Programación y Algoritmia

Un enfoque práctico y didáctico para el diseño de algoritmos

1 Conceptos básicos

Lic. Oscar Ricardo Bruno, MDU

Contenido

Conceptos básicos

Objetivos de aprendizaje
Dominando los temas del presente capitulo Usted podrá.

1. Conocer la terminología propia de la disciplina.
2. Definir y comprender claramente conceptos específicos muchas veces mal definidos
3. Comprender el valor de la abstracción.
4. Dar valor a la eficiencia en las soluciones
5. Introducirse en la notación algorítmica y a la forma e encarar los problemas de programación

Introducción:
Se introducen conceptos fundamentales de algoritmia y programación, los que servirán de base para el desarrollo de los temas a trataren materias de algoritmos y estructurad e datos.

Informática
Disciplina del estudio sistematizado de los procesos algorítmicos que describen y transforman información, su teoría, análisis, diseño, eficiencia, implementación y aplicación.
La informática es una disciplina científica, matemática y una ingeniería; tiene tres formas de pensar propias: Teoría, abstracción y diseño.
Las tres se complementan para la resolución de la mayoría de los problemas.

- ✓ Teoría: Con el pensamiento teórico se describen y prueban relaciones.
- ✓ Abstracción: Recolección de datos y formulación de un modelo, se eliminan los detalles irrelevantes.
- ✓ Diseño: se tienen en cuenta requisitos, especificaciones y se diseñan o analizan mecanismos para resolver problemas. Supone llevar a la práctica los resultados teóricos.

Programación
La programación es una actividad transversal asociada a cualquier área de la informática, aunque es la ingeniería del software el área específica que se ocupa de la creación del software.
En principio la programación se veía como un arte, solo era cuestión de dominar un lenguaje de programación y aplicar habilidades personales a la resolución de problemas, casi en forma artesanal. El software era visto como algo desarrollado a través de la intuición sin la utilización de métodos de diseño con técnicas para proceder en forma sistemática y sin ningún control de su desarrollo. Con el reconocimiento de la complejidad del desarrollo del software nació la ingeniería del software.
Se considero que al igual que cualquier otra disciplina la creación de software debía ser reconocida como una actividad de ingeniería que requería la aplicación de sólidos principios científicos.
La ingeniería del software es la disciplina que se ocupa de la aplicación del conocimiento científico al diseño y construcción de programas de computación y a todas las actividades asociadas de documentación, operación y mantenimiento, lo que proporciona un enfoque sistemático.
La programación es una actividad en la que la creatividad juega un rol primordial

Programa:
Conjunto de instrucciones, ejecutables sobre una computadora, que permite cumplir una función especifica. Se asocia al programa con una determinada función o requerimiento a satisfacer por la ejecución del conjunto de instrucciones que lo forman. En general alcanzan su objetivo en tiempo finito, aunque hay excepciones, por ejemplo los programas de control de un sistema de alarma poseen requerimiento de tiempo infinito. Un programa sin errores que se ejecuta puede no ser correcto si no cumple con los requerimientos.

Definición
Programa: conjunto de instrucciones no activas almacenadas en un computador, se vuelve **tarea** a partir de que se selecciona para su ejecución y permite cumplir una función específica. Un **proceso** es un programa en ejecución.

En principio las tareas más importantes a la que se enfrenta quien debe escribir programas en computadoras son:

1. Definir el conjunto de instrucciones cuya ejecución ordenada conduce a la solución.
2. Elegir la representación adecuada de los datos del problema.

La función esencial del especialista informático es explotar el potencial de las computadoras para resolver situaciones del mundo real. Para esto debe analizar los problemas del mundo real, ser capaz de sintetizar sus aspectos principales y poder especificar la función objetivo que se desee. Posteriormente debe expresar la solución en forma de programa, manejando los datos del mundo real mediante una representación valida para una computadora.

Partes de un programa
Los componentes básicos son las instrucciones y los datos. Las instrucciones o sentencias representan las operaciones que se ejecutaran al interpretar el programa. Todos los lenguajes de programación tienen un conjunto mínimo de operaciones que son las de asignación, selección e iteración. Un lenguaje con solo estas tres instrucciones permite escribir cualquier algoritmo.
Los datos son valores de información de los que se necesita disponer, en ocasiones transformar para ejecutar la función del programa.
Los datos están representados simbólicamente por un nombre que se asocia con una dirección única de memoria.
El contenido de la dirección de memoria correspondiente a un dato constante se asigna solo una vez y solo puede ser modificado en una nueva compilación. En cambio el contenido o valor de la dirección de memoria correspondiente a un dato variable puede ser asignado o modificado en tiempo de ejecución.
Un programa se corresponde con una transformación de datos. A partir de un contexto determinado por las precondiciones.
El programa transforma la información debiendo llegar al resultado esperado produciendo el nuevo contexto caracterizado por las poscondiciones.

Dato
Representación de un objeto el mundo real mediante el cual se pueden modelizar aspectos de un problema que se desea resolver con un programa en una computadora.

Definición
Dato representación de un objeto el mundo real mediante el cual se pueden modelizar aspectos de un problema que se desea resolver con un programa en una computadora.
<dato> -> <objeto><atributo><valor>

Abstracción
Proceso de análisis del mundo real con el propósito de interpretar los aspectos esenciales de un problema y expresarlo en términos precisos.

Modelizacion
Abstraer un problema del mundo real y simplificar su expresión, tratando de encontrar los aspectos principales que se pueden resolver, requerimientos, los datos que se han de procesar y el contexto del problema.

Precondición
Información conocida como verdadera antes de iniciar el programa.

Poscondición
Información que debiera ser verdadera al cumplir un programa, si se cumple adecuadamente el requerimiento pedido.

Especificación
Proceso de analizar problemas del mundo real y determinar en forma clara y concreta el objetivo que se desea. Especificar un problema significa establecer en forma univoca el contexto, las precondiciones el resultado esperado, del cual se derivan las poscondiciones.

Lenguaje de programación
Conjunto de instrucciones permitidas y definidas por sus reglas sintácticas y su valor semántico para la expresión de soluciones de problemas.

Del problema real a su solución por computadoras
Analizando un problema del mundo real se llega a la *modelización* del problema por medio de la *abstracción.*
A partir del modelo se debe elaborar el análisis de la solución como sistema, esto significa la descomposición en módulos. Estos módulos deben tener una función bien definida.
La modularización es muy importante y no solo se refiere a los procesos a cumplir, sino también a la distribución de los datos de entrada, salida y los datos intermedios necesarios para alcanzar la solución.

Estudio de los datos del problema.
Cada módulo debe tener un proceso de refinamiento para expresar su solución en forma ordenada, lo que llevara a la construcción del algoritmo correspondiente.
A partir de los algoritmos se pueden escribir y probar programas en un lenguaje determinado y con un conjunto de datos significativos.

Etapas de resolución de problemas con computadoras.

1. Análisis del problema: en su contexto del mundo real.

2. Diseño de la solución: Lo primero es la modularización del problema, es decir la descomposición en partes con funciones bien definidas y datos propios estableciendo la comunicación entre los módulos.
3. Especificación del algoritmo: La elección adecuada del algoritmo para la función de cada modulo es vital para la eficiencia posterior.
4. Escritura del programa: Un algoritmo es una especificación simbólica que debe convertirse en un programa real sobre un lenguaje de programación concreto.
5. Verificación: una vez escrito el programa en un lenguaje real y depurado los errores sintácticos se debe verificar que su ejecución conduzca al resultado deseado con datos representativos del problema real.

Programación modular – programación estructurada

Se dice modular porque permite la descomposición del problema en módulos y estructurada solo permite la utilización de tres estructuras: Asignación, selección, repetición.

Algoritmo

El termino algoritmo es en honor del matemático árabe del siglo IX, *Abu Jafar Mohamed ibn Musa Al Khowârizmî*. Refiere conjunto de reglas, ordenadas de forma lógica, finito y preciso para la solución de un problema, con utilización o no de un computador.

En la actualidad al término se lo vincula fuertemente con la programación, como paso previo a la realización de un programa de computación aunque en realidad es una metodología de resolución presente en muchas de las actividades que se desarrolla a lo largo de la vida.

Desde los primeros años de escuela se trabaja con algoritmos, en especial en el campo de las matemáticas. Los métodos utilizados para sumar, restar, multiplicar y dividir son algoritmos que cumplen perfectamente las características de precisión, finitud, definición y eficiencia.

Para que el algoritmo pueda ser fácilmente traducido a un lenguaje de programación y luego ser ejecutado la especificación debe ser clara, precisa, que pueda ser interpretada con precisión y corresponda a pocas acciones, si esto no ocurre será necesario acudir a desarrollar un mayor nivel de refinamiento.

La utilización de refinamientos sucesivos es lo que permite alcanzar la solución modular que se propone.

Diseño modular, entonces, es la aplicación del criterio de refinamientos sucesivos, partiendo de un plan de acción, determinando que hacer, por aplicación de los conocimientos estratégicos de resolución pasando luego al como hacerlo con los conocimientos tácticos para la realización del algoritmo.

La programación de algoritmos representa un caso de resolución de problemas que requiere representación mental del mundo real, adaptación para tener una solución computable y criterio para elegir una alternativa eficiente de implementación.

Cuando se analiza un problema, particularmente de programación, y éste es difícil de describir, el plan de acción recomendable para alcanzar la solución es comenzar trazando un esbozo de las formas más gruesas, para que sirvan de andamio a las demás; aunque algunas de ellas se deban cambiar posteriormente. Después, se agregan los detalles, (obteniéndose el algoritmo refinado), para dotar a estos esqueletos de una estructura más realista.

Durante la tarea de integración final, se descartan aquellas primeras ideas provisionales que ya no encajan en la solución. Por lo que, hasta que no se haya visto el conjunto global es imposible encontrarle sentido a ninguna de las partes por sí solas.
Siempre es mejor explicar un misterio en términos de lo que se conoce, pero cuando esto resulta difícil de hacer, se debe elegir entre seguir tratando de aplicar las antiguas teorías, o de descartarlas y probar con otras nuevas. Siguiendo este análisis, se define como reduccionistas a aquellas personas que prefieren trabajar sobre la base de ideas existentes, y como renovadores a los que les gusta impulsar nuevas hipótesis. En programación debe encontrarse un equilibrio entre ambas posturas.
La programación como toda actividad que requiere creatividad necesita que se produzca un salto mental que se puede sintetizar como señala David Perkins en:

1. Larga búsqueda, se requiere esfuerzo en analizar y buscar.
2. Escaso avance aparente: el salto mental sobreviene tras un avance que parece escaso o no muy evidente, pero sobreviene.
3. Acontecimiento desencadenante: El típico proceso de hacer clic comienza con un acontecimiento que lo desencadena.
4. Chasquido cognitivo: De pronto aparece la solución la que sobreviene con rapidez que hace que las piezas encajen con precisión, aun cuando sea necesario todavía ajustar algunos detalles. Pero la idea generadora apareció.
5. Transformación. Este avance nos va modificando nuestro mundo mental.

En síntesis, la practica del salto de pensamiento requiere en primer lugar de buscar analogías, en segundo lugar juegan un papel importante las conexiones lógicas, formulación de una pregunta crucial ocupa un papel decisivo. El repertorio de acciones tras el salto del pensamiento se expande para incluir no solo la analogía sino una extrapolación lógica y la formulación de la pregunta adecuada
Para encontrar una solución muchas veces se necesita desplazarse bastante por el entorno adecuado. Conforme a esto Thomas Edison declaro que la invención significa 99% de transpiración y 1% de inspiración, en contraposición con Platón que sostenía que las soluciones aparecen por inspiración divina.
Muchos problemas son razonables, cabe razonarlos paso a paso para alcanzar la solución. Otros son irrazonables no se prestan a un a reflexión por etapas.

Definición
Algoritmo
Especificación rigurosa (debe expresarse en forma univoca) de la secuencia de pasos, instrucciones, a realizar sobre un autómata para alcanzar un resultado deseado en un tiempo finito. Esto último supone que el algoritmo empieza y termina, en el caso de los que no son de tiempo finito (ej. Sistemas en tiempo real) deben ser de número finito de instrucciones.

En definitiva un algoritmo es una especificación ordenada de la solución a un problema de la vida real. Son el fundamento de la programación de computadores en el paradigma de programación imperativo.
Bajo este paradigma desarrollar un programa significa indicarle al computador, con precisión, sin ambigüedad y en un lenguaje que este pueda entender, todos y cada uno de los pasos que debe ejecutar para lograr el objetivo propuesto.
Previo a la traducción en un lenguaje de programación es necesario poder entender el problema, conocer las pre condiciones, establecer cual debe ser la pos condición, o aquello que debe ser cierto al finalizar la ejecución del algoritmo, en definitiva entender claramente *Que* es lo que se debe hacer para luego avanzar en *Como* hacerlo. Aquí

debe utilizarse todas las herramientas al alcance de la mano para el desarrollo del algoritmo como paso previo a la solución del problema por el computador.
Existen varias técnicas para representar formalmente un algoritmo, una descriptiva llamada pseudo código, y otras graficas como los diagrama de flujo, diagrama Nassi Sneiderman, Diagramas de Lindsay, diagramas de Jackson, entre otros, en este caso se presentara una notación algorítmica similar a la presentada por Piere Scholl en el texto Esquemas algorítmicos fundamentales: Secuencia e iteración.

Definición
Algoritmo:
Secuencia finita de instrucciones, reglas o pasos que describen en forma precisa las operaciones que una computadora debe realizar para llevar a cabo una tarea en tiempo finito [Knuth, 1968].
Descripción de un esquema de comportamiento expresado mediante un repertorio finito de acciones y de informaciones elementales, identificadas, bien comprendidas y realizables a priori. Este repertorio se denomina léxico[Scholl, 1988].

Esta formado por reglas, pasos e instrucciones.
Las reglas especifican operaciones.
La computadora es el agente ejecutor.
La secuencia de reglas y la duración de la ejecución son finitas.

Características de un algoritmo
Un algoritmo debe tener al menos las siguientes características:

1. **Ser preciso**: esto significa que las operaciones o pasos del algoritmo deben desarrollarse en un orden estricto, ya que el desarrollo de cada paso debe obedecer a un orden lógico.
2. **Ser definido**. Ya que en el área de programación, el algoritmo es el paso previo 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 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 deberá ser siempre el mismo.
3. **Ser 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.
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 pseudo código, diagrama de flujo y diagramas de Nassi/Schneiderman, entre otras. En esta publicación se propondrá una notación algorítmica y se darán las equivalencias entre la propuesta y las existentes y también con las sentencias de los lenguajes de programación, en particular Pascal y C.
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. **Eficiencia**: 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.

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. 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 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.

Propiedades de los algoritmos

1. Especificación precisa de la entrada: El algoritmo debe dejar claro el número y tipo de datos de entrada y las condiciones iniciales que deben cumplir esos valores de entrada para conseguir que las operaciones tengan éxito.
2. Especificación precisa de cada instrucción: cada etapa del algoritmo debe estar definida con precisión, no debe haber ambigüedades sobre las acciones que se deben ejecutar en cada momento.
3. Un algoritmo debe ser exacto y correcto: Un algoritmo se espera que resuelva un problema y se debe poder demostrar que eso ocurre. Si las condiciones de entrada se cumplen y se ejecutan todos los pasos el algoritmo entonces debe producir la salida deseada.
4. Un algoritmo debe tener etapas bien definidas y concretas, un número finito de pasos, debe terminar y debe estar claro la tarea que el algoritmo debe ejecutar.
5. Debe ser fácil de entender, codificar y depurar.
6. Debe hacer uso eficiente de los recursos de la computadora

Finitud: en longitud y duración.
Precisión: Determinar sin ambigüedad las operaciones que se deben ejecutar.
Efectividad: las reglas pueden ejecutarse sin el ordenador obteniéndose el mismo resultado.
Generalidad: Resolver una clase de problema y no un problema particular.
Entradas y salidas: puede tener varias entradas pero una sola salida, el resultado que se debe obtener.

Eficiencia de un algoritmo

Se pueden tener varias soluciones algorítmicas para un mismo problema, sin embargo el uso de recursos y la complejidad para cada una de las soluciones puede ser muy diferente.
La eficiencia puede definirse como una métrica de calidad de los algoritmos asociada con la utilización optima de los recursos del sistema de cómputo donde se ejecutara el algoritmo, su claridad y el menor grado de complejidad que se pueda alcanzar. *Hacer todo tan simple como se pueda, no más (Albert Einstein).*
La eficiencia como factor espacio temporal debe estar estrechamente relacionada con la buena calidad, el funcionamiento y la facilidad del mantenimiento.

Medidas de eficiencia para N = 10.0000

Eficiencia		Iteraciones	Tiempo estimado
Logarítmica	$Log_2\ N$	14	Microsegundos
Lineal	N	10.000	0.1 segundo
Logarítmica lineal	$N * Log_2\ N$	140.000	2 segundos
Cuadrática	N^2	10.000^2	15-20 minutos
Poli nómica	N^k	10.000^K	Horas
Exponencial	2^N	$2^{10.000}$	Inmedible

Complejidades más comunes

1. Complejidad constante: se expresa como O(1). Se encuentra en algoritmos sin ciclos, por ejemplo en un intercambio de variables.
2. Complejidad logarítmica: Es una complejidad eficiente, la búsqueda binaria tiene esta complejidad.
3. Complejidad lineal: se encuentra en los ciclos simples.
4. Complejidad logarítmica lineal: Los mejores algoritmos de ordenamiento tienen esta complejidad.
5. Complejidad cuadrática: Aparece en el manejo de matrices de dos dimensiones, generalmente con dos ciclos anidados.
6. Complejidad cúbica: Aparece en el manejo de matrices de tres dimensiones, generalmente con tres ciclos anidados.
7. Complejidad exponencial: es la complejidad de algoritmos recursivos.

Léxico y algoritmo

Para escribir un algoritmo deben seguirse un conjunto de pasos básicos

1. Comprender el problema
2. Identificar los elementos a incluir en el léxico: constantes, tipos, variables y acciones.
3. Encontrar la forma de secuenciar las acciones para obtener el resultado, esto es, alcanzar las poscondiciones a partir de un estado inicial que cumple con la precondición. Para establecer el orden de las acciones los lenguajes de programación proporcionan mecanismos de composición: Secuenciación, análisis de casos, iteración y recursion.
4. Al organizar las acciones en el tercer paso puede ocurrir que se detecte que faltan elementos en el léxico o que algún aspecto del problema no ha sido bien comprendido lo cual requeriría volver a los pasos anteriores.
5. Nunca la solución aparece en el primer intento, en general aparece en un proceso cíclico, entonces se debe:
6. Escribir el léxico,
    a. escribir la primera versión,
    b. incluir en el léxico nuevos elementos que faltaban,
    c. escribir la nueva versión del algoritmo y así sucesivamente

Estructura de un algoritmo

```
LEXICO {Léxico Global del algoritmo}
        {Declaración de tipos, constantes, variables y acciones}
        Acción 1
                PRE {Precondición de la acción 1}
                POS {Poscondición de la acción 1}
                LEXICO {Léxico local, propio de la acción 1}
```

Declaraciones locales
ALGORITMO {Implementación de la acción 1}
{Secuencia de instrucciones de la acción 1}
FIN {Fin implementación algoritmo de la acción 1}

ALGORITMO
PRE {Precondición del algoritmo principal}
POS {Poscondición del algoritmo principal}
{Secuencia de instrucciones del algoritmo principal}
FIN {Fin del algoritmo principal}

Proceso Computacional
Se refiere a un algoritmo en ejecución. La ejecución de las instrucciones origina una serie de acciones sobre elementos de memoria que representan información manejada por el algoritmo. A nivel algorítmico se asigna un nombre a cada información de modo de manejar un par nombre-valor para cada información.
Una variable representa alguna entidad del mundo real, relevante para el problema que se quiere resolver. El efecto que producen las acciones del proceso sobre las variables produce cambio de estados o de sus valores.

Definiciones

Programa: Algoritmo escrito en un lenguaje cuyas instrucciones son ejecutables por una computadora y que están almacenados en un disco.
Tarea: Un programa se vuelve tarea a partir del momento que se lo selecciona para su ejecución y hasta que esta termina.
Proceso: programa en ejecución, se ha iniciado pero aún no ha finalizado.
Lenguajes de programación: notación que permite escribir programas a mayor nivel de abstracción que los lenguajes de máquina. Sus instrucciones deben ser traducidas a lenguaje de máquina.
Lenguaje de máquina: Instrucciones que son ejecutables por el hardware de una computadora.
Paradigmas de programación
Paradigma: Colección de conceptos que guían el proceso de construcción de un programa. Estos conceptos controlan la forma en que se piensan y formulan los programas.
Imperativo – Procedural – Objetos.
Declarativo – Funcional – Lógico.
Dato Información Conocimiento
Dato: <objeto><atributo><valor> sin interpretar.
Información: añade significado al dato.
Conocimiento: Añade propósito y capacidad a la información. Potencial para generar acciones.
Problema
Enunciado con una incógnita, la solución es encontrar el valor de esa incógnita.
Problema computacional o algorítmico: tarea ejecutada por una computadora con una especificación precisa de los datos de entrada y de los resultados requeridos en función de estos.
Clase de problemas
No computables: No existe un algoritmo.

Computables
Tratables: Existe un algoritmo eficiente.
Intratable: No existe algoritmo eficiente.

Expresiones Sentencias Léxico

Expresiones: secuencia de operadores y operandos que se reduce a un solo valor.
Sentencias: acción produce un efecto, puede ser primitiva o no primitiva.
Léxico: Descripción del conjunto de acciones e informaciones a partir de la cual se expresa el esquema de comportamiento del algoritmo.

Pasos para resolver un algoritmo

Comprender el problema.
Identificar información y acciones a incluir en el léxico (constantes, tipos, variables y acciones).
Encontrar un camino de secuenciar las acciones para obtener el resultado, es decir para alcanzar la poscondición a partir del estado inicial que cumple con la precondición.

Acciones primitivas y derivadas

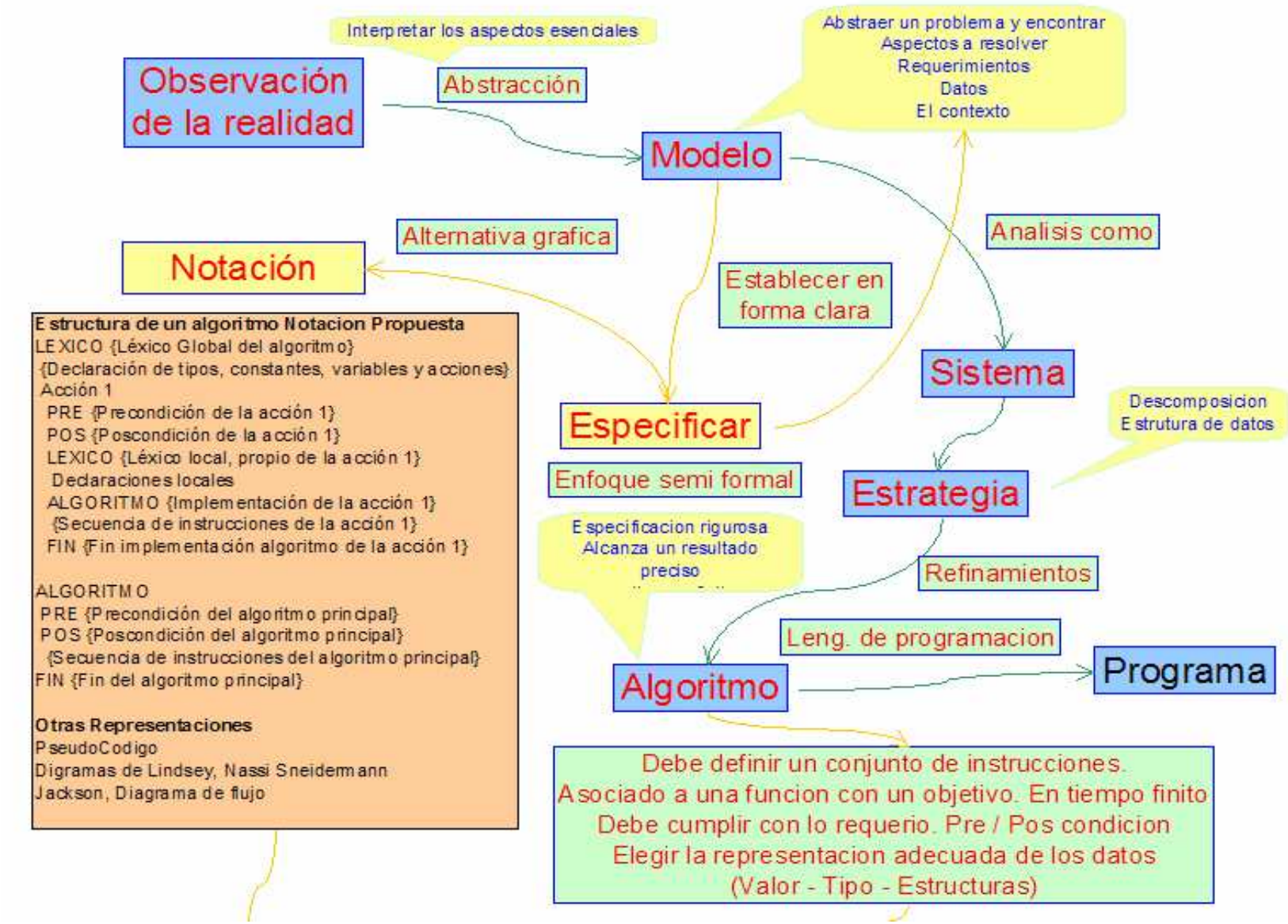
Acciones primitivas: Incorporadas por el lenguaje.
Acciones derivadas: realizadas mediante la combinación de acciones primitivas con el objeto de desarrollar una tarea en particular. Son complementarias y pueden ser desarrolladas por el programador.

Estructura de un algoritmo

```
LEXICO {Léxico Global del algoritmo}
        {Declaración de tipos, constantes, variables y acciones}
        Acción 1
                PRE {Precondición de la acción 1}
                POS {Poscondición de la acción 1}
                LEXICO {Léxico local, propio de la acción 1}
                        Declaraciones locales
                ALGORITMO {Implementación de la acción 1}
                        {Secuencia de instrucciones de la acción 1}
                FIN {Fin implementación algoritmo de la acción 1}

ALGORITMO
        PRE {Precondición del algoritmo principal}
        POS {Poscondición del algoritmo principal}
                {Secuencia de instrucciones del algoritmo principal}
FIN {Fin del algoritmo principal}
```

Resumen:
En el presente capitulo se introdujeron términos y frases de la disciplina en estudio.
Se abordo el problema desde un punto de vista conceptual definiendo con precisión términos y frases para evitar ambigüedades. Se puso especial énfasis en la necesidad de pensar las soluciones antes de escribir código. Se establecieron cuales son los pasos que deben seguirse para una solución correcta de problemas y cuales son los problemas que pueden tratarse en forma algorítmica. Se definió cuales son las características deseables de los algoritmos y se introdujo el concepto de eficiencia de modo de hacer, tal como recomendaba Albert Enstein, las cosas tan simple como se pueda.
Se introdujo, además una representación sem. formal para la descripción de los algoritmos


Interpretar los aspectos esenciales
Observación de la realidad
Abstracción
Modelo
Abstraer un problema y encontrar
Aspectos a resolver
Requerimientos
Datos
El contexto
Alternativa grafica
Notación
Analisis como
Establecer en forma clara
Sistema
Especificar
Descomposicion
Estrutura de datos
Enfoque semi formal
Estrategia
Especificacion rigurosa
Alcanza un resultado preciso
Refinamientos
Leng. de programacion
Algoritmo
Programa
Debe definir un conjunto de instrucciones.
Asociado a una funcion con un objetivo. En tiempo finito
Debe cumplir con lo requerio. Pre / Pos condicion
Elegir la representacion adecuada de los datos
(Valor - Tipo - Estructuras)
Estructura de un algoritmo Notacion Propuesta
LEXICO {Léxico Global del algoritmo}
{Declaración de tipos, constantes, variables y acciones}
Acción 1
PRE {Precondición de la acción 1}
POS {Poscondición de la acción 1}
LEXICO {Léxico local, propio de la acción 1}
Declaraciones locales
ALGORITMO {Implementación de la acción 1}
{Secuencia de instrucciones de la acción 1}
FIN {Fin implementación algoritmo de la acción 1}
ALGORITMO
PRE {Precondición del algoritmo principal}
POS {Poscondición del algoritmo principal}
{Secuencia de instrucciones del algoritmo principal}
FIN {Fin del algoritmo principal}
Otras Representaciones
PseudoCodigo
Digramas de Lindsey, Nassi Sneidermann
Jackson, Diagrama de flujo

PROPIEDADES
Especificacion precisa de entrada
Esp. Precisa de cada instrucción
Etapas bien definidas y concretas
DEBE SER
Preciso dearrollarse en orden
Definido =datos -> =resultados
Finito en tiempo y/o instruciones
Exacto en sus resultados
Correcto alcanzar el objetivo
Formal: Facil de entender y codificar
Eficiente uso de recursos y tiempo

Imperativa
Declarativa

Programacion

Modular

Estructurada

Dato: <objeto><atributo><valor> sin interpretar.
Información: añade significado al dato.
Conocimiento: Añade propósito y capacidad a la Información. Potencial para generar acciones.
Tipos de datos
Estaticos: Simples Estructuras
Dinamicos Punteros

Acciones

Estructura de datos

SECUENCIA
INICIO S1 S2 ... Sn FIN
Ejecuta secuencialmente el bloque de sentencias comprendidas entre S1 y Sn

CONDICIONAL
SI Condicion ENTONCES S SI_NO R FIN_SI
Si la expresión toma el valor verdadero entonces se ejecuta la secuencia S si es falso se ejecuta la secuencia R.

SEGÚN expr V1 : S1 V2 : S2 EN_OTRO_CASO : Sn FIN_SEGUN
Según el valor Vi de la expresión ordinal expr, se ejecuta la secuencia Si(i = 1 ..n)

ITERACION
Mientras Cond. Hacer S FIN_MIENTRAS
Ejecutar la secuencia S mientras la condicion Cond tenga el valor de verdadero

REPETIR S HASTA Cond
Ejecutar la secuencia S hasta que la expresión Cond tome el valor de verdadero

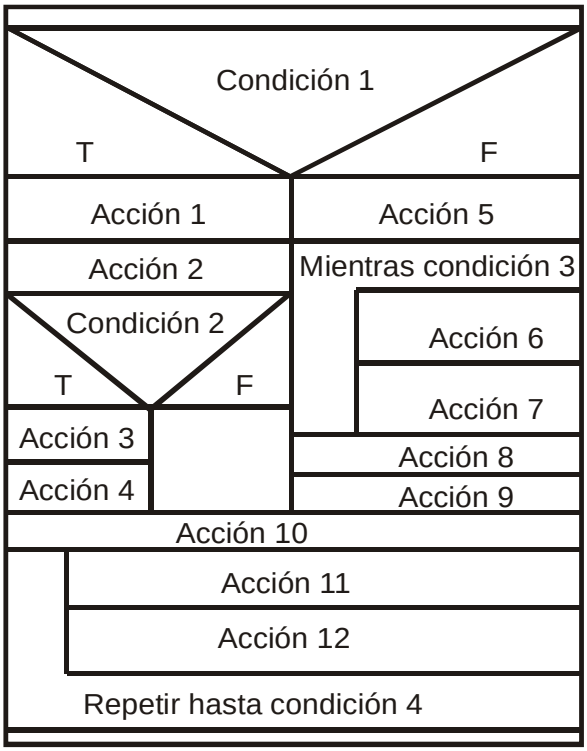
PARA i [Vi..Vf] HACER S FIN_PARA
Ejecutar la secuencia S (Vf – Vi + 1) veces.

Acciones
Nombre (dato ident. Tipo; dato-Resultado ident. Tipo; resultado ident. Tipo): una accion
Se invoca con una incvocacion a si misma

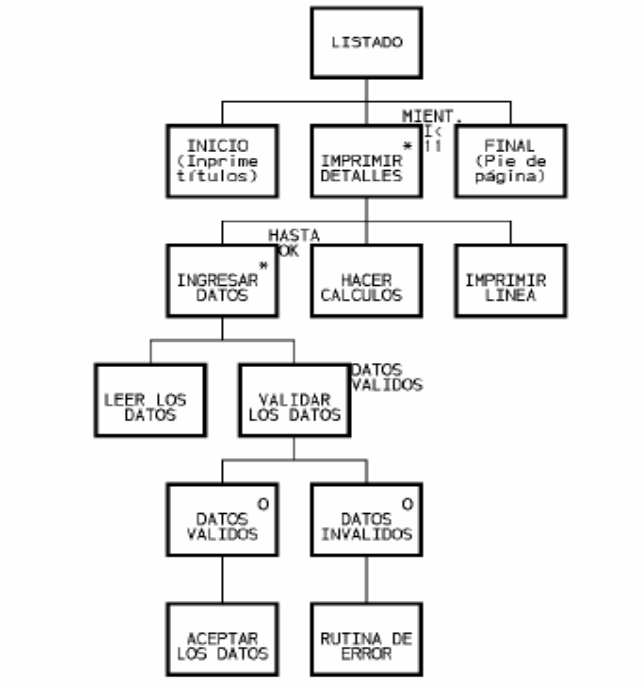
Funciones
Nombre(p1: td1; ...; pn:tdn)->TipoResultado : una funcion
Se invoca en una expresion

Representaciones gráficas de algoritmos

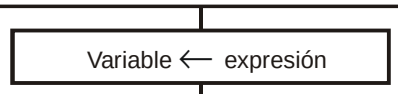
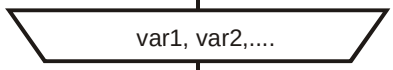
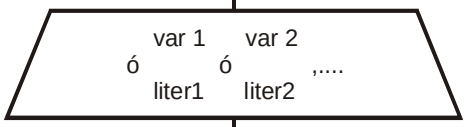
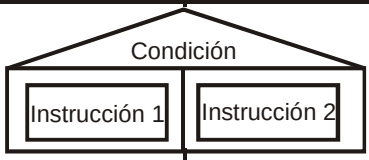
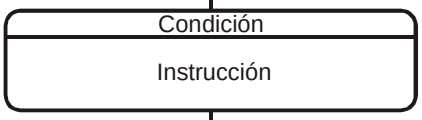
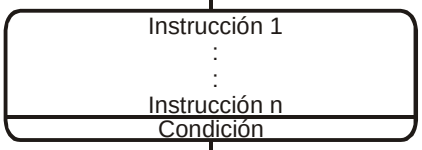
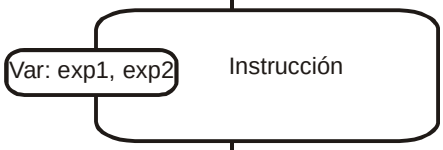
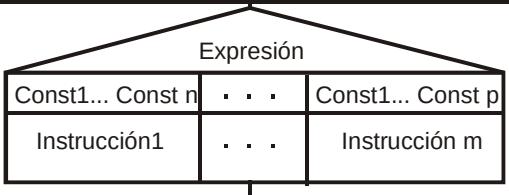
Diagrama de Nassi-Sneiderman



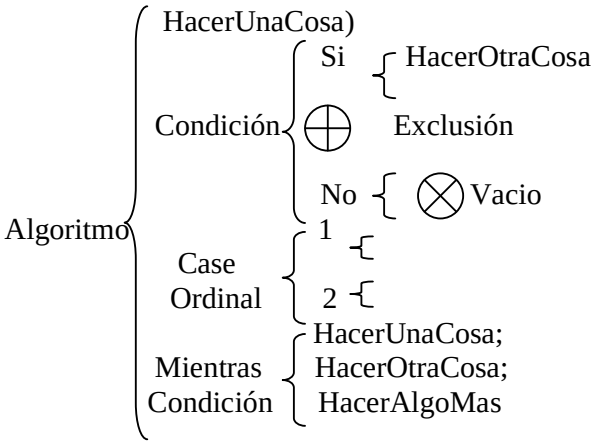
Diagramas de Jackson



Diagramas de Lindsay.

PASCAL	DIAGRAMA DE DETALLE
Variable = expresión	
READ (var1, var2,....var n)	
WRITELN var 1 var 2 (ó ó , ...) WRITE liter1 liter2	
IF condición THEN sentencia 1 ELSE sentencia 2	
WHILE condición DO sentencia	
REPEAT sentencia; : : sentencia n UNTIL condición	
FOR variable: expres1 TO expres2 ó DOWNTO DO sentencia	
CASE expresión OF const1..... const n : instrucción1 : : const1..... const p : instrucción m END	
Cualquier sentencia puede ser reemplazada por un bloque: BEGIN sentencias 1; : : sentencias n END	Correspondiendo en el diagrama de detalle la secuencia: instrucción 1; : : instrucción n

Llaves de Warniel



Notación Algoritmica

LEXICO {Léxico Global del algoritmo}
    {Declaración de tipos, constantes, variables y acciones}
    Acción 1
        PRE {Precondición de la acción 1}
        POS {Poscondición de la acción 1}
        LEXICO {Léxico local, propio de la acción 1}
            Declaraciones locales
        ALGORITMO {Implementación de la acción 1}
            {Secuencia de instrucciones de la acción 1}
        FIN {Fin implementación algoritmo de la acción 1}

ALGORITMO
    PRE {Precondición del algoritmo principal}
    POS {Poscondición del algoritmo principal}
        {Secuencia de instrucciones del algoritmo principal}
FIN {Fin del algoritmo principal}