ESTILOS DE PROGRAMACIÓN.

Se entiende por estilos de programación los métodos que existen para confeccionar y realizar un programa y de esa forma mejorar la calidad de los programas de computación.

Las características de un buen programa son:

1. El programa debe funcionar.
2. El programa no debe tener dificultades. Hay que anticipar las situaciones a las que los usuarios van a poner en el programa, es decir, que este libre de errores.
3. El programa debe estar bien documentado. La documentación puede estar de dos formas: documentación externa, que incluye diagramas de flujo, descripciones de los algoritmos, etc. La documentación interna o comentarios en el propio programa que va dirigido exclusivamente al programador.
4. El programa debe ser eficiente. Que sea un programa fácil de leer y de comprender

PASOS PARA LA SOLUCIÓN DE PROBLEMAS USANDO UNA COMPUTADORA

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 el proceso de diseñar programas es esencialmente un proceso creativo, se pueden considerar una serie de fases o pasos comunes, que generalmente deben seguir todos los programadores.

Las siguientes son las etapas que se deben cumplir para resolver con éxito un problema de programación:

1. **Definición del problema**
2. **Análisis del problema**
3. **Selección de la mejor alternativa**
4. **Diagramación**
5. **Prueba de escritorio**
6. **Codificación**
7. **Transcripción**
8. **Compilación**
9. **Pruebas de computador**
10. **Documentación externa**
11. **Mantenimiento**

1.- DEFINICIÓN DEL PROBLEMA

Está dada por el enunciado del problema, el cúal debe ser claro y completo. Es importante que conozcamos exactamente que se desea del computador; mientras qué esto no se comprenda, no tiene caso pasar a la siguiente etapa.
La primera fase en la resolución de un problema por computadora es la definición o análisis del problema. En donde lo más importante es que conozcamos exactamente lo que debe hacer el programa y "que se desea obtener al final del proceso
Para poder definir con precisión el problema se requiere que las especificaciones de entrada y salida sean descritas con detalle ya que esto es un requisito para lograr una solución eficaz; por lo que es conveniente hacerse las siguientes preguntas:

1.− ¿Qué entradas se requieren? (Tipo Y Cantidad)
2.− ¿Cuál es la salida deseada? (Tipo Y Cantidad)
3.− ¿Qué método produce la salida deseada?

2.- ANÁLISIS DEL PROBLEMA

Entendido el problema (que se desea obtener del computador), para resolverlo es preciso analizar:

- Los datos o resultados que se esperan.
- Los datos de entrada que nos suministran.
- El proceso al que se requiere someter esos datos a fin de obtener los resultados esperados.
- Áreas de trabajo, fórmulas y otros recursos necesarios.

Una recomendación muy práctica es el que nos pongamos en el lugar del computador, y analizar que es necesario que me ordenen y en que secuencia, para poder producir los resultados esperados. También da buenos resultados hacer similitudes con la labor de un empleado que hace el mismo trabajo que deseamos programarle al computador.
Una vez que el problema ha sido definido y comprendido, deben analizarse los siguientes aspectos:

1.− Los resultados esperados.
2.− Los datos disponibles.
3.− Herramientas a nuestro alcance para manipular los datos y alcanzar un resultado.

Esta sería un diagrama de la resolución de un problema en su más mínima expresión.

Y mientras esto no se comprenda no puede pasarse a la siguiente etapa.

3.- SELECCIÓN DE LA MEJOR ALTERNATIVA o DISEÑO DE LA SOLUCIÓN:

Analizado el problema, posiblemente tengamos varias formas de resolverlo; disponemos de varias alternativas o estrategias, lo importante es determinar cúal es la mejor de todas: la que produce los resultados esperados en el menor tiempo y al menor costo. Claro que aquí también es muy válido el principio de que las cosas siempre se podrán hacer de una mejor forma.
Para realizar el diseño de la solución, como todos sabemos, una computadora no tiene capacidad para solucionar problemas más que cuando se le proporcionan los sucesivos pasos a realizar, esto se refiere a la obtención de un algoritmo que resuelva adecuadamente el problema.
En caso de que obtengamos varios algoritmos, seleccionaremos uno de ellos utilizando criterios ya conocidos.
Esta etapa incluye, la descripción del **algoritmo** resultante en un lenguaje natural, en un diagrama de flujo o natural de programación. (en este caso para fijarlo claramente lo separamos en otros pasos más que desarrollamos a continuación)
De esta manera, solo se establece la metodología para alcanzar la solución en forma conceptual, es decir; sin alcanzar la implementación en el sistema de cómputo.
Así tenemos que la información proporcionada constituye su entrada y la información producida por el algoritmo constituye su salida.

Los problemas complejos se pueden resolver más eficazmente por la computadora cuando se dividen en subproblemas que sean más fáciles de solucionar.

Ampliaremos el tema algoritmos más adelante.

4.- DIAGRAMACIÓN

Una vez que sabemos cómo resolver el problema, pasamos a dibujar gráficamente la lógica de la alternativa seleccionada. Eso es precisamente un Diagrama de Flujo: la representación gráfica de una secuencia lógica de pasos a cumplir por el computador para producir un resultado esperado.

La experiencia nos ha demostrado que resulta muy útil trasladar esos pasos lógicos planteados en el diagrama a frases que indiquen lo mismo; es decir, hacer una codificación del programa pero utilizando instrucciones en Español. Como si le estuviéramos hablando al computador. Esto es lo que denominaremos Algoritmo o Pseudocódigo.

Cuando logremos habilidad para desarrollar programas, es posible que no elaboremos el diagrama de flujo; en su lugar podremos hacer directamente el pseudocódigo del programa.

5.- PRUEBA DE ESCRITORIO

Para cerciorarnos de que el diagrama (y/o el pseudocódigo) esta 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 (volver al paso 4). 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.

6.- CODIFICACIÓN

Una vez que hayamos verificado el diagrama mediante las pruebas de escritorio, codificamos el programa en el lenguaje de computador seleccionado. Esto es, colocamos cada paso del diagrama en una instrucción o sentencia, utilizando un lenguaje que el computador reconoce.

Codificación es la escritura en un lenguaje de programación de la representación del algoritmo desarrollada en etapas precedentes. Esto se refiere a la obtención de un programa definitivo que pueda ser comprensible para la máquina.
Cabe destacar que si la codificación original se realizó en papel, previo a la compilación deberá existir un paso conocido como transcripción.

Y posteriormente, una vez que el algoritmo se ha convertido en un programa fuente. Este programa fuente debe ser traducido a lenguaje máquina, este proceso se realiza con el compilador, y se obtiene el programa objeto, (siempre y cuando el programa fuente sea correcto) que posteriormente se vuelve un programa ejecutable.

Todos los lenguajes de programación proveen facilidades para incluir líneas de comentarios en los programas. Estos comentarios aclaran lo que se ordena al computador y facilitan entender el programa. Puesto que estos comentarios no son tenidos en cuenta como instrucciones, y aparecen en los listados del programa, resulta muy conveniente agregar abundantes comentarios a todo programa que codifiquemos. Esto es lo que se denomina Documentación Interna.

7.- TRANSCRIPCIÓN

El programa codificado es necesario que lo llevemos a un medio que sea aceptado como entrada por el computador: lo perforamos en tarjetas, lo grabamos en un disco flexíble o lo grabamos en un disco duro. Este programa es el que se conoce como *Programa Fuente* (Source).

8.- COMPILACIÓN

Utilizamos ahora un programa de computador llamado Compilador o Traductor, el cúal analiza todo el programa fuente y detecta errores de sintaxis ocasionados por fallas en la codificación o en la transcripción. Las fallas de lógica que pueda tener nuestro programa fuente no son detectadas por el compilador. Cuando no hay errores graves en la compilación, el compilador traduce cada instrucción del programa fuente a instrucciones propias de la máquina (Lenguaje de Maquina), creando el Programa Objeto.

Algunos computadores utilizan Interpretadores, (Generalmente para el Lenguaje Basic), en reemplazo de programas compiladores. La diferencia consiste en que el interpretador recibe, desde una terminal, sólo una instrucción a la vez, la analiza y, si esta bien, la convierte al formato propio de la maquina. Si la instrucción tiene algún error, el interpretador llama la atención de la persona para que corrija dicha instrucción.

Como resultado de la corrida del compilador, podemos obtener varios listados:

- Listado del programa fuente
- Listado de los errores detectados
- Listado de campos utilizados, etc.

Los errores los debemos corregir sobre el mismo programa fuente, ya sea reemplazando las tarjetas mal perforadas o regrabando en el disco flexíble o en el disco duro. Este paso de la compilación lo repetimos hasta eliminar todos los errores y obtener el programa ejecutable.

9.- PRUEBAS DE COMPUTADOR

La Prueba se realiza tras la compilación. Cuando tenemos el programa ejecutable (en lenguaje de maquina), ordenamos al computador que lo ejecute, para lo cual suministramos datos de prueba, como lo hicimos en la prueba de escritorio . Los resultados obtenidos los analizamos, luego de lo cual puede ocurrir cualquiera de estas situaciones:

a.- La lógica del programa esta bien, pero hay errores sencillos, los cuales los corregimos modificando algunas instrucciones o incluyendo unas nuevas; el proceso debemos repetirlo desde el paso 6 .

b.- Hay errores ocasionados por fallas en la lógica, lo que nos obliga a regresar a los pasos 4 y 5 para revisión y modificación del diagrama.

c.- Hay errores muy graves y lo más aconsejable es que regresemos al paso 2 para analizar nuevamente el problema, y repetir todo el proceso.

d.- No hay errores y los resultados son los esperados. En este caso, el programa lo podemos guardar permanentemente en una librería o biblioteca del computador, para sacarlo de allí cuando necesitemos ejecutarlo nuevamente.

Si tras la compilación se presentan errores (errores de compilación) en el programa fuente, es preciso volver a editar el programa, corregir los errores y compilar de nuevo, este proceso se repite hasta que no se producen errores.
De esta manera se obtiene el programa objeto, que todavía no es ejecutable directamente, pero si no contiene errores se debe instruir al sistema para que realice la fase de montaje o enlace del programa objeto con las librerías del programa del compilador; este proceso de montaje produce un programa ejecutable.
La Depuración es el proceso de encontrar los errores del programa y corregir o eliminar dichos errores. Cuando se ejecuta un programa, se pueden producir tres tipos de errores:

1.− Errores de compilación. Se producen normalmente por un uso incorrecto de las reglas del lenguaje de programación y suelen ser errores de sintaxis, por lo tanto la computadora no puede comprender la instrucción, y obviamente no se obtendrá el programa objeto, y el compilador imprimirá una lista de todos los errores encontrados durante la compilación.

2.− Errores de ejecución. Estos errores se producen por instrucciones que las computadoras pueden comprender, pero no ejecutar. Ejemplos de éstos son: una división por cero, y raíces cuadradas de números negativos; por lo que en este caso se detiene la ejecución del programa y se imprime un mensaje de error.

3.− Errores lógicos. Se producen en la lógica del programa y la fuente del error suele ser el diseño del algoritmo. Estos errores son los más difíciles de detectar, ya que el programa puede funcionar y no producir errores de compilación ni ejecución, y solo puede detectarse cuando se advierte un error por la obtención de resultados incorrectos.

En este caso se debe volver a la fase del diseño del algoritmo, modificarlo, cambiar el programa fuente, compilar y ejecutar una vez más.

10.- DOCUMENTACIÓN EXTERNA

Cuando el programa ya se tiene listo para ejecutar, es conveniente que hagamos su documentación externa siguiendo las normas de la instalación o las recomendaciones indicadas por el profesor. Una buena documentación incluye siempre:

a. Enunciado del problema
b. Diagrama de pasada
c. Narrativo con la descripción de la solución

d. Relación de las variables o campos utilizados en el programa, cada uno con su respectiva función
e. Diagrama del programa
f. Listado de la última compilación
g. Resultados de la ejecución del programa.

Es un proceso para convertir especificaciones generales de un sistema en instrucciones utilizables por la máquina, que produzcan los resultados deseados. Se le conoce también como desarrollo de software. La documentación de un problema consta de las descripciones de los pasos a dar en el proceso de resolución de un problema. La importancia de la documentación es por su decisiva influencia en el producto final.

Programas pobremente documentados son difíciles de leer, más difíciles de depurar y casi imposibles de mantener y modificar. Por ello la importancia de la documentación, sin la documentación es imposible corregir errores futuros o bien cambiar el programa

11- MANTENIMIENTO

El mantenimiento se define como la modificación del programa por medio de actualizaciones, que mejoran al programa, corrigiendo errores o bien actualizándolos para un mejor funcionamiento.

Por ello la documentación es sin duda muy importante para poder llevar a cabo el mantenimiento

Es uno de los procesos fundamentales dentro de la Ingeniería de Sistemas ya que permite que el sistema continúe con sus características de excelencia y calidad a través del tiempo. En este proceso de mantenimiento se suele dedicar más de la tercera parte de lo invertido en el sistema de programación.

ALGORITMO :

Es una serie de operaciones detalladas a ejecutar paso a paso, que conducen a la resolución de problemas.

Es un conjunto de reglas para resolver determinado problema describiendo de forma lógica su solución.

Cada una de las acciones de que consta un algoritmo es denominada sentencia y éstas deben ser escritas en términos de cierto lenguaje comprensible para el computador, que es el lenguaje de programación.

Para diseñar un algoritmo se debe comenzar por identificar las tareas más importantes para resolver el problema y disponerlas en el orden en que han de ser ejecutadas.

Conjunto de reglas o instrucciones que indican una secuencia lógica de operaciones que proporciona la respuesta a cualquier tipo de problema dado.

Un algoritmo es una serie de pasos lógicos para realizar una acción, programa o tarea ya que es el primer paso para realizar un programa.
Ejemplos de algoritmos son:

- ¨ Instrucciones para montar una bicicleta
- ¨ Hacer una receta de cocina
- ¨ Obtener el máximo común divisor de dos números, etc.

Los algoritmos se pueden expresar por fórmulas, diagramas de flujo, y pseudocódigos. Ésta última representación es la más utilizada en lenguajes estructurados como Turbo Pascal.
La vida cotidiana está llena de soluciones algorítmicas, algunas de ellas son tan comunes que no se requiere pensar en los pasos que incluye la solución. La mayoría de las actividades que se realizan diariamente están compuestas por tareas más simples que se ejecutan en un orden determinado, lo cual genera un algoritmo. Por ejemplo, son tareas comunes, realizar una llamada telefónica, buscar un número en el directorio telefónico, buscar un anuncio en las páginas amarillas del directorio, preparar café, regar las plantas, poner en funcionamiento un automóvil, cambiar una llanta, entre muchas otras.

Muchos de los procedimientos utilizados para desarrollar tareas cotidianas son algorítmicos, sin embargo, esto no significa que todo lo que se hace está determinado por un algoritmo. El cumplimiento

de las características mencionadas anteriormente permitirá determinar si un procedimiento es o no es algorítmico.

Una receta de cocina para preparar un plato cualquiera puede ser un algoritmo, pero también puede no serlo, dependiendo de las especificaciones. Si una de las instrucciones a desarrollar dice "aplicar sal al gusto" ya puede afirmarse que no es un algoritmo, porque contiene un elemento subjetivo, no definido. Esta misma acción podría aparecer de la forma "aplicar 20 gramos de sal", en cuyo caso no se requiere del gusto (subjetivo) de quien lo aplica y por tanto no contradice el principio de la algoritmia.

El primer paso en el diseño de un algoritmo es conocer la temática a tratar, el segundo será pensar en las actividades a realizar y el orden en que deben ejecutarse para lograr el objetivo, el tercero y no menos importante es la presentación formal.

Un algoritmo cotidiano es:
La serie de pasos que realizamos en nuestra vida diaria para realizar las diferentes tareas y actividades comunes, desde los pasos al levantarnos, así como ir de compras, etc.

¿De dónde viene la palabra Algoritmo?
Del árabe tomamos más de cuatro mil palabras. Los árabes estuvieron presentes en España por más de 700 años (de 711 a 1492). Su idioma se extendió en aquellos territorios que ocuparon, especialmente al sur, conviviendo al principio con hablas mozárabes derivadas del latín, y cuando fueron expulsados y las tierras repobladas por cristianos de la España cristiana que hablaban todos lenguas romances, esta lengua árabe se perdió, dejando sin embargo algunos arabismos en castellano y otras lenguas peninsulares. Los árabes trajeron algunas nuevas tecnologías, ciencias y organización política.
La palabra "algoritmo" es de origen árabe. Viene de "al-Khwārizmī", sobrenombre del célebre matemático Mohamed ben Musa. Khwārizmī quiere decir "de Khwārizm", el estado donde nació Ben Musa. Al-Khwārizmī vivió en entre los años 780-840.
Algoritmo originalmente utilizada como alguarismo, guarismo. Posteriomente ´´se produjo en el propio latín medieval la contaminación con el griego aritmo- ´´número´ ´ (ver: aritmética), dando lugar a algoritmo.

Abu Abdallah Muḥammad ibn Mūsā al-Jwārizmī (Abu Yāffar) (أبو عبد الله محمد بن موسى الخوارزمي ابو جعفر), conocido generalmente como al-Juarismi, fue un matemático, astrónomo y geógrafo persa musulmán chií, que vivió aproximadamente entre 780 y 850.
Poco se conoce de su biografía, a tal punto que existen discusiones no saldadas sobre su lugar de nacimiento. Algunos sostienen que nació en Bagdad. Otros, siguiendo el artículo de Gerald Toomer1 (a su vez, basado en escritos del historiador al-Tabari) sostienen que nació en la ciudad corasmia de Jiva, en el actual Uzbekistán. Rashed2 halla que se trata de un error de interpretación de Toomer, debido a un error de transcripción (la falta de la conectiva wa) en una copia del manuscrito de al-Tabari. No será este el último desacuerdo entre historiadores que encontraremos en las descripciones de la vida y las obras de al-Juarismi. Estudió y trabajó en Bagdad en la primera mitad del siglo IX, en la corte del califa al-Mamun. Para muchos, fue el más grande de los matemáticos de su época.

Definición De Lenguaje Algorítmico
Para definir el lenguaje algorítmico, cabe recordar que el conjunto de todas las operaciones a realizar, y el orden en el que deben efectuarse, se le denomina algoritmo.
Así que el lenguaje algorítmico es aquel por medio del cual se realiza un análisis previo del problema a resolver y encontrar un método que permita resolverlo.

Historia Y Aplicación De Los Lenguajes Algorítmicos
Al igual que los idiomas sirven de vehículo de comunicación entre los seres humanos, existen lenguajes que realizan la comunicación entre los seres humanos y las computadoras.
Estos lenguajes permiten expresar los programas o el conjunto de instrucciones que el operador humano desea que la computadora ejecute.
Los lenguajes de computadoras toman diferentes formas; los de las primeras computadoras, como la ENIAC y la EDSAC, se componían en el lenguaje real de las máquinas mismas. La dificultad de programar las máquinas de esta manera limitaba drásticamente su utilidad y proporcionaba un fuerte

incentivo para que se desarrollaran lenguajes de programación más orientados hacia la expresión de soluciones con la notación de los problemas mismos.
Los primeros lenguajes de programación se conocieron como Lenguajes Ensambladores, un ejemplo es: TRANSCODE, desarrollado para la computadora FERUT.
En los lenguajes ensambladores se define un código especial llamado mnemotécnico para cada una de las operaciones de la máquina y se introduce una notación especial para especificar el dato con el cual debe realizarse la operación.

A mediados de los años 60's aparecieron los primeros lenguajes de propósito general como FORTRAN, FORTRAN IV, ALGOL, COBOL, BASIC, PL/I, ADA, C, C++, PASCAL, etc. pero el desarrollo de nuevas tecnologías, tanto en arquitectura de computadoras como en lenguajes de programación, continúa a paso acelerado, cada vez con mayor velocidad, el panorama está cambiando de una etapa de sistemas y lenguajes especialmente desarrollados para aplicaciones individuales. Los lenguajes de programación actuales son los conocidos como Lenguajes visuales, como por ejemplo Visual Fox, Visual Basic, Visual C.

Criterios que debe satisfacer un algoritmo (características):

1. Exactitud o **precisión.** Cada instrucción debe ser clara y sin ambigüedad, tanto en las operaciones como en el orden.
2. **Finito**. Esta característica implica que el número de pasos de un algoritmo, por grande y complicado que sea el problema que soluciona, debe ser limitado. Todo algoritmo, sin importar el número de pasos que incluya, debe llegar a un final. Para hacer evidente esta característica, en la representación de un algoritmo siempre se incluyen los pasos inicio y fin.
3. **Eficiente.** Cada instrucción puede ser verificada por una persona con una prueba manual que satisfaga los requerimientos planteados por el problema.¨ Preciso: Esto quiere decir que debe indicar el orden en cada paso.hablar de eficiencia o complejidad de un algoritmo es evaluar los recursos de cómputo que requiere para almacenar datos y para ejecutar operaciones frente al beneficio que ofrece. En cuanto menos recursos requiere será más eficiente el algoritmo.
4. **Presentación formal**: para que el algoritmo sea entendido por cualquier persona interesada es necesario que se exprese en alguna de las formas comúnmente aceptadas; pues, si se describe de cualquier forma puede no ser muy útil ya que solo lo entenderá quien lo diseñó. Las formas de presentación de algoritmos son: el pseudocódigo, diagrama de flujo y diagramas de Nassi/Schneiderman, entre otras.
5. **Corrección:** el algoritmo debe ser correcto, es decir debe satisfacer la necesidad o solucionar el problema para el cual fue diseñado. Para garantizar que el algoritmo logre el objetivo, es necesario ponerlo a prueba; a esto se le llama verificación o prueba de escritorio.
6. Ser **definido.** Ya que en el área de programación, el algoritmo se desarrolla como paso fundamental para desarrollar un programa, es necesario tener en cuenta que el computador solo desarrollará las tareas programadas y con los datos suministrados; es decir, no puede improvisar y tampoco se inventará o adivinará el dato que necesite para realizar un proceso. Por eso, el algoritmo debe estar plenamente definido; esto es, que cuantas veces se ejecute, el resultado depende estrictamente de los datos suministrados. Si se ejecuta con un mismo conjunto de datos de entrada, el resultado será siempre el mismo.

Partes de un Algoritmo

- **Entrada**. Son cero o más cantidades las cuales son externamente sustituidas.
- **Salida.** Al menos una cantidad es producida.

- **Proceso:** El trabajo que realizamos con los datos para obtener los resultados deseados.

1. **ACTIVIDAD**

Tarea: Investigar en Internet distintos Algoritmos resueltos. Averigua sobre Diagrama de Flujo, las partes de un diagrama de flujo, sus nombres y dibujos.

Cuando hablamos de comprender un problema decimos que podemos definir claramente cuáles son:

1. Los datos
2. El proceso
3. Los resultados

Cuando hablamos de datos nos referimos a la unidad mínima de información. Ellos no tienen "significado" en sí mismos. Sino que su significancia depende del proceso y unión de los mismos en una estructura informativa.

Ejemplo:

Podemos tener como dato: gato, perro, bicicleta, señora, niño. Según el proceso que hagamos podemos tener distintas "informaciones o resultados". Veremos algunas:

- *Una señora fue atropellada por un niño que iba en bicicleta. Este se había asustado al cruzarse en su camino un perro que iba persiguiendo a un gato.*
- *Un niño nos comentó, muy complacido, que una señora pasó en bicicleta llevando un gato y un perro.*

Y así podríamos seguir mucho más….

El resultado obtenido depende del proceso….. Por eso:

si el programa está mal… en realidad está mal el proceso…

Tenemos que tener en cuenta que las computadoras pueden trabajar con dos tipos de datos, y según ellos es el tipo de computadora:

h. **Datos Analógicos** Una computadora analógica manipula "datos analógicos".
i. **Datos Digitales**. Una computadora digital trabaja con "datos digitales".

Los **datos digitales** son magnitudes que sólo pueden tomar valores de un rango discreto. Las variables que pueden tomar un número finito de valores. Por ser de fácil realización los componentes físicos con dos estados diferenciados, es este el número de valores utilizado para dichas variables, que por lo tanto son binarias. Siendo estas variables más fáciles de tratar (en lógica serian los valores V y F) son los que generalmente se utilizan para relacionar varias variables entre si y con sus estados anteriores. Por ejemplo, el número de habitantes de una ciudad, el número de hijos de una persona, etc. De manera gráfica, los datos digitales se pueden representar mediante una línea discontinua de puntos.

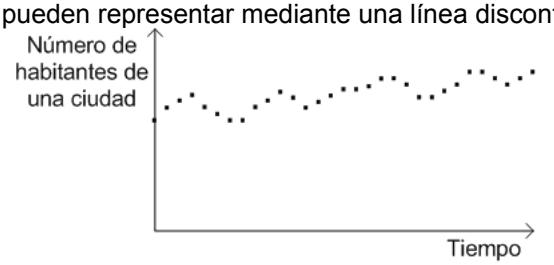


Figura. Dato digital: número de habitantes de una ciudad a lo largo del tiempo.

Matemáticamente, todos los datos digitales pueden ser representados con números enteros. En concreto, las computadoras digitales trabajan con dígitos binarios, llamados bits. Conceptualmente, las computadoras digitales actuales son muy similares entre sí, ya que, todas ellas están basadas en una arquitectura propuesta en 1946 por el estadounidense John von Newmann (1903-1957).

Los **datos analógicos** son magnitudes que pueden tomar valores de un rango continuo. Son aquellas que pueden tomar un número infinito de valores comprendidos entre dos límites. La mayoría de los fenómenos de la vida real dan señales de este tipo. (presión, temperatura, etc.)Por ejemplo, la temperatura de un cuerpo, la altura de una persona, etc. Gráficamente, los datos analógicos se pueden representar con una línea continua.

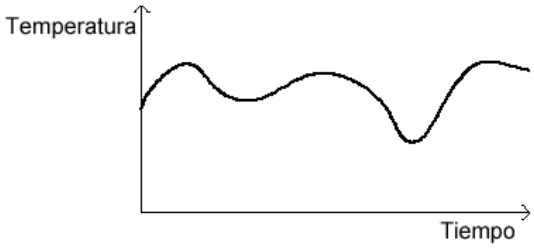


Figura. Dato analógico: temperatura de un cuerpo a lo largo del tiempo.

Matemáticamente, todos los datos analógicos pueden ser representados con números reales. Las computadoras analógicas suelen trabajar con niveles de tensión o presiones hidráulicas.

El **Proceso** es el trabajo, acción, operación, gestión de los distintos datos para llegar al resultado deseado. Éste trabajo se realiza en la computadora gracias al "Programa" que va dando las indicaciones de cómo hacer el "proceso" con los datos.

Un **programa** es un conjunto de instrucciones u órdenes que indican a la máquina las operaciones que ésta debe realizar con unos datos determinados. En general, todo programa indica a la computadora cómo obtener unos datos de salida, a partir de unos datos de entrada.
En la siguiente figura se muestra, gráficamente, el funcionamiento básico de un programa.

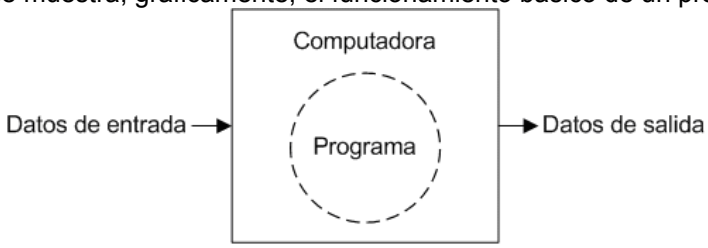


Figura. Funcionamiento básico de un programa en una computadora digital.
Por ejemplo, un programa que sirva para realizar la suma de dos números enteros cualesquiera (por ejemplo, del 3 y el 5), podría representarse de la siguiente manera:

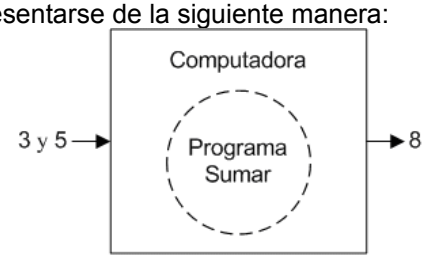
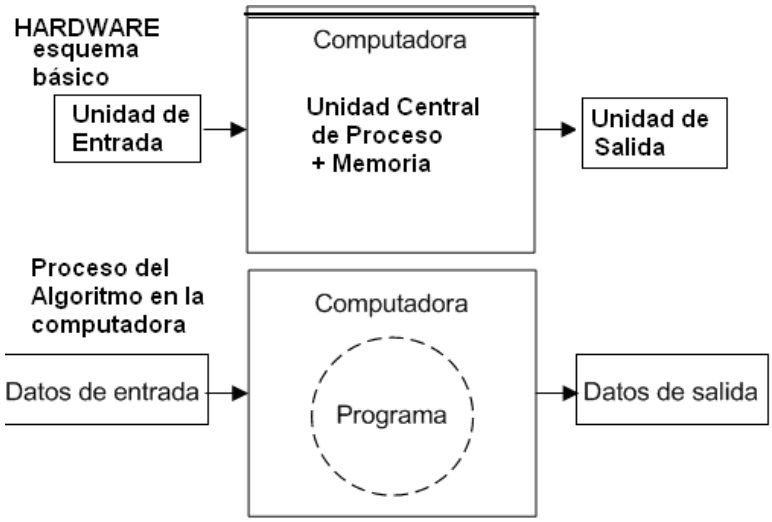


Figura. Programa sumar.
Los programadores son las personas que desarrollan, diseñan, crean los programas. Es decir que los programas surgen de las ideas de los seres humanos expresadas de alguna forma especial para que la máquina las entienda.
Los resultados son la información y/o valores buscados o perseguidos de un proceso con datos. Es lo que se observa en la pantalla o en el papel impreso como resultado del trabajo en la computadora.



En el gráfico observamos el programa alojado en la memoria y funcionando en el procesador de la computadora. Para que pueda estar allí la computadora debe "comprenderlo". Es decir no podemos colocar un algoritmo en la computadora directamente. Antes es necesario "codificarlo" o traducir las instrucciones en un lenguaje. Puede ser : el lenguaje de máquina (binario) o un lenguaje de programación. El lenguaje de programación elegido puede ser más cercano (Alto nivel) o más lejano

(Bajo nivel) al lenguaje humano. Y como todo lenguaje posee "palabras claves" , semántica y sintaxis específica.
Una vez escrito el programa en lenguaje de programación (Programa Fuente) necesitamos hacer un proceso más para que la computadora pueda ejecutarlo. En este paso Traducimos o Compilamos el fuente y obtenemos el Programa Objeto. En este paso se le agregan al programa las indicaciones necesarias para las interrupciones por entrada /salida, y otros elementos necesarios para la computadora.
El Traductor interpreta y ejecuta las ordenes una a una. No obtenemos un programa objeto final. Y cada ves que se tiene que ejecutar se tiene que volver a traducir.
El compilador traduce todo el programa fuente obteniendo un programa objeto final que es el que se va a ejecutar.
Entonces: el programa fuente puede ser modificado, en cambio el programa objeto no. Para modificarlo es necesario modificar el fuente, compilarlo y obtener un nuevo programa objeto.

Ampliemos màs el tema, un programa es una lista de instrucciones que la computadora debe seguir para procesar datos y convertirlos en información. Las instrucciones se componen de enunciados usados en lenguajes de programación como Basic, Pascal o C.

CARACTERÍSTICAS DEL PROGRAMA:

- Debe ser confiable y funcional
- Advertir errores de entrada obvios y comunes
- Documentado adecuadamente
- Ser comprensible
- Codificado en el lenguaje apropiado

Por lo tanto, definiremos los elementos de un proceso en computadora de la siguiente forma:

DATOS:

Son las características propias de cualquier entidad. Por ejemplo: los datos de una persona como su edad, fecha de nacimiento, domicilio, número de teléfono, etc.

INFORMACIÓN:

Es el conocimiento relevante producido como resultado del procesamiento de datos y adquirido por la gente para realzar el entendimiento y cumplir ciertos propósitos.

PROCESAMIENTO DE DATOS:

Consiste en la recolección de datos de entrada que son evaluados y ordenados para ser colocados de manera que produzcan información útil.

ACTIVIDADES DEL PROCESAMIENTO DE DATOS

I. Captura de datos de entrada
II. Manejo de los datos (incluye clasificación, ordenación, cálculo y sumarización de éstos)
III. Administración de la salida resultante.

OBJETIVOS DE LA PROGRAMACIÓN

En la preparación de un programa, el programador puede tener que escoger entre soluciones alternativas en muchos puntos. Cada elección debe hacerse para satisfacer los objetivos y restricciones de la tarea de programación particular. Aquí asumiremos como apropiados para toda tarea de programación los siguientes objetivos:

- Exactitud
- Claridad
- Eficiencia

Esto significa:

- EXACTITUD

Un objetivo obvio en la escritura de cualquier programa de computador es que tiene que satisfacer su especificación exactamente. A menudo, sin embargo, a causa de la complejidad de la labor del programa, y de un entendimiento o cuidado inadecuados de parte del programador, un programa falla en satisfacer alguna parte de su especificación. Un programador tiene que ser en todo tiempo cuidadoso de la exactitud o pertinencia del programa para su propósito especificado.

Un factor clave en el logro de exactitud es la simplicidad. Escogiendo el algoritmo o técnica más simple disponible, es más probable que un programador vea si satisface o no los requerimientos de la especificación del programa, y es menos probable que la describa incorrectamente en su programa. La innecesaria complejidad no cumple propósito alguno en la programación de computadores.

Algunos programas son, por supuesto, inherentemente complejos. Para tales programas, el programador debe adoptar un tratamiento sistemático que controle y limite la complejidad de la que tiene que ocuparse en cada etapa.

- CLARIDAD

Un programa es necesariamente tan complejo como el algoritmo que describe. Sin embargo, es importante que la forma en que el algoritmo esté descrito, por el texto del programa, no sea mas complicada de lo que es necesario. La claridad del programa es una ayuda importante para el programador mismo en el diseño y limpieza del programa; y para otros que puedan tener que leer y alterar el programa en alguna etapa posterior.

La claridad del programa es lograda casi en la misma forma que para cualquier texto escrito, tal como un ensayo o un libro en los cuales se requiere:

a).- Una separación lógica del texto en partes comprensibles (capítulos, secciones, etc.) que reflejen la distinción entre los temas que describen, y su presentación en una secuencia lógica que refleje las relaciones entre ellas;

b).- Una selección cuidadosa de las características del lenguaje, usadas en cada parte para expresar su sentido propuesto tan precisamente como sea posible;

c).- Otra selección cuidadosa de las palabras usadas para denotar los objetos y conceptos involucrados;

d).- La inclusión de comentarios y preámbulos para clarificar el texto principal cuando sea necesario;

e).- Un aprovechamiento de los dispositivos para presentación de textos, tales como las líneas en blanco y la sangría, para enfatizar la relación entre partes componentes de textos.

Un programador podría ser tan diligente en el uso de éstas técnicas para lograr claridad como el autor de cualquier texto. En muchos casos, la utilidad de un programa es determinada tanto por la claridad de su texto como por las cualidades del algoritmo que describe.

- EFICIENCIA

El costo de ejecutar un programa de computador, es medido normalmente en términos de :

a).- El tiempo tomado por el computador para llevar a cabo la secuencia de operaciones involucradas;

b).- La cantidad de memoria de computador usada en hacerlo.

En muchos ambientes (de máquina), el programa competirá con otros programas por el uso de esos recursos del computador, y es, por tanto, lógico minimizar los requerimientos del programa para cada uno.

El tiempo tomado para ejecutar el programa, es directamente proporcional al número de operaciones que el procesador tiene que realizar al hacerlo. El programador debe, por tanto, escoger un algoritmo que minimice las operaciones implicadas, y tener cuidado de evitar cualquier operación redundante al expresar el algoritmo como un programa de computador.

La memoria usada por el programa durante su ejecución, es determinada por la cantidad de datos que tienen que ser guardados, y por el número de instrucciones del procesador requeridas para definir el programa, ya que éstas también tienen que ser guardadas en la memoria. Para minimizar el almacenamiento usado por su programa, el programador debe, por tanto, considerar los datos manipulados y el número de operaciones especificadas por el programa. Para algunos programas, o partes de programas, el uso eficiente del tiempo o del almacenamiento puede ser crítico; para otros

puede serlo menos. El programador debe estar enterado de cualquiera de tales requerimientos de eficiencia cuando escriba su programa.

DATOS

DATO en PROGRAMACIÓN

Es la expresión general que describe los objetos con los cuales opera el programa. Por ejemplo, la edad y el domicilio de una persona, forman parte de sus datos. Los datos se sitúan en objetos llamados variables.
Un elemento a tener en cuenta a la hora de programar es el tipo de dato con el cuál trabajaremos, de esto depende las operaciones que se puedan hacer con ellos. Todos los datos tienen un tipo asociados con ellos que nos servirá para poder conocer con que información trabajaremos. Es decir, cuando ingresemos el sueldo de un trabajador necesitamos que este contenga decimales, o al solicitar la edad de una persona está tiene que estar con números enteros, etc.. Además la suma entre caracteres no tiene sentido.

Las **variables** son zonas de memoria, cuyo contenido cambia durante la fase de procesamiento de información, que tienen un nombre o denominación puesto por el programador y un tipo de dato definido. Son objetos cuyo valor puede ser modificado a lo largo de la ejecución de un programa. Las variables llevan un nombre llamado identificador. Este puede ser una cadena de letras y dígitos, empezando siempre con una letra. Por ejemplo: Pi, curso99, nom_alum,etc.

Los **Identificadores** son palabras creadas por los programadores para dar nombre a los objetos y demás elementos que necesitamos declarar en un programa: variables, constantes, tipos, estructuras de datos, archivos, subprogramas, etc.
Representan los nombres de los objetos de un programa (constantes, variables, tipos de datos, procedimientos, funciones, etc.). Es una secuencia de carácteres que puede ser de cualquier longitud, aunque tiene ciertas reglas que hay que seguir, las cuales son:

- Debe comenzar con una letra en la mayoría de los lenguajes y no puede contener espacios en blanco.
- Letras, dígitos y carácteres subrayados ("_") están permitidos después del primer carácter.

En síntesis un identificador es un método para nombrar a las celdas de memoria en la computadora, en lugar de memorizarnos una dirección de memoria.
Se utilizan para nombrar variables, constantes, procedimientos y funciones.

Es necesario tener en cuenta còmo exige el lenguaje de programación que sean los identificadores o nombres. En C++ las letras mayúsculas se tratan como diferentes y distintas unas de otras. Por ejemplo en este lenguaje, contador, Contador y CONTADOR son tres nombres de identificadores distintos. Un identificador no puede ser igual a una palabra reservada, y no debe tener el mismo nombre que una función, ya sea definida por el usuario o de la biblioteca de C.

Las **Constantes** : Son objetos cuyo valor permanece invariable a lo largo de la ejecución de un programa. Una constante es la denominación de un valor concreto, de tal forma que se utiliza su nombre cada vez que se necesita referenciarlo. Por ejemplo, si se desea obtener un reporte para cada uno de los empleados de una empresa, con sus datos generales, la fecha y cantidad de dinero que recibieron la última semana, el dato fecha puede ser una constante ya que es el mismo para todos.
Las constantes son valores que no pueden cambiar en la ejecución del programa. Recibe un valor en el momento de la compilación del programa y este no puede ser modificado.

Las **Expresiones .** Son representaciones de un cálculo necesario para la obtención de un resultado. Estas pueden ser valores constantes, funciones o combinaciones de valores, variables, funciones y operadores que cumplen determinadas reglas de construcción de una expresión. Son un conjunto de operadores y operandos que producen un valor. Por ejemplo:

Cos (pi * X) + 12.56 * SQR(100)

Tipo : es el conjunto de valores que puede tomar y guardar una variable. Es el conjunto de transformaciones y funciones internas y externas definidas sobre el conjunto de datos. Se tienen dos tipos de datos: Simples como numéricos y alfanuméricos y Estructuras de datos que pueden ser internas o externas.

Tipos de Datos Comunes
Estos son los tipos de datos mas utilizados en los lenguajes de programación:

- Numéricos
- Caracteres y
- Lógicos

Tipos Numéricos

Numéricos son aquellos cuyo contenido es una serie de dígitos (0-9) que en conjunto nos proporcionan un valor numérico ya sea entero o real y pueden ser precedidos de un signo + ó -.Dentro de estos tipos se puede hacer mención de los tipos enteros, reales o de coma flotante, y de los exponenciales. Ellos pueden usarse en cálculos y funciones matemáticas. Son los dígitos del 0 al 9, la coma y el punto.

Tipos Carácter o Alfanuméricos:
O Alfanuméricos, son las Letras, los Dígitos del 0 al 9, los signos, los símbolos. Los tipos carácter se dividen también en caracteres ASCII, como por ejemplo: a A & * , etc.. El otro grupo de caracteres son los strings o cadenas de caracteres, como por ejemplo: "Hola Mundo". No se pueden usar en cálculos. Pero sí se les puede aplicar a ellos las funciones alfanuméricas. Observen que el valor va entrecomillado, es un comentario o texto. Así lo distinguimos de los otros tipos de datos. Tipos de datos Alfanuméricos son aquellos cuyo contenido son letras del abecedario, números o caracteres especiales o bien una combinación de ellos.

Tipos Lógicos o booleanos:
Los tipos lógicos solamente pueden tomar los valores verdadero o falso.

Tipos de Datos

Numéricos: Dígitos del cero al nueve, coma decimal y punto.
- **Enteros:**
- **Reales**
  - **Simple precisión**
  - **Doble precisión**
  - **Punto flotante**
  - **Otros (como: moneda, fecha, etc)**
- **Lógicos o booleanos.**

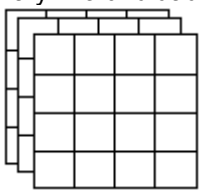
Alfanuméricos: Caracteres ASCII, Letras, dígitos del 0 al 9, Signos, Símbolos.
- Caracteres
- String
- Memo
- Otros.

Lógicos: Verdadero / Falso.

Datos por el espacio de memoria que ocupan

Todo lo que ingresa en la computadora y se pone en ejecución primero se guarda en la memoria.
Cuando estamos ejecutando un programa todos los datos que ingresan se guardan primero en la Memoria RAM o memoria de Trabajo (volátil, se borra al apagar la máquina).
Por lo tanto, los datos también. Cuando los ingresamos por primera vez en la computadora se ubican en la memoria. Luego, podremos guardarlos en un dispositivo de almacenamiento. En este caso diremos que lo guardamos en el disco.
Para poder manejar esta información ubicada en memoria podríamos hacerlo conociendo la ubicación en memoria. Pero podemos hacerlo en forma más sencilla. Cuando el programador escribe el valor en el

programa es una constante. Es un valor que no va a cambiar durante la ejecución del programa. Por ejemplo: 121, "Salta 3439", Verdadero.
Cuando el dato debe ser ingresado por el usuario, durante la ejecución del programa, es una variable. Y al ingresar se guarda en una posición de memoria vacía. No conocemos su contenido pero sí su tipo de dato.
Entonces, Variable es una posición de memoria identificada con un nombre y para un tipo definido de datos.
El nombre de la variable conviene que empiece con una letra y que explicite qué guardamos en ella.
Cada ingreso siempre debe tener una variable (y sólo una).
Estos tipos de datos (constante /variable) ocupan sólo un espacio en memoria. Al ingresar otro se sobrescribe sobre el anterior, perdiéndose este.
Si el programador ve que necesita ingresar y guardar varios datos, y que no es conveniente sobre escribirlos, puede especificar y definir unas variables especiales. Estas variables tienen un nombre y tipo definido por el programador y se llaman Arreglos. Pero también define cuánto espacio quiere que ocupan. Por ejemplo: dos casilleros, 10, etc.
Así tenemos

Variables comunes	
Variable tradicional o única	
Variables definidas por el Programador	
Vector	Línea
	Columna
Matriz	Línea y Columna
	Línea y Columna y Profundidad 

çRecuerda que todos estos tipos de datos especiales son preparadas por el programador. Donde le dice a la computadora cuánto espacio debe ocupar la variable. Seguiremos este tema más adelante.
Cuándo se quiere guardar los datos en el disco hablamos de Archivos.

Los archivos permiten guardar distintos tipos de datos en forma organizada.
Así tenemos que un archivo es un conjunto de Registros. Cada registro es el conjunto de datos de un elemento específico. Por ejemplo los datos de los empleados. Cada registro son los datos de un empleado. Estos son de distinto tipo. Por ejemplo Apellido (alfanumérico), Nombre (alfanumérico), sueldo (numérico real), etc,

Cada registro es un conjunto de campos. Cada campo es un dato específico de un tipo específico. Ejemplo: Mostramos un archivo con dos registros. Uno para Pérez y otro para Ruiz. Los campos son cinco: Apellido, Nombres, Fecha de Pago, Importe pagado, y estado de la cuenta activa (si o no).

Veamos un ejemplo gráfico de un archivo:

Archivo con dos registros		Campo Alfanumérico		Campo Numérico	Campo Lógico
Registro de 5 campos:	Pérez	Juan	17/8/2007	$2.564	Si
Registro	Ruiz	Lila	12/2/2007	$421	No

Este tema también lo veremos más adelante.

- **ACTIVIDAD**

1. Investiga y completa el siguiente cuadro. Tomando como base el lenguaje Visual Basic completa con otros tipos de datos que el lenguaje maneje.

TIPO DE DATO	***OTRO NOMBRE CONOCIDO***	***VALOR MÍNIMO QUE GUARDA***	***VALOR MÁXIMO QUE GUARDA***
ALFANUMÉRICO			
ENTERO			
REAL SIMPLE PRECISIÓN			
REAL DOBLE PRECISIÓN			
BOOLEANO			
FECHA			
PUNTO FLOTANTE			