**ETAPAS PARA EL DESARROLLO DE PROBLEMAS UTILIZANDO EL COMPUTADOR**.

El proceso de la creación de software requiere el uso de una metodología sistemática de desarrollo que permita un acercamiento gradual a la solución del problema que se intenta resolver. Esta metodología, llamada Ciclo de Desarrollo del Software, consta de una serie de pasos lógicos secuenciales denominados Fases. Aunque es posible crear programas sin la aplicación de esta metodología, el producto resultante carece de los beneficios que provee la utilización de este enfoque.

Las Fases de la Programación son:

1. Definición del problema
2. Análisis del problema
3. Diseño de la solución
4. Codificación
5. Prueba y Depuración (Puesta a Punto o Testing)
6. Documentación
7. Implementación (Producción)
8. Mantenimiento
9. **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

1. **Análisis del problema**

Una vez que se ha comprendido lo que se desea de la computadora, es necesario definir: Los datos de entrada. Cuál es la información que se desea producir (salida) Los métodos y fórmulas que se necesitan para procesar los datos.

Dado que se busca una solución se precisan especificaciones de entrada y salida. Para poder definir bien un problema es conveniente responder a las siguientes preguntas: ¿Qué entradas se requieren? (cantidad y tipo) ¿Cuál es la salida deseada? (cantidad y tipo) ¿Qué método produce la salida deseada?

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.

**Mientras no se comprenda con claridad el problema por resolver no puede pasarse a la fase siguiente.**

1. **Diseño de la solución.**

En esta fase se diseña la lógica de la solución a usar, o sea, cómo hará el programa la tarea que se desea automatizar usando los **datos de entrada** para generar los **datos de salida**, enfatizándose los diseños limpios, sencillos y claros. Pueden plantearse diferentes alternativas de solución al problema y elegir la más adecuada, la que produzca los resultados esperados en el menor tiempo y al menor costo. El proceso de diseño se realiza en dos pasos:

* 1. **Elaboración del Algoritmo:**

Una vez definido y analizado el problema, se procede a la creación del algoritmo (Diagrama de flujo o pseudocódigo), en el cual se da la serie de pasos ordenados que nos proporcione un método explícito para la solución del problema

En el caso de emplear diagramas de flujo pueden emplearse herramientas de software tales como el DFD v1.0. Si se opta por el pseudocódigo, se recomienda escribirlos a doble interlínea para efecto de facilitar modificaciones o adición de acciones no consideradas y dotarlos de las siguientes características.

La definición de un algoritmo debe describir con claridad las tres partes fundamentales del problema: Entrada, Proceso y Salidas encontradas en las fases de Definición y Análisis del problema.

Se deja sentado que todos los programas empiezan primero en papel, no directamente frente a la computadora. Aún los programadores más experimentados plasman en papel sus ideas y soluciones antes de programarlas.

* 1. **Prueba de escritorio:**

Para cerciorarnos de que el diagrama (y/o el pseudocódigo) está bien, y, para garantizar que el programa que codifiquemos luego también funcione correctamente, es conveniente someterlo a una Prueba de Escritorio . Esta prueba consiste en que damos diferentes datos de entrada al programa y seguimos la secuencia indicada en el diagrama, hasta obtener los resultados. El análisis de estos nos indicará si el diagrama esta correcto o si hay necesidad de hacer ajustes. Se recomienda dar diferentes datos de entrada y considerar todos los posibles casos, aun los de excepción o no esperados, para asegurarnos de que el programa no producirá errores en ejecución cuando se presenten estos casos.

1. **Codificación**: En este paso se traduce el algoritmo ya estructurado, verificado y comprobado a mano, al **lenguaje de programación** que vaya a utilizarse. Sólo se convierten las acciones del algoritmo en **instrucciones de computadora** usando la sintaxis de un lenguaje particular, pero requiere de conocimientos del lenguaje y de sumo cuidado en la colocación de las instrucciones, las que deben apegarse y seguir fielmente a la lógica del algoritmo y la semántica y sintaxis del lenguaje.

**La digitación**, el acto de teclear el algoritmo codificado, se lleva a cabo para almacenar el programa en la memoria de la computadora (virtual o física) y pueda ser aceptado por esta. Con frecuencia los programadores realizan la codificación y la digitación al mismo tiempo a fin de ahorrar tiempo, pero esto puede conducir a errores debido a la pérdida de concentración que implica el uso de un editor.

**La compilación**, o corrección de los errores sintácticos y semánticos del código, es la eliminación de los errores "gramaticales" según las reglas de construcción de instrucciones particulares del propio lenguaje (la sintaxis). Puede hacerse a medida que se traduce, pero es mejor al final para no perder la secuencia de la codificación. Al terminar debe tenerse el código libre de los errores antes mencionados.

1. **Prueba y Depuración (Puesta a Punto o Testing):** Los errores humanos dentro de la programación de computadoras 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 depuración. La depuración o prueba resulta una tarea tan creativa como el mismo desarrollo de la solución, por ello se debe considerar con el mismo interés y entusiasmo.

Las pruebas que se aplican al programa son de diversa índole y generalmente dependen del tipo de problema que se está resolviendo. Comúnmente se inicia la prueba de un programa introduciendo datos válidos, inválidos e incongruentes y observando cómo reacciona en cada ocasión.

Los resultados obtenidos en las pruebas pueden ser cualquiera de los siguientes:

* La lógica del programa está bien, pero hay errores sencillos, los cuales los corregimos eliminando o modificando algunas instrucciones o incluyendo nuevas.
* Hay errores ocasionados por fallas en la lógica, lo que nos obliga a regresar a las fases de Diseño y Codificación para revisión y modificación del diagrama.
* Hay errores muy graves y lo más aconsejable es que regresemos a la fase 2 para analizar nuevamente el problema, y repetir todo el proceso.
* No hay errores y los resultados son los esperados. En este caso guardamos el programa permanentemente en un medio de almacenamiento.

Puede ser necesario en la mayoría de los casos retroceder a fases previas de desarrollo, revisar el algoritmo otra vez en caso de errores de análisis y/o lógica (que son los más difíciles de detectar, a diferencia de los de sintaxis y semántica), realizar ajustes al código y una serie de nuevas ejecuciones de prueba para que el programa funcione correctamente. Si no existen errores en el programa, puede entenderse la depuración como una etapa de refinamiento en la que se ajustan detalles para optimizar el desempeño del programa.

1. **Documentación:** Es la fase más ignorada por la mayoría de los programadores noveles, por razones de tiempo, costos o simple pereza. Pero no documentar los programas es un mal hábito en programación y un gran error. Será muy difícil a los usuarios entender un programa si no cuentan con un manual de operaciones (el Manual de Usuario). También para los programadores que necesiten darle mantenimiento o hacerle modificaciones si no existe ninguna documentación acerca de sus fases de desarrollo. Incluso será difícil de entender para el mismo autor, algún tiempo después.

Es la guía o comunicación escrita es sus variadas formas, ya sea en enunciados, procedimientos, dibujos o diagramas. A menudo un programa escrito por una persona, es usado por otra. Por ello la documentación sirve para ayudar a comprender o usar un programa o para facilitar futuras modificaciones (mantenimiento).

**Manual del Usuario:** Describe paso a paso la manera cómo funciona el programa, con el fin de que el usuario obtenga el resultado deseado.

1. **Implementación (Producción)**

El programa ya probado, revisado y mejorado se considera terminado y puede utilizarse con un alto grado de confianza para resolver los problemas que dieron origen a su creación. Si se está automatizando alguna tarea manual, ésta última se desecha para emplear solamente el programa

1. **Mantenimiento:** Se lleva acabo después de terminado el 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.