

Técnico Universitario en Programación
UNIVERSIDAD TECNOLÓGICA NACIONAL
Facultad Regional Gral. Pacheco

Apuntes de clase de la asignatura

Programación I

CONCEPTOS GENERALES

2024

Abel Oscar Faure

Lorena Raquel Palermo



INTRODUCCIÓN

La Informática se ha constituido en el pilar fundamental, de la tecnología moderna y del desarrollo de la civilización misma, siendo su elemento central la computadora.

La computadora no solamente es una máquina que puede realizar procesos para darnos resultados, sin que tengamos la noción exacta de las operaciones que realiza para llegar a esos resultados. Con la computadora además de lo anterior también podemos diseñar soluciones a la medida, de problemas específicos que se nos presenten. Más aun, si estos involucran operaciones matemáticas complejas y/o repetitivas, o requieren del manejo de un volumen muy grande de datos. El diseño de soluciones a la medida de nuestros problemas requiere como en otras disciplinas una metodología que nos enseñe de manera gradual, la forma de llegar a estas soluciones. A las soluciones creadas por computadora se les conoce como **programas**.

Aprender a programar una computadora, es aprender a sacar el máximo provecho de las capacidades que esta posee. La programación de computadoras centra todo su trabajo en el manejo de conjuntos de instrucciones estructuradas en forma lógica, que traen como consecuencia una lógica de funcionamiento. Es por esto que una de las mejores formas de conocer mejor como funciona una computadora, es conocer como se la programa. Las personas que aprenden a programar una computadora se sumergen en toda la estructura lógica que rige su funcionamiento, y esto facilita una mejor comprensión de esta.

Hoy en día la habilidad de programar no solo tiene campo de aplicación en el mundo de las computadoras, también existen infinidad de dispositivos electrónicos, que basan su funcionamiento en un programa insertado en un circuito integrado. El control remoto de nuestro televisor es un buen ejemplo de esta situación. Esto le ha permitido a la electrónica dominar un amplio rango de aplicaciones, sobretodo de gran complejidad, ya que todo depende de la habilidad para programar y al mismo tiempo de la creatividad que tenga quién va a desarrollar una determinada solución.



¿QUÉ ES PROGRAMAR?

Programar una computadora consiste en indicarle a la máquina lo que queremos que haga. Para cumplir con este objetivo, se debe establecer un medio de comunicación entre la persona y la máquina, para poder darle las instrucciones necesarias que le permitan hacer su tarea. Aunque en la actualidad se han desarrollado medios para instruir a la computadora por medio de la voz, el medio de comunicación más efectivo se establece a través de instrucciones escritas.

Como resultado de programar una computadora, se obtiene un programa que puede realizar infinitas veces la tarea programada. Pero, la necesidad de pedirle a la computadora que nos ayude a realizar una determinada tarea esta dada por que existen uno o varios problemas que deseamos que esta nos ayude a resolver. No olvidemos que una de las características fundamentales de la computadora consiste en ser un ayudante del cerebro, capaz de realizar cálculos a gran velocidad.

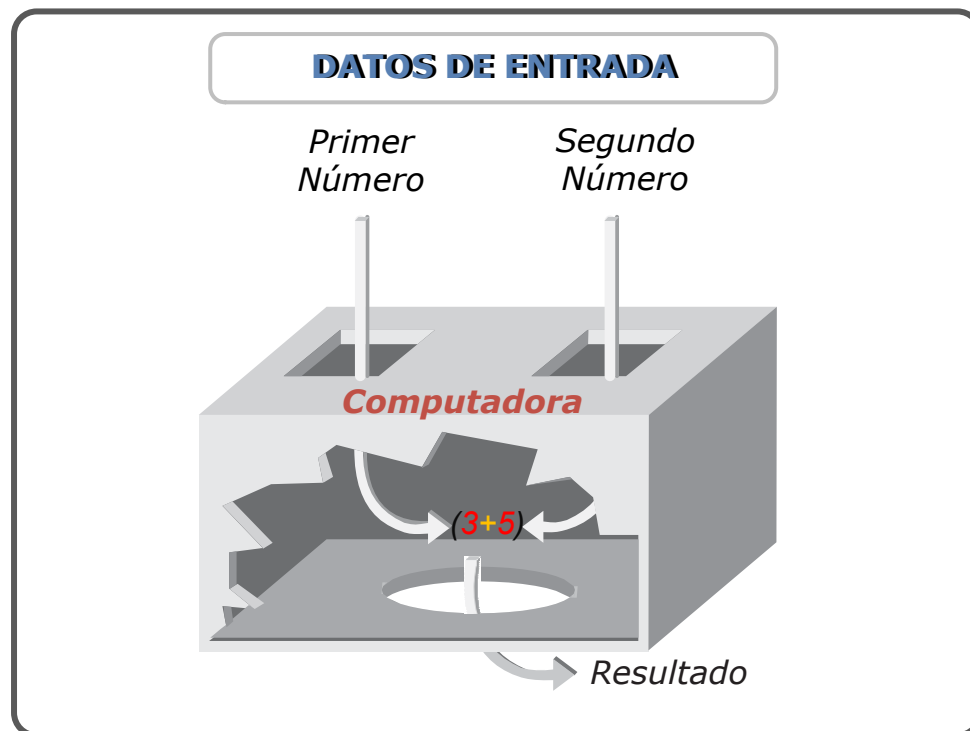
Un **programa de computadora** es un conjunto independiente de instrucciones usado para operar una computadora con el fin de producir un resultado específico.

El proceso de escribir un programa, se llama **programación**, mientras al conjunto que puede usarse para construir un programa se llama **lenguaje de programación**.



IDENTIFICADORES DE MEMORIA

Los datos para procesar, así como los resultados de operaciones producidos por una computadora, sin importar de qué tipos de datos sean (Simples o Estructurados) deben almacenarse en la computadora, de esta manera podrán ser utilizados por el programa y cumplir con la tarea para la cual fueron creados.

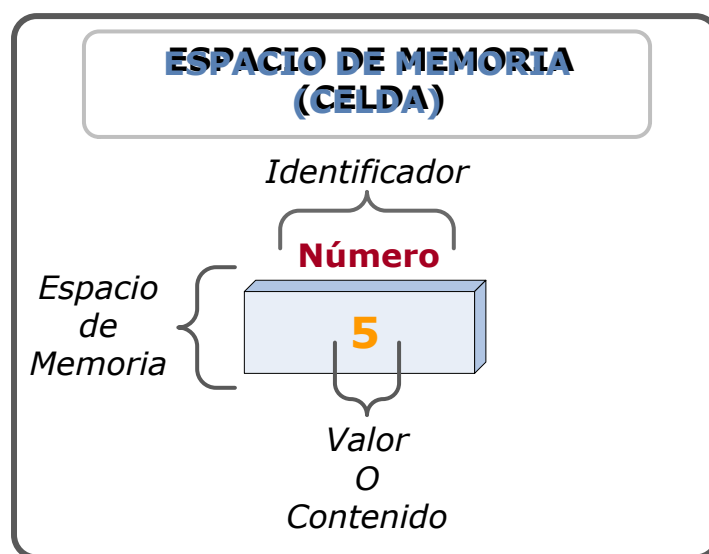


Un identificador es una secuencia o cadena de caracteres que sirven para nombrar constantes, variables o funciones, dentro de un algoritmo.

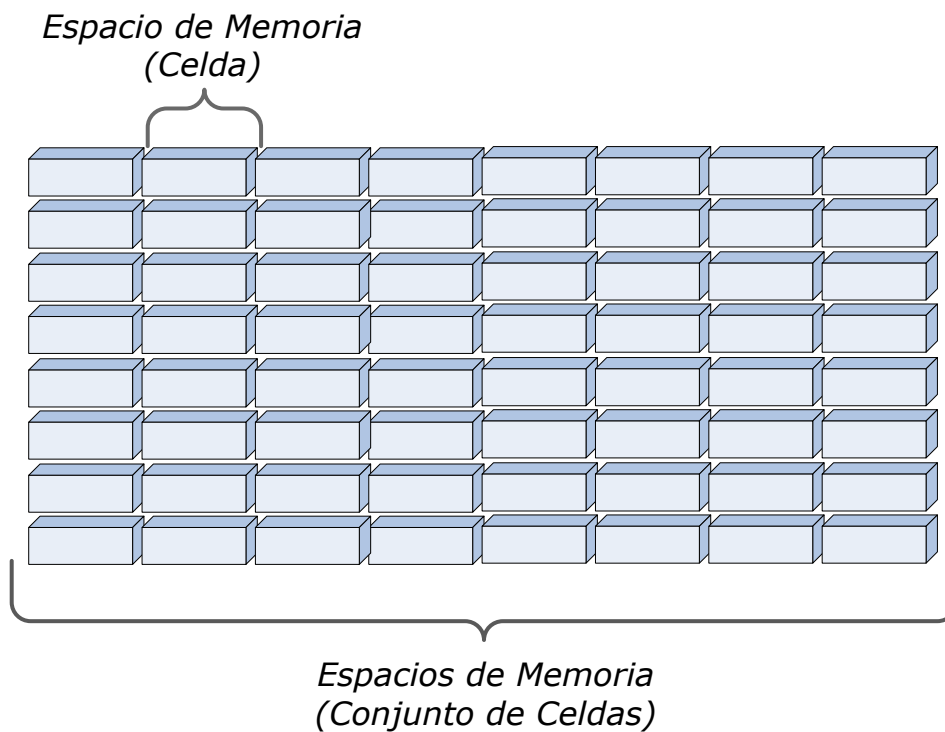
Los identificadores permiten asignar nombres (A Constantes, Variables o Funciones) a las secciones de memoria del computador, para de esta manera poder almacenar datos en ella.

La memoria de la computadora se puede considerar como un conjunto de celdas (o cajas) formando un panel.

A cada celda se le puede asignar un identificador, de tal manera que cuando se desee colocar o leer un dato, de esa sección de memoria, bastará con referirse al nombre asignado.



ESPACIOS DE MEMORIA (CONJUNTO DE CELDAS)





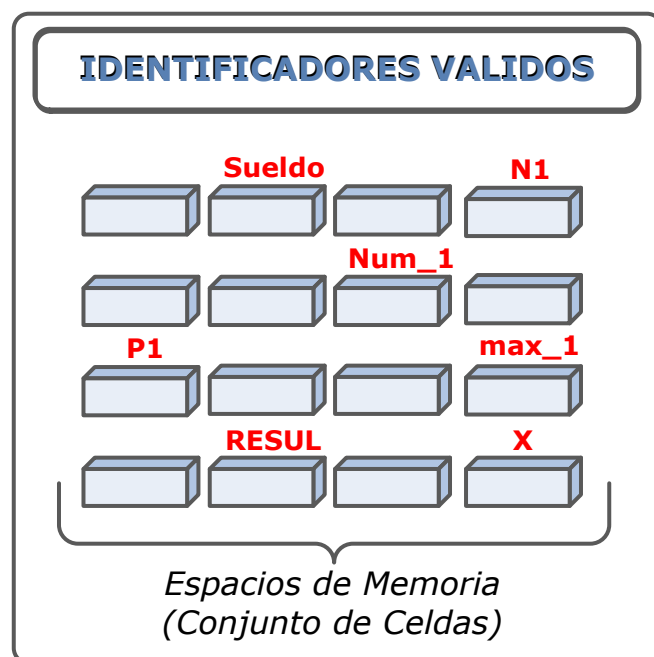
Reglas para la creación de los identificadores

La secuencia únicamente puede estar formada cumpliendo las siguientes reglas:

- ❖ Debe comenzar con una letra (De la **A** a la **Z**, mayúsculas o minúsculas) y no deben contener espacios en blanco.
- ❖ Letras, dígitos y caracteres como la subraya (**_**) están permitidos después del primer carácter.

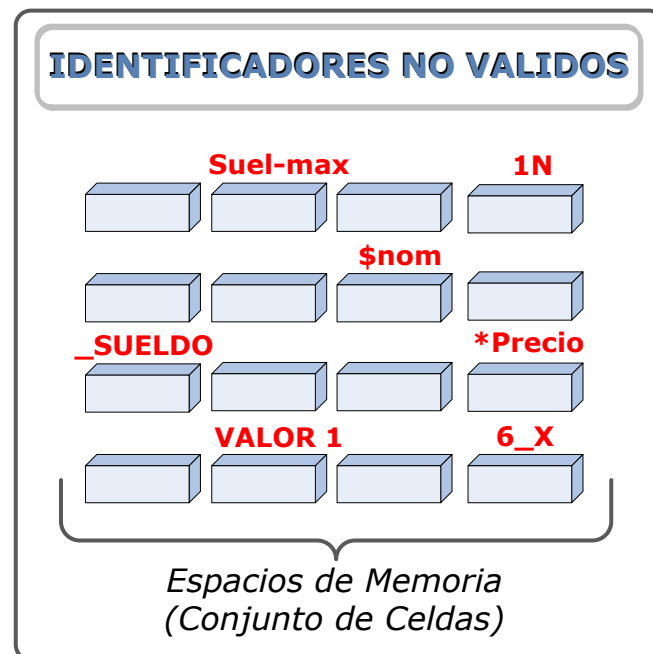
Las mismas pueden tener algunas variantes dependiendo del lenguaje de programación que se utilice, como caracteres especiales que pueden utilizarse y longitud.

Por ejemplo, los siguientes identificadores si son válidos:





En cambio, los siguientes identificadores no son permitidos:

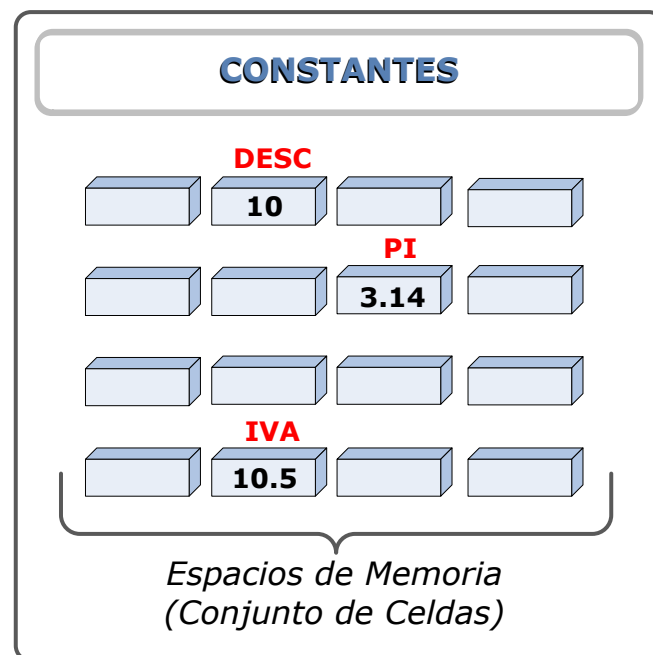


En algunos lenguajes de programación se diferencian, en los identificadores, las mayúsculas de las minúsculas. Por lo tanto, los identificadores **num**, **NUM** y **Num** son diferentes Identificadores a pesar de llamarse iguales. Durante el curso haremos esta diferencia entre Mayúscula y Minúsculas.

CONSTANTES

Una constante es un espacio de memoria en el cual se almacena un dato que no puede ser modificado, y se mantiene durante la ejecución del algoritmo que la creó. Para nombrar las constantes utilizamos los Identificadores que mencionamos anteriormente. Es muy importante que los nombres de las constantes sean representativos de la función que tienen las mismas dentro del algoritmo.

Si en alguna parte del algoritmo se define o establece una constante llamada PI, el sistema toma una de las celdas de memoria disponibles y la marca con el identificador PI, de tal manera que en ella se puede almacenar un valor solo en el momento de crearla. Después de esto no se podrá cambiar ese valor y se mantendrá durante toda la ejecución del algoritmo.



A partir de aquí en la memoria del computador existirá una sección llamada PI que no podrá ser modificada por ninguna sentencia del algoritmo que la creó.

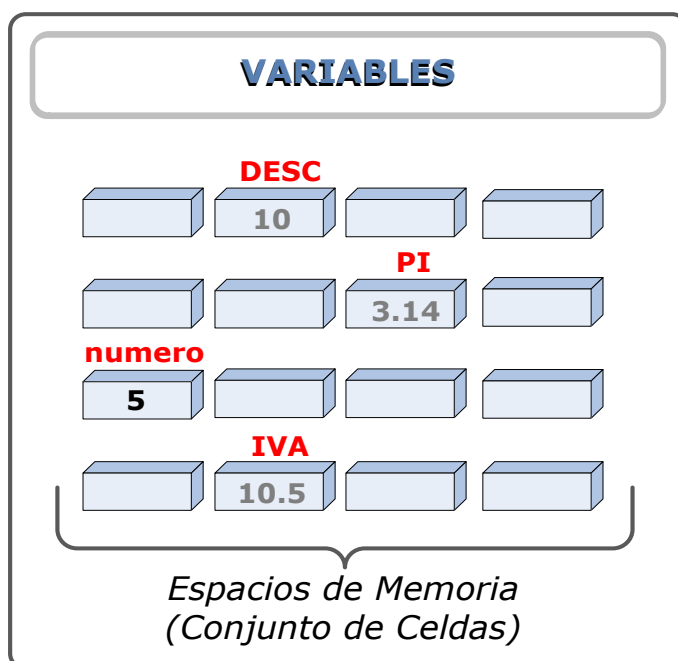
Si en alguna parte del algoritmo se intenta modificar el valor de la constante PI, se generará un error, que puede, incluso, impedir la ejecución de todo el programa.

VARIABLES

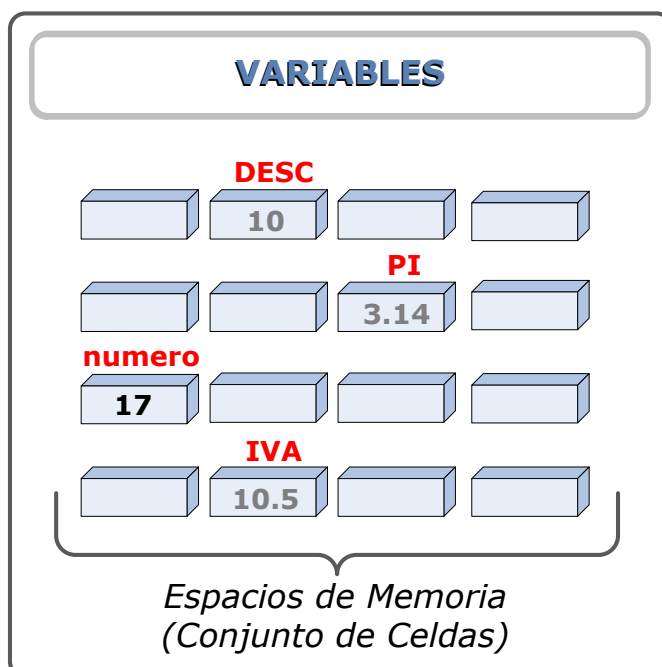
Una variable es una sección de memoria cuyo contenido se puede modificar en cualquier momento durante la ejecución del algoritmo. Para nombrar las variables utilizamos los Identificadores que mencionamos anteriormente. Es muy importante remarcar que los nombres de las variables sean representativos de la función que tienen las mismas dentro del algoritmo.

Por ejemplo, la sentencia **numero** = 5 crea la variable **numero**, y le asigna el valor **5**.

En este punto, el sistema asigna el nombre **numero** a una sección de memoria que se encuentre libre y coloca en ella el valor **5**. El esquema de memoria se vería así:



Si en otro punto del algoritmo aparece la sentencia **numero** = 17, inmediatamente se elimina el valor **5** y se coloca el valor **17**. El resultado sería el siguiente:





TIPOS DE DATOS

Todos los datos tienen un tipo asociado con ellos. Un dato puede ser un simple carácter, tal como **"B"**, un valor entero, tal como **35**. El tipo de dato determina la naturaleza del conjunto de valores que puede tomar una variable. Por esta razón los datos se agrupan en conjuntos de datos llamados **tipo de datos**. En este curso vamos a hablar únicamente de cuatro tipos de datos: enteros, reales, cadenas de caracteres y booleanos. Pero solo se harán referencias informativas sobre los mismos, ya que no requiere la definición de tipos de datos en la construcción de algoritmos.

TIPOS DE DATOS	SIMPLES	<ul style="list-style-type: none">• Numéricos• Lógicos• Alfanuméricos (String)
	ESTRUCTURADOS (Def. por el Usuario)	<ul style="list-style-type: none">• Arreglos (Vectores, Matrices)• Registros• Archivos• Apuntadores

Tipos de datos simples

- ❖ **Datos Numéricos:** *Permiten representar valores escalares de forma numérica, esto incluye a los números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.*
- ❖ **Datos Lógicos:** *Son aquellos que solo pueden tener dos valores (Verdadero o Falso) ya que representan el resultado de una comparación entre otros datos (numéricos o alfanuméricos).*
- ❖ **Datos Alfanuméricos (String):** *Es una secuencia de caracteres alfanuméricos que permiten representar valores identificables de forma descriptiva, esto incluye nombres de personas, direcciones, etc. Es posible representar números como alfanuméricos, pero estos pierden su propiedad matemática, es decir no es posible hacer operaciones con ellos. Este tipo de datos se representan encerrados entre comillas.*



CLASIFICACIÓN DE VARIABLES

Las variables pueden clasificarse en dos grupos, dependiendo de su contenido (Lo que guardan) y por su uso (Como las empleamos).

VARIABLES	POR SU CONTENIDO	<ul style="list-style-type: none">• Numéricas• Lógicas• Alfanuméricas (String)
	POR SU USO	<ul style="list-style-type: none">• De Trabajo• Contadores• Acumuladores

Por su contenido

- ❖ **Variable Numéricas:** Son aquellas en las cuales se almacenan valores numéricos, positivos o negativos, es decir almacenan números del 0 al 9, signos (+ y -) y el punto decimal. **Ejemplo:**

IVA = 21

Costo = 2500

PI = 3.14

- ❖ **Variables Lógicas:** Son aquellas que solo pueden tener dos valores (verdadero o falso) estos representan el resultado de una comparación entre otros datos.
- ❖ **Variables Alfanuméricas:** Esta formada por caracteres alfanuméricos (letras, números y caracteres especiales). **Ejemplo:**



Nombre = 'Juan Perez'

Direccion = 'Libertad 2500 #P1'

Universidad = 'UNIVERSIDAD TECNOLÓGICA NACIONAL'

Por su uso

- ❖ **Variables de Trabajo:** Variables que reciben el resultado de una operación matemática completa y que se usan normalmente dentro de un diagrama.
- ❖ **Contadores:** Se utilizan para llevar el control del número de ocasiones en que se realiza una operación o se cumple una condición. Con los incrementos generalmente de uno en uno.
- ❖ **Acumuladores:** Forma que toma una variable y que sirve para llevar la suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente.

ASIGNACIÓN DE DATOS A LA MEMORIA

La asignación o almacenamiento de datos en memoria se realiza utilizando las constantes o variables definidas en los algoritmos del programa. Como se vio anteriormente, a las constantes se les asigna un valor únicamente en el momento de su creación, seguida del nombre de la constante, el signo igual (=) y el valor que se desea asignar.

Nombre Constante = valor



La asignación de un valor a una variable se hace colocando su identificador a la izquierda del signo "igual" y a la derecha de este se escribe el valor que se quiere almacenar en esa localidad de memoria. Así, por ejemplo,

INTERES = 2.5

Significa que en la sección de memoria llamada **INTERES** se coloca el valor **2.5**, el cual se mantendrá ahí hasta que termine la ejecución del programa.

Pero, en forma general una variable puede recibir valores directamente o de otras variables o constantes definidas con anterioridad, en el algoritmo. Estas se colocan al lado derecho del signo "igual". La siguiente sentencia es un ejemplo de este caso:

Precio = PU

Aquí, la variable que se encuentra al lado izquierdo del signo "igual" (**Precio**), se toma como una celda de memoria en la cual se va a almacenar un valor. La variable que se encuentra al lado derecho del signo "igual"(en este caso **PU**), únicamente representa el valor contenido en esa celda de memoria.

Como regla general, se establece lo siguiente: la variable escrita al lado izquierdo del signo de asignación (en este caso el "igual" representa una celda de memoria, la cual recibe un valor para almacenar, y todas las variables y/o constantes que aparecen al lado derecho son tomadas como los valores de las celdas de memoria a las cuales identifican.

Por ejemplo, en la siguiente sentencia, **Suma** es la celda de memoria en la cual se va a almacenar un valor; **Numero1** y **Numero2** son los valores contenidos en las celdas de memoria identificadas con **Numero1** y **Numero2**, respectivamente.

Suma = Numero1 + Numero2

Al final, en la variable **Suma** se almacena el resultado de realizar las operaciones indicadas a la derecha del signo igual.



EJEMPLO:

Analizar el esquema de memoria para determinar el valor de las variables A y B después de ejecutar las siguientes sentencias:

A = 100

B = 20

B = 50

A = 30

Es importante tener en cuenta que la computadora lee las instrucciones en estricto orden, de arriba hacia abajo y de izquierda hacia derecha. Por lo tanto, el análisis se debe hacer siguiendo esta regla.

SENTENCIA	DESCRIPCION DE LA EJECUCION	RESULTADO DE LA EJECUCION
A = 100	Coloca en la sección de memoria A , el valor 100 .	<div><div>A</div><div>100</div></div> <div><div>B</div><div></div></div>
B = 20	Coloca el valor 20 en la sección de memoria llamada B .	<div><div>A</div><div>100</div></div> <div><div>B</div><div>20</div></div>
B = 50	Coloca el valor 50 en la sección de memoria B .	<div><div>A</div><div>100</div></div> <div><div>B</div><div>50</div></div>
A = 30	Coloca el valor 30 en A .	<div><div>A</div><div>30</div></div> <div><div>B</div><div>50</div></div>

Observe que, cuando una sentencia cambia el valor de una variable, esto no le afecta en nada a la otra variable, por que corresponden a dos secciones de memoria totalmente independientes una de otra.

En conclusión, la variable **A** queda con el valor **30** y la variable **B** con el valor **50**.



EJEMPLO:

Analizar el esquema de memoria para las variables **Num1** y **Num2**, y determinar su valor final, en la ejecución de las siguientes sentencias:

Num1 = 23

Num2 = 10

Num1 = Num2

Num2 = Num1

A primera vista, parece lógico, que la ejecución de las anteriores sentencias intercambia el valor de las variables, dejando a **Num1** con el valor **10** y a **Num2** con el valor **23**. Pero esto no es cierto, ya que la misma lógica nos demuestra, en un análisis detallado, que el valor **23** de **Num1** se pierde cuando se ejecuta la asignación de la tercera línea, **Num1 = Num2** donde **Num1** pierde el valor que traía y toma el valor de **Num2**. Analicemos la ejecución de las sentencias y que sucede en los espacios de memoria para comprender la situación.

SENTENCIA	DESCRIPCION DE LA EJECUCION	RESULTADO DE LA SENTENCIA
Num1 = 23	Coloca el valor 23 en la variable Num1 .	<div><div>Num1</div><div>23</div></div> <div><div>Num2</div><div></div></div>
Num2 = 10	Coloca el valor 10 en la variable Num2 .	<div><div>Num1</div><div>23</div></div> <div><div>Num2</div><div>10</div></div>
Num1 = Num2	Coloca una copia del contenido actual de Num2 , el valor 10 , en la variable Num1 (pero Num2 no pierde su valor).	<div><div>Num1</div><div>10</div></div> <div><div>Num2</div><div>10</div></div>
Num2 = Num1	Coloca una copia del contenido de Num1 , el valor 10 , en Num2 .	<div><div>Num1</div><div>10</div></div> <div><div>Num2</div><div>10</div></div>



Observe que, el hecho de que en la línea 3 se asigne el contenido de **Num2** a **Num1**, esto no significa que **Num2** vacíe su contenido, ya que este solo puede perderse cuando se sustituye por otro o se elimine mediante alguna sentencia de asignación del tipo **Num2 = valor**. En conclusión, en **Num1** y en **Num2** queda almacenado el mismo valor, el número **10**.

EXPRESIONES

Las expresiones son combinaciones de constantes, variables, símbolos de operación, paréntesis y nombres de funciones especiales. Por ejemplo:

$$A + (B + 3) / C$$

Cada expresión toma un valor que se determina tomando los valores de las variables y constantes implicadas y la ejecución de las operaciones indicadas. Una expresión consta de operadores y operandos. Según sea el tipo de datos que manipulan, se clasifican las expresiones en:

- ❖ **Aritméticas**
- ❖ **Relacionales**
- ❖ **Lógicas**

OPERADORES

Los operadores son caracteres especiales que le dicen al computador que se desea realizar una operación, sobre unos operandos. Los operandos pueden ser variables, expresiones o valores literales. Existen básicamente los siguientes operadores: aritméticos, relacionales y lógicos. En esta primera parte analizaremos solo los operadores Aritméticos.

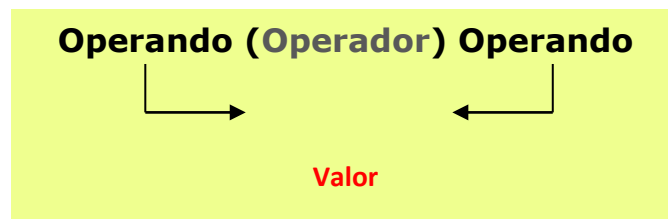
Tipos de Operadores

- *Aritméticos*
- *Relacionales*
- *Lógicos*



Operadores aritméticos

Los operadores aritméticos permiten la realización de operaciones matemáticas con los valores literales, *variables o constantes*, de la misma forma como lo hacen en el álgebra. Los operandos, necesariamente deben ser valores numéricos, enteros o reales. Si ambos son enteros, el resultado es entero; si alguno de ellos es real, el resultado es real.



OPERADOR	RESULTADO
+	Suma
-	Resta
*	Multiplicación
/	División
% (Mod)	Resto (Modulo)
++	Incremento
--	Decremento

JERARQUIA DE LOS OPERADORES ARITMETICOS		
OPERADOR	JERARQUIA	OPERACION
\wedge	MAYOR ↓ MENOR	Potencia
$*$, $/$, $\%$		Multiplicación, división, Resto (Modulo).
$+$, $-$		Suma, resta.



Prioridad de los operadores aritméticos

- ❖ Todas las expresiones entre paréntesis se evalúan primero. Las expresiones con paréntesis anidados se evalúan de dentro hacia fuera, el paréntesis más interno se evalúa primero.
- ❖ Dentro de una misma expresión los operadores se evalúan en el siguiente orden.

ORDEN	OPERADOR	OPERACION
1	\wedge	Potencia
2	$*$, $/$, $\%$	Multiplicación, división, modulo.
3	$+$, $-$	Suma, resta

- ❖ Los operadores en una misma expresión con igual nivel de prioridad se evalúan de izquierda a derecha.

Ejemplos	
Expresión	Resultado
$4 + 2 * 5$	14
$23 * 2 / 5$	9.2
$3 + 5 * (10 - (2 + 4))$	23
$3.5 + 5.09 - 14.0 / 40$	8.24
$2.1 * (1.5 + 3.0 * 4.1)$	28.98



Suma, resta, multiplicación y división

La suma, resta, multiplicación y división se aplican exactamente como en el álgebra, conservando sus propiedades y su nivel de precedencia. La precedencia hace referencia a que operación se realiza primero, cuando se combinan varias de ellas.

Así, por ejemplo, en $3 + 5 \times 4$, la multiplicación se realiza primero y al final la suma, de tal manera que se obtiene lo siguiente:

$$3 + 5 \times 4 = 3 + 20 = 23$$

En general la multiplicación y la división tienen más alta precedencia que la suma y la resta, lo cual implica que estas operaciones se realizan primero en una combinación de operaciones. Pero, esto únicamente se aplica, así directamente, cuando no existan paréntesis de agrupación, ya que en estos casos la operación u operaciones que se realizan primero son aquellas que se encuentren dentro del paréntesis.

Por ejemplo, en $(3 + 5) \times 4$, la operación que se debe realizar primero es la suma contenida entre los paréntesis, obteniéndose:

$$(3 + 5) \times 4 = 8 \times 4 = 32$$

Operador Resto

El operador *Resto* devuelve el residuo de dividir dos números enteros entre sí. Sus operandos deben ser números enteros. Para aplicar el operador Resto se puede utilizar el comando **Mod**, o también el carácter %, colocado entre los números que se dividen. Así, $5 \% 3$ calcula el residuo que se obtiene al dividir 5 entre 3. En este caso la operación devuelve el valor **2**

Operador incremento

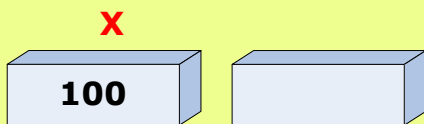
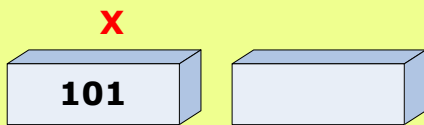
El incremento consiste en modificar el valor de una variable, sumándole un cierto valor al dato almacenado en la variable.

Por ejemplo, si en la variable **X** está guardado el valor **100**, se puede desear sumarle **1**, a ese valor, para que en la variable **X** quede almacenado el valor **101**. En es-



te caso se dirá que la variable **X** fue incrementada en **1**.

Analicemos con más detalle el incremento del ejemplo. Supongamos que en la variable **X** se almacena, inicialmente, el valor **100**, mediante la sentencia **X = 100**. Para sumarle, posteriormente, el valor **1** a ese dato, bastará con aplicar la sentencia, **X = X + 1**. El siguiente análisis de asignación muestra lo que ocurre.

SENTENCIA	DESCRIPCION DE LA EJECUCION	RESULTADO DE LA SENTENCIA
X = 100	Coloca el valor 100 en la variable X .	
X = X + 1	Coloca en X el resultado de sumar el valor contenido en la celda X más 1 .	

Es importante tener bien claro el concepto de asignación. Al ejecutar la sentencia **X = X + 1** primero se realiza la operación del lado derecho del signo "igual" (**100 + 1**) y al final se realiza la asignación del resultado, a la variable **X**. Y solo a partir de esta asignación, en la variable **X** existirá el valor **101**.

Para incrementar en 1 el valor contenido en una variable, se puede aplicar la sentencia analizada con anterioridad o el operador **++**. Así, la sentencia **X = X + 1** es equivalente a la sentencia **X++**.

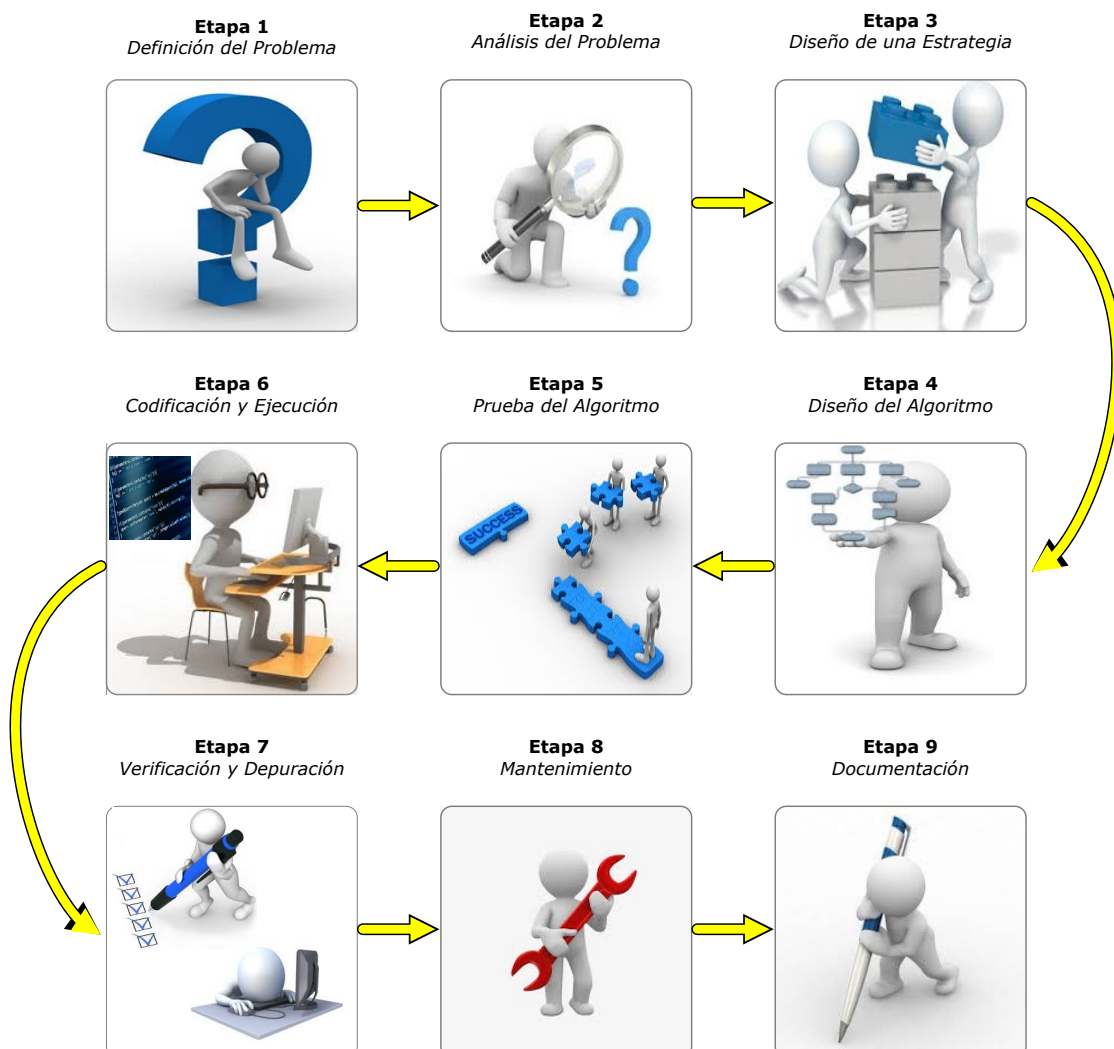
En general, si se desea incrementar en un cierto *valor* una variable **X**, se puede aplicar la sentencia **X + valor** donde *valor* es cualquier número entero o real. Por ejemplo, la sentencia **X + 12** incrementa en **12** el valor contenido en la variable **X**.

Operador decremento

El decremento se aplica en forma análoga a como se hace con el incremento, con la diferencia obvia de que aquí de lo que se trata es de disminuir el valor contenido en una variable. La sentencia **X = X - 1** disminuye en **1** el valor contenido en la variable **X** y esta misma operación se puede realizar aplicando el operador **--** de la siguiente forma **X--** para disminuir la variable **X** en un valor cualquiera, se puede aplicar la sentencia, **X - Valor**.

METODOLOGÍA PARA LA RESOLUCIÓN DE PROBLEMAS

La etapa inicial comienza con el planteamiento de un problema o con una solicitud específica de un programa, a la cual se la conoce como **requerimiento de programa**. Una vez que se ha planteado un problema o se ha hecho una solicitud específica para un programa, es necesario aplicar una Metodología que nos permita desarrollar una solución de forma ordenada. 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 diseñar un programa, es esencialmente un proceso creativo, es necesario considerar una serie de etapas o pasos comunes, que generalmente deben seguir todos los programadores. Las etapas de resolución de un problema con computadora son:





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 la computadora; mientras esto no se conozca del todo no tiene mucho caso continuar con la siguiente etapa.

ANÁLISIS DEL PROBLEMA

Una vez que se ha comprendido lo que se desea de la computadora, es necesario definir:

- ❖ **¿Cuáles serán los datos de entrada?** (*Tipo de datos y cantidad*).
- ❖ **¿Cuál es la información que se desea producir?** (*salida*)
- ❖ **Los métodos y fórmulas que se necesitan para procesar los datos.**
- ❖ **Requisitos o requerimientos adicionales y restricciones a la solución.**

Una recomendación muy práctica es el que nos pongamos en el lugar de la computadora y analicemos que es lo que necesitamos que nos ordenen y en que secuencia para producir los resultados esperados.

DEFINICIÓN DE UNA ESTRATEGIA

Consiste en elaborar un plan general sobre el cual nos basaremos para luego desarrollar el algoritmo. En esta etapa definimos **"como resolveremos el problema"**.

Esta etapa es la más creativa y también la más difícil en la resolución de los problemas. Es donde se aprecia la experiencia e intuición de quien resuelve el problema. No se trata tampoco de ser siempre original, conviene basarse en problemas anteriores o en experiencias similares.



EJEMPLO: *Determinar la edad promedio de los 5 alumnos de un curso.*

1	DEFINICIÓN DEL PROBLEMA (¿Qué debo hacer?)	<i>Calcular la edad promedio de los alumnos de un curso.</i>
2	ANÁLISIS DEL PROBLEMA (¿Qué datos, Que Proceso....?)	Datos: <i>Edad de cada alumno del curso.</i> Resultados: <i>Edad promedio.</i> Proceso: <i>Calculo del promedio.</i>
3	DISEÑO DE UNA ESTRATEGIA (¿Como lo voy a hacer?)	A. <i>Ingresar las edades de cada uno de ellos.</i> B. <i>Calcular el promedio (sumar todas las edades y dividirla por la cantidad de alumnos)</i> C. <i>Informar el resultado</i>

DISEÑO DEL ALGORITMO

Todas las etapas enumeradas anteriormente son importantes y necesarias, algunas las desarrollaremos en este curso y otras serán desarrolladas durante la carrera. Pero durante el curso nos centraremos particularmente en la Etapa de Diseño del Algoritmo, de esta manera al ser considerada unas de las etapas más importantes durante el desarrollo del curso, la misma tendrá un apartado especial que nos guiará en la creación de los Algoritmos.

¿Qué es un Algoritmo?

Casi inconscientemente, cotidianamente efectuamos una serie de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema.

Esta serie de pasos, procedimientos o acciones, comenzamos a aplicarlas muy temprano en la mañana cuando, por ejemplo, decidimos tomar un baño o posteriormente cuando pensamos en desayunar también seguimos una serie de pasos que nos permiten alcanzar un resultado específico que sería tomar el desayuno. La historia se repite innumerables veces durante el día. Continuamente seguimos una serie de pasos o conjunto de acciones que nos permiten alcanzar un resultado. Estamos en realidad aplicando un *algoritmo para resolver un problema*.



"Formalmente definimos un algoritmo como un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema".

Muchas veces aplicamos el algoritmo de manera inadvertida inconsciente o automáticamente. Esto generalmente se produce cuando el problema que tenemos enfrente lo hemos resuelto con anterioridad un gran número de veces.

Por otra parte, existe una gran cantidad de problemas que requieren de un análisis profundo y de un pensamiento flexible y estructurado para su solución. No existen reglas específicas que nos permitan resolver un problema. Sin embargo, se pueden ofrecer un conjunto de técnicas y herramientas metodológicas que permitan flexibilizar y estructurar el razonamiento utilizado en la solución de un problema. Eso llevara finalmente a la construcción de algoritmos eficientes.

PRUEBA DEL ALGORITMO

Los errores humanos dentro de un Algoritmo de soluciones para un problema son muchos y aumentan considerablemente con la complejidad del problema. El proceso de identificar y eliminar errores, para dar paso a una solución sin errores se le llama ***Prueba del Algoritmo***.

La ***Prueba del Algoritmo*** resulta una tarea tan creativa como el mismo desarrollo de la solución, por ello se debe considerar con el mismo interés y entusiasmo.

CODIFICACIÓN Y EJECUCIÓN

Este paso, el cual también se conoce como escribir el programa y poner en práctica la solución. La *codificación* es la escritura en un lenguaje de programación de la representación del algoritmo desarrollada en las etapas precedentes.

Dado que el diseño de un algoritmo es independiente del lenguaje de programación utilizado para su implementación, el código puede ser escrito con igual facilidad en un lenguaje o en otro. Si los pasos de análisis y solución se han realizado en forma correcta, el paso de codificación se vuelve bastante mecánico.



VERIFICACIÓN Y DEPURACIÓN

El propósito de probar es verificar que el programa funciona en forma correcta y en realidad cumple con sus requerimientos. En teoría, las pruebas revelarían todos los errores del programa.

En terminología de computación, un error de programa se conoce como **bug**. En la práctica, esto requeriría comprobar todas las combinaciones posibles de ejecución de las instrucciones. Debido al tiempo y al esfuerzo requeridos, esto por lo general es una meta imposible, excepto para programas simples.

Para detectar y corregir errores en un programa es importante desarrollar un conjunto de datos de prueba por medio de los cuales determinar si el programa proporciona respuestas correctas.

Desarrollar buenos procedimientos y datos de prueba para problemas complejos puede ser más difícil que escribir el código del programa en sí.

MANTENIMIENTO

Se lleva a cabo después de terminada la creación del programa, cuando se detecta que es necesario hacer algún cambio, ajuste o complementación al programa para que siga trabajando de manera correcta.

Para poder realizar este trabajo se requiere que el programa este correctamente documentado.

DOCUMENTACIÓN

Es la guía o comunicación escrita en sus variadas formas, ya sea en enunciados, procedimientos, dibujos o diagramas.

A menudo un diagrama dibujado por una persona es usado por otra. Por ello la documentación sirve para ayudar a comprender o usar un diagrama o para facilitar futuras modificaciones (mantenimiento).



La **documentación** se divide en tres partes:

- ❖ Documentación Interna
- ❖ Documentación Externa
- ❖ Manual del Usuario

DOCUMENTACIÓN INTERNA

Son los comentarios o mensaje que se añaden al código fuente para hacer mas claro el entendimiento de un proceso.

DOCUMENTACIÓN EXTERNA

Se define en un documento escrito los siguientes puntos:

- ❖ Descripción del Problema
- ❖ Nombre del Autor
- ❖ Algoritmo (diagrama de flujo o pseudocodigo)
- ❖ Diccionario de Datos

MANUAL DEL USUARIO

Describe paso a paso la manera como funciona el diagrama, con el fin de que el usuario obtenga el resultado deseado.



DISEÑO DE ALGORITMO

Para la construcción 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 de la computadora que los ejecuta. En cada problema el algoritmo se puede expresar en un lenguaje diferente de programación y ejecutarse en una computadora distinta; sin embargo, el algoritmo será siempre el mismo. Así, por ejemplo, haciendo una comparación 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 las computadoras. Un lenguaje de programación es tan sólo un medio para expresar un algoritmo y una computadora es sólo un procesador para ejecutarlo. Tanto el lenguaje de programación como la computadora son los medios para obtener un fin, conseguir que el algoritmo se ejecute y se efectúe el proceso correspondiente.

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

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

¿Qué es un algoritmo?

Un algoritmo es una serie de pasos organizados que describe el proceso que se debe seguir, para dar solución a un problema específico.

Aunque la popularización del término ha llegado con la era informática, **algoritmo** proviene de *Mohammed al-KhoWârizmi*, matemático persa 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 en la palabra *algorismus* derivó posteriormente en algoritmo.

Otras definiciones de Algoritmo pueden ser:

Un Algoritmo es un conjunto finito de instrucciones u operaciones que ejecutadas en un orden determinado permiten resolver un problema.

Un Algoritmo es una secuencia de instrucciones precisas que llevan a una solución.

Palabras equivalentes a Algoritmo:

- ❖ *RECETA.*
- ❖ *METODO.*
- ❖ *INSTRUCCIONES.*
- ❖ *PROCEDIMIENTO.*
- ❖ *RUTINA.*

PARADIGMAS DE PROGRAMACIÓN

En la etapa de análisis en La Metodología de Resolución de Problemas se determina qué debe hacer el programa. En la etapa de diseño del algoritmo se determina *cómo* hace el programa la tarea solicitada. Previamente debemos definir el enfoque que utilizaremos para su diseño. Existen enfoques alternativos a los procesos de la solución de problema, se los conoce como Paradigmas de Programación.

Un paradigma de programación representa fundamentalmente enfoques diferentes para la construcción de soluciones a problemas y por consiguiente afectan al proceso completo del desarrollo de programas.

Los paradigmas de programación clásicos son: Procedimental (o Imperativo), funcional, declarativo y orientado a objetos. Durante el curso centraremos nuestro diseño de algoritmos basado en el enfoque procedimental.



El **paradigma imperativo o procedimental** representa el enfoque o método tradicional de programación. Este paradigma define el proceso de programación como el desarrollo de una secuencia de órdenes (*comandos*) que manipulan los datos para producir los resultados deseados. Por consiguiente, el paradigma procedimental señala un enfoque del proceso de programación mediante la realización de un algoritmo que resuelve de modo manual el problema y a continuación expresa ese algoritmo como una secuencia de órdenes.

Dentro de este Paradigma de Programación se encuentran la **programación modular** y la **programación estructurada**. Dichos enfoques o métodos son los más eficaces para el proceso de diseño en esta parte de introducción a la programación.

La **Programación Modular** se basa en el conocido divide y vencerás. Es decir, la resolución de un problema complejo se realiza dividiendo el problema en subproblemas y a continuación dividiendo estos subproblemas en otros de nivel más bajo, hasta que pueda ser *implementada* una solución en la computadora. Este método se conoce técnicamente como diseño descendente (**top-down**) o **modular**. El proceso de romper el problema en cada etapa y expresar cada paso en forma más detallada se denomina *refinamiento sucesivo*.

Cada subprograma es resuelto mediante un **módulo** (*subprograma*) que tiene un solo punto de entrada y un solo punto de salida.

Cualquier programa bien diseñado consta de un *programa principal* (el módulo de nivel más alto) que llama a subprogramas (módulos de nivel más bajo) que a su vez pueden llamar a otros subprogramas. Los programas estructurados de esta forma se dice que tienen un *diseño modular* y el método de romper el programa en módulos más pequeños se llama *programación modular*.

La utilización de la técnica de diseño **Top-Down** tiene los siguientes objetivos básicos:

- ❖ *Simplificación del problema y de los subprogramas de cada descomposición.*
- ❖ *Las diferentes partes del problema pueden ser programadas de modo independiente e incluso por diferentes personas.*



- ❖ *El programa final queda estructurado en forma de bloque o módulos lo que hace más sencilla su lectura y mantenimiento.*

La **Programación Estructurada** se basa en el diseño modular (Programación Modular), donde cada módulo es diseñado en forma descendente (Top-Down) y cada módulo se diseña basándose en la combinación de tres estructuras básicas.

La Programación Estructurada nos dice que:

Todo Algoritmo puede ser construido en forma correcta utilizando únicamente tres estructuras básicas. Secuencial, Decisión e Iteración.

El proceso que convierte los resultados del análisis del problema en un diseño modular con refinamientos sucesivos que permitan una posterior traducción a un lenguaje se denomina **diseño del algoritmo**.

CARACTERÍSTICAS DE LOS ALGORITMOS

Las características fundamentales que debe cumplir todo algoritmo son:

- ❖ *Un algoritmo debe ser preciso, cada instrucción utilizada debe ser clara y sin ambigüedades.*
- ❖ *Un algoritmo debe producir al menos un resultado.*
- ❖ *Un algoritmo debe estar compuesto por una cantidad finita de instrucciones.*
Si se sigue un algoritmo, se debe terminar en algún momento; o sea, debe tener un número finito de pasos.
- ❖ *Un algoritmo debe estar compuesto por instrucciones que sean suficientemente básicas para que pueda ser resuelta por cualquier computadora.*

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



LENGUAJES ALGORÍTMICOS

Los lenguajes algorítmicos son herramientas que nos permiten diseñar y escribir los algoritmos. Son una serie de símbolos y reglas que se utilizan para describir de manera explícita un proceso.

Tipos de Lenguajes Algorítmicos:

- ❖ **Gráficos:** *Es la representación gráfica de las operaciones que realiza un algoritmo. (**Diagrama de Flujo**)*
- ❖ **No gráficos:** *Representa en forma descriptiva las operaciones que debe realizar un algoritmo. (**Pseudocódigo**)*

Diagrama de flujo

Un diagrama de flujo es la representación gráfica de un algoritmo. También se puede decir que es la representación detallada en forma gráfica de cómo deben realizarse los pasos en la computadora para producir resultados.

Esta representación gráfica se da cuando varios símbolos (que indican diferentes procesos en la computadora), se relacionan entre sí mediante líneas que indican el orden en que se deben ejecutar los procesos.

PSEUDOCODIGO

Mezcla de lenguaje de programación y español (o inglés o cualquier otro idioma) que se emplea, dentro de la programación estructurada, para realizar el diseño de un programa. En esencial, el pseudocódigo se puede definir como un lenguaje de especificaciones de algoritmos.

Es la representación narrativa de los pasos que debe seguir un algoritmo para dar solución a un problema determinado.

El pseudocódigo utiliza palabras que indican el proceso a realizar.



ESTRUCTURAS ALGORITMICAS

Las estructuras de operación de programas son un grupo de formas de trabajo, que permiten, mediante la manipulación de variables, realizar ciertos procesos específicos que nos lleven a la solución de problemas. Estas estructuras se clasifican de acuerdo con su complejidad en:

ESTRUCTURAS ALGORÍTMICAS	
SECUENCIALES	<ul style="list-style-type: none">• Asignación.• Entrada.• Salida.
SELECTIVAS (Decisión)	<ul style="list-style-type: none">• Simples.• Dobles.• Múltiples.
ITERACIÓN (Ciclos)	<ul style="list-style-type: none">• Ciclo Exacto.• Ciclo Inexacto.



BIBLIOGRAFÍA

- ❖ Osvaldo Cairó Battistutti (2005). Metodología de la Programación. Editorial Alfaomega.
- ❖ Pedro Vicente Rosero Montaña (2006). Introducción a la Programación de Computadores.
- ❖ Luis Joyanes Aguilar (2008). Fundamentos de Programación. Cuarta edición. Editorial McGRAW-HILL.
- ❖ Francisco Javier Pinales Delgado, César Eduardo Velázquez Amador. Problema-rio de algoritmos resueltos con diagrama de flujos y pseudocódigo. Universidad Autónoma de Aguascalientes.