|  |  |
| --- | --- |
| PROGRAMA DE FORMACIÓN | Implementación y gestión de bases de datos |

|  |  |  |  |
| --- | --- | --- | --- |
| COMPETENCIA | 220501096 - Desarrollar la solución de *software* de acuerdo con el diseño y las metodologías de desarrollo. | RESULTADOS DE APRENDIZAJE | 220501096-01- Diseñar algoritmos de acuerdo con el problema a solucionar. |

|  |  |
| --- | --- |
| NÚMERO DEL COMPONENTE FORMATIVO | CF15 |
| NOMBRE DEL COMPONENTE FORMATIVO | Algoritmos, estructuras y operaciones. |
| BREVE DESCRIPCIÓN | Las bases de datos se han convertido en elementos indispensables que precisan manejar grandes o pequeños volúmenes de información, a través de estas se pueden solucionar problemas por medio de instrucciones secuenciales de lenguaje natural, las cuales pueden ser llevadas posteriormente a un programa basado en el lenguaje de programación definido o seleccionado. |
| PALABRAS CLAVE | Algoritmo, estructuras, programación, pseudocódigo, diagrama de flujo. |

|  |  |
| --- | --- |
| ÁREA OCUPACIONAL | Ciencias naturales, aplicadas y relacionadas. |
| IDIOMA | Español |

# **Tabla de contenidos**

**Introducción**

**1. Definición y conceptos de algoritmos**

1.1. Metodología para la solución de problemas usando computadoras

1.2. Concepto de algoritmo

1.3. Construcción de expresiones algorítmicas

1.4. Variables y constantes

1.5. Operaciones aritméticas, relacionales y lógicas

1.6. Representación gráfica de algoritmos

**2. Estructuras de control y funciones**

2.1. Conceptos y aplicación de condicionales

2.2. Estructuras de control

2.3. Contadores y acumuladores

**3. Definición de funciones y procedimientos**

**Introducción**

|  |
| --- |
| Cuadro de texto |
| Estimado aprendiz bienvenido, el presente componente contiene información sobre algoritmos, estructuras y operaciones, temáticas muy importantes en el desarrollo del programa que están cursando. Les invito a ver el video de presentación para identificar el contexto de aprendizaje. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video animación 2D | | | |
| **NOTA** | **La totalidad del texto locutado para el video no debe superar las 500 palabras aproximadamente** | | | |
| **Título** | **Bienvenido al componente formativo** | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración** | **Texto** |
| **1** | Imagen de una persona realizando la bienvenida  welcome to digital  Imagen de referencia <https://cutt.ly/NXFSYy1>  228130\_i1 | N/A | Estimado aprendiz, bienvenido a este componente formativo que se ha denominado **Algoritmos, estructuras y operaciones**. | Algoritmos, estructuras y operaciones. |
| **2** | Colocar imágenes de computador con código y apuntando al centro de este.  Green darts on center of virtual screen of dartboard  Imagen de referencia <https://cutt.ly/iXFSH7p>  228130\_i2 | N/A | El objetivo principal de este componente es entregar información de valor agregado, para aprender a solucionar problemas de la vida cotidiana por medio de *software* de acuerdo a los requerimientos recibidos. | Diseño y metodologías de desarrollo. |
| **3** | Imagen de un cerebro recibiendo órdenes  2d illustration Concept of thinking, background with brain, Abstract Artificial intelligence. Technology web background  Imagen de referencia <https://cutt.ly/hXFS1x1>  228130\_i3 | N/A | Al igual que una persona realiza una acción al recibir una serie de órdenes transmitidas desde su cerebro, cuando le indica que debe hacer y cómo hacerlo. | Órdenes emitidas a través de su cerebro. |
| **4** | Imagen de una persona escribiendo en un computador.  Back-end project architecture development. Database diagram, markup. Integrated Development Environment  Imagen de referencia <https://cutt.ly/XXFDejH>  228130\_i4 | N/A | Algo similar sucede con las acciones que puede realizar un computador, solo que, en ese caso el cerebro es un ser humano que programa las acciones y el paso a paso de lo que debe realizar el computador según las indicaciones dadas. | Humano programa acciones del computador. |
| **5** | Imagen de una persona que realiza una serie de pasos en el computador.  Programming code writing, software coding developing, hands typing computer script..  Imagen de referencia <https://cutt.ly/oXFDkcj>  228130\_i5 | N/A | El computador necesita que se le indiquen los pasos con los que se puede resolver un problema, sin dichas indicaciones, no es posible realizar ninguna acción con estas máquinas. | Paso a paso para resolver un problema. |
| **6** | Imagen de un Computador con un algoritmo en su pantalla.  Personajes aprendiendo programación. Estudiantes resolviendo algoritmos matemáticos, codificando código de programa, mejorando habilidades. Aprender a codificar el concepto. Ilustración vectorial de dibujos animados planos.  Imagen de referencia  <https://n9.cl/nwui4>  228130\_i6  Automatización de procesos de negocio y flujo de trabajo con diagrama de flujo. Esquema de gestión jerárquica de la gestión corporativa y de procesamiento.  Imagen de referencia  <https://n9.cl/jnvyr>  228130\_i7 | N/A | Las instrucciones que se la dan al computador para solucionar un problema se denominan algoritmo. Esto corresponde a una serie de indicaciones que tienen un inicio y un solo final. Además, tienen un orden lógico y secuencial para indicar cómo lograr el objetivo que se propone realizar con el computador. Las partes que componen un algoritmo son: la entrada que hace referencia a los iniciales, el proceso que corresponde a los pasos que se deben realizar para llevar a la solución del problema y, la salida que presenta los resultados obtenidos en el proceso como la solución final. | Algoritmo |
| **7** | Imagen de un Computador que contenga un diagrama de flujo  Diagrama de diagrama de flujo de dibujo de BusinessMan para mostrar el flujo de trabajo y el procedimiento del proceso en el lugar de trabajo, el concepto de flujo de trabajo del proceso  Imagen de referencia  <https://n9.cl/pc97j>  228130\_i8 | N/A | La construcción de los algoritmos se puede realizar a través de gráficos que indican la secuencia de pasos a realizar, los cuales se denominan diagramas de flujo. | Diagrama de flujo, representación gráfica de un algoritmo. |
| **8** | Imagen de una pantalla que presente un código de programación.  Pseudocódigo | ¿Qué es el pseudocódigo? | Parte 1 - YouTube  Imagen de referencia  <https://n9.cl/65esu>  228130\_i9 | N/A | También se pueden construir como instrucciones en lenguaje natural, formando lo que se denomina pseudocódigo, este último es lo más parecido al código que se escribe en un lenguaje de programación y que constituye lo que se denomina programa de computador. | Lenguaje natural, seudocódigo. |
| **9** | Imagen de un computador que presente en su pantalla “Variables”.  Se cambia la información de la pantalla por estructuras de control (condicionales y ciclos).  ✓ Fundamentos de Programación: Variables, Constantes y Palabras Reservadas  Imagen de referencia  <https://n9.cl/ic9zd>  228130\_i10 | N/A | Para la construcción de los algoritmos se utilizan múltiples herramientas que permiten resolver problemas con diferentes niveles de complejidad, como lo son: el uso de las variables, las instrucciones de asignación, las operaciones con sus operadores y las estructuras de control, como los condicionales y los ciclos; los cuales se pueden desglosar en estructuras de decisión simple o estructuras de decisión compuesta. Todos ellos constituyen los recursos básicos para poder representar la solución de problemas a través de la programación. | Variables, estructuras de control, condicionales, ciclos. |
| **10** | Presentar a manera de secuencia el proceso que está en la imagen.  Estructura de control Mientras - Definición y sintaxis  Imagen de referencia  <https://n9.cl/90jdl>  228130\_i11 | N/A | El uso de las secuencias de instrucciones es lo que se conoce como programación estructurada, lo cual presenta un flujo único de pasos con las alternativas que permiten las estructuras de control. Sin embargo, para la solución de problemas más complejos se utilizan diferentes paradigmas que permiten su solución. | Programación estructurada. |
| **11** | Teniendo en cuenta la imagen de referencia colocar dentro de ella: algoritmo, programación, problemas.  Programación y diseño de arquitectura de software con árbol de nodos. Resumen del concepto de tecnología con el lenguaje de modelado unificado UML en persona en el equipo. Visualización de relaciones de datos.  Imagen de referencia  228130\_i12 | N/A | A través de este componente se presenta el uso de los recursos de la programación, para realizar la construcción de los algoritmos. También, se muestran los principios de la programación modular, la cual permite dividir un problema en pequeños subproblemas y abordar la construcción de los algoritmos desde esa subdivisión.  Que este proceso de aprendizaje sea exitoso. | Construcción de algoritmos. |
| **Nombre del archivo** | 228130\_v1 | | |  |

1. **Definición y conceptos de algoritmos**

|  |
| --- |
| Cuadro de texto |
| Se presentan a continuación algunos conceptos relacionados con los algoritmos para tenerlos en cuenta. |

**1.1. Metodología para la solución de problemas usando computadoras**

|  |  |
| --- | --- |
| **Tipo de recurso** | Acordeón tipo 1 |
| **Introducción** | Joyanes (2020) indica que “El proceso de resolución de un problema con una computadora conduce a la escritura de un programa y a su ejecución en la misma. Aunque el proceso de diseñar un programa es - esencialmente- un proceso creativo, se puede considerar una serie de fases o pasos comunes, que generalmente deben seguir todos los programadores” (p. 68).  Para solucionar problemas de una manera efectiva en programación se necesita trabajar con una metodología fácil de entender que indique el paso a paso hasta llegar a la solución, este es básicamente, un hábito o guía que puede aplicarse cada vez que se necesite resolver un problema.  Los pasos de la metodología son: |
| Imagen de referencia <https://n9.cl/2aqif>  228130\_i13 | |
| **1. Análisis previo del problema**  Se debe analizar el escenario inicial de los requerimientos del problema para conocer los datos de entrada, el origen de estos datos, cómo se obtienen, igualmente cómo se relacionan entre ellos y definir o establecer las tareas que se deben realizar en el programa, para llevar a cabo la solución del mismo. De este paso depende la solución de un problema de manera exitosa, ya que hacerlo incorrectamente conlleva a un diseño insuficiente y, posteriormente a un resultado erróneo; convirtiéndose así en uno de los pasos más críticos. | |
| **2. Diseño de la solución**  Después de haber aclarado los requerimientos del problema y de haber definido las entradas y salidas necesarias en el programa, al igual que las tareas a realizar, es el momento de diseñar la solución. En este paso se define qué y cómo lo hará el programaa través de dos formas que son: como unpseudocódigo o como un diagrama de flujo y esto debe ser entendible para cualquier persona. | |
| **3. Codificación**  Es el paso donde se escribe o transcribe una serie de instrucciones específicas, de acuerdo a la lógica establecida en el diseño, utilizando unlenguaje de programación con sentencias y sintaxis propias del mismo. Esta serie de instrucciones son también conocidas como código fuente, el cual se escribe en un lenguaje de alto nivel. | |
| **4. Prueba y Depuración**  Es un proceso dividido en dos tareas. La primera es la prueba del código elaborado en el paso anterior con el fin de ingresar o registrar datos, hasta que el programa no presente errores de tipo sintáctico y lógico; la segunda es la depuración, donde se deben detectar los errores del programa para corregirlos o eliminarlos. | |
| **5. Documentación**  Es un registro detallado de manera escrita de todo lo relacionado con el *software,* por ejemplo, conceptos, solicitudes o acuerdos, procedimientos y diagramas. La documentación tiene como fin ayudar a comprender el uso de un programa y facilitar futuros mantenimientos; no solo para quien que lo diseñó, sino también, para cualquier persona.  Existen tres tipos de documentación:   * Documentación interna: es cualquier tipo de comentario que se escribe en el código fuente, con el fin de hacer más clara la comprensión en un proceso. * Documentación externa: esta documentación se encuentra resumida en el manual técnico del programa y describirá́:   Como instalar el sistema, como usarlo y operarlo, los requisitos y el diseño, la administración del sistema y los procedimientos de prueba. Todo ello, para poderles dar el adecuado mantenimiento y que será útil durante todo el tiempo de vida del sistema.  Adicionalmente, el manual técnico incluirá también: la descripción del problema, el nombre de cada módulo, la fecha de creación o modificación, el nombre de los autores, el algoritmo y/o el diagrama de flujo, el listado de los módulos y procedimientos del programa; así como, el diccionario de datos y el código fuente (programa).   * Manual del usuario: está orientado a los usuarios finales que utilizarán el sistema. Este explica paso a paso el funcionamiento del programa, con el fin de que el usuario final conozca de manera detallada el programa para obtener el resultado deseado. | |

**1.2. Concepto de algoritmo**

|  |
| --- |
| Cajón de texto de color |
| Al hablar de un algoritmo, se hace referencia a un conjunto de instrucciones de forma ordenada, que buscan solucionar un determinado problema y que están relacionadas lógicamente. La palabra algoritmo deriva de la traducción al latín de la palabra árabe *alkhowarizmi*, nombre de un matemático y astrónomo árabe que escribió un tratado sobre manipulación de números y ecuaciones en el siglo IX.  En resumen, un algoritmo es una serie de instrucciones organizadas que describen el proceso que se debe seguir, para solucionar un problema por medio de una computadora. Por tanto, todo algoritmo tiene unas características primordiales que debe llevar a cabo, estas son:  “Debe ser **preciso** e indicar el orden de realización de cada paso.  Debe estar bien **definido**, se debe obtener el mismo resultado cada vez que se lleve a cabo.  Debe ser **finito**, se debe terminar en algún momento”. (Joyanes, 2022, p.60).  Estas tres características permiten definir y estructurar de manera clara y legible un algoritmo confiable que ayuda a resolver el problema.  Por otra parte, la estructura de un algoritmo está compuesta por tres partes: entrada, proceso y salida.   * La entrada se refiere a los datos que se le suministran al algoritmo o programa para iniciar y pueden ser asignados directamente en el programa o suministrados por el usuario por medio de una solicitud de ingreso desde el teclado. * El proceso es la parte donde se llevan a cabo las diferentes acciones con los datos de entrada tales como ciclos, operaciones, cálculos, etc. * La salida es la visualización de los resultados obtenidos en la parte del proceso.   **Representación gráfica de los algoritmos**  Para representar un algoritmo, es necesario seleccionar algún método que permita desligarlo del lenguaje de programación, permitiendo así que el mismo algoritmo pueda ser codificado, sin importar cual sea el lenguaje.  Los métodos más conocidos para representar gráficamente un algoritmo son: diagrama de flujo y lenguaje de especificación de algoritmos: pseudocódigo.  **Diagrama de flujo**  Para Joyanes (2020) un diagrama de flujo o *flowchart* en inglés, es uno de los métodos gráficos de representación de algoritmos más antiguos, que a su vez, es uno de los más utilizados. Aunque en la actualidad su uso se ha reducido, sobre todo, con la aparición de lenguajes de programación estructurados. Un diagrama de flujo es un diagrama que utiliza los símbolos y que tiene los pasos de algoritmo escritos en cajas unidas por flechas, denominadas líneas de flujo, que indican la secuencia en que se debe llevar a cabo.  **Figura 1**  *Ejemplo de un diagrama de flujo*    228130\_i14  Imagen de referencia  *Nota:* Adaptada de *Wikimedia commonds* (2007). <https://es.wikipedia.org/wiki/Diagrama_de_flujo#/media/Archivo:LampFlowchart_es.svg>  **Pseudocódigo**  Según Joyanes (2020), el pseudocódigo (seudocódigo) es un lenguaje de especificación o de descripción de algoritmos. El uso de este, hace que el paso de codificación a un lenguaje de programación sea relativamente más fácil. Es considerado como un primer borrador, dado que el pseudocódigo tiene que traducirse posteriormente a un lenguaje de programación. El pseudocódigo no puede ser ejecutado por una computadora, presenta como ventaja que al planificar un programa, el programador se puede concentrarse en la lógica y en las estructuras de control y no preocuparse de las reglas de un lenguaje específico.  Ejemplo de un pseudocódigo:  INICIO  Escriba “Nombre”  Lea Nombre  Escriba Nombre  FIN  **Algoritmos**  Se definen como una serie de pasos a seguir para solucionar un problema.  Utilizando Elementos de secuencia:   * Entrada * Proceso * Salida   Creando estructuras:   * De decisión: * Simples * Dobles * De control   Secuenciales  Cíclicas o repetitivas   * Estructura Para * Estructura Mientras * Estructura Repita   Emplean la siguiente metodología:   1. Análisis del problema 2. Diseño de algoritmo  * Constantes * Variables * Operadores   + Aritméticos   + Relacionales   + Lógicos |

**1.3. Construcción de expresiones algorítmicas**

|  |  |
| --- | --- |
| **Tipo de recurso** | Tarjetas Animadas |
| **Introducción** | Para realizar la construcción de expresiones algorítmicas se deben tener en cuenta las siguientes consideraciones: |
| 2d illustration Concept of thinking, background with brain, Abstract Artificial intelligence. Technology web background  Imagen de referencia <https://cutt.ly/rXFGFJD>  228130\_i15 | **Analizar el problema**  Analizar la situación teniendo en cuenta los requerimientos del cliente y organizarlos en un orden lógico de pasos para resolver el problema. |
| API, application programming interface to connect between software or application, coding to exchange information for website or software development, programmer connect socket between coding program.  Imagen de referencia <https://cutt.ly/LXFGL0C>  228130\_i16 | **Diseñar el algoritmo**  En este paso se describe la secuencia ordenada de pasos del algoritmo, para diseñar una solución que conduce a resolver el problema citado a través de un diagrama de flujo o pseudocódigo. |
| Concept of computer programming or developing software or game. Vector 3d illustration with coding symbols and programming windows. Concept of Information technologies and computer engineering.  Imagen de referencia <https://cutt.ly/JXFG4bY>  228130\_i17 | **Expresar el algoritmo**  Se debe expresar como un programa en un lenguaje de programación adecuado. También conocido como fase de codificación. |
| Abstract technology background, design circle, digital hi-tech technology with innovative concept start button. Vector illustration.  Imagen de referencia <https://cutt.ly/3XFHQu5>  228130\_i18 | **Ejecución y validación**  Se pone en ejecución el programa realizado en un computador, verificando y eliminando los errores *(bugs).* |

**1.4. Variables y constantes**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tipo de recurso** | Video spot animado | | | |
| **NOTA** | **La totalidad del texto locutado para el video no debe superar las 500 palabras aproximadamente** | | | |
| **Título** | Variables y constantes | | | |
| **Escena** | **Imagen** | **Sonido** | **Narración (voz en off)** | **Texto** |
| **1** | Presentar una persona frente a un computador donde se vea la pantalla que aparezca un algoritmo apareciendo las palabras variables, almacenar valores, identificadores  Cerrar hombre escribiendo código en la computadora portátil  Imagen de referencia <https://acortar.link/U4ULe4>  228130\_i19 | N/A | Siempre que se da inicio al análisis de un problema se deben identifican los datos iniciales, estos datos se estructuran o se definen como **variables**; en estas se pueden almacenar valores, siendo nombradas con identificadores, es decir los nombres que se le asigna poder identificarlas en el algoritmo. | Variables  Almacenar valores  Identificadores |
| **2** | De la imagen que se presentar resaltar base y altura detallando que son las variables.  Interfaz de usuario gráfica, Texto  Descripción generada automáticamente  Imagen anexa como 228130\_i20 | N/A | Tomando, por ejemplo, el problema de calcular el área de un triángulo, el cual presenta como datos iniciales la base y la altura, estos dos datos anteriores se llamarían variables y los nombres que se le asignarán serán “base” y “altura”. | Datos iniciales  Base  Altura |
| **3** | Para cada una de las imágenes presentadas señalar el cambio en los datos (base y altura)  Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente  Imagen anexa como 228130\_i21  Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente  Imagen anexa como 228130\_i22 | N/A | Un elemento importante de estas variables es que pueden ir cambiando su valor durante la ejecución del algoritmo. Se debe tener en cuenta que una variable puede ser declarada, asignada o solicitada, mediante un algoritmo para poder ser utilizada. | Cambiar de valor  Declarada, asignada o solicitada |
| **4** | De la imagen ir mostrando los tipos de datos correspondientes numéricos, carácter y lógicos.  DATOS DE ALGORITMOS | Serpientesenbotas Wiki | Fandom  Imagen de referencia  228130\_i23 | N/A | Una **variable** no es más que un espacio de memoria que un programa utiliza, para conservar un valor que puede cambiar durante la ejecución del mismo. Cuando el programador desarrolla el programa, será el encargado de indicar el nombre o identificador que le asignará a esa zona de memoria. Al disponer esa zona de memoria, hay que detallar que tipo de dato será el que se almacenará en ella y ese tipo no cambiará nunca. | Zona de memoria  Programa  Tipo de dato |
| **5** | Ir presentando la declaración de variables, mostrando cada línea a medida que se describe el texto.  8. Algoritmos || Conceptos básicos de los algoritmos - Aprender a programar  PRO!!  Imagen de referencia  <https://acortar.link/nBLGzA>  228130\_i24 | N/A | Por ende, el espacio se puede ocupar con distintos valores a medida que el programa se ejecuta, aunque todos ellos serán del mismo tipo, que es el tipo de dato que el programador haya definido al crear la variable. Por tanto, cuando el programador desarrolle, establecerá las variables que necesite y el tipo de dato que almacenará cada una. | Valores  Tipo de dato |
| **6** | Presentar la información de cada línea a medida que se describe el texto  Texto  Descripción generada automáticamente  Imagen anexa como 228130\_i25  Interfaz de usuario gráfica, Texto  Descripción generada automáticamente  Imagen anexa como 228130\_i26 | N/A | **Declaración de una variable**  Para utilizar una variable, ya sea a través de pseudocódigo o en un diagrama de flujo, siempre es necesario declararla, es decir, indicarle al algoritmo o diagrama que va a utilizar una variable. Por ejemplo, Nom\_Persona, que es para indicar que se necesita una variable que almacene los nombres de las personas. Ejemplo Nom\_Persona=Nombre de las personas. | Declaración  Pseudocódigo  Diagrama de flujo |
| **7** | Presentar la información de cada línea a medida que se describe el texto  Interfaz de usuario gráfica, Texto  Descripción generada automáticamente  Imagen anexa como 228130\_i27 | N/A | **Asignación de valor a una variable**  Luego de declarar una variable, se puede necesitar que esta inicie por defecto con un valor, por ejemplo, Saldo = 40000, lo que quiere decir que se va a utilizar la variable saldo, pero que esta inicializará con un valor de 40000. Tener en cuenta que se puede inicializar una variable, según el valor que se necesite para solucionar el problema. | Asignación  Inicializar  Solucionar problema |
| **8** | Presentar la información de cada línea a medida que se describe el texto  Texto  Descripción generada automáticamente  Imagen anexa como 228130\_i28  Texto  Descripción generada automáticamente  Imagen anexa como 228130\_i29 | N/A | **Solicitar una variable**  Cuando se necesita que el valor de la variable pueda ser ingresado, por un usuario al computador, entonces debe solicitarse este valor de la variable. | Solicitar  Variable |
| **9** | Presentar la información de cada línea a medida que se describe el texto  Texto  Descripción generada automáticamente  Imagen anexa como 228130\_i30 | N/A | Una **constante** en cambio, es un valor que se almacena en una zona de la memoria, pero que no varía durante la ejecución del programa. Un ejemplo, puede ser el número pi, que siempre permanecerá con el mismo valor. | Constante  No varía  Memoria |
| **10** | Presentar el recuadro de diagrama de flujo y posteriormente la información de pseudocódigo    Imagen anexa como 228130\_i31  Presentar la información de cada línea a medida que se describe el texto  Interfaz de usuario gráfica, Texto, Aplicación  Descripción generada automáticamente  Imagen anexa como 228130\_i32 | N/A | Las constantes también llevan asociado un nombre, es decir, un identificador. Cuando se crea una constante, se especifica el identificador o nombre de la constante y el valor que va a tener (y por tanto también el tipo de dato). Luego, a medida que se vaya necesitando dicha constante en el programa, sólo hay que poner el nombre o identificador de esa constante. Las constantes almacenan datos al igual que las variables, pero su gran particularidad es que sus datos no cambian durante la ejecución del algoritmo; es decir, siempre el valor de la constante va a ser el mismo. | Identificador  Constante |
| **11** |  | N/A |  |  |
| **Nombre del archivo** | **228130\_v2** | | |  |

**Información a tener en cuenta**

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| Las constantes se deben declarar e inicializar. A la acción de crear una variable (o una constante), por primera vez en un programa, se le denomina **declaración de variable (o constante)**. La vida de las variables y constantes dentro de un programa, abarca desde que son declaradas por primera vez hasta la finalización del programa. Todo ese tiempo es lo que se llama **ámbito de la variable (o constante)**. El programador podrá hacer referencia a las variables y constantes utilizando los identificadores, pero siempre dentro del ámbito de las mismas (o sea, desde que son declaradas hasta que el programa termina).  Interfaz de usuario gráfica, Texto  Descripción generada automáticamente  Imagen anexa como  228130\_i33 | |

**1.5. Operaciones aritméticas, relacionales y lógicas**

|  |
| --- |
| Cuadro de texto |
| **Expresiones**  Joyanes (2020) indica que: “Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales”.  Los operadores son elementos que relacionan de forma diferente los valores de una o más variables y/o constantes. Es decir, los operadores permiten manipular valores, para obtener nuevos valores en nuevas o diferentes variables. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Slider Imagen | |
| **Introducción** | A continuación, pueden conocerse los tipos de operadores, lo que representan y una ejemplificación de cada uno de ellos: | |
| **Operadores aritméticos**  Permiten la realización de operaciones matemáticas con las variables y las constantes.  Son utilizados de dos formas con los dos tipos de datos:   * Enteros: si ambos son enteros, el resultado es entero. * Reales: si alguno de ellos es real, el resultado es real. | | **Tabla 1**  *Operadores aritméticos*   |  |  |  |  | | --- | --- | --- | --- | | **Operador** | **Nombre** | **Ejemplo** | **Significado** | | **^** | Exponenciación | x^y | x elevado a y | | **\*** | Multiplicación | x\*y | x multiplicado por y | | **/** | División | x/y | x dividido entre y | | **%** | Módulo o resto | x%y | resto de la división x/y | | **+** | Suma | x+y | x más y | | **–** | Resta | x-y | x menos y | |
| **Prioridad de los operadores aritméticos**  Es el orden lógico establecido como se ejecutan las operaciones aritméticas.  Se evalúan de la siguiente manera:   1. Todas las expresiones que están en paréntesis. 2. Todas las expresiones con paréntesis anidados de adentro hacia afuera. 3. Todas las expresiones dentro del paréntesis interno se evalúan primero.   En una misma expresión, los operadores aritméticos se evalúan en el siguiente orden:   1. ^ Exponenciación. 2. \*, /, mod Multiplicación, división, módulo. 3. +, – Suma y resta.   En una misma expresión con igual nivel de prioridad, los operadores aritméticos se evalúan de izquierda a derecha. | | **N/A** |
| **Operadores relacionales**    El Departamento de Sistemas Informáticos Escuela Técnica Superior de Ingeniería ICAI (2020) indica que “Los operadores se utilizan para formar expresiones lógicas”. | | **Tabla 2**  *Operadores relacionales*   |  |  |  |  | | --- | --- | --- | --- | | **Operador** | **Nombre** | **Ejemplo** | **Significado** | | **<** | Menor que | x<y | x menor que y | | **>** | Mayor que | x>y | x mayor que y | | **==** | Igual a | x==y | x igual a y | | **!=** | Distinto a | x!=y | x distinto de y | | **<=** | Menor o igual que | x<=y | x menor o igual a y | | **>=** | Mayor o igual que | x>=y | x mayor o igual a y | |
| **Operadores lógicos** Se utilizan para establecer relaciones entre valores lógicos. Estos valores pueden ser resultado de una expresión relacional. | | **Tabla 3**  *Operadores lógicos*   |  |  |  |  | | --- | --- | --- | --- | | **Operador** | **Nombre** | **Ejemplo** | **Significado** | | **&&** | AND | (x<y)&&(x<z) | ‘x menor que y’ Y ‘x menor que z’ | | **||** | OR | (x<y)||(x<z) | ‘x menor que y’ O ‘x menor que z’ | | **!** | NOT | !(x<y) | x NO es menor que y | |
| Los operadores AND y OR combinan expresiones relacionales, cuyo resultado viene dado por la última columna de sus tablas de verdad. | | **Ejemplo:**  (a<b) && (b<c)  es verdadero (**true**), si ambas son verdaderas. Si alguna o ambas son falsas el resultado es falso (**false**). En cambio, la expresión  (a<b) ||(b<c) es verdadera, si una de las dos comparaciones lo es. Si ambas, son falsas, el resultado es falso.  La expresión» NO *a* es menor que *b*»  !(a<b)  es falsa si *(a<b)* es verdadero, y es verdadera si la comparación es falsa. Por tanto, el operador NOT actuando sobre *(a<b)* es equivalente a  (a>=b)  La expresión «NO *a* es igual a *b*»  !(a==b)  es verdadera si *a* es distinto de *b*, y es falsa si *a* es igual a *b*. Esta expresión es equivalente a  (a!=b) |

|  |  |
| --- | --- |
| **Tipo de recurso** | Infografía estática |
| **Texto introductorio** | Tipos de operadores |
| **Figura 2**  *Tipos de operadores*    Describir y/o colocar una imagen de referencia, de la infografía solicitada. Incluir cualquier observación o solicitud específica frente al diseño gráfico de la infografía. | |
| **Código de la imagen** | 228130\_i34 |

**1.6. Representación gráfica de algoritmos**

|  |
| --- |
| Cuadro de texto |
| A través de la siguiente imagen se puede ver la explicación de los algoritmos, de acuerdo a su representación gráfica:  **Tabla 4**  *Representación gráfica de algoritmos*    *Nota:* Adaptado de *Olimpiada Oaxaqueña de informática. Gitbook* (2019) (<https://ooi.gitbook.io/courses/temario/introduccion-a-la-programacion/representacion-grafica-de-un-algoritmo>)  228130\_i35 |

**2. Estructuras de control y funciones**

|  |
| --- |
| Cuadro de texto |
| También es conocido como secuenciación o control del flujo. El flujo de control hace referencia a la secuencia en que se ejecutan las instrucciones de un algoritmo o de un programa; el flujo normal de control de todos los programas es secuencial, a menos que se indique expresamente lo contrario o se especifique algo diferente. Esto quiere decir que las sentencias se llevan a cabo en secuencia, seguida una de otra, en el orden en que se encuentran ubicadas en el algoritmo o en el programa.  Las estructuras de selección, repetición o iteración, permiten que el flujo de control del programa sea alterado de una forma precisa y establecida con anterioridad. Estas se usan como lo indica su palabra para seleccionar, elegir o escoger, las instrucciones que se han de llevar a cabo a continuación. Por su parte, las estructuras de repetición (repetitivas o iterativas), se usan para que un conjunto de instrucciones se repita.  En la siguiente información al respecto, se encuentra la indicación de las utilizaciones, las estructuras de control, así como los contadores y acumuladores. |

**2.1. Conceptos y aplicación de condicionales**

|  |
| --- |
| Cuadro de texto |
| Los condicionales se suelen utilizar para tomar decisiones lógicas, son llamadas estructuras selectivas o alternativas, a través de ellas se evalúa una condición y, en función del resultado de la misma, se realiza una opción u otra.  En la siguiente información al respecto, se encuentra la indicación de las utilizaciones, las estructuras de control, así como los contadores y acumuladores. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Pestañas o *tabs* horizontales | |
| **Introducción** | Las estructuras de decisión pueden ser de las siguientes formas: estructuras simples y estructuras dobles. | |
| **Estructuras simples** | Las estructuras simples permiten que se lleven a cabo una o varias acciones o instrucciones determinadas cuando se cumple una condición específica; esta estructura es comúnmente llamada “si-entonces”.  La estructura “si-entonces” analiza la condición y si esta es verdadera, entonces lleva a cabo las acciones que están dentro de la estructura; si la condición es falsa entonces no realiza nada. | **Figura 3**  *Estructuras simples*  Imagen anexa como 228130\_i36  Recrear la imagen, los textos son:  (Tabla)  **Diagrama de flujo**  verdadera – falsa  condición  acción S2  **Pseudocódigo**  Inicio  Si <condición>  <acción> else  fin-si  (Gráfico)  Nota >=6 – si – “APROBADO”  no |
| **Estructuras dobles** | Las estructuras dobles permiten llevar a cabo una o más acciones entre dos opciones o alternativas posibles, si se cumple o no una determinada condición; esta estructura es comúnmente llamada “si-entonces-si\_no”.  La estructura “si-entonces-si\_no” analiza la condición y si esta es verdadera, entonces ejecuta las acciones que están dentro de la estructura Si; si la condición es falsa, entonces realiza las acciones que están dentro de la estructura si\_no. | **Figura 4**  *Estructuras dobles*    Imagen anexa como 228130\_i37  Recrear la imagen, los textos son:  **Diagrama de flujo**  si – no  condición  acción S2  acción S2  **Pseudocódigo**  Si <condición>entonces  <acción S1>si\_no  <acción S2>  fin-si |

**2.2. Estructuras de Control**

|  |
| --- |
| Cuadro de texto |
| Las estructuras de control son acciones o instrucciones que permiten, en general darle solución al problema, pero de una forma lógica, es decir, la que permite dar el orden secuencial para lograr la solución del problema.  Las estructuras de control pueden ser:   * Estructuras secuenciales   Las estructuras secuenciales son conocidas también como estructura lineal, son de las más sencillas y se ejecutan tal cual en el orden que van apareciendo, de arriba hacia abajo.  **Figura 5**  *Estructuras secuenciales*    Imagen anexa como 228130\_i38   * Estructuras cíclicas y/o repetitivas   Las estructuras cíclicas también llamadas repetitivas o bucles, como su nombre lo indica, se repiten realizando la misma acción un determinado número de veces, dependiendo del requerimiento del problema. Existen algunas ocasiones donde se conoce previamente la cantidad de veces que se necesitan llevar a cabo las acciones o instrucciones de un ciclo. |

**Tipos de estructuras cíclicas**

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Pestañas o *tabs* horizontales | |
| **Introducción** | Dentro de los tipos de estructura cíclicas se pueden encontrar: la estructura Para, la estructura Mientras y la estructura Repita. | |
| **Estructura Para** | La estructura Para permite llevar a cabo una o varias acciones o instrucciones, un determinado número de veces.  Es necesario que la variable contadora de control del ciclo sea de tipo entero, tenga un valor inicial, definir el valor limite y el valor de como aumenta o disminuye el contador. | **N/A** |
| **Estructura Mientras** | La estructura Mientras permite llevar a cabo una o varias acciones o instrucciones, mientras el resultado de la condición, sea verdadera.  Cuando se ejecuta la estructura Mientras, se evalúa la condición, si esta es falsa, ninguna acción se realiza y el programa continuo; si es verdadera, entonces se llevan a cabo las instrucciones del bucle.  Este proceso se repite una y otra vez, mientras la condición sea verdadera, durante la ejecución del programa. | **Figura 6**  *Estructuras Mientras*    Imagen anexa como 228130\_i39  Recrear la imagen, los textos son:   |  |  | | --- | --- | | **Diagrama de flujo**  Inicio  Contador = 0  Número  Número >0  Contador = Contador + 1  Número  Cierre  "El número de enteros positivos es" Contador  Fin | **Pseudocódigo**  Inicio  Contador=0  Leer número  Mientras número >0 hacer  Contador=Contador+1  Leer número  Fin mientras  Escribir “El número de enteros positivos es” contador  Fin | |
| **Estructura Repita** | La estructura Repita permite llevar a cabo una o varias acciones o instrucciones en un número indeterminado de veces, hasta que el resultado de la condición sea falso.  A diferencia de la estructura Mientras, que evalúa la condición al inicio del ciclo, la estructura Repita lo hace al final del mismo. Esto es importante, ya que garantiza la ejecución de las instrucciones, mínimo una vez cuando se ejecuta el algoritmo o programa. | **Figura 7**  *Estructura Repita*    Imagen anexa como 228130\_i40  Recrear la imagen, los textos son:   |  |  | | --- | --- | | **Diagrama de flujo**  Inicio suma = 100  Contador = 0  sumatoria = 0  Repite 1,100  Contador = contador + 1  Sumatoria = sumatoria + contador  Imprimir Contador  Imprimir sumatoria  Imprimir [El valor de la suma 1+2+3…+100 es] sumatoria  Fin | **Pseudocódigo**  Inicio  Suma=100  Contador=100  Sumatoria=0  Para 1 hasta 100  Contador=contador=1  Sumatoria=sumatoria+contador  Mostrar contador  Mostrar sumatoria  Fin para  Mostrar “el valor de la suma 1+2+3…+100 es” Sumatoria  Fin | |

**2.3. Contadores y acumuladores**

|  |
| --- |
| Cuadro de texto |
| **Contadores**  Como su palabra lo dice permiten contar, para poder utilizar un contador es necesario inicializarlo en un valor y luego incrementar su valor de una manera constante, para permitir realizar el conteo. Se considera que es una variable, a través de la cual su valor se incrementa o decrementa en una cantidad constante, cada vez que se produce un determinado suceso o acción. La finalidad que estos tienen es la de contar sucesos o acciones internas de un bucle; realizando una operación de inicialización y posteriormente las sucesivas de incremento o decremento del mismo. La inicialización consiste en asignarle al contador un valor. Se situará antes y fuera del bucle.  **Figura 8**  *Ejemplo de estructura de contadores*  Imagen anexa como 228130\_i41  Representación:  <nombre del contador> = <nombre del contador> + <valor constante>  Si en vez de incremento es decremento se coloca un menos en lugar del más.  Ejemplo: i = i + 1  **Figura 9**  *Ciclo Para*    Imagen anexa como 228130\_i42  **Acumuladores**  Como su palabra lo dice permiten acumular el valor de una variable, para poder utilizar un acumulador es necesario inicializarlo en un valor y luego iniciar con la acumulación del valor.  El algoritmo que se presenta anteriormente se esquematiza a través de un diagrama de flujo y lo que hace es inicializar un valor en 1 hasta 100, hasta imprimir la suma de cada dígito, acumulándolo hasta 100. En este caso sería 1+2+3+4+5.....+100. |

**3. Definición de funciones y procedimientos**

|  |
| --- |
| Cuadro de texto |
| **Procedimientos**  Rojas (2017) dice que los “procedimientos son módulos que se utilizan para elaborar un conjunto de instrucciones, independientes y reutilizables, pero definiendo claramente el significado y la utilidad de esa porción de programa” (Cohesión) (p. 35). En otras palabras, se puede decir que los procedimientos son fragmentos o pedazos de códigos que se pueden reutilizar dentro de la ejecución de un programa.  **Funciones**  Rojas (2017) dice que “Son módulos que tienen la misma definición de un procedimiento, pero con la particularidad de que permite devolver un valor en una variable” (p. 36).  Las funciones prácticamente son lo mismo que los procedimientos, pero, con la diferencia de que estas devuelven un valor al programa que las llamó.  **Declaración de funciones**  En pseudocódigo:  **SINTAXIS**  Función nombre\_función (lista de parámetros a recibir):  Tipo  declaraciones locales  Hacer  cuerpo ejecutable de la función  nombre\_función = <expresión o valor calculado dentro de la función> // 1 //  Fin Hacer  Fin Funcion  **Descripción**  **nombre\_función:** es el nombre de la función.  **lista de parámetros:** es la lista de parámetros formales y no puede estar vacía.  **Tipo:** es el tipo de dato que devuelve la función.  // 1 // En el cuerpo de la función debe existir una sentencia que asigne valor explícitamente al nombre de la función.  **Invocación a las funciones**  Para que una función sea ejecutada, ésta debe ser llamada desde un programa principal, otras funciones o procedimientos haciendo referencia a su nombre.  Joyanes (2020) indica que “Cada vez que se llama a una función desde el algoritmo principal, se establece automáticamente una correspondencia entre los parámetros formales y los reales. Debe haber exactamente el mismo número de parámetros reales que de formales en la declaración de la función” (p. 214).  Ejemplo en pseudocódigo:  nombre\_función (lista de parámetros).  La sentencia nombre\_función con los parámetros, es la que realiza el llamado de la función. El control de ejecución lo toma la función, lleva a cabo secuencialmente las instrucciones y cuando termina, le devuelve el control al programa llamador, ejecutando el paso inmediatamente siguiente a la de la llamada. El resultado lo devuelve en el nombre de la función. |

|  |  |
| --- | --- |
| **Tipo de recurso** | Rutas / Pasos. Verticales 1 |
| **Introducción** | Una llamada a una función implica los siguientes pasos: |
| Programming code icon. 3D Low polygonal abstract programming code symbol in hand. Coding or Hacker background. Development and software concept. Vector Illustration.  Imagen de referencia <https://cutt.ly/xXHFCdu>  228130\_i43 | |
| **Paso 1** | **Asignar valor**  A cada parámetro formal se le asigna el valor real de su correspondiente parámetro actual (cabe destacar que se dice “real” haciendo referencia al valor verdadero, con el cual va a trabajar el subprograma, y no al tipo de dato). |
| **Paso 2** | **Ejecución**  Se ejecuta el cuerpo de acciones o instrucciones de la función. |
| **Paso 3** | **Devolver valor**  Se devuelve el valor de la función al nombre de la función y se retorna al punto de llamada. |

**Procedimientos**

|  |
| --- |
| Cuadro de texto |
| Los procedimientos son fragmentos o pedazos de códigos que se pueden reutilizar dentro de la ejecución de un algoritmo o programa. |

|  |  |
| --- | --- |
| **Tipo de recurso** | Rutas / Pasos. Verticales 1 |
| **Introducción** | Todo procedimiento, al igual que un programa principal, consta de una cabecera, que proporciona su nombre y sus parámetros de comunicación; de una sección de declaraciones locales y el cuerpo de sentencias ejecutables. Las ventajas más destacables de usar procedimientos son: |
| Two 3D low-poly hands touching each other.  Imagen de referencia <https://cutt.ly/fXHZpCt>  228130\_i44 | |
| **Ventaja 1** | Facilitan el diseño *top-down*. |
| **Ventaja 2** | Se pueden llevar a cabo más de una vez en un programa, con sólo llamarlos las veces que así se desee. Con esto se ahorra tiempo de diseño y programación. |
| **Ventaja 3** | El mismo procedimiento se puede usar en distintos programas. |
| **Ventaja 4** | Su uso facilita la división de tareas entre los diferentes programadores de un equipo. |
| **Ventaja 5** | Se pueden probar individualmente e incorporarlos en librerías o bibliotecas, para ser usados posteriormente por quien lo necesite. |

|  |
| --- |
| Cuadro de texto |
| **Declaración de procedimiento:**  En pseudocódigo:  **SINTAXIS**  procedimiento nombre procedimiento (lista de parámetros):  Tipo  declaraciones locales  Hacer  cuerpo ejecutable del procedimiento  Fin procedimiento  **Ámbito variables locales y globales**  Las variables utilizadas en los programas principales y subprogramas se clasifican en dos tipos: variables locales y variables globales. |

|  |  |
| --- | --- |
| **Tipo de recurso** | Tarjetas Conectadas |
| **Introducción** | A continuación, se muestra una breve explicación de ambos tipos de variables: |
| Abstract background with binary code. Hackers, darknet, virtual reality and science fiction concept.  Imagen de referencia <https://cutt.ly/rXHX7br>  228130\_i45 | |
| Colombia map abstract schematic from blue ones and zeros binary digital code with space stars for banner, poster, greeting card. Vector illustration.  Imagen de referencia <https://cutt.ly/GXHCC70>  228130\_i46 | **Variables locales**  Joyanes (2020) indica que “Una variable local es aquella que está declarada y definida dentro de un subprograma, en el sentido de que está dentro de este subprograma y es distinta de las variables con el mismo nombre declaradas en cualquier parte del programa principal” (p. 223). |
| Futuristic global network and tacit digital data transfer 3D graphic . Concept of smart digital transformation and technology disruption that changes global trends in new information era .  Imagen de referencia <https://cutt.ly/YXHCi8W>  228130\_i47 | **Variables globales**  Joyanes (2020) indica que “Es aquella que está declarada para el programa o algoritmo principal, del que dependen todos los subprogramas” (p. 233). |

|  |
| --- |
| Cuadro de texto |
| **Comunicación con subprogramas**  **Un parámetro:** es un valor (puede ser una variable, una constante o una expresión) que se pasa desde el programa principal al subprograma (función o procedimiento) para que puedan ser usados en este.  **Un subprograma:** tiene su propia estructura (nombre, declaración de variables y sus respectivos parámetros).  **Paso de parámetros**  Es la forma en que los diferentes subprogramas se comunican, la información entre ellos, es a través de la lista de parámetros y se establece una correspondencia entre los diferentes parámetros. |

|  |  |  |
| --- | --- | --- |
| **Tipo de recurso** | Infografía interactiva Punto caliente | |
| **Texto introductorio** | Los parámetros pueden ser clasificados como: | |
| Salidas (S)  Entradas/Salidas (E/S)  Importante  Entradas (E) | | |
| **Código de la imagen** | 228130\_i48 | |
| **Punto caliente 1** | Las entradas proporcionan valores desde el programa que llama y se utilizan dentro del procedimiento. | Entradas (E) |
| **Punto caliente 2** | Las salidas proporcionan los resultados del subprograma. | Salidas (S) |
| **Punto caliente 3** | Un solo parámetro se utiliza para mandar argumentos a un programa y para devolver resultados. | Entradas/Salidas (E/S) |
| **Punto caliente 4** | Los métodos más empleados para realizar el paso de parámetros son:   * Paso por valor (parámetro valor). * Paso por referencia o dirección (parámetro variable). * Paso por nombre. * Paso por resultado. | Importante |

|  |
| --- |
| Cuadro de texto |
| **Paso por valor**  “No se hace diferencia entre un parámetro que es variable; constante o expresión, ya que solo importa el valor del parámetro” (Joyanes, 2020, p. 227).  **Paso por referencia**  Son los parámetros que están precedidos por la palabra reservada ref., que indica que sólo reciben valor en el subprograma, o bien proporcionan valor al subprograma y reciben un valor nuevo en el mismo. Así que todo cambio realizado sobre los parámetros formales precedidos por “ref.”, se refleja en los parámetros reales correspondientes. Se los considera como parámetros bidireccionales o variables, ya que son de Entrada y/o Salida.  **Recursión (recursividad)**  La recursión es un método por medio del cual un subprograma puede ser llamado por el mismo. La escritura de un procedimiento o función recursiva es similar a sus homónimos no recursivos. Sin embargo, para evitar que la recursión continúe indefinidamente, es preciso incluir una condición de terminación dentro del subprograma. |

|  |
| --- |
| Cuadro de texto |
| Para complementar la temática abordada en este componente, relacionado con los algoritmos, se puede descargar el siguiente documento infográfico que contiene la explicación al respecto:  **Descargar** |

|  |  |
| --- | --- |
| **Tipo de recurso** | Síntesis |
| Tecnólogo en implementación y gestión de bases de datos  Síntesis: Algoritmos, estructuras y operaciones | |
| **Introducción** | En este componente de Algoritmos, estructuras y operaciones, se dio una inmersión en los diferentes temas y conceptos necesarios para aprender a programar y desarrollar la mentalidad algorítmica que se requiere como programador. Esto permitirá al aprendiz enfrentar cualquier tipo de problema dentro de una organización, para plantear y llevar a cabo una solución óptima para la misma. Estructurar la mente de tal forma que entrenará capacidades muy demandadas en el mercado laboral actual, tales como el análisis de requerimientos, la resolución de problemas bajo mucha presión, la organización y gestión del trabajo en forma de micro tareas, entre muchas otras. La mente analítica como programador será uno de los mayores activos.  La siguiente figura presenta a manera resumen la temática abordada: |
| **Figura 10**  *Síntesis*    Imagen anexa como 228130\_i49 | |

|  |
| --- |
| Cuadro de texto |
| Se invita a explorar los demás recursos que se encuentran disponibles en este componente navegando a través del menú principal, en donde se encuentra la síntesis, una actividad didáctica, el material complementario, entre otros. |

**Actividad interactiva**

|  |  |
| --- | --- |
| Tipo de recurso | Cajón de texto de color |
| El respectivo archivo de la actividad didáctica se encuentra en la carpeta Anexos / Anexo - Actividad interactiva CF15. | |

**Material complementario**

|  |  |  |  |
| --- | --- | --- | --- |
| Tipo de recurso | Material complementario | | |
| Tema | Referencia APA del material | Tipo | Enlace |
| Funciones y procedimientos | Garcia, M. (2020). *Fundamentos de programación, algoritmos y estructuras de datos.* | Página web | <http://www.profmatiasgarcia.com.ar/uploads/tutoriales/Algoritmos2.pdf> |
| Algoritmos | Guerrero, L., Villegas E. (2016). *Análisis y diseño de algoritmos, un enfoque práctico*. Editorial Universidad Nacional de Colombia. | Libro PDF | <https://sena-primo.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=sena_elibroELB129991&context=L&vid=SENA&lang=es_ES&search_scope=sena_completo&adaptor=Local%20Search%20Engine&tab=sena_completo&query=any,contains,ALGORITMOS&offset=0> |
| Estructuras de control | López, M, y Silva, E. (2018). *Verificación formal de algoritmos. Ejercicios resueltos.* Universidad de Cádiz. | Libro PDF | <https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_elibroELB33887> |
| Operadores | Muro, A. (2018, mayo 29) *Informática, Operadores, Programación.* | Página web | <https://procomsys.wordpress.com/tag/programacion/> |
| Síntesis | Respuestas rápidas (2020, julio 31). *Cuáles son las principales herramientas ofimáticas.* | Página web | <https://respuestasrapidas.com.mx/cuales-son-los-principales-herramientas-ofimaticas/> |
| Estructuras simples | Rincón, R. (2017). *Introducción y construcción de algoritmos*. Creative commons. | Libro PDF | <https://pdfcookie.com/documents/oa-introduccion-y-construccion-de-algoritmospdf-025ek0myqk21> |
| Algoritmos | Sznajdleder, P. (2012). *Algoritmos a fondo.* Alfaomega. | Libro PDF | <https://sena-primo.hosted.exlibrisgroup.com/primo-explore/fulldisplay?docid=sena_alfaomega17554&context=L&vid=SENA&lang=es_ES&search_scope=sena_completo&adaptor=Local%20Search%20Engine&tab=sena_completo&query=any,contains,ALGORITMOS> |

**Glosario**

|  |  |
| --- | --- |
| **Tipo de recurso** | Glosario |
| **Algoritmo:** | es un conjunto de instrucciones que están relacionadas lógica y ordenadamente, que conllevan a la solución de un problema determinado. |
| **Constante:** | es un valor que se almacena en una zona de la memoria y que no varía durante la ejecución del programa. |
| **Diagrama de flujo:** | representación gráfica de un algoritmo, el cual presenta a través de símbolos los pasos o procesos a seguir para alcanzar la solución de un problema. Su correcta construcción es sumamente importante porque, a partir del mismo se escribe un programa en algún lenguaje de programación. |
| **Estructura cíclica:** | es parte de un algoritmo necesario para repetir una o varias acciones en un número determinado. |
| **Función:** | es un subprograma que recibe como argumentos o parámetros, datos de tipo numérico o no numérico y devuelve un único resultado. |
| **Operadores:** | son elementos que relacionan de forma diferente, los valores de una o más variables y/o constantes. Es decir, los operadores nos permiten manipular valores. |
| **Procedimiento:** | es un subprograma que ejecuta una tarea determinada. |
| **Pseudocódigo:** | es una descripción informal de alto nivel de un algoritmo, que utiliza las convenciones estructurales de un lenguaje de programación verdadero, pero que está diseñado para la lectura. |
| **Sintaxis:** | es el conjunto de reglas que deben seguirse al escribir el código fuente de los programas y así, considerarse como correctos para ese lenguaje de programación. |
| **Variable:** | es un espacio en la memoria para almacenar un valor determinado que puede cambiar durante la ejecución dentro del programa y son definidas por nombres, para identificarlas dentro del algoritmo. |

**Referentes bibliográficos**

|  |  |
| --- | --- |
| **Tipo de recurso** | **Bibliografía** |
| Ayala, G. (2020). *Algoritmos y programación. Mejores prácticas.* Fundación Universidad de las Américas. Puebla UDLAP. <https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_elibroELB180290> | |
| Joyanes, L. (2020). *Fundamentos de programación.* McGraw-Hill. <https://www-ebooks7-24-com.bdigital.sena.edu.co/stage.aspx?il=&pg=&ed=> | |
| López, M, y Silva, E. (2017). *Corrección de algoritmos complejos*: *verificación formal*. Universidad de Cádiz. <https://sena-primo.hosted.exlibrisgroup.com/permalink/f/1j5choe/sena_elibroELB33887> | |
| Rojas, C. (2017). *Algoritmia y estructura de datos. Manual autoformativo*. <https://es.slideshare.net/jdelacruzpalacios/algoritmia-y-estructura-de-datos> | |