalles financieros o información de identificación.
- Antes de hacer clic en enlaces o descargar archivos, verifica la autenticidad de las fuentes para evitar caer en trampas de ***phishing*** o ***malware.***
- Cuando accedas a sitios web sensibles, asegúrate de que la conexión sea segura (HTTPS) y evita conectarte a redes Wi-Fi públicas no seguras.
- Piensa antes de publicar en redes sociales y considera cómo tus acciones en línea pueden afectar tu reputación y privacidad a largo plazo.

Al seguir estas buenas prácticas y mantener una actitud vigilante y responsable en línea, puedes disfrutar de una experiencia segura y protegida en internet. La educación y la conciencia son claves para navegar de manera segura en el mundo digital actual. ¡Protege tu información y disfruta de todo lo que internet tiene para ofrecer de forma segura!

3.5 Uso seguro de redes sociales: riesgos y medidas de seguridad en redes sociales

Cuando se trata del uso seguro de redes sociales, es importante estar consciente de los riesgos asociados y tomar medidas de seguridad adecuadas para proteger tu privacidad y seguridad en línea. Los riesgos más comunes en redes sociales son:

- Compartir demasiada información personal en redes sociales puede exponerte a riesgos de robo de identidad, acoso en línea o fraudes.
- Los ciberdelincuentes pueden utilizar redes sociales para enviar enlaces maliciosos o mensajes de ***phishing*** con el objetivo de obtener información confidencial o infectar tu dispositivo con ***malware.***
- Los perfiles falsos y la suplantación de identidad son riesgos comunes en redes sociales, lo que puede llevar a la difusión de información falsa o la manipulación de la reputación en línea.

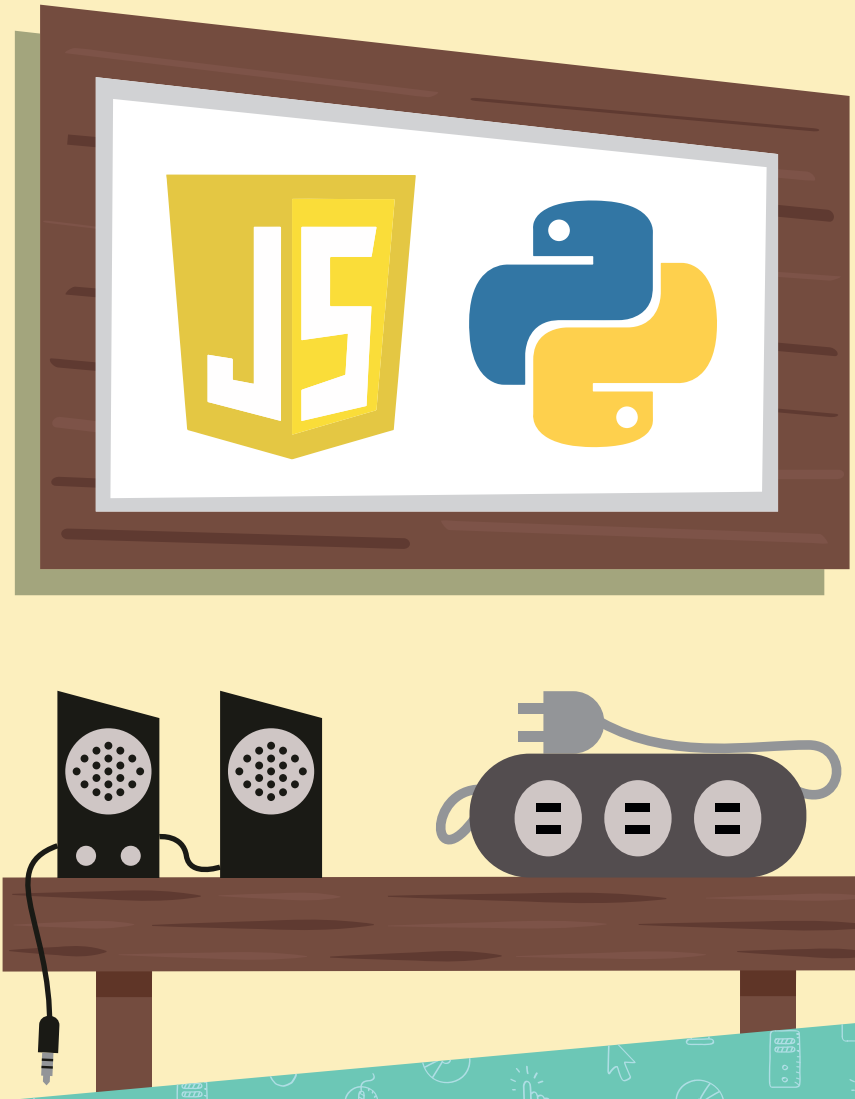
Para evitar estos incidentes en redes sociales es preciso tener en cuenta las siguientes medidas de seguridad en redes sociales:

- Revisa y ajusta la configuración de privacidad de tus cuentas en redes sociales para controlar quién puede ver tu información y publicaciones.
- Antes de aceptar solicitudes de amistad, verifica la autenticidad de la persona y evita conectar con desconocidos.
- Limita la cantidad de información personal que compartes en línea, como tu dirección, número de teléfono o detalles financieros.
- Asegúrate de mantener actualizados tus dispositivos y aplicaciones para protegerte contra posibles vulnerabilidades de seguridad.
- Informa a tus amigos y familiares sobre los riesgos en redes sociales y promueve prácticas seguras en línea, como la verificación de la autenticidad de las fuentes y la protección de la información personal.

Al ser consciente de los riesgos y seguir estas medidas de seguridad en redes sociales, puedes disfrutar de una experiencia más segura y protegida en línea. Recuerda que la precaución y la educación son fundamentales para mantener tu seguridad y privacidad en el entorno digital de las redes sociales.

Comprueba lo aprendido

1. ¿Qué es la ciberseguridad?
2. ¿Quiénes son los atacantes?
3. ¿Qué tipos de ataques cibernéticos existen?
4. ¿Cuáles pueden ser las consecuencias de un ciberataque?
5. ¿Cómo podemos protegernos?
6. ¿Cómo crear una contraseña segura?



EDITORIAL
PUEBLO Y EDUCACIÓN

9 789591 351807