I. NOCIONES DE PROGRAMACIÓN (CONCEPTOS BÁSICOS)

1.1. INTRODUCCIÓN.

Introducción.

Se pueden utilizar muchos lenguajes para programar una computadora. El más básico es el lenguaje de máquina, una colección de instrucciones muy detallada que controla la circuitería interna de la máquina. Este es el dialecto natural de la máquina. Muy pocos programas se escriben actualmente en lenguaje de maquina por dos razones importantes: primero, porque el lenguaje de maquina es muy incómodo para trabajar y segundo porque la mayoría de las maquinas se pide programar en diversos tipos de lenguajes, que son lenguajes de alto nivel, cuyas instrucciones son más compatibles con los lenguajes y la forma de pensar humanos como lo es el lenguaje c que además es de propósito general.

Debido a que los programas diseñados en este lenguaje se pueden ejecutar en cualquier maquina, casi sin modificaciones. Por tanto el uso del lenguaje de alto nivel ofrece tres ventajas importantes, sencillez, uniformidad y portabilidad.

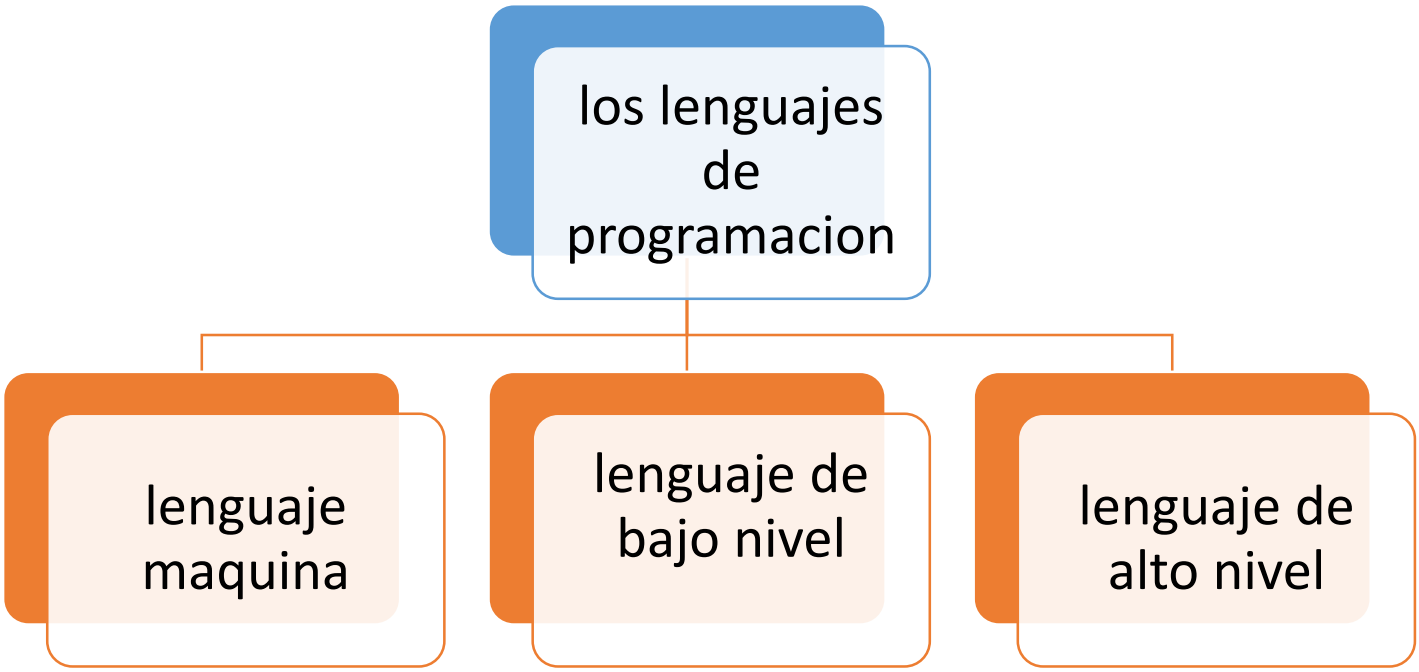
1.1.1. Definición de lenguaje de programación.

Definición

Sistema de símbolos y reglas que permite la construcción de programas con los que la computadora puede operar así como resolver problemas de manera eficaz.

Estos contienen un conjunto de instrucciones que nos permiten realizar operaciones de entrada, salida, calculo, manipulación de textos, lógica, comparación y almacenamiento, recuperación y etc.

Los lenguajes se clasifican en



Lenguaje máquina.

Son aquellos cuyas instrucciones son directamente entendibles por la computadora y no necesitan traducción posterior para que la CPU pueda comprender y ejecutar el programa. Las instrucciones en lenguaje maquina se expresan en términos de la unidad de memoria más pequeña el bit (dígito binario 0 ó 1).

Lenguaje de bajo nivel.

(Ensamblador): En este lenguaje las instrucciones se escriben en códigos alfabéticos conocidos como mnemotécnicos para las operaciones y direcciones simbólicas.

Lenguaje de alto nivel.

Los lenguajes de programación de alto nivel (BASIC, pascal, cobol, fortran, etc.) son aquellos en los que las instrucciones o sentencias a la computadora son escritas con palabras similares a los lenguajes humanos (en general en inglés), lo que facilita la escritura y comprensión del programa.

1.1.2. Definición de algoritmos.

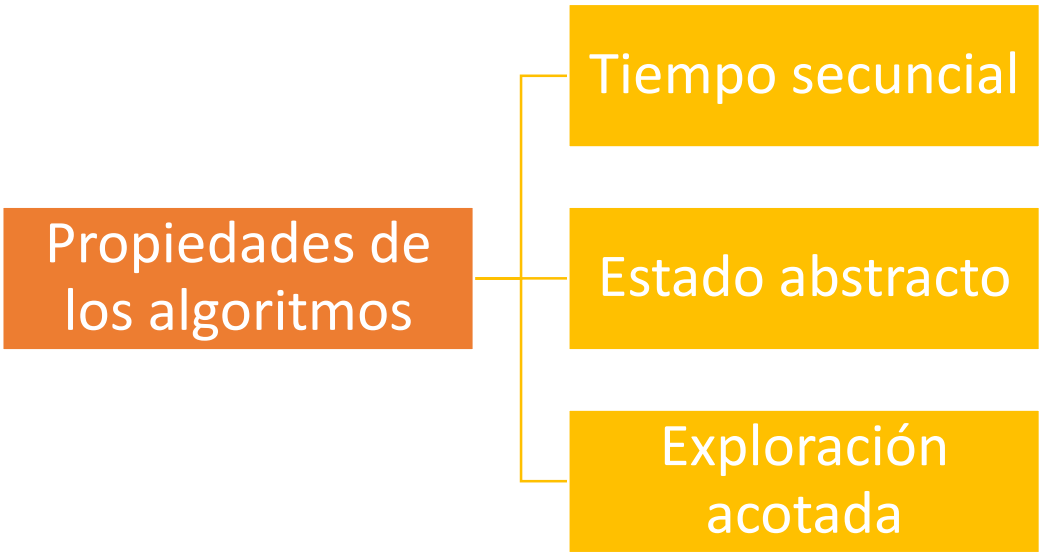
En matemáticas, ciencias de la computación y disciplinas relacionadas, un algoritmo (del griego y latín, dixit algorithmus y este a su vez del matemático persa Al-Juarismi) es un conjunto preescrito de instrucciones o reglas bien definidas, ordenadas y finitas que permite realizar una actividad mediante pasos sucesivos que no generen dudas a quien deba realizar dicha actividad.2 Dados un estado inicial y una entrada, siguiendo los pasos sucesivos se llega a un estado final y se obtiene una solución. Los algoritmos son el objeto de estudio de la algoritmia.

En la vida cotidiana, se emplean algoritmos frecuentemente para resolver problemas. Algunos ejemplos son los manuales de usuario, que muestran algoritmos para usar un aparato, o las instrucciones que recibe un trabajador por parte de su patrón. Algunos ejemplos en matemática son el algoritmo de la división para calcular el cociente de dos números, el algoritmo de Euclides para obtener el máximo común divisor de dos enteros positivos, o el método de Gauss para resolver un sistema lineal de ecuaciones.

Definición formal.

En general, no existe ningún consenso definitivo en cuanto a la definición formal de algoritmo. Muchos autores los señalan como listas de instrucciones para resolver un problema abstracto, es decir, que un número finito de pasos convierten los datos de un problema (entrada) en una solución (salida).1 2 3 4 5 6 Sin embargo cabe notar que algunos algoritmos no necesariamente tienen que terminar o resolver un problema en particular. Por ejemplo, una versión modificada de la criba de Eratóstenes que nunca termine de calcular números primos no deja de ser un algoritmo.7

A lo largo de la historia varios autores han tratado de definir formalmente a los algoritmos utilizando modelos matemáticos como máquinas de Turing entre otros.8 9 Sin embargo, estos modelos están sujetos a un tipo particular de datos como son números, símbolos o gráficas mientras que, en general, los algoritmos funcionan sobre una vasta cantidad de estructuras de datos.3 1 En general, la parte común en todas las definiciones se puede resumir en las siguientes tres propiedades siempre y cuando no consideremos algoritmos paralelos



Tiempo secuencial.

Un algoritmo funciona en tiempo discretizado –paso a paso–, definiendo así una secuencia de estados "computacionales" por cada entrada válida (la entrada son los datos que se le suministran al algoritmo antes de comenzar).

Estado abstracto.

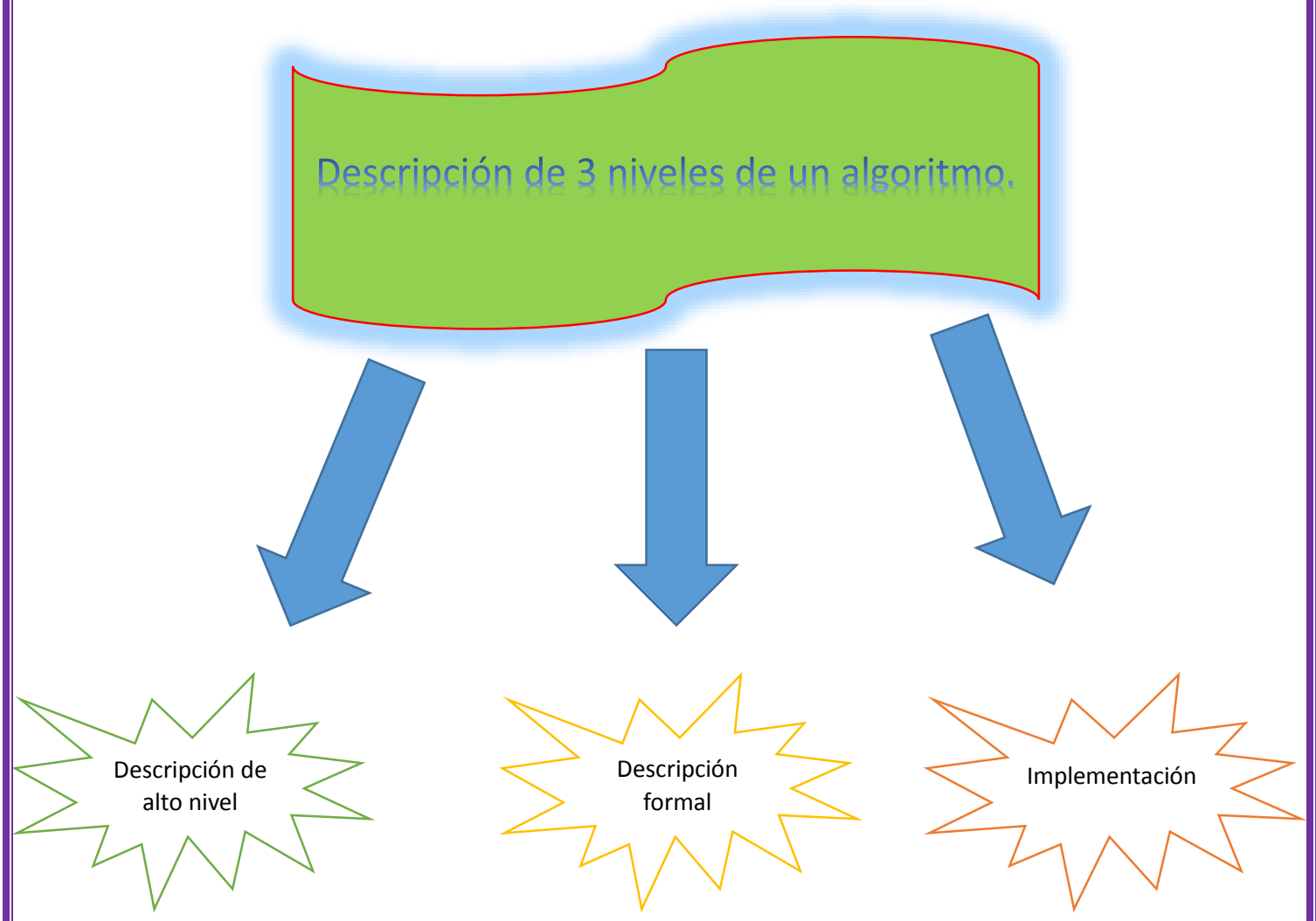
Cada estado computacional puede ser descrito formalmente utilizando una estructura de primer orden y cada algoritmo es independiente de su implementación (los algoritmos son objetos abstractos) de manera que en un algoritmo las estructuras de primer orden son invariantes bajo isomorfismo.

Exploración acotada.

La transición de un estado al siguiente queda completamente determinada por una descripción fija y finita; es decir, entre cada estado y el siguiente solamente se puede tomar en cuenta una cantidad fija y limitada de términos del estado actual.

Medios de expresión de un algoritmo.

Los algoritmos pueden ser expresados de muchas maneras, incluyendo al lenguaje natural, pseudocódigo, diagramas de flujo y lenguajes de programación entre otros. Las descripciones en lenguaje natural tienden a ser ambiguas y extensas. El usar pseudocódigo y diagramas de flujo evita muchas ambigüedades del lenguaje natural. Dichas expresiones son formas más estructuradas para representar algoritmos; no obstante, se mantienen independientes de un lenguaje de programación específico.



Descripción de alto nivel.

Dado un conjunto finito C de números, se tiene el problema de encontrar el número más grande. Sin pérdida de generalidad se puede asumir que dicho conjunto no es vacío y que sus elementos están numerados como $c_0, c_1, \ldots, c_n$.

Es decir, dado un conjunto $C = \{c_0, c_1, \ldots, c_n\}$ se pide encontrar m tal que $x \leq m$ para todo elemento x que pertenece al conjunto$C$.

Para encontrar el elemento máximo, se asume que el primer elemento (c_0) es el máximo; luego, se recorre el conjunto y se compara cada valor con el valor del máximo número encontrado hasta ese momento. En el caso que un elemento sea mayor que el máximo, se asigna su valor al máximo. Cuando se termina de recorrer la lista, el máximo número que se ha encontrado es el máximo de todo el conjunto.

Descripción formal.

El algoritmo puede ser escrito de una manera más formal en el siguiente pseudocódigo:

Algoritmo Encontrar el máximo de un conjunto
función max(C) //C es un conjunto no vacío de números// $n \leftarrow |C|$ //$|C|$ es el número de elementos de C// $m \leftarrow c_0$ **para** $i \leftarrow 1$ **hasta** n **hacer** **si** $c_i > m$ **entonces** $m \leftarrow c_i$ **devolver** m

Sobre la notación:

- "$\leftarrow$" representa una asignación: $m \leftarrow x$ significa que la variable m toma el valor de x;
- "**devolver**" termina el algoritmo y devuelve el valor a su derecha (en este caso, el máximo de C).

Implementación.

En lenguaje C++:

```
int max(int c[], int n)
{
   int i, m = c[0];
   for (i = 1; i < n; i++)
      if (c[i] > m) m = c[i];
   return m;
}
```

1.1.3. Definición de programas.

Definición de programas.

Un programa informático es un conjunto de instrucciones que una vez ejecutadas realizarán una o varias tareas en una computadora. Sin programas, estas máquinas no pueden funcionar. Al conjunto general de programas, se le denomina software, que más genéricamente se refiere al equipamiento lógico o soporte lógico de una computadora digital.

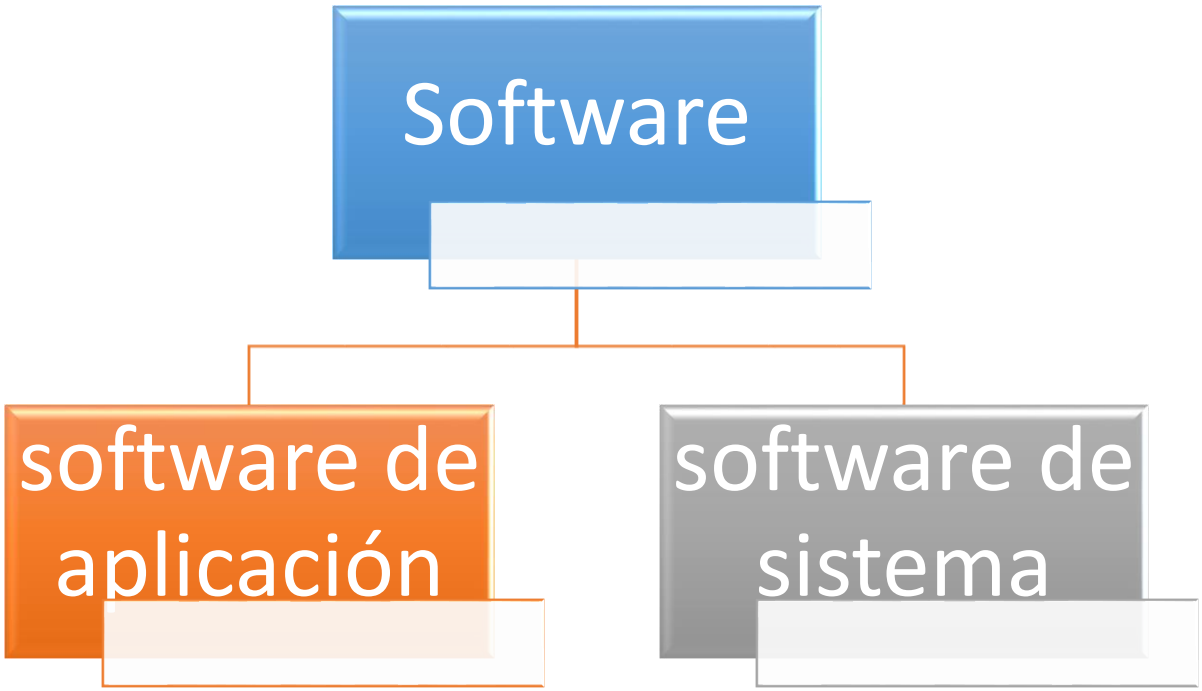
En informática, se los denomina comúnmente binarios, (propio en sistemas Unix, donde debido a la estructura de este último, los ficheros no necesitan hacer uso de extensiones. Posteriormente, los presentaron como ficheros ejecutables, con extensión .exe, en los sistemas operativos de la familia Windows) debido a que una vez que han pasado por el proceso de compilación y han sido creados, las instrucciones que se escribieron en un lenguaje de programación que se usan para escribirlos con mayor facilidad, se han traducido al único idioma que la máquina comprende, combinaciones de ceros y unos llamada código máquina. El mismo término, puede referirse tanto a un programa ejecutable, como a su código fuente, el cual es transformado en un binario cuando es compilado.

Generalmente el código fuente lo escriben profesionales conocidos como programadores. Se escribe en un lenguaje que sigue uno de los siguientes dos paradigmas: imperativo o declarativo y que posteriormente puede ser convertido en una imagen ejecutable por un compilador. Cuando se pide que el programa sea ejecutado, el procesador ejecuta instrucción por instrucción.

1.1.3. Definición de programas.

Clasificaciones del software.

De acuerdo a sus funciones, se clasifican en software de sistema y software de aplicación. En los computadores actuales, al hecho de ejecutar varios programas de forma simultánea y eficiente, se le conoce como multitarea.



Software de aplicación.

En informática, una aplicación es un tipo de programa informático diseñado como herramienta para permitir a un usuario realizar uno o diversos tipos de trabajos. Esto lo diferencia principalmente de otros tipos de programas como los sistemas operativos (que hacen funcionar al ordenador), las utilidades (que realizan tareas de mantenimiento o de uso general), y los lenguajes de programación (con el cual se crean los programas informáticos).

Software de sistema.

En terminología informática el software de sistema, denominado también software de base, consiste en programas informáticos que sirven para controlar e interactuar con el sistema operativo, proporcionando control sobre el hardware y dando soporte a otros programas; en contraposición del llamado software de aplicación. Como ejemplos cabe mencionar a las bibliotecas como por ejemplo OpenGL para la aceleración gráfica, PNG para el sistema gráfico o demonios que controlan la temperatura, la velocidad del disco duro, como hdparm, o la frecuencia del procesador como CPU dyn.

Multitarea.

La multitarea es la característica de los sistemas operativos modernos de permitir que varios procesos sean ejecutados (en apariencia) al mismo tiempo, compartiendo uno o más procesadores.

1.2. FASES DE LA CREACION DE UN PROGRAMA.

Ciclo de vida del software

El término **ciclo de vida del software** describe el desarrollo de software, desde la fase inicial hasta la fase final. El propósito de este programa es definir las distintas fases intermedias que se requieren para **validar** el desarrollo de la aplicación, es decir, para garantizar que el software cumpla los requisitos para la aplicación y **verificación** de los procedimientos de desarrollo: se asegura de que los métodos utilizados son apropiados.

Estos programas se originan en el hecho de que es muy costoso rectificar los errores que se detectan tarde dentro de la fase de implementación. El ciclo de vida permite que los errores se detecten lo antes posible y por lo tanto, permite a los desarrolladores concentrarse en la calidad del software, en los plazos de implementación y en los costos asociados.

El ciclo de vida básico de un software consta de los siguientes procedimientos:

- **Definición de objetivos**: definir el resultado del proyecto y su papel en la estrategia global.
- **Análisis de los requisitos y su viabilidad**: recopilar, examinar y formular los requisitos del cliente y examinar cualquier restricción que se pueda aplicar.
- **Diseño general**: requisitos generales de la arquitectura de la aplicación.
- **Diseño en detalle**: definición precisa de cada subconjunto de la aplicación.
- **Programación** (programación e implementación): es la implementación de un lenguaje de programación para crear las funciones definidas durante la etapa de diseño.
- **Prueba de unidad**: prueba individual de cada subconjunto de la aplicación para garantizar que se implementaron de acuerdo con las especificaciones.
- **Integración**: para garantizar que los diferentes módulos se integren con la aplicación. Éste es el propósito de la *prueba de integración* que está cuidadosamente documentada.
- **Prueba beta** (o *validación*), para garantizar que el software cumple con las especificaciones originales.
- **Documentación**: sirve para documentar información necesaria para los usuarios del software y para desarrollos futuros.
- **Implementación**
- **Mantenimiento**: para todos los procedimientos correctivos (mantenimiento correctivo) y las actualizaciones secundarias del software (mantenimiento continuo).

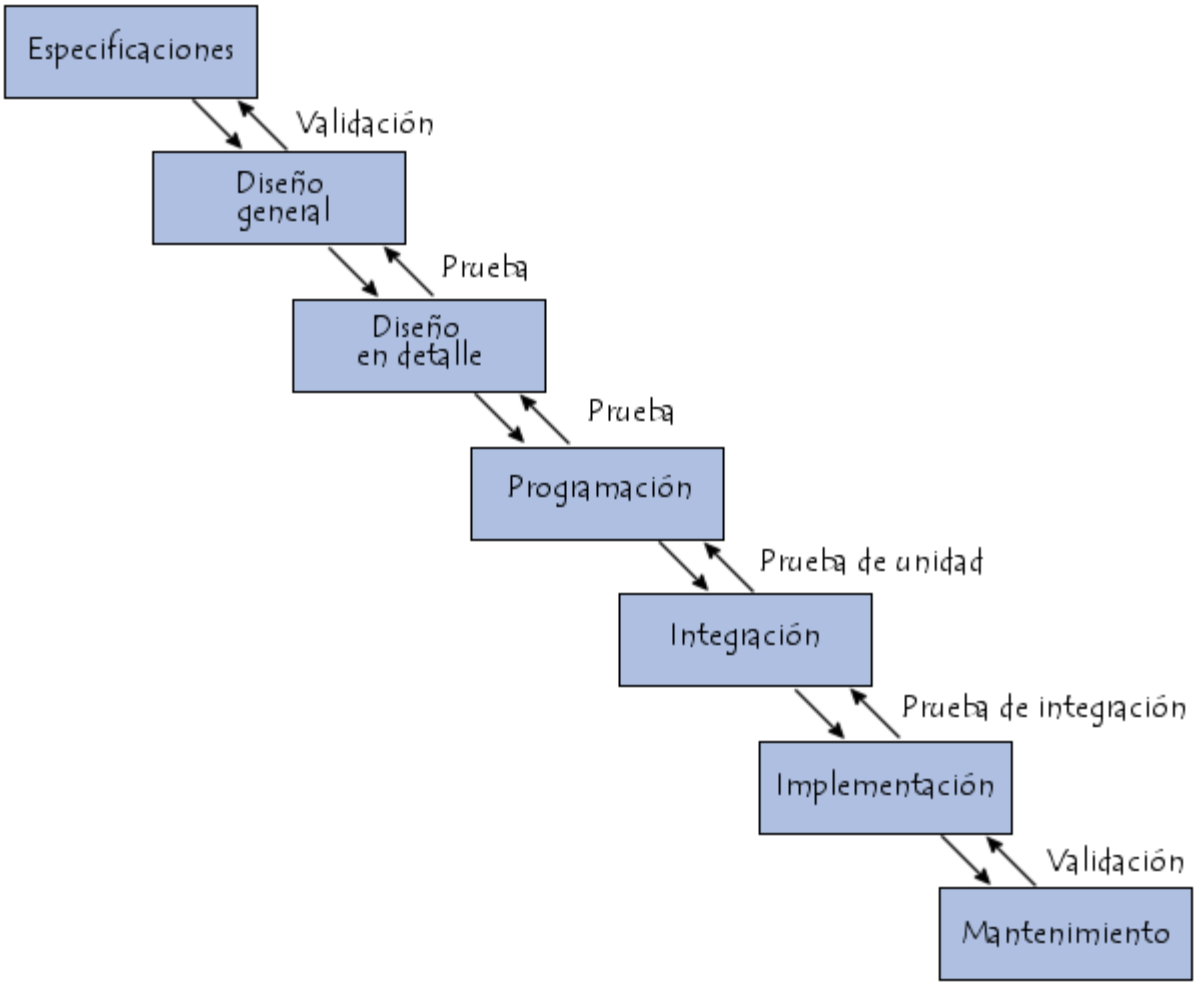
El orden y la presencia de cada uno de estos procedimientos en el ciclo de vida de una aplicación dependen del tipo de modelo de ciclo de vida acordado entre el cliente y el equipo de desarrolladores.

Modelos de ciclo de vida

Para facilitar una metodología común entre el cliente y la compañía de software, los modelos de ciclo de vida se han actualizado para reflejar las etapas de desarrollo involucradas y la documentación requerida, de manera que cada etapa se valide antes de continuar con la siguiente etapa. Al final de cada etapa se arreglan las revisiones de manera que *(texto faltante)*.

Modelo en cascada

El modelo de ciclo de vida en cascada comenzó a diseñarse en 1966 y se terminó alrededor de 1970. Se define como una secuencia de fases en la que al final de cada una de ellas se reúne la documentación para garantizar que cumple las especificaciones y los requisitos antes de pasar a la fase siguiente:



Modelo V

El modelo de ciclo de vida V proviene del principio que establece que los procedimientos utilizados para probar si la aplicación cumple las especificaciones ya deben haberse creado en la fase de diseño.