Universidad Autónoma de Entre Ríos
Facultad de Ciencia y Tecnología
Sede: Oro Verde

FUNDAMENTOS DE PROGRAMACIÓN

RESUMEN DE CONTENIDOS N° 1

Introducción

RESUMEN DE CONTENIDOS N° 1

Introducción

RESOLUCIÓN DE PROBLEMAS COMPUTACIONALES

En la vida diaria nos enfrentamos continuamente a problemas que debemos resolver en lo posible felizmente. Así como cada individuo tiene formas de encarar un problema y su propia manera de solucionarlo, computacionalmente hablando podemos hacer un paralelo. Ante la presentación de un problema encarar la mejor forma de resolverlo para arribar al resultado esperado y correcto es un desafío. Para ello debemos comprender exactamente qué se pide, qué resultados se pretenden y que restricciones y/o condiciones existen. Para realizar lo antes dicho dividiremos la resolución de un problema en etapas, las cuales enunciamos y definimos a continuación.

ETAPAS PARA LA RESOLUCIÓN DE PROBLEMAS

a. *Definición del problema*

Está dada por la formulación del problema en forma correcta y completa. Esta enunciación de lo que se desea es primordial para el éxito de la resolución.

b. *Análisis del problema*

A partir del estudio del problema se deberá identificar y conocer las partes principales del mismo y de esa manera determinar los siguientes conjuntos:

- de ***DATOS*** : es la información con que contamos para resolver el problema.
- de ***RESULTADOS*** : es lo que se desea obtener.
- de ***CONDICIONES***: una o más relaciones que vinculan los dos conjuntos anteriores y que permitirán plantear la solución del problema.

c. *Programación*

Esta etapa consiste en obtener la solución del problema dado. Se divide en dos subetapas:

c.1. Elección y creación del método

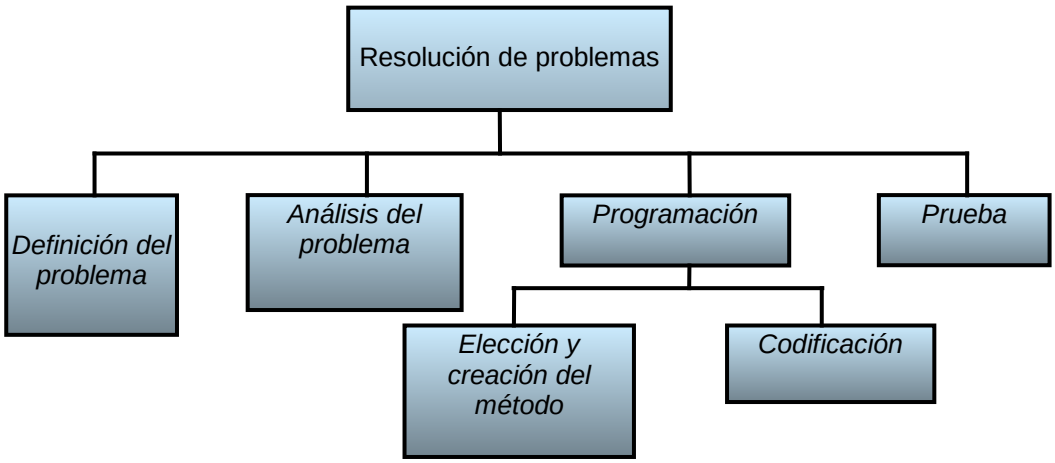
Se trata de buscar un procedimiento o método general que permita resolver el problema planteado utilizando una computadora. Es muy factible que se encuentren varios métodos para hacerlo, lo importante es determinar la “mejor alternativa”, de acuerdo a distintos parámetros que se establezcan para esta selección. Esta puede ser la que produzca los resultados esperados en el menor tiempo y al menor costo o sólo en el menor tiempo u otras.

c.2. Codificación

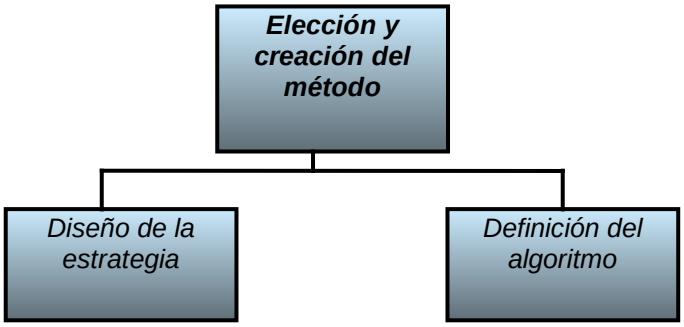
Consiste en expresar el método elegido en un lenguaje, llamado lenguaje de programación, que pueda ser interpretado por la computadora. Esta subetapa será objeto de estudio en años superiores.

d. *Prueba*

Esta etapa consiste en la ejecución del código del método elegido, es decir, suministrar los datos al computador, y obtener los resultados. Luego se analizarán los mismos determinando si son realmente los esperados. Caso contrario, deberán analizarse las etapas previas, comenzando por la última hacia atrás, y realizar las modificaciones necesarias, repitiendo este proceso hasta obtener los resultados esperados.
Observemos gráficamente las etapas descriptas



La etapa de elección y creación del método se puede dividir a su vez en el diseño de la estrategia y la definición del algoritmo, y puede graficarse de la siguiente manera:



Ahora definamos el concepto de estrategia y de algoritmo.

ESTRATEGIA

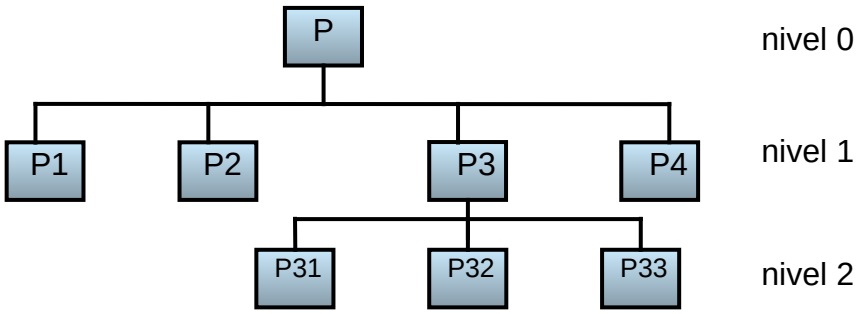
El diseño de la estrategia consiste en encontrar un método que nos permita llegar a resolver el problema planteado. Como primer paso de esta etapa, debemos preparar un plan o esquema general de las tareas que deben realizarse para llegar a la solución. Este esquema se denomina estrategia y debe ser una lista de **QUÉ** hacer.

¿Cómo se diseña una estrategia?

Por lo dicho, diseñar una estrategia consiste en dividir o descomponer el problema original en una sucesión de problemas más simples, de tamaño suficientemente pequeño como para que cada uno de ellos pueda ser comprendido en su totalidad. Ésto, permitirá atacar la solución de cada problema simple por separado e independientemente de los demás, volviendo a aplicar este enfoque a cada uno de los subproblemas hasta llegar a subproblemas de solución simple. Una vez que todos ellos han sido resueltos, se puede decir que el problema original ha sido resuelto.

Este proceso de descomposición de un problema partiendo de la formulación completa del problema hasta llegar a problemas elementales de simple solución, se llama **diseño descendente**, también conocido como **top-down**, método de **refinamiento sucesivo** o **diseño compuesto**.

Gráficamente, dado el problema P lo dividiremos en subproblemas Pi. Cada subdivisión implica un descenso de nivel.



Cada Pi representa un enunciado o subproblema. Para cada uno existen 2 posibilidades:

- que Pi sea un subproblema o una tarea simple, dando por finalizada la descomposición
- que Pi sea un subproblema o una tarea compuesta y por lo tanto sea posible su descomposición en una nueva secuencia de subproblemas

Las características generales de este tipo de diseño se basan en:

- ir de lo general a lo particular
- no existe una única descomposición de subproblemas
- en cada nivel puede verificarse que el esquema sea el correcto

Finalmente se realiza un trabajo de recomposición del esquema completo, resolviendo cada subproblema hasta lograr la solución del problema.
El diseño de una estrategia y su posterior refinamiento, constituyen las etapas más creativas y quizás más dificultosas de todo el proceso de resolución de un problema.

ALGORITMO

Planteada una estrategia indicando **QUÉ** tareas hacer, debemos especificar una lista detallada de **CÓMO** hacerlas, llegando así a definir una solución paso a paso del problema llamada algoritmo. La descripción de la solución detallada por medio de un algoritmo constituye el segundo paso en la etapa de elección del método.
La palabra algoritmo se utiliza, en general, como sinónimo de procedimiento, método o técnica. Pero en el área de computación tiene un significado más específico.

Un algoritmo es un conjunto finito de operaciones (instrucciones - pasos) que seguidos en un determinado orden permiten resolver un tipo de problema.

Las características principales de un algoritmo son:

- **Finito**: permite arribar a la solución de un problema después de la ejecución de un número finito de pasos.

- **Definido**: cada paso debe ser enunciado en forma clara y precisa, y no debe dar lugar a ambigüedades. Para los mismos datos el algoritmo debe dar siempre los mismos resultados

- **General:** la solución debe ser aplicable a un tipo de problemas y no a un problema particular.

Teniendo en cuenta las características mencionadas previamente podemos decir que: un algoritmo es una secuencia ordenada y finita de pasos que constituyen un método general para resolver un tipo de problemas.

Es de notar que esta definición, se refiere a '...resolver un tipo de problemas .....' y no hace hincapié en el uso del computador como herramienta para su resolución. Esto se debe a que el concepto de algoritmo se aplica a problemas computacionales que van a ser resueltos por medio de un computador y a problemas no computacionales, en cuya resolución no interviene esta herramienta. En ambos casos el lenguaje usado en la descripción del algoritmo debe ser comprensible para el destinatario o para quien lo va a ejecutar. Por lo visto, para cualquier problema para el que pueda especificarse un método finito de solución puede definirse un algoritmo.

Ejemplos que se pueden presentar en la vida diaria:

- una receta de cocina
- las instrucciones para utilizar un aparato electrónico
- el camino para llegar a un lugar determinado desde un punto de partida

Ejemplos de algoritmos computacionales:

- Calcular los sueldos de los empleados de una empresa
- Actualizar el stock de un comercio
- Calcular las raíces de una ecuación

Desarrollemos el siguiente ejemplo de la vida diaria

Problema: Preparar un taza de café instantáneo
El grado de detalle que deberemos usar en la definición del método, dependerá de la persona que sea la ejecutante de la solución.
Si el ejecutante es un ama de casa, probablemente con el enunciado sea suficiente, pero si se trata de alguien que nunca preparó un café podríamos detallar los siguientes pasos:

ALGORITMO Cafe1

- Calentar una taza de agua sin llegar al punto del hervor;
- Poner en un taza tres cucharaditas de azúcar, dos de café instantáneo y media cucharadita de soda;
- Batir hasta que la mezcla se torne marrón claro;
- Llenar con el agua caliente la taza;
- Revolver para disolver la mezcla en el agua

FINALGORITMO

Obsérvese que para indicar el inicio y el fin del algoritmo se han utilizado las palabras **ALGORITMO** Y **FINALGORITMO** respectivamente y que los pasos han sido lo suficientemente simples para un principiante en el arte de preparar café.
Otro aspecto que es importante considerar es que contamos con una serie de elementos para poder preparar el café como por ejemplo: recipiente para calentar el agua, azúcar, café, cucharita, taza, soda.
Supongamos que no se tiene la certeza de que en el momento de hacer el café se tenga soda, por ende, este elemento se podrá reemplazar con agua, con lo cual el algoritmo será:

ALGORITMO Cafe2

- Calentar una taza de agua sin llegar al punto del hervor;
- Poner en un taza tres cucharaditas de azúcar , dos de café instantáneo
- **SI** se tiene soda

  ENTONCES agregar en la taza media cucharadita de soda
  SINO agregar en la taza media cucharadita de agua fría
- **FINSI**
- Batir hasta que la mezcla se torne marrón claro;
- Llenar con el agua caliente la taza;
- Revolver para disolver la mezcla en el agua

FINALGORITMO

Las dos primeras instrucciones se ejecutan una a continuación de otra, luego se presentan dos alternativas: o se agrega media cucharadita de soda o se agrega media cucharadita de agua fría. Para describirlas se ha usado las palabras **SI ENTONCES SINO FINSI** que se analizarán en detalle más adelante.

En este caso el algoritmo cubre ya mayor cantidad de posibilidades, no previstas en la versión anterior.

A partir de esto se pueden realizar las siguientes observaciones:

- El algoritmo debe estar compuesto por acciones tales que el ejecutante sea capaz de realizar
- El algoritmo debe ser enunciado en un lenguaje comprensible para el ejecutante, hombre o computador. En este último caso, estará restringido a un juego de instrucciones perfectamente determinado.
- El algoritmo deberá representar todo el conjunto de posibles resultados del problema, inclusive el caso de que no tenga solución

- Para un mismo problema se pueda describir más de un algoritmo y con cualquiera de ellos se deberá llegar a la/s misma/s solución/es; un algoritmo será más eficaz que otro. La eficacia del algoritmo depende de los recursos con que se cuente y los factores que se consideren: costos, tiempo, etc.

Ejemplo de resolución de problemas de la vida cotidiana

Ejemplo: Preparar un licuado de frutas
Recursos: Licuadora. Fruta con cáscara. Taza con leche. Taza con azúcar. Cuchillo. Plato. Todos los elementos están sobre la mesada. Se cuenta con las medidas necesarias de todos los ingredientes.

Algoritmo:

ALGORITMO Licuado
- Tomar el vaso de la licuadora
- Colocar el vaso en la base
- **SI** la licuadora no está enchufada
  ENTONCES enchufarla
- **FINSI**
- Tomar el cuchillo
- **REPETIR**
  tomar la fruta
  pelarla
  cortarla sobre el plato
  colocar la fruta cortada en el vaso
- **HASTAQUE** no haya más frutas
- Dejar el cuchillo
- Tomar la taza con la leche
- Echar la leche en el vaso
- Dejar la taza de la leche sobre la mesada
- Tomar la taza con el azúcar
- Colocar el azúcar en el vaso
- Dejar la taza del azúcar sobre la mesada
- Tapar el vaso
- Mover la perilla de encendido hacia la derecha
- **REPETIR**
  esperar
- **HASTAQUE** la mezcla esté licuada
- Mover la perilla de encendido hacia la izquierda

FINALGORITMO

En este caso ciertas acciones como la de Tomar la fruta, pelarla, cortarla, colocar..... se repiten mientras se tiene fruta para hacerlo. Aparecen aquí las palabras **REPETIR** y **HASTAQUE**, que veremos en capítulos posteriores.

FORMALIZACIÓN

Formalizaremos algunos conceptos vistos anteriormente.
Hemos mencionado que la forma de enunciar la solución a un problema planteado depende del **ejecutante** o también llamado **procesador**. Por lo tanto llamaremos así a toda entidad capaz de entender un enunciado y ejecutar los pasos descriptos en un algoritmo. Si bien en los ejemplos vistos el ejecutante se trataba de una persona en la resolución de problemas computacionales debemos pensar que el procesador será la computadora.
También hemos notado que para poder realizar su tarea el ejecutante debe contar con los recursos adecuados. El conjunto de estos recursos existentes en el momento de la ejecución de un trabajo constituye el **ambiente** del problema.

El método que se elija para proponer la solución de un tipo de problema depende del ejecutante y de los recursos o elementos con que se cuenta (ambiente). Cuando definimos algoritmo hemos hablado de un conjunto de pasos o acciones.

Una acción es un evento que modifica el ambiente y puede ser:

- Primitiva
- No-primitiva

Una acción es **primitiva** cuando para un ejecutante dado su enunciado es suficiente para que pueda ser ejecutada sin información adicional.
Una acción **no-primitiva** es aquella que puede ser descompuesta en acciones primitivas para un ejecutante dado.
También hemos visto que en los ejemplos se nos presentan situaciones que indican alternativas: “SI se tiene soda ...”, esta no es una acción porque no modifica el ambiente, pero son elementos que el ejecutante debe saber interpretar. A estos enunciados se los denomina **condición**.
Por lo tanto:

Una **condición** es una afirmación lógica sobre el estado de algún recurso del ambiente, que puede tomar valor verdadero o falso en el momento de la observación.

El ejecutante determina en el momento de la ejecución del algoritmo las acciones a seguir, dependiendo de que la condición sea satisfecha o no.

PROGRAMACIÓN MODULAR

Si bien este tema será visto con profundidad en teorías posteriores, plasmamos ahora un resumen del mismo.
La programación modular es un método de diseño y tiende a dividir el problema en partes perfectamente diferenciadas que puedan ser analizadas, resueltas y puestas a punto por separado.
Para atacar el análisis de un problema, y siguiendo el diseño *Top-Down*, se pueden utilizar criterios de programación modular para dividirlos en partes independientes, probando cada uno por separado y realizando su recomposición ascendente.
Cada una de las partes independientes se llama **Módulo** y para su determinación se deben tener en cuenta los siguientes criterios:

- un módulo debe corresponder a una función lógica perfectamente bien definida.
- los módulos deben ser pequeños para que sean claros y de poca complejidad.
- un módulo debe tener una estructura de caja negra, es decir la salida debe ser exclusivamente función de la entrada.
- cada módulo deber tener una única entrada y una única salida.

Objetivos de la programación modular

La programación modular tiende a:

- **disminuir complejidad**: disminuye la complejidad del problema original, dividiendo un problema en partes más simples.

- **aumentar la claridad**: el problema original es planteado ahora como una sucesión de módulos que resulta más fácil de comprender inclusive para terceras personas.

- **aumentar la fiabilidad**: como consecuencia de los dos puntos anteriores, aumenta la confiabilidad en todo proceso de resolución.

- **facilitar modificaciones y conexiones**: cada módulo puede realizarse y probarse por separado, minimizándose los problemas de puesta a punto al final.