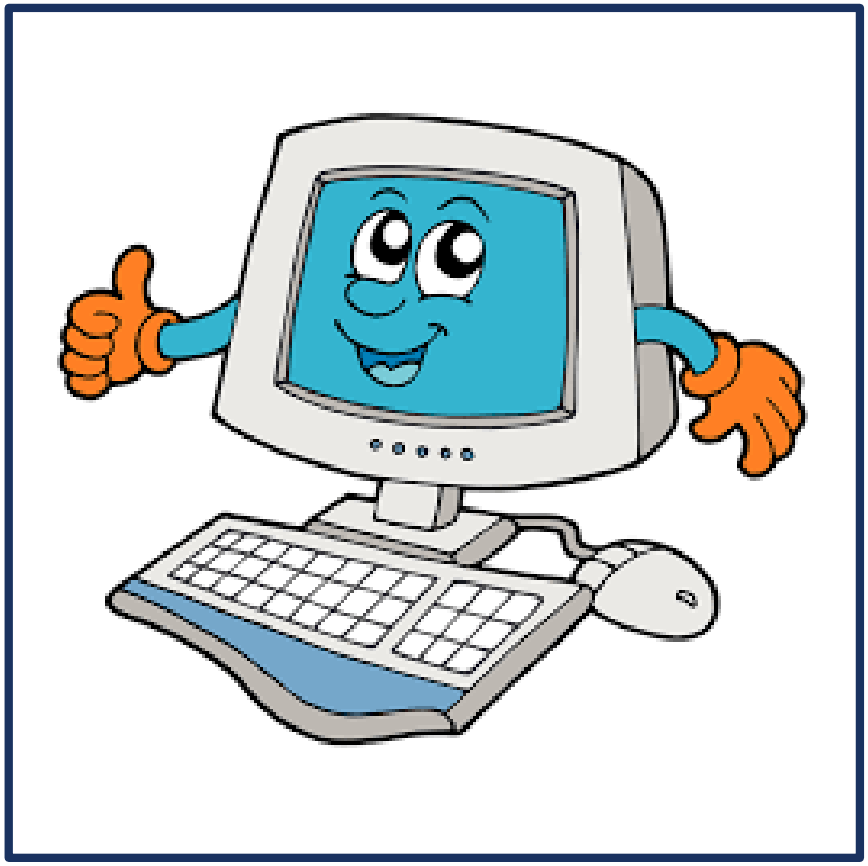
# INTRODUCCIÓN A LA PROGRAMACIÓN

ING. GIANKARIS G. MORENO R., M.SC

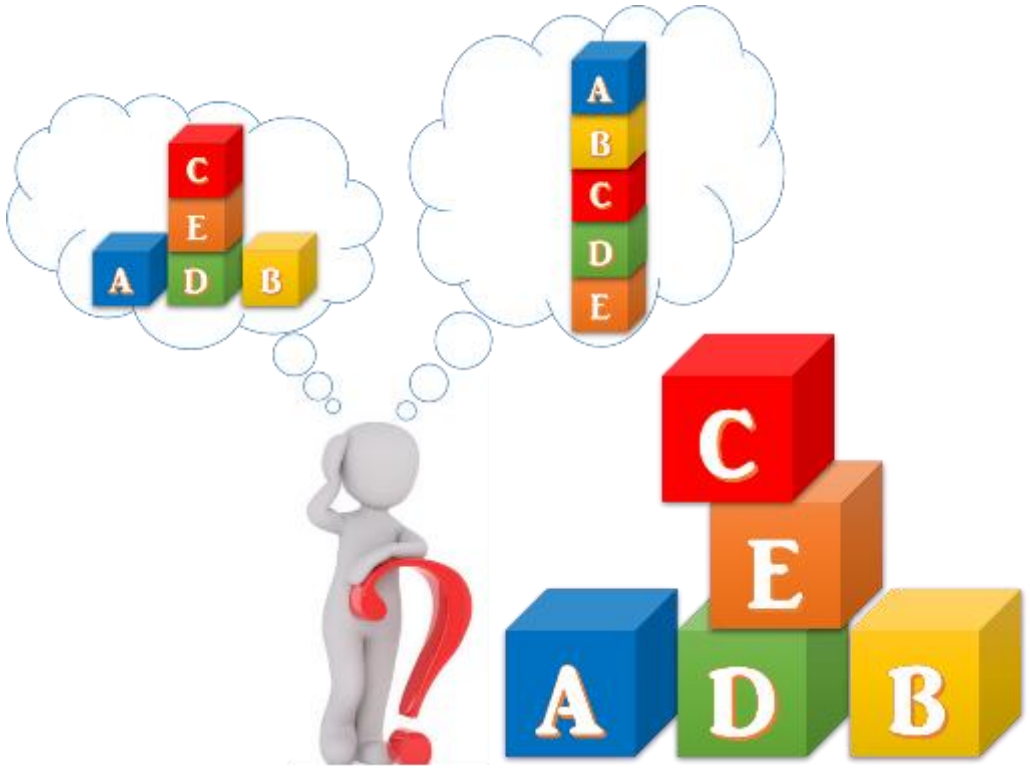
# INTRODUCCIÓN A LA PROGRAMACIÓN



- Las computadoras son herramientas esenciales en casi todos los campos de nuestra vida.
- Una computadora sin un programa, sin instrucciones, es virtualmente inútil.
- Los lenguajes de programación nos permiten escribir esos programas, haciendo posible la comunicación con las computadoras.

# ALGORITMO

- Un algoritmo puede definirse como una técnica de solución de problemas, que consiste en una serie de instrucciones paso a paso, que produce resultados específicos para un problema determinado.



# PROGRAMA

- Es una secuencia lógica de instrucciones escritas en un determinado lenguaje de programación, que establece las operaciones que van a ser realizadas por la computadora.

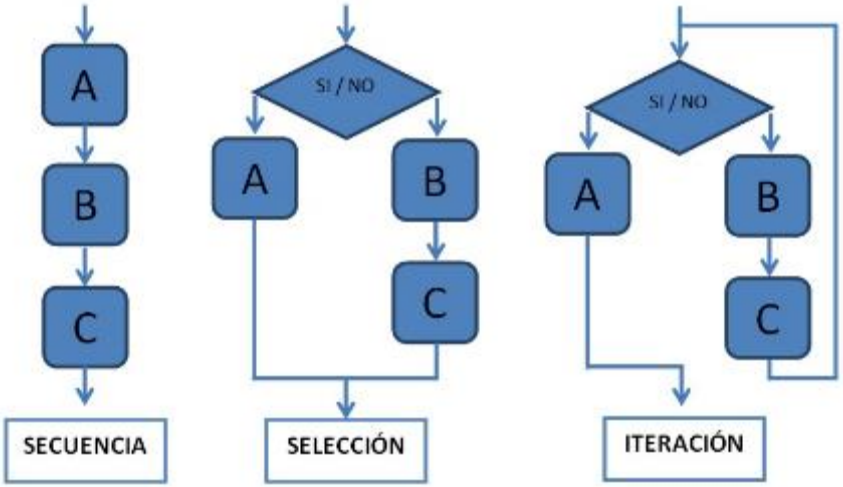
# PARADIGMAS DE PROGRAMACIÓN

- Los paradigmas de programación son estilos de desarrollo de programas. Es decir, son modelo para resolver problemas computacionales.
- Entre ellos podemos mencionar:
  - Programación Estructurada
  - Programación Orientada a Objetos
  - Programación Orientada a Eventos

PARADIGMAS DE PROGRAMACIÓN

## Programación Estructurada:

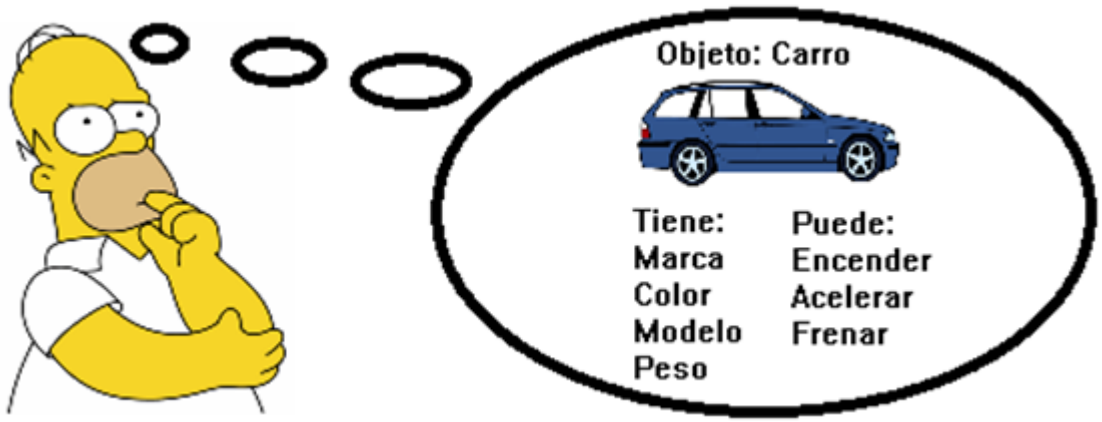
- Es un paradigma de programación orientado a mejorar la claridad, calidad y tiempo de desarrollo de un programa.
- Se fundamenta en la utilización de las tres estructuras básicas de control: ***secuencia, selección (alternativas)*** e ***iteración (ciclos)***.
- Luego incorpora la segmentación, que no es más que programar utilizando ***funciones***.



PARADIGMAS DE PROGRAMACIÓN

# Programación Orientada a Objetos (POO):

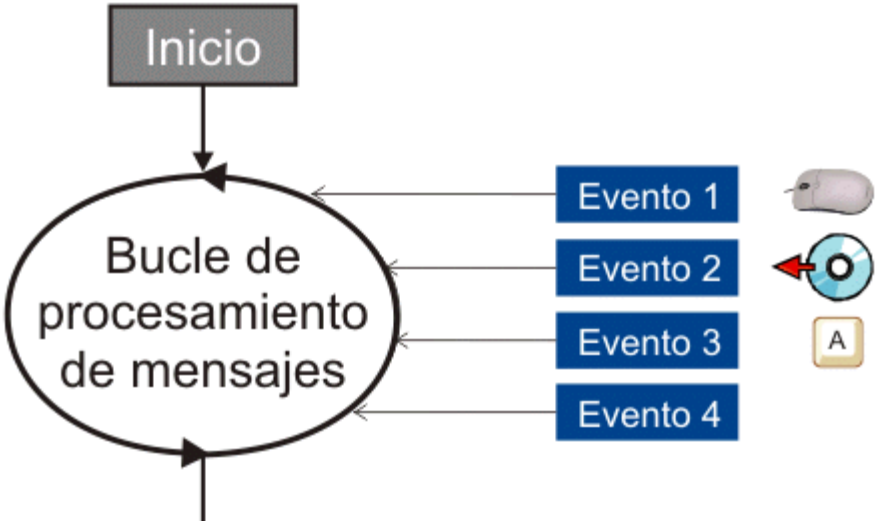
- Es un paradigma de programación que utiliza objetos como bloque esencial de construcción.
- El objeto es una unidad que contiene datos y las funciones que operan sobre esos datos.
- sobre esos datos
- La POO tiene cuatro pilares: abstracción, encapsulamiento, herencia y polimorfismo.



PARADIGMAS DE PROGRAMACIÓN

## Programación Orientada a Eventos:

- Es un paradigma de programación en el que tanto la estructura como la ejecución del programa depende de la ocurrencia de un suceso por lo general definido por el usuario.
- El flujo del programa en este caso es dirigido por el usuario.



# ETAPAS PARA LA RESOLUCIÓN DE PROBLEMAS POR COMPUTADORA

# 1. DEFINICIÓN DEL PROBLEMA

- Es la descripción en forma narrativa o esquemática de modo claro y concreto en un lenguaje corriente el problema que ha de resolverse.
- En pocas palabras, no es más que el enunciado del problema.
- Ejemplo:

Elabore un programa que lea dos lados de un triángulo y calcule e imprima la hipotenusa del triángulo. La fórmula para calcular la hipotenusa es: $H = \sqrt{a^2 + b^2}$

# 2. ANÁLISIS Y DISEÑO DEL PROBLEMA

- Consiste en identificar y describir los elementos en el dominio del problema.
  - ¿Qué datos de entrada me proporciona el problema?
  - ¿Cuál es la salida del problema?
  - ¿Qué procesos debo realizar para obtener la salida?

# 2. ANÁLISIS Y DISEÑO DEL PROBLEMA

| Análisis y Diseño | | |
|---|---|---|
| Entradas | Proceso | Salida |
| (escribir de manera narrativa las entradas. Colocar el valor numérico cuando es una constante) | (escribir de manera narrativa el proceso) | (escribir de manera narrativa la salida) |

- Ejemplo:

| Análisis y Diseño | | |
|---|---|---|
| Entradas | Proceso | Salida |
| ladoA<br>ladoB | Calcular la hipotenusa<br>$H = \sqrt{ladoA^2 + ladoB^2}$ | *H* |

# 3. PROGRAMACIÓN

- Esta etapa consiste en escribir un programa para resolver un problema, hacerlo correctamente no es sólo redactar código en un lenguaje de programación específico.
- La correcta ejecución de esta etapa implica seleccionar una técnica que permita tener una visión más clara y detallada de los pasos lógicos a seguir, como por ejemplo:
  - Algoritmo
  - Codificación
  - Prueba de escritorio
  - Compilación
  - Ejecución

# 3.1. ALGORITMO

- Los algoritmos permiten esquematizar los pasos a seguir usando un lenguaje de especificaciones llamado pseudocódigo.
- Desarrollar software sin un buen algoritmo es como construir una casa sin los planos necesarios. Los resultados podrían ser catastróficos.

- Ejemplo: Algoritmo escrito en lenguaje natural (hipotenusa)
  - Paso 1: Solicitar al usuario los dos lados del triángulo (ladoA y ladoB)
  - Paso 2: Calcular la hipotenusa ($H = \sqrt{ladoA^2 + ladoB^2}$)
  - Paso 3: Desplegar en pantalla el valor de la hipotenusa (H)

# 3.2. PRUEBA DE ESCRITORIO

- Consiste en examinar la solución exhaustivamente con el fin que produzca los resultados deseados; al mismo tiempo, podemos con la prueba detectar, localizar y eliminar errores.
- Debemos considerar varias posibilidades de valores de entrada para garantizar que éstos produzcan siempre un resultado correcto.

| Constantes en memoria | | Variables en memoria | | | | | | Pantalla |
|---|---|---|---|---|---|---|---|---|
| | | var1 | var2 | var3 | ..... | | | |
| | | | | | | | | |

# 3.3. CODIFICACIÓN

- La codificación implica la transcripción del algoritmo resultante a un lenguaje de programación.
- Al resultado de la codificación se le denomina ***Programa Fuente*** o ***Código Fuente***.
- Ejemplo de codificación en el lenguaje C:

```
//nombre del archivo que contiene este programa:
ProgH.c
   #include <math.h>
  #include  <iostream.h>
  #include  <conio.h>
   void main( )      {
        float a, b;
        float  H;
        clrscr( );
 //Paso 1
        cout << "Entrar el valor de a : ";
        cin>> a;
        cout << "Entrar el valor de b : ";
        cin>> b;
 //Paso 2
          H=sqrt (pow(cat1,2) + pow (cat2, 2));
//Paso 3
         cout<<"Hipotenusa es = "<< H;
         getch();
    }
```

# 3.4. COMPILACIÓN

- Una vez que el algoritmo ha sido convertido a un programa fuente, es preciso introducirlo a la máquina. Luego, el programa debe ser compilado o traducido al ***programa objeto***.
- Esta tarea la ejecuta un software que da el fabricante, llamado compilador.
- Si tras la compilación se presentan errores (de sintaxis) en el programa fuente, éstos se deben corregir y proceder a compilar otra vez el programa.

# 3.4. COMPILACIÓN – TIPOS DE ERRORES

- Errores de Compilación
  - Son errores sintácticos, es decir que algo está mal escrito o hay una instrucción incompleta. Si existe un error de sintaxis, la computadora no puede comprender la instrucción, no se obtendrá el programa objeto y el compilador imprime una lista de todos los errores encontrados durante la compilación.
- Errores de Ejecución
  - Estos errores se producen por instrucciones que la computadora puede comprender pero no ejecutar, ejemplo: la división por cero, o calcular raíces cuadradas de número negativos.
- Errores Lógicos
  - Ocurren cuando el programa se ejecuta pero los resultados no son correctos. Son los más difíciles de detectar.

# 3.5. EJECUCIÓN

- La ejecución de un programa consiste en que el computador procese cada una de las instrucciones del programa.
- Al ejecutarse el programa debe hacerse con una amplia variedad de datos de entrada, llamados datos de prueba, que determinarán si el programa tiene errores.

# 3.6. DOCUMENTACIÓN

- Consiste en describir los pasos a dar en el proceso de resolución de un problema.
- Programas pobremente documentados son difíciles de leer, más difíciles de depurar y casi imposibles de mantener y modificar.
- La documentación de un programa se hace a través de las líneas de comentarios, y se incluyen tantas como sean necesarias para aclarar o explicar el significado de las líneas de código.
- Ejemplo de comentarios en el lenguaje C:

```
//Esto es un comentario

/* Para escribir en varias
líneas hay que hacerlo así */
```

# ¿PREGUNTAS?