GUIA DE ESTUDIO PARA FUNDAMENTOS ALGORITMICOS

PROPÓSITO

ESTUDIAR, ANALIZAR Y APLICAR TODOS LOS CONCEPTOS FUNDAMENTALES NECESARIOS PARA RESOLVER PROBLEMAS REALES USANDO LAS TÉCNICAS ALGORITMICAS, QUE EN UN FUTURO SERÁN TRADUCIDAS A UN LENGUAJE DE PROGRAMACIÓN PARTICULAR.

Referencias recomendadas:

- NORTON, Peter. Introducción a la Informática. Tercera Edición.
- Fundamentos de programación. Luis Joyanes Aguilar
- Peña Tresancos / M.C. Vidal Fernández. Introducción a la Informática. McGraw- Hill, edición 2004.
- Internet.

INTRODUCCIÓN

La computadora no debe ser considerada, solo como una herramienta que nos permite solucionar problemas, podemos pasar de ser **usuarios finales** a ser **programadores** y/o **diseñadores de sistemas**, entonces también podemos ser creadores de soluciones, usando como herramienta de trabajo la computadora, como materia prima nuestra inteligencia, capacidad de análisis, habilidad para detectar soluciones a problemas y como forma de expresión de esas soluciones utilizaremos los algoritmos y los lenguajes de programación, con estos últimos crearemos programas como resultado de un previo análisis y diseño de algoritmos.

Lenguaje: Es una serie de símbolos que sirven para transmitir uno o mas mensajes (ideas) entre dos entidades diferentes. A la transmisión de mensajes se le conoce comúnmente como **comunicación**.

La **comunicación** es un proceso complejo que requiere una serie de reglas simples, pero indispensables para poderse llevar a cabo. Las dos principales son las siguientes:

- Los mensajes deben correr en un sentido a la vez.
- Debe forzosamente existir 4 elementos: Emisor, Receptor, Medio de Comunicación y Mensaje.

GUIA DE ESTUDIO PARA FUNDAMENTOS ALGORITMICOS

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

Lenguajes Algorítmicos: Es un conjunto ordenado 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**).

CARACTERÍSTICAS GENERALES DE UN PROGRAMA O ALGORITMO (Más adelante se hablará de programas)

Con el fin de facilitar la explotación y el mantenimiento de un algoritmo, es fundamental reunir un conjunto de características generales para obtener su máximo rendimiento en el menor plazo de tiempo y esfuerzo. Estas características son:

Legibilidad: Un algoritmo o programa, debe ser claro y sencillo para facilitar su lectura y comprensión a las personas ajenas al programador (autor) de la aplicación. De este modo existe la posibilidad de repartir las tareas de mantenimiento y pruebas.

Fiabilidad: Un programa debe ser "robusto". Es decir, capaz de recuperarse frente a errores o usos inadecuados por parte del usuario y controlar todo posible error que pueda producirse en las distintas operaciones que realicen los dispositivos que se utilizan en la aplicación.

Portabilidad: El diseño del algoritmo debe permitir la codificación en diferentes lenguajes utilizando para ello un diseño único y universal, sin entrar en sentencias u operadores específicos de algún lenguaje en especial.

GUIA DE ESTUDIO PARA FUNDAMENTOS ALGORITMICOS

Modificabilidad: Debe ser fácil y posible su instalación en distintas máquinas. Facilitar al máximo su mantenimiento, modificación y actualización para adaptarlo o mejorarlo a nuevas situaciones.

Eficiencia: Aprovechar y no derrochar los recursos de la máquina. Esto lo conseguimos minimizando el uso de la memoria, el tiempo de proceso y el de ejecución. No debemos dejarnos llevar por los últimos y potentes máquinas que surgen día a día en el mercado. Siempre que podamos debemos pensar que disponemos de una máquina con pocos recursos.

EN GENERAL:

- Un algoritmo debe ser preciso e indicar el orden de realización de cada aso.
- Un algoritmo debe estar definido. Si se sigue un algoritmo dos veces, se debe obtener el mismo resultado cada vez.
- Un algoritmo debe ser finito: si se sigue un algoritmo, se debe terminar en algún momento (número finito de pasos).

METODOLOGÍA PARA LA SOLUCIÓN DE PROBLEMAS POR MEDIO DE COMPUTADORA

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 la computadora; mientras esto no se conozca del todo no tiene mucho caso continuar con la siguiente etapa. En el caso de los ejercicios propuestos en clases, el enunciado de los mismos, será la definición del problema.

2. **Análisis del Problema: Se refiere a una descomposición del problema en tres aspectos fundamentales: ENTRADA-PROCESO-SALIDA.** Una vez que se ha comprendido lo que se desea de la computadora, es necesario definir:

   ENTRADA: Identificar en el problema planteado todos los datos de entrada para su posterior procesamiento y obtener los resultados esperados. Los datos de entrada son aquellos que se requieren ingresar como usuario para alimentar el sistema.

PROCESO: Se refiere a una identificación y descripción del proceso a seguir para poder obtener el resultado deseado. En esta etapa de la metodología se usan métodos, fórmulas y similares para poder procesar los datos de entrada.

SALIDA: Se refiera a la identificación del resultado final esperado a partir de la entrada de datos, luego su procesamiento y finalmente la salida resultante. Es decir, la información que se desea producir.

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.

3. **Diseño del Algoritmo:** Las características de un buen algoritmo son:
   - Debe tener un punto particular de inicio.
   - Debe ser definido, no debe permitir dobles interpretaciones.
   - Debe ser general, es decir, soportar la mayoría de las variantes que se puedan presentar en la definición del problema.
   - Debe ser finito en tamaño y tiempo de ejecución.

4. **Codificación:** Es la operación de escribir la solución del problema (de acuerdo a la lógica del algoritmo), en una serie de instrucciones detalladas, en un código reconocible por la computadora, la serie de instrucciones detalladas se le conoce como código fuente, el cual se escribe en un lenguaje de programación o lenguaje de alto nivel.

5. **Prueba y Depuración:** 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 la misma importancia. Resulta conveniente observar los siguientes principios al realizar una depuración, ya que de este trabajo depende el éxito de nuestra solución.

6. **Documentación:** 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). La documentación se divide en tres partes:

   - **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 pseudocódigo)
     - Diccionario de Datos
     - Código Fuente (programa)

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

7. **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.

TÉCNICAS DE DISEÑO

Los modelos de programación (o metodología de programación), consiste en los métodos y técnicas disciplinadas para desarrollar algoritmos cumpliendo con las características de los programas anteriormente expuestas. Entre los modelos mas comunes tenemos:

GUIA DE ESTUDIO PARA FUNDAMENTOS ALGORITMICOS

"¡Divide y vencerás! ese es el secreto"

SECUENCIAL: Modelo descendente, se basa en una serie de descomposiciones sucesivas del algoritmo inicial que describen las instrucciones que constituyen el programa. El programa quedara formado por una serie de módulos consecutivos que realizaran una parte del algoritmo y en conjunto funcionaran como un programa único.

ESTRUCTURADA: Programación basada en la secuencial añadiendo subprogramas y estructuras secuenciales, de repetición y condicionales, consiguiendo un programa con módulos independientes y reutilizables desde cualquier parte del programa. Los lenguajes de alto nivel tienden a este tipo de programación.

A OBJETOS: Los módulos del programa son independientes y reutilizables, no solo dentro del mismo programa, también en otros programas. Posee las mismas características que los lenguajes estructurados, añadiendo nuevas funciones y mecanismos de trabajo, como los objetos (conjunto de variables y funciones), el encapsulamiento, el polimorfismo y la herencia. A través de estos lenguajes se consigue un mejor rendimiento de los programas y una mayor seguridad en el tratamiento de los datos. Los lenguajes de última generación tienden a ser orientados a objetos.

TOP DOWN: También conocida como de arriba-abajo, consiste en establecer una serie de niveles de mayor a menor complejidad (arriba-abajo) que den solución al problema. Consiste en efectuar una relación entre las etapas de la estructuración de forma que una etapa jerárquica y su inmediato inferior se relacionen mediante entradas y salidas de información.

Este diseño consiste en una serie de descomposiciones sucesivas del problema inicial, que recibe el refinamiento progresivo del repertorio de instrucciones que

van a formar parte del programa. 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.

BOTTOM UP: El diseño ascendente se refiere a la identificación de aquellos procesos que necesitan computarizarse con forme vayan apareciendo, su análisis como sistema y su codificación, o bien, la adquisición de paquetes de software para satisfacer el problema inmediato.

Cuando la programación se realiza internamente y haciendo un enfoque ascendente, es difícil llegar a integrar los subsistemas al grado tal de que el desempeño global, sea fluido. Los problemas de integración entre los subsistemas son sumamente costosos y muchos de ellos no se solucionan hasta que la programación alcanza la fecha límite para la integración total del sistema. En esta fecha, ya se cuenta con muy poco tiempo, presupuesto o paciencia de los usuarios, como para corregir aquellas delicadas interfaces, que en un principio, se ignoran.

Aunque cada subsistema parece ofrecer lo que se requiere, cuando se contempla al sistema como una entidad global, adolece de ciertas limitaciones por haber tomado un enfoque ascendente. Uno de ellos es la duplicación de esfuerzos para acceder al software y más aun al introducir los datos. Otro es, que se introducen al sistema muchos datos carentes de valor. Un tercero y tal vez el mas serio inconveniente del enfoque ascendente, es que los objetivos globales de la organización no fueron considerados y en consecuencia no se satisfacen.