

	FACULTAD DE INGENIERÍAS ÁREA DE PROGRAMAS INFORMÁTICOS	Guía # 3
		Versión: 01

ASIGNATURA: Algoritmos y Programación 1	CÓDIGO: ING01179
DOCENTE: Luis Fernando González Alvarán	FECHA: marzo 6/2023

SEMINARIO DE ENCUENTRO NÚMERO 2

PRESEMINARIO # 2

Con el fin de afianzar los conocimientos recibidos en clase, el estudiante debe realizar la siguiente lectura y complementarla con la bibliografía, o los enlaces sugeridos, buscando temas relacionados con la unidad en estudio, posteriormente realizará en el chat de la asignatura, una socialización de lo estudiado manifestando en éste, el tema que considera más importante en la resolución de problemas por computador.

I. METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS POR COMPUTADOR.

Existen una serie de principios que gobiernan el comportamiento de los computadores y programas; los cuales establecen la forma correcta de abordar un problema cuando se pretende resolverlo a partir de un computador. Este conjunto de principios recibe el nombre de Resolución de problemas algorítmicos.

En consecuencia, de lo anterior se hace necesario fundamentar y adquirir un conocimiento firme sobre lo que es un algoritmo, y lo que es un problema algorítmico; y así poder establecer claramente la forma de resolver dichos problemas.

1. CONCEPTO DE ALGORITMO.

Un algoritmo es un método para resolver un problema, aunque formalmente se define como un conjunto de instrucciones, procedimientos o acciones que realizan una descripción paso a paso y precisa de cierto proceso; garantizando alcanzar un resultado o resolver un problema determinado.

Aunque la popularización del termino ha llegado con el advenimiento de la era informática, algoritmo proviene de Mohammed al-KhoWârizmi, matemático persa que vivió durante el siglo IX y alcanzó gran reputación por el enunciado de las reglas paso a paso para sumar, restar, multiplicar y dividir números decimales; la traducción al latín del apellido es la palabra algorismus la cual derivó posteriormente en algoritmo. Euclides, el gran matemático griego (del siglo IV a.C.) que inventó un método para

 <p>POLITÉCNICO COLOMBIANO JAIME GARZÓN</p>	FACULTAD DE INGENIERÍAS ÁREA DE PROGRAMAS INFORMÁTICOS	Guía # 3 Versión: 01
--	---	-------------------------

encontrar el máximo común divisor de dos números, se considera con Al-Khowarizmi el otro gran padre de la algoritmia (ciencia que trata de los algoritmos).

El profesor Niklaus Wirth —inventor de Pascal, Modula-2 y Oberon— tituló uno de sus más famosos libros, Algoritmos + Estructuras de datos = Programas, indicando que solo se puede llegar a realizar un buen programa con el diseño de un algoritmo y una correcta estructura de datos. Esta ecuación será una de las hipótesis fundamentales consideradas en este libro.

Para llegar a la realización de un programa es necesario el diseño previo de un algoritmo, de modo que sin algoritmo no puede existir un programa.

Los algoritmos son independientes tanto del lenguaje de programación en que se expresan como del computador que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en un computador distinto; sin embargo, el algoritmo será siempre el mismo. Así, por ejemplo, en una analogía con la vida diaria, una receta de un plato de cocina se puede expresar en español, inglés o francés, pero cualquiera que sea el lenguaje, los pasos para la elaboración del plato se realizarán sin importar el idioma del cocinero.

En la ciencia de la computación y en la programación, los algoritmos son más importantes que los lenguajes de programación o los computadores. Un lenguaje de programación es tan solo un medio para expresar un algoritmo y un computador es solo un procesador para ejecutarlo. Tanto el lenguaje de programación como el computador son los medios para obtener un fin: conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

Dada la importancia del algoritmo en la ciencia de la computación, un aspecto muy importante será el diseño de algoritmos.

El diseño de la mayoría de los algoritmos requiere creatividad y conocimientos de las técnicas de programación. En esencia, la solución de un problema se puede expresar mediante un algoritmo.

Muchas veces se aplican algoritmos de manera inadvertida, inconsciente o automáticamente. Esto generalmente se produce cuando el problema al que nos enfrentamos se ha resuelto con anterioridad varias veces.

1.1. CARACTERÍSTICAS DE LOS ALGORITMOS.

Las características fundamentales que debe cumplir todo algoritmo son:

- Debe tener un punto particular de inicio.
- Un algoritmo debe ser preciso e indicar el orden de realización de cada paso.
- Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez. No debe permitir dobles interpretaciones.
- Un algoritmo debe ser finito. Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos. Debe ser finito en tamaño y tiempo de ejecución.
- Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.

La definición de un algoritmo debe incluir tres partes: Entrada, Proceso y Salida.

En el ejemplo de una receta de cocina, del cual se habló anteriormente se pueden identificar:

Entrada:	Ingredientes y utensilios empleados.
Proceso:	Elaboración de la receta en la cocina.
Salida:	Terminación del plato (por ejemplo, "Pato a la Naranja")

La resolución de un problema exige el diseño de un algoritmo que resuelva el problema propuesto.

El programador de computadores es antes que nada una persona que resuelve problemas, por lo que para llegar a ser un programador eficaz se necesita aprender a resolver problemas de un modo riguroso y sistemático. A lo largo de todo este libro se hablará de la metodología necesaria para resolver problemas mediante programas, concepto que se denomina metodología de la programación. El eje central de esta metodología es el concepto de algoritmo.

1.2. PROBLEMA ALGORÍTMICO.

Un problema algorítmico es cualquier problema conceptual o práctico cuya solución puede expresarse mediante un algoritmo.

A diario se encuentran muchos problemas algorítmicos, no solamente en el campo de la informática, es decir, muchas de las acciones rutinarias de una persona, obedecen a un problema algorítmico.

Ejemplo.

PROBLEMA ALGORÍTMICO	ALGORITMO
Dar la vuelta a oriente	Itinerario particular del recorrido.
Cambiar una llanta estallada	Conjunto de pasos para el cambio de la llanta

Es importante tener en cuenta que no todos los algoritmos pueden ser ejecutados por un computador, pues los computadores solo pueden ejecutar algoritmos que se componen de acciones individuales que se pueden entender y realizar, por ejemplo la preparación del pollo a la cazadora implica acciones como "deshuesar el pollo", tarea para la que un computador aún no está preparado. Por lo anterior es necesario conocer bien cuáles son las tareas que puede realizar un computador, de forma que se diseñen algoritmos que contengan solo este tipo de tareas, obedeciendo al modelo computacional que se plantea en la figura 1. y que es congruente con las partes de un algoritmo.

Figura 1. Modelo de computación



	FACULTAD DE INGENIERÍAS ÁREA DE PROGRAMAS INFORMÁTICOS	Guía # 3 Versión: 01
---	---	-------------------------

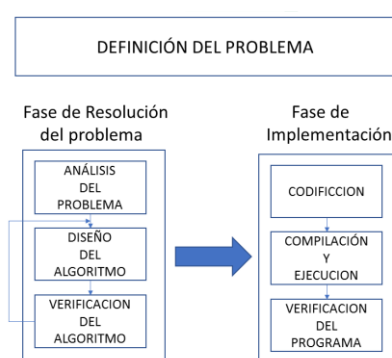
Este modelo plantea que cualquier proceso de cálculo se compone de tres partes: una entrada un proceso y una salida; donde **la entrada** se constituye por un conjunto de información que necesitan los pasos de los cuales se compone el algoritmo para llevar a cabo la tarea; **el proceso** contiene (una descripción de) los pasos del algoritmo; y finalmente **la salida** constituye el resultado que se obtiene ejecutando los pasos con los datos de entrada.

Por ejemplo el procedimiento para calcular la calificación media es un algoritmo que si se puede ejecutar en un computador, donde las entradas corresponden a una lista de valores numéricos dentro del rango 0 a 5; la salida será la media de esos valores; y el proceso es (una descripción de) el conjunto de pasos individuales que deben seguirse para obtener el resultado.

Para llegar a identificar claramente las entradas, procesos y salidas de un algoritmo e implementarlas en un computador se ha definido una metodología la cual se describe a continuación.

II. RESOLUCION DE PROBLEMAS POR COMPUTADOR.

La solución de un problema algorítmico con la ayuda del computador recibe el nombre de resolución de problemas por computador y es una metodología que consta de dos fases cada una compuesta de ciertas etapas y que todas en conjunto conducen a la escritura de un programa y a la ejecución de éste; pero para poder abordar el proceso detalladamente se hace necesario cumplir con una fase preliminar que no hace parte del proceso de resolución como tal, aunque muchos autores la incluyen en éste y que es la definición del problema, ésta apunta al entendimiento claro del problema. Un problema no se puede pretender solucionar, sin tener un claro entendimiento de éste y lo que se espera en su solución.



1.1. DEFINICIÓN DEL PROBLEMA: Esta fase está dada por el enunciado del problema, el cual requiere una definición clara y precisa. Es importante que se conozca lo que se desea que realice el computador; mientras esto no se conozca del todo no tiene mucho caso continuar con el proceso. La definición del problema va estrechamente ligada con el entendimiento detallado de éste. No es posible definir con certeza una solución, si no se tiene claridad acerca de la dimensión del problema.

Las etapas para la resolución de un problema por computador se agrupan en dos grandes fases:

1.2. FASE RESOLUCIÓN DEL PROBLEMA: Esta corresponde a todas las actividades previas al uso del computador en la solución de un problema. Se compone de las siguientes actividades:

1.2.1. Análisis del Problema: En esta etapa se hace una clara interpretación del problema, donde se contempla exactamente lo que debe hacer el programa y el resultado o solución deseada. Una vez que se ha comprendido lo que se desea del computador, es necesario definir:

1.2.1.1. Modelamiento: Obedece a la identificación y descripción de los datos de entrada, salida y auxiliares; cada uno de estos asociados a un respectivo identificador. También se puede relacionar cierta información auxiliar que ayudará a la solución del problema, pero que no requiere estar asociada con un identificador.

Es de aclarar que no necesariamente tienen que existir los datos auxiliares, e incluso en ciertas ocasiones puede suceder con los de entrada.

- **Datos de entrada.** Corresponde a los datos que se requieren para la solución del problema. Son aquellos que no se conocen y que no existe forma de deducirlos en el transcurso de la solución.
- **Datos Auxiliares e información auxiliar o adicional.** Los datos auxiliares son aquellos que son conocidos desde el enunciado del problema, o que pueden obtenerse a partir de los datos de entrada y una información adicional, pero que no obedecen a los resultados esperados y que van a ser almacenados en una variable o definidos como una constante. Estos sirven para el logro de la solución.

La información auxiliar o adicional como su nombre lo indica, obedece a cierta información que se suministra en el enunciado y que es fundamental para la definición de procesos, condiciones o validaciones que se utilizarán para la solución. Estos no requieren ser almacenados en una variable.

- **Datos de Salida.** Obedece a la información que se desea producir como resultado, es decir corresponden a la solución del problema planteado.

Una recomendación muy práctica, es que, a la hora de analizar el problema planteado, la persona que analiza se coloque en el lugar del computador y piense que es lo que se necesita que se le ordene y en que secuencia, para producir los resultados esperados. Este ejercicio ayuda a identificar los datos de entrada, salida y auxiliares

Ejemplo:

Dado un número positivo obtener el resultado de elevar al cuadrado el producto de dicho número multiplicado por 3.

DATO DE ENTRADA

Descripción
Numero dado

Identificador
Num

DATOS AUXILIAR	
Descripción	Identificador
Producto del número por 3	Producto
DATO DE SALIDA	
Descripción	Identificador
Cuadrado del producto	Cu_del_P

1.2.1.2. Especificaciones: Obedece a un análisis intuitivo del estado de los identificadores definidos, específicamente de aquellos que almacenarán los diferentes datos que van a hacer parte de la solución, donde se describen como están éstos antes y después de que se haya ejecutado un paso del algoritmo. Existen dos pasos de interés especial: el estado inicial que recibe el nombre de *precondición* del algoritmo (o pre); y el estado final que recibe el nombre de *postcondición* del algoritmo (o post)

- **Precondiciones.** Corresponde a una descripción del estado inicial de los datos de entradas, es decir antes de que se ejecute el primer paso del algoritmo; por lo tanto, corresponde a las especificaciones de los identificadores de entrada. En esta descripción se indica el tipo de dato de cada una de las variables, e incluso se deben establecer los límites de los datos, factor que posteriormente determinará las condiciones de validación
- **Postcondiciones.** Corresponde a la descripción de las variables auxiliares, de salida y se establece el valor de las constantes, definiendo claramente el tipo de dato de éstos.

Ejemplo: (basado en el modelamiento anterior)	
Precondiciones	
Num	$\in \{\text{Reales} > 0\}$
Postcondiciones	
Producto	$\in \{\text{Reales}\}$
Cu_del_P	$\in \{\text{Reales}\}$

1.2.1.3. Procesos y/o formulas: Corresponde a la definición de métodos y/o fórmulas que se necesitan utilizar para procesar los datos y llegar a obtener el valor de las variables auxiliares y de salida, se deben plantear en estricto orden de uso.

Ejemplo (basado en el anterior)	
Fórmulas	
Producto	$\leftarrow \text{Num} * 3$
Cu_del_P	$\leftarrow \text{Producto} ^ 2$

El estricto orden de planteamiento se refiere a que (según el ejemplo) no se puede plantear el proceso de **Cu_del_P** sin previamente haber definido a **Producto**, ya que la primera implica haber calculado la segunda.

1.2.2. Diseño del Algoritmo: Una vez se tenga claridad del problema, el modelamiento, las especificaciones y los procesos y/o fórmulas; se diseña una solución que obedece a un algoritmo que resuelva el problema, el cual consiste en hacer una descripción de los pasos lógicos que dan solución al problema. Dicha solución se puede realizar de forma gráfica (diagrama) o no grafica (pseudocódigo) como se puede observar en la figura

Es de aclarar que el diseño corresponde a una solución basada en el análisis previamente realizado.



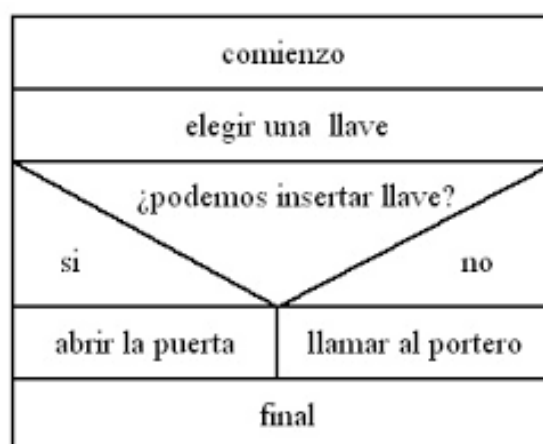
Aunque este curso se centra en el diseño de soluciones a través de pseudocódigos, a continuación, se da una breve descripción de los algoritmos gráficos.

1. Diagrama NASSI – SCHNEIDERMAN

Conocido como diagrama N-S, se considera como otra forma de representar algoritmos. Se basa en escribir las instrucciones en bloques o cuadros de texto. Este diagrama también se conoce con el nombre de diagrama de Chapin y es una combinación de la escritura en pseudocódigo y del diagrama de flujo.

En el diagrama N-S, las flechas que indican el flujo del algoritmo, se reemplazan por las cajas de texto, en las cuales se escriben las instrucciones respectivas. La sintaxis general se indica a continuación:

Como se puede apreciar, cada bloque contiene una instrucción, en el caso de definición de variables, se recomienda utilizar un bloque por cada tipo de datos distinto que se genere, es decir si existen varios datos enteros, su definición abarcaría un solo bloque, si por el contrario se requieren definir datos dobles, ellos ocuparían otro bloque y así sucesivamente.








2. Diagrama de flujo

El diagrama de flujo es una forma de representar una serie de instrucciones, con símbolos estandarizados por el Instituto Nacional Americano de Estándares, ANSI (por sus siglas en inglés), mediante los cuales se representan de manera gráfica un

algoritmo y permiten visualizar el flujo de datos cuando se procesan y hacia donde se dirigen dichas salidas de cada proceso para ingresar en el siguiente, hasta que termine el algoritmo.

Los símbolos que se emplean en un diagrama de flujo se definen en la figura que se presenta a continuación:

Símbolo	Nombre	Función
	Inicio / Final	Representa el inicio y el final de un proceso
	Línea de Flujo	Indica el orden de la ejecución de las operaciones. La flecha indica la siguiente instrucción.
	Entrada / Salida	Representa la lectura de datos en la entrada y la impresión de datos en la salida
	Proceso	Representa cualquier tipo de operación
	Decisión	Nos permite analizar una situación, con base en los valores verdadero y falso

3. Pseudocódigo

Es la herramienta de programación para la representación no gráfica de algoritmos donde las instrucciones se escriben en palabras similares al lenguaje natural, facilitando así la escritura como la lectura de programas. Aunque no existen reglas para la escritura de un pseudocódigo en español hay una serie de notación estándar que se utilizarán durante el curso y que ya está muy empleada por algunos libros.

3.1. Sintaxis General de un Pseudocódigo

Algoritmo_<nombre>

Inicio

//Variables.

Entero : <lista de variables enteras>

Real : <lista de variables enteras>

Cadena : <lista de variables enteras>

Carácter : <lista de variables enteras>

Lógica : <lista de variables enteras>

} Diferentes estructuras

Fin

	FACULTAD DE INGENIERÍAS ÁREA DE PROGRAMAS INFORMÁTICOS	Guía # 3 Versión: 01
---	---	-------------------------

- Donde <nombre> corresponde al nombre que se le asigna a la solución (lo escoge el usuario y debe cumplir con las reglas para identificadores).
- En variables solo se enumeran los tipos de variables que hayan resultado en el análisis. Las variables, como se verá más adelante, también se pueden definir antes del inicio, pero en casos muy específicos.
- Entre inicio y fin irán las diferentes estructuras que se van trabajando, las cuales pueden ser:
 - Secuenciales
 - Selectivas o de decisión
 - Cíclicas o repetitivas

Nota. Obsérvese que en la estructura de un pseudocódigo se debe conservar una adecuada indentación, que indica la dependencia de una instrucción o un bloque de instrucciones con relación a la anterior.

Indentación. Este término significa mover un bloque de texto hacia la derecha insertando espacios o tabuladores, para así separarlo del margen izquierdo y distinguirlo mejor del texto adyacente.

3.2. Estructuras Secuenciales

La estructura secuencial es aquella en la que una acción (instrucción) 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. Una estructura secuencial se representa de la siguiente forma:

```

Inicio
  //Variables
  -----
  -----
  Accion1
  Accion2 } Estructuras secuenciales
  .
  .
  AcciónN
  Fin
  
```

Las estructuras secuenciales se dividen en tres:

- **Lectura:** La lectura consiste en recibir desde un dispositivo de entrada (p.ej. el teclado) un valor. Esta operación se representa en un Pseudocódigo como sigue:

```

Leer (A)
Leer (B)

```

	FACULTAD DE INGENIERÍAS ÁREA DE PROGRAMAS INFORMÁTICOS	Guía # 3 Versión: 01
---	---	-------------------------

Donde “A” y “B” son variables previamente declaradas y que almacenarán los valores ingresados por el usuario.

- **Asignación:** La asignación consiste, en el paso de valores o resultados a una zona de la memoria. Dicha zona será reconocida con el nombre de variable que recibe el valor. La asignación se puede representar de la siguiente forma: \leftarrow
 - Simple: Consiste en pasar un valor constante a una variable ($a \leftarrow 15$)
 - Contador: Consiste en usarla como un verificador del número de veces que se realiza un proceso ($a \leftarrow a + 1$)
 - Acumulador: Consiste en usarla como un sumador en un proceso ($a \leftarrow a + b$)
 - De trabajo: Donde puede recibir el resultado de una operación matemática que involucre muchas variables ($a \leftarrow c + b * 2 / 4$).
- **Escritura:** Consiste en mandar por un dispositivo de salida (por ejemplo, monitor o impresora) un resultado o mensaje. Este proceso se representa en un Pseudocódigo como sigue:

Escribir (“El resultado es:”, R)

Donde “**El resultado es:**” corresponde a un mensaje que se desea que aparezca y R es una variable que contiene un valor previamente ingresado o asignado.

Ejemplo

Calcular la nota definitiva a partir de tres notas parciales que tienen igual porcentaje.

Asumiendo que ya se tiene previamente el análisis el diseño sería el siguiente:

Algoritmo_Nota

Inicio

//Var

Real : Nota_1, Nota_2, Nota_3, Def

Escribir (“Ingrese la Nota uno: “)

Leer (Nota_1)

Escribir (“Ingrese la Nota dos: “)

Leer (Nota_2)

Escribir (“Ingrese la Nota tres: “)

	FACULTAD DE INGENIERÍAS ÁREA DE PROGRAMAS INFORMÁTICOS	Guía # 3 Versión: 01
---	---	-------------------------

Leer (Nota_3)

Def $\leftarrow (Nota_1 + Nota_2 + Nota_3) / 3$

Escribir ("La nota definitiva es: ", Def)

Fin

Problemas Secuenciales: Para los siguientes ejercicios realice el análisis y diseño de la solución.

1) Suponga que un individuo desea invertir su capital en un banco y desea saber cuánto dinero ganara después de un mes si el banco paga a razón de 2% mensual.

2) Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuánto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.

3) Una tienda ofrece un descuento del 15% sobre el total de la compra y un cliente desea saber cuánto deberá pagar finalmente por su compra.

4) Un alumno desea saber cuál será su calificación final en la materia de Algoritmos. Dicha calificación se compone de los siguientes porcentajes:

55% del promedio de sus tres calificaciones parciales.

30% de la calificación del examen final.

15% de la calificación de un trabajo final.

5) Un maestro desea saber qué porcentaje de hombres y qué porcentaje de mujeres hay en un grupo de estudiantes.

6) Realizar un algoritmo que calcule la edad de una persona.

1.2.3. Verificación del algoritmo: Cuando se crea un algoritmo es posible que se cometan errores de tipo lógico; estos errores no pueden ser detectados por el computador, es decir, los resultados que éste arroje simplemente son errados. El proceso de verificación permite detectar los posibles errores que se cometan en el diseño de un algoritmo y así corregirlos antes de continuar con el proceso. Esta verificación se conoce como prueba de escritorio y corresponde a un seguimiento de las secuencias del algoritmo, a partir de datos extremos. Los resultados obtenidos se analizan comparativamente con resultados reales y así se puede establecer el correcto funcionamiento o no del algoritmo. En caso de encontrarse inconsistencias, se procede a la búsqueda y corrección del error.

1.3. FASE DE IMPLEMENTACIÓN.

1.3.1. Codificación: La solución sin errores lógicos se escribe en un lenguaje de programación de alto nivel.

	FACULTAD DE INGENIERÍAS ÁREA DE PROGRAMAS INFORMÁTICOS	Guía # 3
		Versión: 01

Simplemente se convierten uno a uno los pasos del algoritmo a instrucciones del lenguaje seleccionado, lo anterior se realiza en un editor de textos o en el propio compilador del lenguaje; cumpliendo con la sintaxis específica de éste. El resultado de este proceso recibe el nombre de programa fuente.

1.3.2. Compilación y ejecución: La compilación es un proceso de traducción de programas fuentes a programas objeto, durante el cual se realiza el análisis de cada una de las instrucciones escritas en el lenguaje de programación seleccionado, acompañado de una rigurosa detección de errores de sintaxis (denominados bugs). El programa resultado después de la compilación objeto corresponde a un código de máquina que no posee errores y el cual será ejecutado.

1.3.3. Verificación del Programa: Esta fase está estrechamente ligada con la anterior y corresponde a un proceso de pruebas y refinamiento del código, acorde con un periodo donde se analizan los resultados obtenidos y se comparan con resultados reales.