

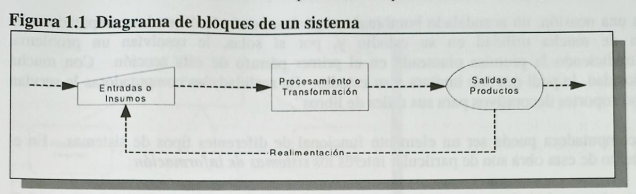
Glosario

SUBTÍTULO DEL INFORME

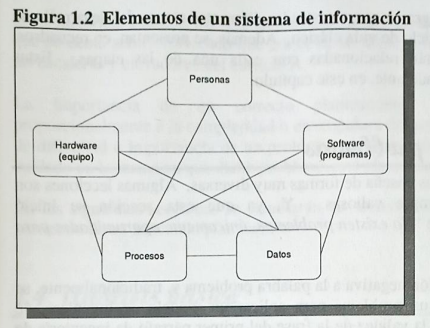
Nombre | Nombre del curso | Fecha

# Sistema

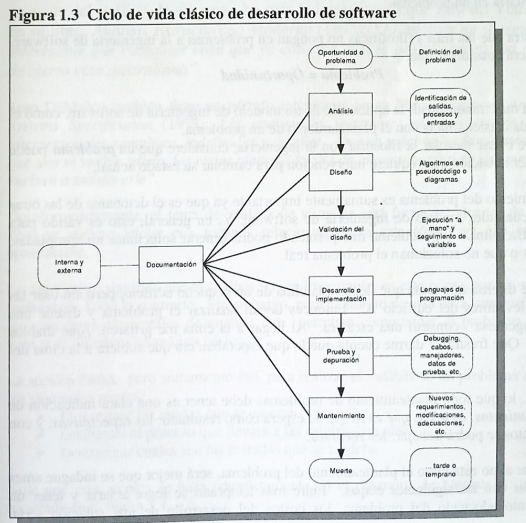
Conjunto de elementos que interactúan entre sí para lograr un objetivo común.



# Sistemas de Información:

En los sistemas de información, el elemento funcional es la computadora.

# Modelo clásico de ingeniería de software:

Un modelo ampliamente difundido y utilizado para fines académicos es el denominado: ciclo de vida clásico. 

# Planteamiento de Problemas

Problema = Oportunidad

Un problema es cualquier situación que requiere intervención para cambiar su estado actual

En general, lo que todo planteamiento de problemas debe tener es una clara indicación de los requerimientos, es decir, que es lo que se espera como resultado

Requerimientos para resolver un problema:

Expectativas

Resultados

Recursos

Retroalimentación

# Análisis Básico

La técnica básica pero sumamente útil para realizar el análisis de un problema cosiste en los siguientes puntos

* Determinar las salidas que se esperan
* Establecer el proceso que llevara a las salidas deseadas
* Determinar cuáles serán las entradas que se tendrán

En un análisis básico sugerido, tanto para cada salida como para cada entrada, deberá determinar las siguientes propiedades:

* Nombre
* Descripción
* Tipo

Para los procesos existen dos cosas importantes que un análisis básico debe proporcionar:

* Descripción de los procesos que generan las salidas, a partir de las entradas.
* Dibujos o diagramas libres.

# Diseño de la Solución

Una de las metodologías más didácticas y útiles para proponer y desarrollar soluciones es el Diseño Descendente, es decir, dividir un problema grande en problemas más pequeños

Finalmente, se sugiere realizar cualquier dibujo, diagrama, tabla o garabato que permita gráficamente comprender la naturaleza del problema.

En programación, existen diferentes herramientas para expresar el diseño de una solución.

En esta sección se presentarán dos temas básicos: algoritmos y diagramas de flujo.

# Algoritmos

Se puede definir algoritmo como un conjunto ordenado y finito de operaciones que plantea una solución a un problema determinado.

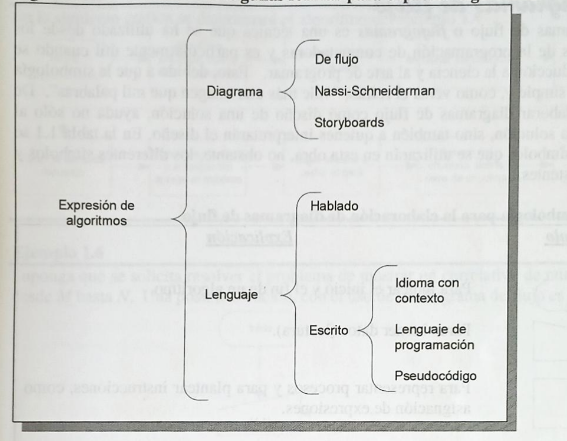
la palabra algoritmo tiene su origen en el nombre de un famoso matemático árabe llamado Al-Khoreoni, quien nació a finales del siglo VIII en Khorezm, al

Los algoritmos pueden ser expresados de diferentes maneras, las cuales se pueden agrupar, básicamente, en las siguientes categorías:

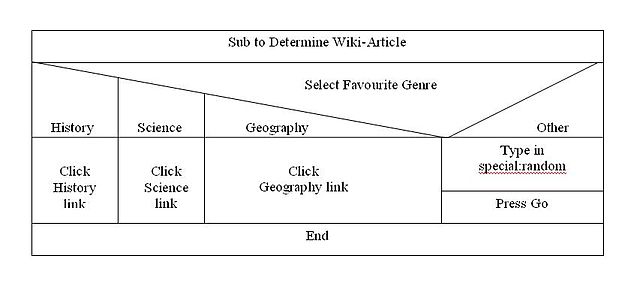
Lenguaje

Diagramas

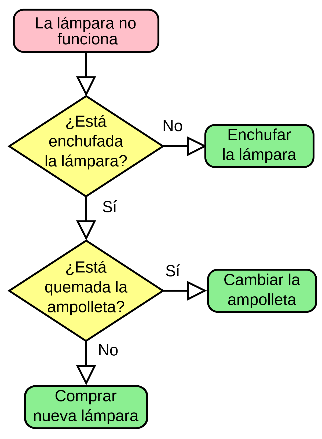
La categoría Lenguaje se puede subdividir en leguaje hablado y lenguaje escrito. Los algoritmos expresados en lenguaje hablado no tienen más restricciones que las impuestas por un idioma en un contexto determinado.



Modelo de diagrama de nassi schneiderman



Modelo de diagrama de flujo



Escriba un algoritmo para cambiar la llanta de un carro

cambiar la llanta de un auto (versión chapina) :

1. Aflojar los chuchos, con la llave de chuchos .
2. Levantar eI carro con el gato.
3. Quitar los chuchos.
4. Quitar la llanta.
5. Poner la Llanta de repuesto.
6. Poner los chuchos y enroscar sin apretar a1 máximo.
7. Bajar el carro y quitar eI gato.
8. Apretar los chuchos, con la llave de chuchos .

ALGORITMO DE COMO PREPARAR UN CAFÉ CON LECHE

1. Coger una taza
2. Cogerel café
3. Coger una cuchara
4. Coger una pava
5. Abrir la llave del agua
6. Llenar la pava con agua
7. Cerrar la llave del agua
8. Coger un encendedor
9. Si no tiene encendedor, puede comprar/usar fósforos (DECISIÓN)
10. Con el encendedor/fósforo prenderla hornalla
11. Poner a calentar la pava con el agua
12. Retirar la pava con agua cuando eI agua hierva
13. Poner con la cuchara la cantidad deseada de café en la taza
14. Verter agua caliente en la taza
15. Coger el azúcar (opcional)
16. Si quiere que su café quede dulce, agréguele azúcarcon una cuchara (DECISIàN2
17. Guardare/ azúcar
18. Revolver(REPITICION)

La suma de dos numeros:

1. Agarramos la primera cantidad y la segunda cantidad
2. Ponemos una cantidad de lado derecho y otra del lado izquierdo
3. Agregamos el signo + en medio de ambas cantidades
4. Operamos
5. Escribimos el resultado

# 1.5.2 Diagrama de Flujo

Dibujar diagramas de flujo o flujogramas es una técnica que se ha utilizado desde los inicios mismos de la programación de computadoras y es particularmente útil cuando se hace una introducción a la ciencia y al arte de programar. Esto, debido a que la simbología es sumamente similar , como versa el refrán, "vale más una imagen que mil palabras". De

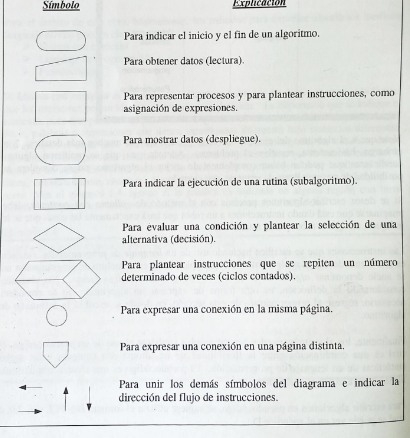
Los diagramas de flujo son una representación gráfica de un algoritmo. Muestran

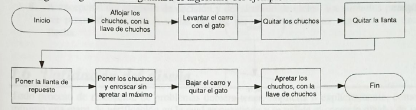
los pasos a seguir para la solución de un problema, y se valen de símbolos

estandarizados. En las siguientes líneas se listarán algunos lineamientos básicos sugeridos

para elaborar diagramas de flujo.

* Un diagrama de flujo debe ser simple, claro y general.
* Todo diagrama de flujo debe indicar claramente su inicio y su fin.
* Las flechas, o líneas de flujo, deben ser rectas, verticales u horizontales.
* EI flujo del diagrama debe ir de aniba hacia abajo y/o de izquierda a derecha,
* siempre que sea posible.
* Cuando sea posible, los símbolos deben dibujarse proporcionalmente.
* No deben existir partes por las cuales nunca pase la lógica del algoritmo en ejecución.
* Use los conectores, en la misma o diferente página, para evitar que las flechas del flujo entorpezcan la claridad del diagrama, pero no abuse de ellos.
* A excepción del símbolo de decisión y del símbolo de ciclos, los símbolos pueden ser precedidos de una o varias flechas, pero sólo pueden ser seguidos por una flecha.





# 1.6 Validacion del Diseño

Todo proceso de resolución de problemas debiera tener, implícita o explícitamente, como mínimo una etapa de verificación en la cual se cuestione y se valide el correcto diseño de la solución.

Para validar los diseños, en eI contexto de esta obra, se sugiere realizar una ejecución a mano, completa o parcial, de sus algoritmos. Para ello se debe simular la ejecución siguiendo la secuencia de instrucciones escritas en el diseño.

# 1.6.1 Datos de prueba

La selección de los datos de prueba es la base para una correcta validación del diseño. Básicamente, existen cuatro tipos de datos de prueba:

1. Típicos
2. Ilegales
3. Extremos
4. Aleatorios

Los datos de prueba típicos son los valores más usuales dentro del dominio del problema. Inicie siempre la validación de sus diseños con datos típicos. Estos datos de prueba deben ser sencillos, de modo que los resultados sean verificables en forma rápida.

Los datos ilegales son aquellos valores que no están permitidos para el problema en particular, pero que su diseño debe contemplar a efecto de reaccionar adecuadamente a esta posibilidad. EI ejemplo típico de estos datos de prueba se da en la selección de opciones alfanuméricas en un menú. Si, por ejemplo, el menú principal de un sistema contempla las opciones:

1. Ingreso

2. Proceso

3. Reporte

4. Salir

Se debería probar el sistema con valores que estuvieran fuera del rango válido de opciones. Los datos extremos son aquellos valores que están en la frontera del rango válido de opciones. Particularmente los contadores y las dimensiones de los arreglos de datos son susceptibles a errores en sus valores extremos.

EI último tipo de datos de prueba son valores tomados de forma aleatoria, es decir, al azar.

# 1.7 Desarrollo

En esta obra se Ie llama desarrollo a la etapa del ciclo de vida del software en la que el diseño realizado se transcribe a uno o más lenguajes de programación, con eI objeto de crear un programa que se ejecute en una computadora en panicular. En esta etapa se realizan varias actividades, sin embargo, es común que se haga referencia a ella como etapa de codificación o simplemente programación.

# 1.7.1 Paradigmas de Programación

Como su nombre 10 indica, la programaci6n secuencial se caracteriza por una secuencia de instrucciones, las cuales son ejecutadas una tras Otra.

En la programación estructurada se proponen básicamente, tres tipos de estructuras las cuales se pueden agrupar en rutinas. Las estructuras  
básicas propuestas en este paradigma son: estructuras secuenciales, estructuras selectivas y estructurarse repetitivas.

Uno de los principales conceptos que introduce la programaci6n estructurada es la abstracción.

La programación orientada a objetos surge como un nuevo paradigma que permite acoplar el diseño de programas a situaciones del mundo real. Las entidades centrales son IOS objetos, que son tipos de datos que encapsulan estructuras de datos y las operaciones o algoritmos que manipulan esos datos. Este paradigma de programaci6n se fundamenta en los siguientes conceptos: objeto, clase, métodos, mensajes, herencia, polimorfismo, abstracción y encapsulamiento.

Un fenómeno interesante relacionado con los paradigmas básicos de programación es que ninguno ha anulado al anterior, es más, cada nuevo paradigma ha utilizado el anterior y ha ampliado las posibilidades para Ios programadores.

Paradigmas de programación. De manera que es probable que en sus investigaciones  
encuentre Otros Tipos de programación que sean clasificados como paradigmas, entre otros:  
Programación declarativa  
Programación funcional

Programación lógica  
Programación orientada a eventos

# 1.7.2 Lenguaje de Programacion

Lenguaje de programación es un conjunto de reglas e instrucciones estandarizadas para la creación de programas de computadora.

Es pertinente iniciar esta sección definiendo algunos términos. Primero es el de programa de computadora. Básicamente, es un conjunto de instrucciones detalladas que le indican a una computadora c6mo resolver un problema.

Existen muchas formas de clasificar los lenguajes de programaci6n. Una de ellas es clasificarlos según su grado de dependencia de la máquina, de la siguiente manera:

Lenguaje maquina: Consiste en sucesiones de dígitos binarios. Todas las instrucciones se escriben haciendo uso de secuencias de estos dígitos. Aún en la actualidad, es el único lenguaje interno que entiende la computadora; los programas se escriben en lenguajes de mayor nivel y se traducen a lenguaje de máquina para que esta los pueda ejecutar,

Lenguajes ensambladores: Se diferencian de los lenguajes de máquina ya que, en lugar de usar c6digos binarios, las instrucciones se representan con símbolos fáciles de reconocer, conocidos como mnemotécnicos. A1 igual que los lenguajes de la máquina, los lenguajes ensambladores son únicos para una computadora particular.

Lenguaje de medio nivel: Combinan la abstracci6n de los lenguajes de alto nivel con el acceso cercano al hardware. Algunos programadores consideran que el lenguaje C, por ejemplo, puede ser clasificado como lenguaje de medio nivel.

Lenguaje de alto nivel: Son independientes de la máquina y se pueden utilizar en una e variedad de computadoras. Los lenguajes de alto nivel no ofrecen necesariamente mayores capacidades de programación, pero si ofrecen una interacción más avanzada entre programador y computadora.

# 1.7.4 Programacion

Las computadoras no pueden hacer mucho sin un programa. Los programas deben ser escritos por programadores. Así que estas personas tienen una gran responsabilidad para lograr que esa máquina llamada computadora funcione según las expectativas planteadas.

Se requiere un ingrediente adicional para convertirse en un auténtico programador de computadoras: voluntad.

# 1.8 Prueba y Depuración

Los tipos de errores en un programa se pueden clasificar como sigue:

* Errores de compilación: léxicos, sintácticos y semánticos.
* Errores en tiempo de ejecución.
* Errores de funcionamiento: aquellos en los que dada una entrada, no se obtiene el resultado correcto.

# 1.8.1 Depuradores

Un depurador es un programa especial utilizado para encontrar errores en la ejecución de otros programas. En general, el depurador permite que el programador interrumpa la ejecución de un programa en cualquier punto, de manera que pueda examinar y modificar los valores que en ese momento tienen elementos como variables y parámetros, por ejemplo. Entre otras, éstas son algunas de las funciones características de los depuradores:

* Iniciar la ejecución de un programa.
* Establecer los valores iniciales de los parámetros del programa.
* Ejecutar las instrucciones en secuencia. Instrucción por instrucción.
* Ejecutar resumidamente una rutina. Es decir, ejecutar la rutina completa.
* Establecer los diferentes puntos en donde el programa se detendrá.
* Avanzar directamente hasta el siguiente punto de parada.
* Visualizar el valor actual de variables, parámetros o estructuras.
* Evaluar y modificar los valores de las expresiones del programa.

# 1.8.2 Estrategias Básicas de Prueba y Depuración

Caja Negra: La mayor parte de los usuarios de programas no tienen interés en los detalles del funcionamiento interno de los mismos. Su interés está centrado en las respuestas que pueden obtener del programa, 10 cual no necesariamente es malo. Es decir, ven el programa como una caja negra. Los datos de prueba se seleccionan tomando como base el planteamiento del problema, sin importar los detalles internos del programa.

Caja de Cristal: La segunda estrategia básica parte del hecho que un programa no puede considerarse como probado por completo si parte de su código nunca ha sido ejecutado. En el método de caja de cristal, se analiza la estructura lógica del programa y se seleccionan los datos de prueba, de manera que se considere cada alternativa posible.

Caja de Pandora: La estrategia de caja de Pandora consiste en abstenerse de realizar pruebas, luego de creer que cl programa está listo para ser entregado al usuario. No está de más resaltar que usualmente es una bomba de tiempo.

Cabos: Para evitar que el compilador produzca errores de sintaxis, por la invocación de rutinas que aún no se han detallado, es preciso utilizar rutinas sustitutas a las cuales se les da el nombre de cabos.

Manejadores: Una manera de probar y depurar una rutina en particular consiste en escribir un programa auxiliar. Breve cuya finalidad es proporcionar la entrada necesaria para la rutina y comprobar así la salida que produce. A este programa auxiliar se le da el nombre de manejador.

# 1.9 Mantenimiento

EI mantenimiento surge por la modificación de los requerimientos o por la necesidad de resolver errores no detectados previamente.

Luego que un programa ha sido codificado, probado y depurado puede requerir algún cambio motivado por las causas anteriores. En cualquiera de los casos se considerará como un mantenimiento al programa.

# 1.10 Documentación

# 1.10.1 Documentación Interna

Poner comentarios a nuestros códigos

# 1.10.2 Documentación Externa

# Existen muchas metodologías y propuestas para realizar la documentación de sistemas o programas. En los siguientes párrafos se describirán las secciones que se sugiere incluir en la documentación externa de sus proyectos. Portada: Como mínimo debe indicar el nombre del proyecto, la versión, la fecha de realización y el autor. Tabla de contenido: Es inaceptable omitirla, pues se obliga al lector a recon•er toda la documentación cada vez que necesita consultar algún detalle puntual. La mayoría de los procesadores de palabras ayudan mucho en esto. Planteamiento del problema: En un ambiente académico esto puede reducirse a incluir el enunciado del problema otorgado por el profesor. En general, se debe definir el objetivo de su programa, no sea que termine programando otra cosa. Como referencia, se sugiere revisar la sección 1.3. Planteamiento de la solución: Como su nombre lo indica, en esta sección debe describirse la solución empleada. EI programa es una expresión detallada del método de solución, por lo que este planteamiento debe hacerse en un alto nivel de abstracción, o en varios, si fuera necesario. EI objetivo es lograr que el lector comprenda rápidamente el enfoque usado para lograr el desarrollo de la solución. Puede usar gráficas o tablas para aumentar del documento producido. "Una imagen vale por mil palabras" Análisis básico: En esta sección se debe especificar el método utilizado en el análisis básico, descrito en la sección 1.4. Definición de módulos: Si su proyecto está compuesto de varios algoritmos y módulos, es necesario que todas estas relaciones queden claramente especificadas. En esta parte es conveniente incluir diagramas que ilustren la interacción entre módulos. Diseño algorítmico: Explicación de la metodología empleada en la elaboración del algoritmo que plantea la solución. EI diseño algorítmico fue descrito en la sección 1.5. Validación del diseño: En cuanto el diseño de una solución ha sido concluido, se puede validar que, en efecto, resuelve el problema en cuestión. En esta sección de la documentación se debe especificar la forma en que se realizó la validación del diseño. Este tema fue tratado en la sección 1.6. Código fuente: No es más que la incorporación del código fuente que escribió como parte de la solución. Para proyectos complejos y extensos esta parte podría ser igualmente extensa, de manera que puede substituirse con una detallada indicación de los archivos que contienen todo el código fuente.

# Manual de Uso: La mayoría de programas no funcionan solos, deben ser operados por usuarios y ellos necesitan instructivos completos. EI instructivo debe contestar todas las preguntas del usuario. A1 escribir un manual de uso, se debe preguntar ¿Contiene el manual de uso las respuestas a todas las preguntas que le surgirán a un usuario? ¿Será necesario alguna vez llamar al programador a media noche cuando un usuario esté lidiando con el programa? Descripción de datos de prueba: Aquí se agrupan todos los casos de prueba que se usaron para verificar el correcto funcionamiento del programa. Desgraciadamente es común omitir esta sección de la documentación, tal vez porque generalmente los programadores se avergüenzan de los errores cometidos en los programas y tratan de ocultar todo aquello que pueda demostrarlo. Pero construir programas correctos es un proceso iterativo. Es difícil lograrlo al primer intento, de manera que encontrar un error es un signo, no de nuestra incompetencia, sino más bien de que el programa se acerca a estar terminado. Incluir los datos de prueba ayuda a demostrar que el programa funciona correctamente, fomenta la programación disciplinada y no desordenada y evita el rediseñar todo los casos de prueba cada vez que se le da mantenimiento al programa. Vea la sección 1.8.2. Glosario: EI glosario no es más que un catálogo de palabras o términos especiales, con su respectiva definición o explicación. Referencias: Para finalizar el documento, puede indicar todas las referencias realizadas incluyendo libros, manuales, revistas e internet.

# 