Introducción a la Programación

Jessica Rodríguez

August 22, 2003

0.1 Introducción.. Bienvenida

La programación es una herramienta mentalmente poderosa, no sólo para aquellos que se benefician por un trabajo final, sino también para quienes se dedican a desarrollar soluciones prácticas, cómodas y confiables a problemas cotidianos, pues programar es una ciencia que exige minucioso cuidado y previsión del detalle, no da lugar a imprevistos, ve y va más allá, permitiendo y obligando al programador desarrollar un pensamiento analítico, ordenado, creativo, recursivo, práctico, modular y congruente que le permitirá salir adelante con cualquier tipo de problema que requiera de un análisis "paso a paso" y de una solución puntual.

Este texto es una ventana al inmenso mundo de posibilidades y beneficios que existe detrás del simple hecho de desarrollar una aplicación; Es un primer paso hacia un camino de múltiples destinos que cada día es más innovador, más exigente y más variado; Es una herramienta teórica y práctica que permitirá al estudiante familiarizarse con muchos conceptos .

Chapter 1

Conceptos Básicos de Computación

1.1 Algoritmos, Máquinas y Programas

Algoritmos

"Paso a paso" es quizá la expresión más apropiada para empezar a hablar de algoritmos, o mejor dicho, una buena definición para ellos. Teóricamente, Algoritmo se define como *conjunto finito y ordenado de pasos que se realizan en pro de resolver una tarea específica en un tiempo finito.* Prácticamente, un algoritmo es todo proceso "paso a paso" que realizamos para obtener un resultado, sin ir más allá un algoritmo es como una receta de cocina, una rutina en un gimnasio o la ruta fija para llegar a algún punto partiendo desde una ubicación específica.

El término Algoritmo, por reciente que parezca, en verdad proviene de un matemático árabe del siglo IX, Mohammed ibn Musa Djefar Al-Khwarizmi, quien publicó un importante tratado de Aritmética y álgebra titulado "tratado de cálculo por restablecimiento y equilibrio", (Kitab al-gebr we' I mukabala), de este también se deriva el término "álgebra" que significa restablecimiento, y en él se presentó el sistema indoárabe de numeración decimal. Cuando esta obra fue traducida al latín, el sistema de numeración europeo usado hasta ahora en las matemáticas, se vio afectado y la conversión de las matemáticas al sistema de numeración decimal, o arábigo cómo se le denominó, no dio

mucha espera. En ese entonces, y durante algún tiempo la aritmética decimal fue llamada algorismo, (El arte de Al-Khwarizmi); Finalmente el término adquirió el significado que le damos hoy en día.

Entonces, paso a paso es posible analizar y resolver un problema, muchos problemas se resuelven usando los algoritmos adecuados, pero cada problema tiene una forma específica, no existe un método general para interpretar, abstraer y generar una solución a algo, es ahí donde se encuentra la esencia, el reto. Con las herramientas conceptuales suficientes es posible generar un estilo propio para la solución de problemas, una forma de pensar que sólo se moldeará e irá surgiendo con la práctica.

Pero para no confundirnos con ideas tan abstractas veamos algunos ejemplos.

- PUDIN CREMOSO, Ingredientes para 6 personas:

  Harina: 80 g. Huevos: 4. Leche: $\frac{3}{4}$ litros. Mantequilla: 80 g. Azúcar en polvo: 80 g. Azúcar para el caramelo: 50 g. Corteza de un limón.

  - Hervir en un cazo la leche junto con la corteza de limón.
  - Agregar el azúcar. Retirar la corteza.
  - Derretir la mantequilla en una cazuela. Dorar la harina.
  - Verter la leche hirviente. Batir enérgicamente con un batidor. Retirar del fuego cuando vuelva a hervir. Dejar enfriar.
  - Separar las claras de las yemas. Montar las claras a punto de nieve.
  - Mezclar la leche, las yemas y las claras. Encender el horno.
  - Poner los 50 g de azúcar en un molde. Colocar sobre el fuego hasta que se disuelva. Con las manos protegidas, mover el molde para que el caramelo se extienda uniformemente. Rellenar con la mezcla.
  - Introducir el molde en otra cazuela. Verter agua en ésta hasta la mitad de la altura del molde. Llevar al horno, y cocer durante 1 hora. Desmoldar en el momento de llevar a la mesa. Servir tibio.

- INSTRUCCIONES DE INSTALACIÓN:
  - Cierre todas las aplicaciones en curso antes de iniciar la instalación de esta aplicación.
  - Indique la ruta en la cual desea que quede ubicado el programa.
  - Seleccione el tipo de instalación que desea, (Completa, económica, avanzada).
  - Espere mientras se copian los archivos...
  - Ha finalizado la instalación, reinicie su equipo para actualizar los cambios.

Máquinas

Ahora bien, sabemos que los algoritmos son una herramienta capaz de dar solución de una forma "mecánica" a gran número de problemas, pero qué pasa cuando en pro de resolver estos problemas resultan cálculos tediosos, procesos repetitivos, o labores que implican un esfuerzo humano mayor?. Esto mismo se pensó hace cientos de años, debía haber una forma de minimizar el esfuerzo humano o animal, no sólo en procesos mentales, sino también en procesos industriales que requerían mucha mano de obra para sacar una producción. En las últimas décadas del siglo XVIII la revolución industrial se produjo como respuesta a estas necesidades, primero con molinos para obtener energía del viento o del agua y luego con máquinas de vapor; de este modo muchas labores agrícolas e industriales se vieron impulsadas por la nueva tecnología. Pero los avances no daban abasto, habían muchos problemas sin resolver que necesitaban urgente atención: cartas de navegación imprecisas, control de censos, control exacto de mercancías etc. Grandes pensadores idearon algoritmos para trabajar con esta clase de problemas, soñaban con una máquina capaz de trabajar con datos, instrucciones que los manejaran y finalmente producir un resultado, (máquina analítica de Charles Babbage); lamentablemente, la tecnología de la época no proporcionaba las herramientas suficientes para esta clase de proyectos, sin embargo en esta época se idearon muchos algoritmos para máquinas capaces de trabajar de esta forma, abriendo así las puertas hacia un pensamiento que involucraba mucho más a las máquinas en procesos de cómputo, ampliando la visión sobre las mismas.

Muchas veces se ha juzgado la automatización como un productor de problemas sociales, ya que con su llegada un trabajo que antes era efectuado por varias personas ahora puede ser resuelto con mayor eficacia y exactitud por una máquina; respecto a este pensar hay que tener algo muy claro: La automatización exige una capacitación cada vez mayor, nos exige dar más y conocer más; es una herramienta que si sólo se usa para depender de ella, generará estancamiento mental. Por el contrario, si entendemos el proceso detrás de cada aplicación, si conocemos el alcance y las limitaciones de las mismas, y si ampliamos la visión hacia la infinidad de posibilidades que surgen al poder realizar procedimientos que antes no eran concebibles, el horizonte no se limita, se expande, dando así a nuestra mente la posibilidad de ir más allá, de querer y poder producir más.

Programas

Los programas surgen como la respuesta inmediata ante la idea de construir una máquina y hacer que esta lleve a cabo un conjunto de instrucciones, ya que un programa es la codificación de esas instrucciones de modo que puedan ser interpretadas y efectuadas de forma correcta por la máquina.

Para la implementación de algoritmos en el computador se usan programas. Pero no todos los programas son del mismo tipo; Existen diversas formas de codificar instrucciones, diversos **lenguajes de programación** que nos permiten hacer de formas distintas las mismas aplicaciones dependiendo de los requerimientos de estas y del alcance y limitaciones de cada lenguaje.

El lenguaje que veremos a lo largo de este texto es **Scheme**. La razón por la cual este lenguaje es ideal para comenzar a programar es, entre otras, su expresividad, es decir lo sencillo que resulta entender su sintaxis y acoplarse a ella debido a su naturalidad, lo que permite implementar algoritmos de una forma muy cómoda.

En el siguiente ejemplo veremos cómo una misma labor puede ser implementada por lenguajes distintos y cómo cada uno es diferente a los demás es decir, veremos un mismo programa en código Assembler,(de bajo nivel); y lenguajes de Alto nivel como C, Java y finalmente en **Scheme**. Queda a responsabilidad del lector sacar sus propias conclusiones...

Programa en Assembler

```
STACK     SEGMENT STACK
          DW     64 DUP (?)
STACK     ENDS

DATA      SEGMENT

MENSAJE      DB    "Hola mundo!!",13,10,"$"

DATA      ENDS

CODE      SEGMENT
          ASSUME CS:CODE, DS:DATA, SS:STACK
INICIO:
          MOV  AX,DATA
          MOV  DS,AX
          MOV  DX,OFFSET MENSAJE
          MOV  AH,09H
          INT     21H
          MOV  AH,4CH
          INT     21H
CODE      ENDS
          END  INICIO
```

Programa en C

```
#include <stdio.h>

        main()
        {
                printf ("\nHola mundo");
        }
```

Programa en Java

```
public class HolaMundo {
    public static void main(String arg[]){
       system.out.println("Hola Mundo!!!);
    }
}
```

Programa en Scheme

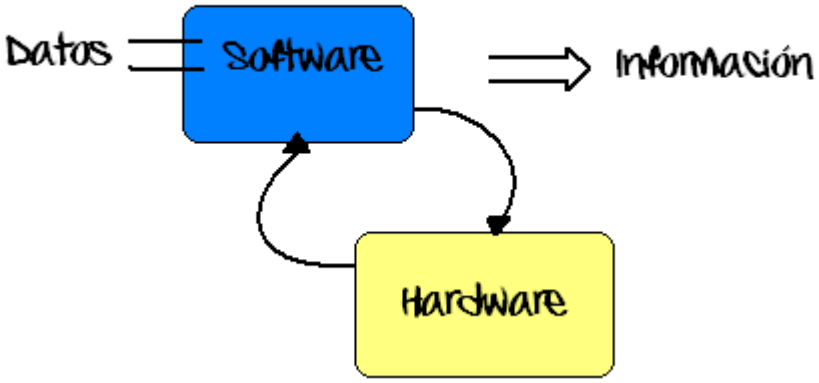
```
(define (programa)
    (write "hola Mundo!!"))
```

1.1.1 Sistemas de Cómputo

Junto a la necesidad de realizar cálculos grandes surgió la idea de desarrollar un método capaz de disminuir el tiempo en el que estos cálculos se realizaban, un sistema más confiable y eficiente, es decir, se vio la necesidad de desarrollar lo que conocemos como un sistema de cómputo .

Es así cómo se inicia la historia de la computación. A diferencia de lo que muchos podrían pensar, esta no comienza con el primer computador, comienza miles de años atrás con el *Ábaco* que fué la primera herramienta creada para agilizar operaciones aritméticas básicas. Desde la época del ábaco hasta hoy, el hombre ha creado miles de herramientas cada vez más sofisticadas para satisfacer sus necesidades en cuanto al manejo de información, y es así como llegamos al computador,una máquina moderna programable capaz de representar gran variedad de datos, recibirlos y generarlos como salida despues de procesarlos y ejecutar las instrucciones que se hallan asignado para ellos. Por ello el computador es un Sistema de procesamiento de información.

Para que los datos de entrada que se proporcionan a un computador puedan ser procesados y finalmente convertidos en información, debe ocurrir una interacción entre dos componentes fundamentales del computador: **Hardware y Software**. El término **Hardware** hace referencia a todos los componentes físicos de la máquina tales como los dispositivos de Entrada/Salida, La unidad Central de Procesamiento, La memoria central y los dispositivos de almacenamiento secundario en los que se almacenan grandes cantidades de datos e información. Cuando nos referimos a **Software**, estamos hablando del conjunto de instrucciones y programas que hacen uso de los dispositivos físicos para realizar con éxito el procesamiento de los datos.



1.1.2 Almacenamiento de Datos

Pero... cómo maneja la información un computador?, Cómo se representan los datos internamente?. Resulta difícil de comprender en un principio, pero toda la información que maneja un computador es internamente representada a partir de cadenas de bits,(unos y ceros,), y un conjunto de operaciones lógicas que pueden realizarse entre ellos a través del hardware. Para no ir muy lejos veamos un ejemplo sencillo: La representación de un número decimal (base diez) en bits (base binaria)

$$(456)_{10} \Rightarrow 111001000$$

De modo similar ocurre con las imágenes, por ejemplo, si se manejan como Mapa de Bits, una imagen se puede ver como un conjunto de puntos donde cada punto es un pixel y cada pixel ocupa determinado número de bits de acuerdo al formato de colores que maneje, desde 1 hasta 16 bits.

De este modo son convertidos todos los tipos de datos, desde números hasta imágenes y música son representados a través de bytes, cada tipo de archivo tiene una codificación especial y un manejo distinto de modo que podamos disfrutar del resultado final.

Variables

Dejando a un lado el tema de la representación interna de los datos,

(de seguro se estudiará más adelante en un curso especializado en ello), pasemos a ver cómo pueden ser manejados los datos en un nivel más alto. Cuando diseñamos un algoritmo que tiene determinado número de datos y que durante su ejecución los manipula pueden ocurrir dos cosas:

1. El dato permanece inmutable durante la ejecución del algoritmo
2. El valor relacionado con el dato se actualiza o cambia.

Para comprender mejor estos sucesos analicemos la siguiente situación:

- Una carrera de Autos en la cual existen los siguientes Datos de entrada:
  - Piloto en la primera posición (PrimerLugar).
  - Nombre del piloto a (NombreA).
  - Nombre del piloto b (NombreB).
    ⋮
  - Nombre del piloto n (NombreN).

  Durante el transcurso de la carrera los datos referentes a los nombres de los pilotos no cambiarán, pero el dato, (o variable), PrimerLugar cambiará a medida que la carrera evolucione, pues durante este tiempo cualquier piloto puede tomar la delantera y ser luego rebasado por cualquier contrincante; Así, en un tiempo t esta variable puede contener la información de NombreA, y en un tiempo $t + h$ la información de NombreX.

Estos dos *tipos de Variables* reciben nombres Especiales:

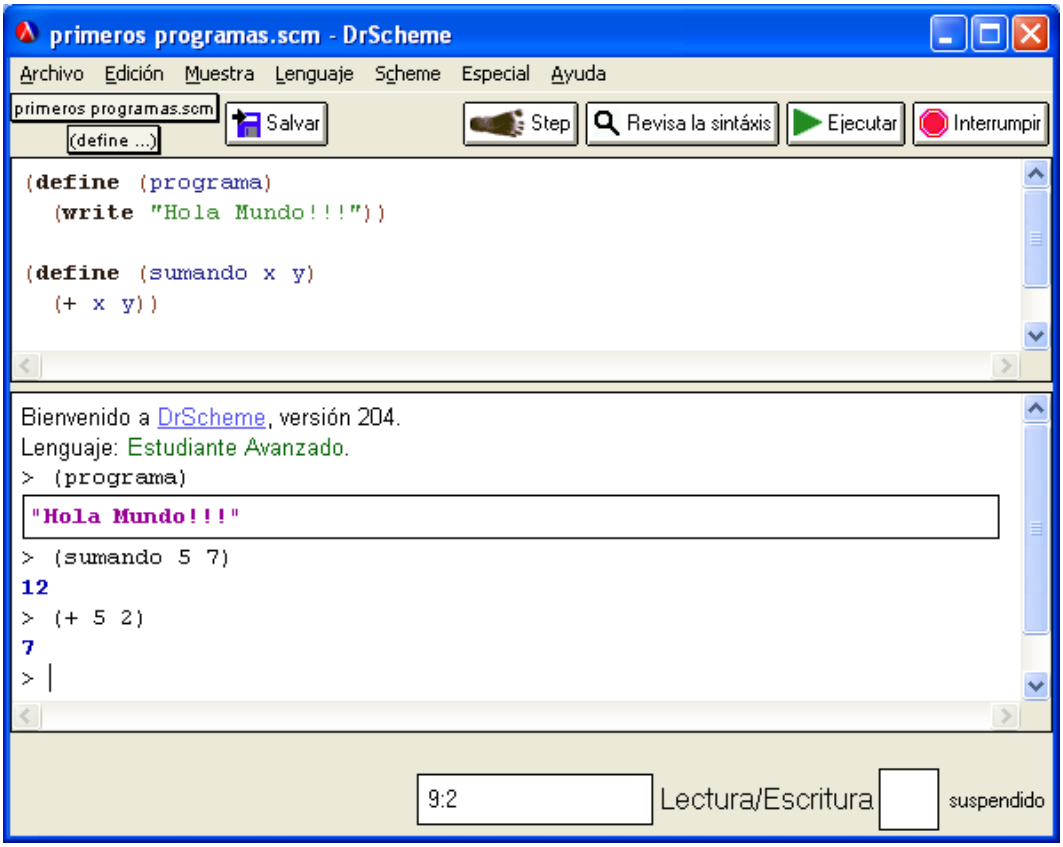
- ***Variables Declarativas Lógicas:*** Son aquellas que se declaran por primera vez de una forma y conservan el mismo valor asociado siempre, como los nombres de los pilotos.
- ***Variables que usan memoria:*** Son aquellas que a pesar de ser inicialmente declaradas de una forma pueden cambiar su valor a lo largo de la ejecución del programa.

Siempre que se diseña un algoritmo, es necesario el uso de **variables** para contener la información de entrada y/o la usada a lo largo de la ejecución, (como contadores o registros). Sin importar la naturaleza de esta dichas variables, por orden y disciplina, es recomendable hacer un conjunto de declaraciones en el mismo lugar del código,(preferiblemente al comienzo), de este modo es más fácil planear el manejo que se les va a dar a las mismas, y a su vez tener en claro la función de cada una desde el principio.

Por ejemplo si fuéramos a preparar alguna comida en particular, debemos seguir la receta adecuada, en la cual lo que primero necesitamos mirar es la lista de ingredientes.

1.1.3 Programas en Scheme

Como pudimos apreciar al principio de esta sección, **Scheme** ofrece una claridad en su sintaxis que nos permite comprender e implementar más fácilmente cualquier programa o algoritmo. Ahora nos corresponde conocer el amable entorno que **DrScheme** nos ofrece para empezar a trabajar!.



- ***Ventana de Definiciones:***
  En este espacio van las definiciones de los programas y funciones, en este caso vemos la definición del primer programa usado como ejemplo anteriormente y del programa sumando que se encarga de sumar dos valores.

- ***Ventana de Interacciones:***
  En este panel se evalúan las expresiones de Scheme interactivamente, es decir, además de poder evaluar los programas de la ventana de definiciones, es posible efectuar otras operaciones de **Scheme**, en la imagen se evalúan los programas definidos y se realiza una suma. En esta ventana también se indica el lenguaje usado al evaluar las expresiones, en este caso es el de Estudiante Avanzado, aunque en principio trabajaremos con Estudiante Principiante

- ***Step***  ***:***

Este botón es una ayuda didáctica para ver cómo son evaluadas las expresiones paso a paso por **Scheme**, Al ser accionado se activa el Stepper, que es el encargado de mostrar el proceso de evaluación paso a paso.

- ***Revisa la Sintaxis***  ***:***
  Este botón se encarga de resaltar varias cosas en caso de que las definiciones sean correctas, las más significativas son:

  - **Sintaxis:** Palabras Reservadas, variables ligadas, no ligadas y constantes. Para estos casos se usan colores y estilos de letra distintivos.
  - **Estructura Léxica:** Si se posiciona el mouse sobre una variable inmediatamente se dibujará una flecha hacia el lugar en el que fué declarada, se se sitúa el mouse sobre la variable en el momento de su declaración se dibujarán flechas hacia todos los puntos del código en el que esta es referenciada.
    Además haciendo click derecho sobre la misma, se despliega un menú que permite saltar por cada punto donde se referencia la variable, anclar o liberar las flechas y renombrar la variable.

- ***Ejecutar***  ***:*** Este botón evalúa las definiciones y reinicia la ventana de interacciones de modo que se puedan ejecutar los programas evaluados.

- ***Interrumpir***  ***:*** Es de gran ayuda si al evaluar las expresiones se llega a un ciclo infinito o a un cálculo muy pesado, pues interrumpe las evaluaciones si si presiona una vez, y las mata inmediatamente si se presiona dos veces.

- ***Nombre de la aplicación actual:*** En este pequeño botón a la izquierda de la pantalla se encuentra el nombre de la aplicación en curso, y al presionarlo se despliega un menú con ciertas rutas de directorios, al escoger una de ellas se abre un archivo nuevo desde la ubicación especificada.

- ***(define....):*** Al hacer click sobre este botón se despliega un menú con la lista de definiciones hechas en el archivo, en el caso del ejemplo, se mostrarían *Programa* y *Sumando*

- ***CREANDO Y GUARDANDO ARCHIVOS***  ***:***
  Cuando se abre la aplicación de **DrScheme**, inmediatamente quedan disponibles los dos paneles de edición e interacción para comenzar a trabajar en nuevas definiciones y programas. El botón *Salvar* permanece habilitado mientras el panel de definiciones presente cambios y estos no sean guardados, al hacer click en este botón se despliega una ventana para escoger la ubicación del archivo que se va a guardar; y cada vez que desee guardarse una actualización del mismo trabajo lo guarda en dicha dirección automáticamente.
  Si se desea crear un nuevo archivo, al hacer click en la barra de herramientas superior en el botón *Archivo* y seleccionando la opción *nuevo*, inmediatamente se abrirá otra ventana de **DrScheme** lista para comenzar a trabajar en ella.
  Además, existen otras opciones de guardado en el menú *Archivo*, es posible guardar no sólo las definiciones sino también las interacciones, y lo que es **MUY** importante para quienes trabajan con versiones de DrScheme distintas, existe la opción de *Guardar Otros...* que permite guardar las definiciones como texto de modo que si se está trabajando en una versión reciente de **DrScheme**, y luego es necesario acceder los archivos desde una versión anterior, no se presente ningún problema de compatibilidad entre las versiones.

- ***ABRIR ARCHIVOS:***
  Dentro del menú *Archivo* existe la opción de *Abrir...* De este modo es posible buscar entre los directorios disponibles la ubicación del archivo que se desea abrir.

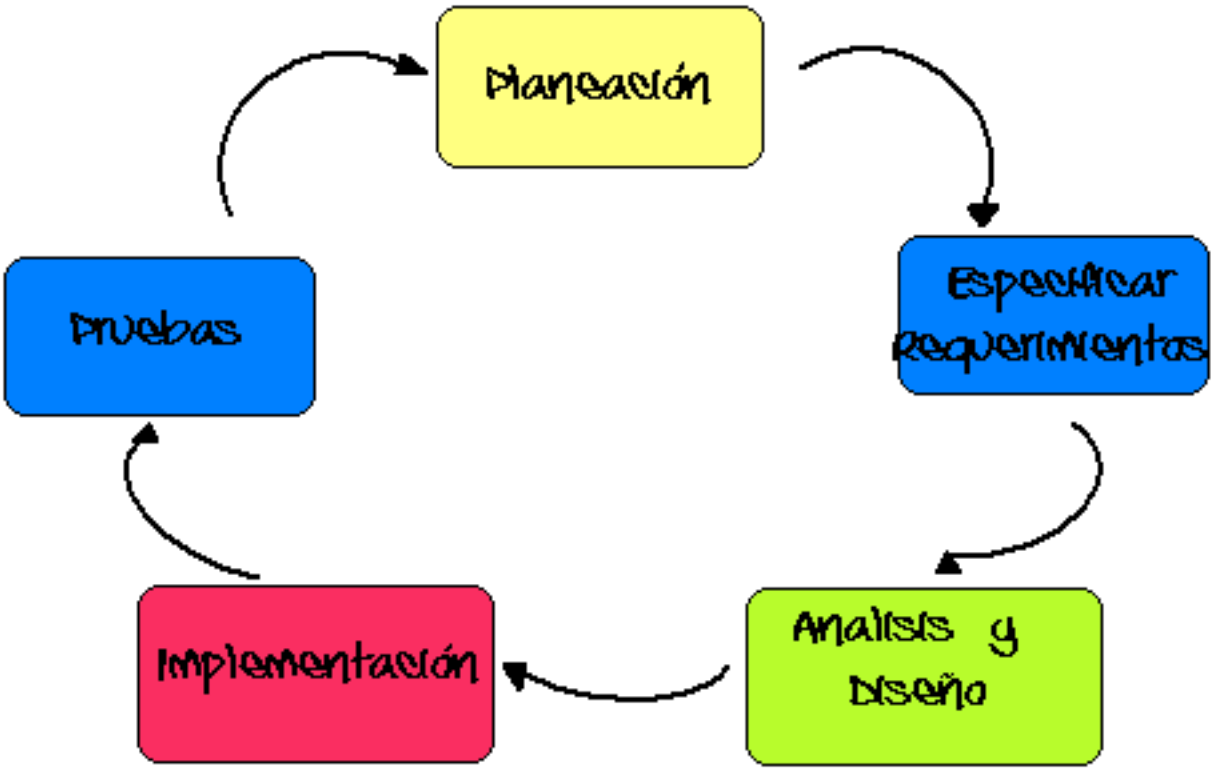
1.1.4 Preguntas y Ejercicios

1.2 El Proceso de Programación

1.2.1 Por qué es importante el diseño en la creación de Programas

Cuando se nos plantea un problema y debemos encontrar el algoritmo adecuado para resolverlo e implementarlo por medio de un programa, es de vital importancia tener en claro lo que se desea hacer y los recursos con los cuales se va a trabajar, Esto nos permite saber cuáles son los objetivos específicos de cada parte del programa; de este modo podemos fácilmente resolver cada uno de los requerimientos del mismo sin olvidar como interactuan unos con otros y así, finalmente, *"paso a paso"* resolver el problema principal.

Modelar y entender el problema planteado de este modo es ***diseñar*** el programa para resolverlo. Cuando nos encontramos en la labor de desarrollar una aplicación para resolver un planteamiento o problema debemos empezar por determinar qué es relevante y qué no lo es dentro del enunciado del problema, además, debemos saber qué es lo que se espera como producción o salida del mismo y la relación que guarde esta salida con los datos de entrada que también deben ser identificados desde el principio. Teniendo en cuenta estos aspectos desde un comienzo, debemos preocuparnos por ver el manejo que debemos darle a los datos que tenemos para procesarlos y las herramientas que nos ofrece el lenguaje en el que trabajemos, (**Scheme**), para hacer estos procedimientos de una forma más sencilla; en caso de que no se cuente con las herramientas para manejar los datos directamente o que las existentes no sean suficientes, debemos desarrollar los programas auxiliares que sean necesarios para cumplir con los requerimientos de nuestra aplicación. Finalmente, cuando se ha completado el proceso de desarrollo de la aplicación, (implementación) y esta se encuentra lista para ejecutarse, debemos cerciorarnos de que funciona de la manera esperada y que la salida producida sea la correcta.



Tal vez en este momento no parezca muy comprensible y aplicable este hecho, pero resulta indispensable a la hora de resolver problemas más complejos, en los que es necesario modular la solución de modo que el resolverla resulte sencillo por difícil que esta parezca en un comienzo.

1.2.2 Clarificar el problema

Es obvio que no podríamos siquiera pensar en generar una solución a un problema que no entendemos, o a uno que no estamos seguros de haber interpretado de la manera correcta. Por ello, es fundamental asegurarse de comprender todos los requerimientos que existan a la hora de realizar una aplicación, de modo que cualquier problema de interpretación quede desechado desde el comienzo.

Por ejemplo si tratamos un problema que nos pide ordenar un determinado grupo de datos, procesarlos y también realizar búsquedas especiales sobre ellos podríamos pensar varias cosas:

- Seguramente debemos ordenar los datos de modo que al realizar la búsqueda guiados por algún parámetro, este sea más fácil de encontrar para así procesar este dato y extraer la información necesaria.
- También podríamos pensar que debemos procesar los datos, luego realizar la búsqueda y finalmente ordenar los resultados que arroje la búsqueda.
- Además, se podría pensar que el problema consiste en realizar la búsqueda, luego procesar los datos arrojados por la búsqueda y finalmente ordenarlos según algún tipo de criterio establecido previamente.

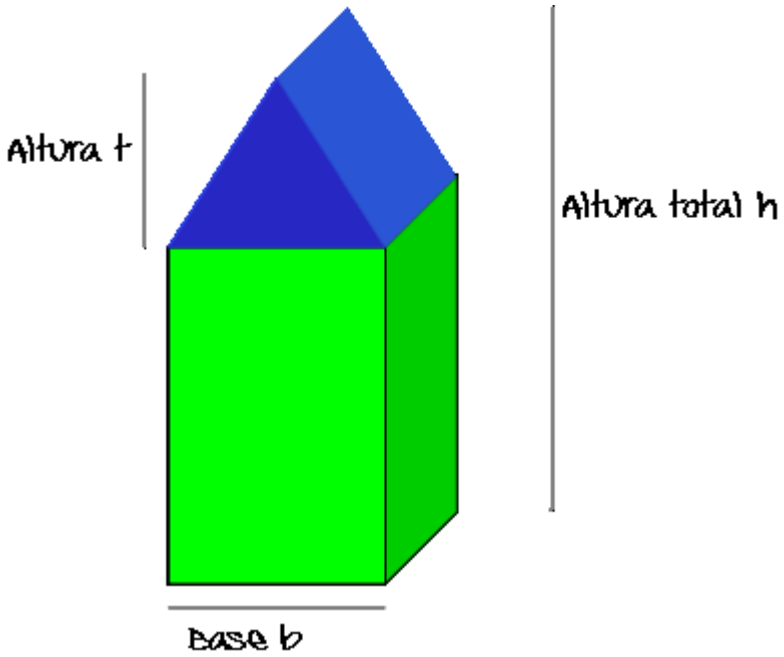
Como podemos ver existen distintas formas de interpretar un mismo problema. La importancia de la interpretación radica en el hecho de que si hacemos una interpretación incorrecta en ultimas la salida de nuestro programa será incorrecta, por esto debemos siempre asegurarnos de qué es lo que tenemos que hacer desde un principio, de este modo será más fácil visualizar la solución correcta.

1.2.3 Dividir el problema hasta hacerlo trivial y resolverlo

Ahora que tenemos en claro cuál es el problema a tratar, lo que se pretende con él y los resultados que se esperan, debemos buscar la manera de resolverlo fácilmente. Una técnica muy útil es la de *Dividir y Conquistar*, que consiste en tomar un problema grande y verlo como un conjunto de problemas más pequeños fáciles de manejar y resolver que al relacionarse unos con otros,

(podría ser compartiendo datos de E/S), generan la solución global del problema inicial.

Un pequeño ejemplo de esto puede ser averiguar el volumen total contenido en una figura como esta:



con los valores suministrados es muy fácil realizar este cálculo, vamos a ver cómo hacerlo paso por paso, dividiendo el problema para así hallar la solución.

1. El paralelepípedo (en verde) principal está formado por la proyección del cuadrado de lado b por la altura h - t, por lo tanto si se halla el área del cuadrado es posible calcular el volumen del paralelepípedo.

2. El volumen de la pirámide (en azul), está dada por la proyección del triángulo de base b y de altura t en una distancia b.

3. El área del cuadrado y del triángulo son entonces:

   (a) Área Cuadrado (Ac): b^2

   (b) Área Triángulo (At): $\frac{b*t}{2}$

4. Con estos valores ahora hallamos los volúmenes:

   (a) Paralelepípedo (*vpar*): *Ac * (h-t)*

   (b) Pirámide(*vpir*): *At * b*

5. Finalmente se suman los dos volúmenes y se obtiene como resultado el volumen total.

   - Volumen total: *vpar + vpir*

Con este mismo modelo es posible hallar soluciones a problemas grandes que requieren de muchos pasos intermedios para llegar a una solución. Hemos visto como *"paso a paso"* se puede resolver un problema.

1.2.4 Buscar otras soluciones al problema

La mayoría de las veces no basta con generar una solución a un planteamiento. El propósito de nuestros algoritmos no sólo es dar soluciones, además deberán dar la solución más adecuada y eficiente, pues los recursos de un computador son limitados y entre menos recursos de máquina ocupe un programa, menos tiempo demorará su ejecución, haciendo de él un programa eficiente. Los programas eficientes son producto de algoritmos eficientes, y estos se logran evadiendo operaciones innecesarias que pueden ser sustituidas por un número menor de cálculos.

Por ahora no nos corresponde fijar nuestra atención a este objetivo. Lo más importante en este momento es poder buscar varias soluciones a un mismo problema desde puntos de vista diferentes, para así comprobar que en muchos casos no existe un ÚNICO camino, y además desarrollar un pensamiento más abierto, (*"divergente"*), hacia las posibilidades de solución de un mismo planteamiento, es decir, desarrollar un pensamiento divergente y convergente a la vez.

Para ilustrar esta situación, analicemos un caso muy sencillo y común:

- Se tiene un conjunto de números desordenados y se quiere sacar el menor de ellos.

Este es un problema trivial, pero vamos a ver dos maneras de resolverlo:

1. Ordenar el conjunto de números y sacar el elemento que ocupe la primera o última posición de acuerdo a la configuración del ordenamiento, (de menor a mayor o de mayor a menor).

2. escoger un elemento cualquiera y denominarlo el *"menor"* del conjunto. Luego, compararlo con cada uno de los demás elementos, de modo que: si es menor que el elemento con el que se compara se sigue comparando con los demás; de lo contrario, se denominará *"menor"* al elemento con el que se comparó. Cuando se haya recorrido todo el conjunto el elemento denominado como **"menor"**, efectivamente lo será.

Como este caso hay muchos, es más, casi todos los problemas pueden ser resueltos de diversas formas. Lo importante es no cerrarse ante un sólo método, pues normalmente existen muchos caminos para lograr un mismo fin, y entre más caminos conozcamos mayor será nuestra posibilidad de llegar por el mejor.

1.2.5 Diseño guiado por Recetas

Cómo hemos visto a lo largo de esta sección es de gran importancia seguir un diseño a la hora de resolver un planteamiento. Ahora, veremos un modelo o receta de diseño que nos ayudará a plantear nuestras soluciones y a llevar en orden el desarrollo de su implementación.

Esta receta es una guía que paso a paso nos indica lo que se debe hacer y el orden en el que deben llevarse acabo cada uno de los pasos. Tiene una estructura muy sencilla y lleva el siguiente orden:

1. ***Contrato:*** Aquí va el nombre del programa y la relación entre su entrada y su salida, es decir se especifica de qué tipo de datos es cada una. Por ejemplo, recibe dos números y devuelve un número.

2. ***Propósito:*** Se especifica cuál es el objetivo del programa, por ejemplo sumar dos números

3.

4. ***Cabecera:*** Como ya se conoce el objetivo del programa y el tipo de datos que recibe como parámetros de entrada, entonces es posible crear la primera línea del programa, *"la cabecera"*.

5. ***Ejemplos:*** En este punto se especifican algunos ejemplos en los cuales se relacionarán los datos de entrada y la salida; es decir, si se van a sumar dos números, en el ejemplo irán esos dos números y el resultado esperado de su suma.

6. ***Cuerpo:*** Aquí se indica cómo van a ser computados los datos por el programa desarrollándolo junto a las funciones auxiliares y/o variables, (si son necesarias).

7. ***Pruebas:*** Como ya hemos hecho el programa, y además contamos con un grupo de ejemplos de los cuales conocemos los valores que deben producirse como resultados, se ejecuta el programa con cada uno de los ejemplos y se comparan los resultados obtenidos con los esperados. De este modo podremos saber si se ha cumplido, o no, el objetivo del programa.

El siguiente es el ejemplo que vimos antes para sumar dos números. Esta vez siguiendo los pasos de la receta de diseño. En Scheme, las líneas precedidas por ***";"*** son comentarios que no son tomados en cuenta a la hora de compilar y ejecutar el programa.

```
;EJEMPLO:
;CONTRATO: sumando: numero numero --> numero
;PROPOSITO:Producir el resultado de la suma de dos numeros.
;CABECERA:
; (define (sumando x y)
;         ...)

;EJEMPLOS: (sumando 6 5) debe producir:
11
;(sumando 100 456) debe producir
556

;CUERPO:
      (define (sumando x y)
          ( + x y))

;PRUEBAS:
```

```
(sumando 6 5)
(sumando 100 456)
```

Al compilar estas líneas de código deberán producirse en el panel de interacción las salidas esperadas.

Veamos ahora un programa para hacer conversiones de dólares a pesos:

```
;EJEMPLO 2
;CONTRATO: dolares-pesos: numero --> numero
;PROPOSITO: Determinar el valor en pesos de cierta cantidad dada en dolares.
;CABECERA:
; (define (dolares-pesos x )
;         ...)

;EJEMPLOS: (dolares-pesos 6 ) debe producir:
16800
;(dolares-pesos 100 ) debe producir
280000

;CUERPO:
      (define DOLAR 2800)
      (define (dolares-pesos m)
          (* DOLAR m))

;PRUEBAS:
(dolares-pesos 6 )
(dolares-pesos 100 )
```

1.2.6 Ejercicios

Para los siguientes ejercicios implemente, si es posible, las funciones y definiciones auxiliares que sean pertinentes.

1. Implementar en **Scheme** el ejemplo del volumen de la figura compuesta por un paralelepípedo y una pirámide explicado anteriormente. Desarrollar paso a paso siguiendo la receta de diseño.

2. Desarrollar de acuerdo a la receta de diseño un programa en **Scheme** que dadas las coordenadas x y y de dos puntos en el plano cartesiano determine la distancia total entre ellos.

3. Desarrollar de acuerdo a la receta de diseño un programa en **Scheme** que dados los radios internos y externos de un cilindro hueco en pulgadas, calcule el volumen máximo que pueda contener este en milímetros cúbicos.

4. Implemente la siguiente Ecuación conocida como la Ley de Coulomb, de modo que dada la distancia r, pueda determinarse el valor de F:

   $$F = k \times \frac{|q_1||q_2|}{r^2}$$

   Donde la constante $k = 8.9875 \times 10^9$.
   r, es la distancia dada
   q_1 yq_2 son los valores de la carga de un electron y un proton respectivamente, conocidos como $\pm e$ ($|e| = 1.60219 \times 10^{-19}$).
   Recuerde seguir los pasos de la receta de diseño.