Universidad Nacional del Litoral
Facultad de Ingeniería y Ciencias Hídricas
Departamento de Informática

Tecnicatura en Informática Aplicada al Diseño Multimedia y Sitios Web

Elementos de Programación

Unidad 1: Introducción a la Programación - Parte 1

Docente: Ing. Risso, Oscar Luis
Año: 2024

Índice

Unidad 1: Introducción a la Programación

Parte 1

1. En esta primera sesión de estudio

Esta unidad temática constituye una introducción a la programación. En esta introducción nos encargaremos de aprender algunos conceptos básicos y aproximarnos a la terminología técnica de programación.

En primer lugar estudiaremos el concepto de algoritmo y luego el de programa. Después, veremos de que se trata la ejecución y prueba de los programas. Como en general los programas no funcionan correctamente desde la primera vez estudiaremos los errores típicos en el proceso de programar.

2. Objetivos

Al finalizar este tema Ud. debe lograr:

- Conocer las etapas de resolución de problemas mediante algoritmos computacionales y programas.
- Comprender los principales conceptos involucrados en la construcción de programas.

3. Etapas para la Resolución de Problemas

En el proceso de resolución de problemas mediante algoritmos computacionales utilizaremos la siguiente serie de etapas:

- Definición del problema
- Análisis del problema
- Elección de un modelo
- Diseño de la solución
- Codificación
- Prueba
- Depuración
- Documentación

Analizaremos a continuación cada una de estas etapas utilizando el siguiente problema como ejemplo:

Problema: Calcular las raíces reales de una ecuación de segundo grado $(ax^2 + bx + c = 0)$ *conociendo los coeficientes a,b y c como datos.*

3.1. Definición del Problema

La definición del problema no es más que la completa enunciación del problema. Es fundamental conocer QUE se desea realizar a través de la definición.

En el caso a resolver de la ecuación de 2do grado, podemos observar que su enunciado es claro y plantea toda la información necesaria para encarar la resolución del problema. Solo podríamos agregar que existe la posibilidad de que no existan raíces (soluciones) en el campo real, para lo cual deberíamos proponer una respuesta que contemple ese caso.

Importante

Enunciado del Problema
Este paso puede parecer obvio, pues más de un lector se preguntará: ¿cómo voy a resolver un problema si no tengo enunciado?. Pues, en informática (y en especial en programación y análisis de sistemas) suele ser común que los propios solucionadores tengan que recabar importante información acerca del caso a resolver y plantear ellos mismos el enunciado del problema.

3.2. Análisis del Problema

En cualquier problema se debe poder determinar tres conjuntos perfectamente definidos:

1. Un conjunto de DATOS. (¿ qué información tengo ?)
2. Un conjunto de RESULTADOS (¿qué quiero obtener ?)
3. Una o más relaciones que vinculen a los dos conjuntos anteriores (¿cómo puedo obtener un resultado a partir de los datos ?) La identificación de estos conjuntos es fundamental para la resolución del problema.

Datos: a, b, c (coeficientes de la ecuación).
Resultados: r1,r2 (raíces o soluciones de la ecuación).
Relaciones entre datos y resultados:
$r1 := (-b + \sqrt{b^2 - 4ac}/(2a))$
$r2 := (-b - \sqrt{b^2 - 4ac}/(2a))$
Si $b^2 - 4ac$ *es negativo no existen raíces reales*

3.3. Elección de un Modelo

Debemos proponer o aplicar un modelo para lograr sistematizar la búsqueda de la solución. Esto es importante, porque de otro modo, dejaríamos librado a la habilidad del solucionador la resolución del problema. Y lo que proponemos es una metodología sistemática que esté más allá del arte o habilidad de una persona. Como estudiaremos más adelante existen ciertos modelos o paradigmas a partir de los cuales se comienza a bosquejar la solución computacional de un problema.

Importante

Estudiaremos en esta materia el modelo o paradigma de la programación orientada a objetos.

3.4. Diseño de la solución

En base al modelo o paradigma elegido, debemos plantear el diseño de la solución.
Esta etapa está muy ligada al modelo en cuestión: existen técnicas de diseño para cada modelo.

En el ejemplo, la solución es casi inmediata (problema casi trivial), y por tanto no requiere un diseño previo

---

3.5. Codificación

Consiste en describir los pasos que deben ejecutarse para resolver una instancia del problema. Es común utilizar en esta etapa un lenguaje algorítmico. Si se ha elegido un lenguaje interpretable por una computadora (lenguaje de programación) estaremos codificando un programa.

```
Comienzo
 Conocer los coeficientes a,b,c
 Calcular discriminante d=b^2-4ac
 Calcular r1
 Calcular r2
 Informar r1 y r2
Fin
```

3.6. Prueba

Esta etapa implica realizar la ejecución de los pasos propuestos en la codificación, suministrando los datos de un caso, para obtener los correspondientes resultados. Esta etapa implica la verificación del funcionamiento de la solución propuesta y detectar los errores que se presenten.
La presencia de errores implica pasar a la etapa de depuración.

En el ejemplo, si efectuamos la prueba con la ecuación $x^2 - 3x + 2 = 0$ *los datos serán: 1,-3,2 Podemos ver que el discriminante d es 1 y las raíces serán r1=l y r2=2. Pero si probamos el algoritmo con la ecuación* $x^2 - 2x + 5 = 0$*, vemos que el discriminante es –16 y no podemos aplicar la fórmula resolvente pues no existe solución real para* $\sqrt{-16}$

3.7. Depuración

Es la etapa donde deben subsanarse los errores detectados en la prueba. Esta depuración debe realizarse revisando las etapas anteriores. La depuración debe continuar, hasta lograr una prueba libre de errores.

```
Comienzo
 Conocer los coeficientes a,b,c
 Calcular discriminante d=b^2-4ac
 Si d<0 informar 'No existen raíces reales'
   sino
     calcular r1
     calcular r2
     informar r1 y r2
Fin
```

3.8. Documentación

Finalizado la resolución del problema mediante un programa, es necesario y muy conveniente documentarlo. Soluciones bien documentadas facilitan su uso general y su mantenimiento.

Como veremos más adelante, los programas pobremente documentados son difíciles de leer, corregir y prácticamente imposible de mantener.

La documentación puede ser externa o interna. La interna se refiere a líneas de texto en el código con comentarios descriptivos. La documentación externa incluye análisis, diseño, código, manuales de usuario con instrucciones para el uso del programa e interpretación de resultados.

Como veremos, esta etapa no debe dejarse para el final, pues casi todas las etapas anteriores requieren su correspondiente documentación a medida que se van desarrollando.

4. Algoritmo

Mencionamos varias veces la palabra algoritmo sin definir su significado. Es común encontrar el término de algoritmo como sinónimo de procedimiento, técnica o método. Pero expresaremos su significado más específicamente.

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

4.1. Características de un algoritmo

Las características principales de todo algoritmo son:

- **Finitud**: permite arribar a la solución de un problema después de una cantidad finita de pasos.
- **Precisión**: cada paso debe expresarse en forma clara y precisa y no debe dar lugar a ambigüedades.
- **Generalidad**: la solución debe ser aplicable a un conjunto de problemas del mismo tipo y no a un problema particular.

Obsérvese que en la definición y en las características no se hace mención alguna de resolución computacional de un algoritmo. Efectivamente, veremos que en ciertos problemas su solución es correcto, pues nos encontramos a diario con innumerables algoritmos que no requieren para su ejecución la presencia de una computadora. Por ejemplo:

- Una receta de cocina.
- Un manual de instalación de un artefacto.
- La multiplicación de dos números.
- Comprar 1 Kg. de pan.

Como observación, podemos decir que un algoritmo debe estar expresado en un lenguaje comprensible para el ejecutante. Y las acciones o pasos descriptos deben ser tales que el ejecutante sea capaz de realizarlas.

También es observable que un problema determinado puede ser resuelto con varios algoritmos diferentes. Es deseable encontrar una solución eficaz. Podemos afirmar que un algoritmo es eficaz si resuelve el problema con un mínimo de recursos y en el menor tiempo posible.

Problema: Hallar el área de un triángulo rectángulo, cuya altura es 10cm y su base 5cm.
Análisis del problema

- *Datos: base (10cm) y altura (5cm) del triángulo.*
- *Resultado: área del triángulo.*

- *Relaciones: área triángulo = base * altura /2*

Algoritmo:

```
Comienzo
 Multiplicar 10 por 5 (Base por altura)
 Dividir 50 por 2 (área de un triángulo)
 Informar el valor del cociente
Fin
```

Nota

La resolución del problema anterior tiene un defecto importante, de acuerdo a las características deseables de un algoritmo. El mismo consiste en que resuelve un problema particular. Es decir, calcula el área de un triángulo de base 10 cm y altura 5 cm, y no el área de cualquier triángulo. Para un diseño más general es necesario modificar el algoritmo de la forma siguiente:

```
Comienzo
 Conocer base y altura del triángulo.
 Multiplicar base por altura.
 Dividir el producto anterior por 2.
 Informar el valor del cociente.
 Fin
```

5. Programa

Un algoritmo que ha sido codificado empleando un lenguaje de programación interpretable por una computadora constituye un programa.

Las etapas de resolución de problemas correspondientes a la **Codificación**, **Prueba** y **Depuración** constituyen el proceso de **Programación**.

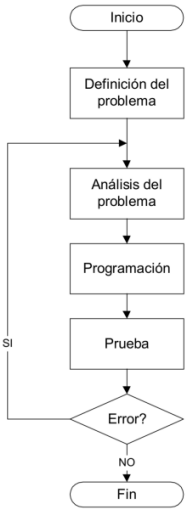

Figura 1: Diagrama de las Etapas de Resolución de Problemas

En este diagrama no se incluye el proceso de documentación ya que, como se verá luego, debe realizarse durante todas las etapas de resolución del problema.

6. Ejecución (prueba) del programa

Para poder probar un programa escrito en un lenguaje de programación de alto nivel -como el que usaremos en la materia- es necesario generar un código ejecutable. Este proceso puede efectuarse mediante la Compilacion o mediante la Interpretación del código fuente que editó el programador.

6.1. El Proceso de Compilación

El proceso de compilación es el que da como resultado un archivo o imagen ejecutable. Los archivos ejecutables pueden ser directamente utilizados por el usuario mediante una simple llamada desde el sistema operativo (por ejemplo un doble clic en Windows). En el proceso de compilación se procesa todo el programa y se genera un archivo ejecutable que es independiente del compilador utilizado.

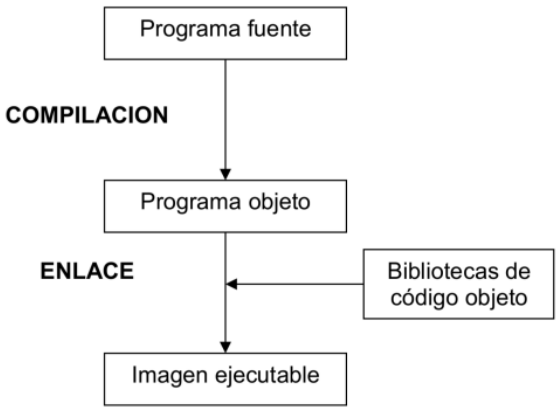

Figura 2: Diagrama del Proceso de Compilación

Importante

Los programas que escribimos usando lenguajes de alto nivel tienen una sintaxis similar (en inglés) a la usada por las personas para comunicarse entre sí. Pero las computadoras no entienden el inglés; son máquinas electrónicas que responden a señales eléctricas de encendido y apagado (ceros y unos en un modelo matemático).
La compilación traduce el programa escrito en un lenguaje de alto nivel al código que interpreta una computadora.

6.2. El Proceso de Interpretación

En este caso el intérprete procesa instrucción por instrucción, sin generar un código ejecutable completo. Cada vez que el usuarios necesita el ejecutar el programa deberá llamar al intérprete para que lo ejecute línea por línea.

6.3. Ventajas y Desventajas de Compiladores e Intérpretes

6.3.1. Ventajas

Compilación	Interpretación
Errores de sintaxis antes de la ejecución. Velocidad de ejecución Protección del código.	Ejecución en una sola etapa Proceso interactivo de depuración

6.3.2. Desventajas

Compilación	Interpretación
En algunos lenguajes es un proceso en varias etapas y a menudo engorroso. (*)	Errores de sintaxis detectados durante la ejecución Baja velocidad de ejecución. Código abierto.

Nota (*)

Existen compiladores con entornos de desarrollo integrados (IDE) que simplifican esta tarea.

6.4. Las Máquinas Virtuales

Una máquina virtual es un software que emula a una computadora y tiene la capacidad de ejecutar programas como si fuera una computadora real.

6.5. La Máquina Virtual de Java

Java es un lenguaje que corre dentro de su propia máquina virtual. Cuando compilamos el programa fuente obtenemos un código intermedio bycode (código de bytes) este código es entendido por la JVM (Java Virtual Machine) o JRE (Java Runtime Environment).

7. Depuración de Programas

Depuración o "debugging" significa eliminar los errores de un programa. En ciertos casos esa depuración es sencilla, pero a menudo constituye un proceso penoso. Esto depende de los tipos de errores de un programa.

7.1. Errores de un Programa

- **Errores en Tiempo de Compilación:** son aquellos en los que se infringen las reglas que definen la estructura de las declaraciones y sentencias. Estos errores son denominados también errores de sintaxis. Los errores de sintaxis más comunes son: errores tipográficos, falta del punto y coma final y utilización de variables que no han sido declaradas previamente.
- **Errores en Tiempo de Ejecución:** los programas contienen estos errores cuando, a pesar de contar con sentencias sintácticamente válidas, se producen errores al ejecutar estas sentencias. Por ejemplo, un programa podría intentar utilizar un archivo que no existe en el disco o dividir un número por cero.
- **Errores de Lógica:** en muchos casos el programa arroja resultados incorrectos a pesar de que no posea errores de sintaxis y tampoco de errores al ejecutarse. En general se trata de los casos en que el programa no realiza las tareas que originalmente teníamos en mente. Por ejemplo, puede que un programa que debería multiplicar números, los sume. Observaríamos que no hay errores de sintaxis, tampoco errores de ejecución y, sin embargo, al darle dos números al programa nos devolvería su suma y no su producto.

8. Resumen

1. Las etapas para la resolución de problemas constituyen una forma sistemática de plantear soluciones algorítmicas a dichos problemas y resolver los casos propuestos.

2. Es fundamental entender claramente lo que plantea el problema (1er etapa: definición); conocer los datos, los resultados a obtener y las relaciones entre datos y resultados (2da etapa: Análisis); seleccionar un modelo o paradigma (3er etapa: modelo); elegir un método (4ta etapa: método).

3. La quinta etapa para la resolución de problemas se denomina codificación y en ella se plantea el algoritmo. Para codificar un algoritmo se debe utilizar un lenguaje adecuado que describa las acciones o pasos a realizar. Ese lenguaje está ligado al ejecutante encargado de procesar las acciones. El objetivo de esta asignatura es aprender a construir algoritmos computacionales por lo cual nos centraremos en esta etapa.

4. Podemos escribir un algoritmo usando nuestro lenguaje natural. Ese algoritmo podrá ser ejecutado por una persona la cual entiende nuestro lenguaje (y que sepa cómo realizar cada acción). En tal caso estamos en presencia de un algoritmo no computacional.

5. Al escribir un algoritmo usando un lenguaje similar al que puede interpretar una computadora estamos en presencia de un algoritmo computacional. Si usamos un lenguaje de programación, el algoritmo constituye un programa.

6. Un algoritmo debe:

   a) completar siempre un número determinado de pasos durante su ejecución, independientemente de los datos del problema (Finitud).

   b) tener acciones que puedan ser interpretadas por el ejecutante sin dar lugar a ambigüedades (Precisión).

   c) resolver un grupo de casos similares o de igual tipo (Generalidad).

7. Al ejecutar un algoritmo estamos realizando la prueba del mismo (6ta etapa). Si detectamos errores debemos corregirlos o depurarlos (7ma etapa). Por último es importante documentar la solución propuesta (8va etapa).

Parte I
Segunda Sesión de Estudio

9. En esta segunda sesión de estudio

Existe una gran cantidad de lenguajes de programación. Nosotros estudiaremos una en particular (JAVA). Sin embargo, en esta unidad veremos las características generales que distinguen a unos lenguajes de otros y las ventajes relativas.

También veremos algunos consejos acerca de un proceso que nunca debe faltar en la programación: la documentación.

10. Objetivos

Al finalizar este tema Ud. debe lograr:

- Conocer los paradigmas de programación existentes en el actual estado del arte de la ingeniería del software.

11. Clasificación de los Lenguajes de Programación

En base a la similitud de los lenguajes de programación de computadoras respecto de nuestro lenguaje natural podemos clasificarlos en:

- **Lenguajes de Bajo Nivel**

  Si bien estos lenguajes pueden ser interpretados por un humano, esta tarea es altamente compleja. Los lenguajes de bajo nivel se utilizan en situaciones muy particulares, por ejemplo para la programación de sistemas electrónicos sencillos (algunos aparatos electrónicos de uso cotidiano, por ejemplo). Son lenguajes de de bajo nivel:

  - Lenguaje Máquina
  - Lenguaje Ensamblador

- **Lenguajes de Alto Nivel**

  Los lenguajes de alto nivel son más accesibles a ser leídos por un humano. Generalmente consisten en un conjunto de palabras y símbolos que tienen una relación directa con su significado. Los lenguajes más ampliamente utilizados hoy en día son de alto nivel. Estos lenguajes facilitan el entrenamiento de recursos humanos (programadores) y permiten resolver más rápidamente problemas de gran complejidad. Los entornos de desarrollo para lenguajes de alto nivel poseen un amplio espectro de herramientas para la edición, compilación y depuración de los programas.

Según la metodología empleada para abordar el diseño del programa:

- **Procedimentales:** estos se basan en un conjunto de subprogramas o subrutinas que resuelven diferentes partes del problema. Por ejemplo, para resolver el problema del cálculo del volumen de un cilindro, podríamos tener un programa principal que pide los datos al usuario y muestra los resultados y un subprograma que realiza el cálculo.
- **Imperativos y Declarativos:** en los lenguajes imperativos las líneas del programa le dicen a la computadora que debe hacer. A lo largo de la evolución de los lenguajes de alto nivel se ha pasado desde los lenguajes imperativos hacia los lenguajes declarativos. En los lenguajes declarativos son aquellos en que en las líneas del programa encontramos las mismas abstracciones del problema en cuestión. Un buen ejemplo de lenguajes declarativos son los lenguajes orientados a objetos.

- **Orientados a objetos:** en este caso, las representaciones que se plasman en el programa tiene una relación directa con la realidad del problema que resuelven. Siguiendo el ejemplo anterior, nuestro programa contaría con un objeto denominado interfaz de usuario y otro cilindro. Estos objetos se encargaría de realizar las operaciones de carga de datos y muestra de resultados (el objeto interfaz de usuario) y de cálculo del volumen (el objeto cilindro).
- **Manejados a eventos:** los lenguajes manejados por eventos son aquellos en los que todas las acciones del programa son disparadas por algún suceso disparador o evento. Este evento puede provenir del usuario (como apretar un botón) o del sistema (como el aviso de que hay espacio insuficiente en el disco). En cualquier caso, el evento es enviados al programa y este podrá las acciones específicas para este evento.

Esta clasificación no es excluyente. Podríamos encontrar, por ejemplo, lenguajes orientados y manejados por eventos. De forma similar, algunos lenguajes son procedimentales y manejados por eventos. En particular, las versiones más recientes del lenguaje Pascal permiten programar según cualquiera de las clasificaciones anteriores e incluso combinarlas a todas en un solo programa. Sin embargo generalmente en una programa en particular se utiliza de forma consistente la metodología para abordar el problema. Es recomendable que cuando se programa orientado a objetos no se mezclen líneas de programas procedimentales.

12. Documentación de Programas

La documentación es un aspecto muy importante en la programación. Sin embargo, generalmente los programadores principiantes suelen dejar de lado el proceso de documentación por falta de dedicación o simple desconocimiento de su relevancia. La documentación es todo el material anexo a un programa que no constituye parte del código que se compila. Todo esto sirve para que el mantenimiento, la reparación y la actualización del programa sea rápida y segura. En el caso de que estas tareas deba desarrollarlas otro programador la documentación será imprescindible. Pero más aún, cuando hayan pasado varios meses de hecho el programa, para usted mismo será imprescindible la documentación porque ya no podrá recordar claramente lo que hizo y le constará mucho volver a entender el programa para poder hacer las modificaciones necesarias. La documentación puede realizarse en el mismo archivo de texto del programa mediante cometarios. Los comentarios poseen una sintaxis particular y son totalmente ignorados por el compilador. Esta documentación es denominada interna ya que se encuentra en el mismo archivo de texto del programa. Por otra parte, cualquier tipo de información que no se encuentre en el mismo programa se denomina documentación externa. La documentación externa puede consistir por ejemplo en gráficas, texto explicativo, manual de usuario, algoritmos en pseudocódigo, etc. A continuación podemos ver unas recomendaciones generales en cuanto a la documentación de programas:

12.1. Documentación Interna

- Objetivo del programa o módulo.
- Datos de autoría.
- Fechas: iniciación, finalización, modificaciones.
- Finalidad y uso de cada constante, estructura de datos y variable.
- Finalidad y modo de uso de cada procedimiento: entradas, procesos y salidas.
- Comentarios aclaratorios en las líneas de código y bloques.
- Buen uso de la tipografía: mayúsculas y minúsculas.
- Usar correctamente la indentación y separación de bloques lógicos.
- Utilizar nombres significativos para los identificadores.
- Documentar a medida que se realiza la implementación: ¡NO AL FINALIZARLA!

12.2. Documentación Externa

- Información general: propósitos, modo de uso.
- Estructura modular
- Árboles jerárquicos de módulos y clases Documentación de las módulos y clases
- Nombre y archivo que la contiene
- Diagrama de herencia con atributos y métodos
- Breve descripción de la clase.
- Atributos: nombre, estructura de datos
- Funcionalidades: nombre, cabecera, finalidad, modo de uso, método y algoritmo
- Fechas y datos de autoría

Puede que algunos términos de esta lista le resulten poco familiares pero los comprenderá a medida que avancemos en el curso.

13. Algunos Lenguajes de Programación de Alto Nivel

Existen muchos lenguajes de alto nivel. Cada cual tiene su propio campo de aplicación. Sin embargo, existen lenguajes de propósitos generales como el Pascal, el Basic o el C++. A continuación podemos ver una lista con los lenguajes más difundidos en la actualidad, listados en orden de aparición aproximadamente

- **FORTRAN**: FORmua TRANslator. Lenguaje para cálculos científicos. Posee especiales características para el tratamiento de números. nació a mediados de los 50 y todavía se lo emplea.
- **COBOL**: COmmon Business Oriented Language. Lenguaje orientado a los negocios comunes. Útil para tareas administrativas. Fue creado a fines de los 50.
- **BASIC**: Lenguaje para todo propósito de gran sencillez de aprendizaje. Fue creado a mediados de los 60 y fue muy popular con la aparición de las primeras PCs a principios de los 80.
- **ALGOL**: Lenguaje Algorítmico. De uso científico.
- **PASCAL**: Primer lenguaje estructurado. Nació a principios de los 70 y fue popular por su modularidad, elegancia y sencillez. Respondía al paradigma de la programación estructurada, aunque hoy existen versiones orientadas a objetos.
- **C, C++**: El lenguaje C nació a principios de los 70. Es algo críptico (difícil) para principiantes, pero genera programas muy eficientes. C++ es una extensión de C para el paradigma de objetos. Es muy popular a nivel profesional. En 1998 los comités ANSI e ISO publicaron un standard de este lenguaje de gran potencia, que facilita el desarrollo de aplicaciones.
- **PROLOG**: Programación lógica. Empleado en inteligencia Artificial.
- **SMALTALK**: lenguaje empleado exclusivamente para el modelo de objetos.
- **VISUAL C, VISUAL BASIC, VISUAL PASCAL (Delphi)**: lenguajes de programación visual. Facilitan el diseño de interfaces gráficas a través de herramientas visuales, el código se escribe automáticamente.
- **JAVA**: lenguaje multiplataforma de estructura similar a C++. Empleado para aplicaciones en Internet.
- **HTML**: lenguaje de hipertexto empleado en el diseño de páginas web.

En esta materia se estudiará el lenguaje Java.

14. Resumen

1. Al programar se suelen emplear varios tipos de lenguajes. Los de alto nivel se denominan así pues poseen una sintaxis similar a la de cualquier lenguaje natural empleado por las personas para comunicarse. Un programa escrito en un lenguaje de programación se denomina programa fuente.

2. Al ejecutar un algoritmo estamos realizando la prueba del mismo (6ta etapa). Si detectamos errores debemos corregirlos o depurarlos (7ma etapa). Por último es importante documentar la solución propuesta (8va etapa).

3. Los programas fuente escritos en lenguajes de alto nivel no son interpretables por una computadora, por ello debemos compilarlos para lograr un programa ejecutable. Es posible usar intérpretes en lugar de compiladores, pero estos últimos son más eficientes.

4. Los lenguajes de alto nivel se denominan así pues poseen una sintaxis similar a la de un lenguaje natural empleado por personas para comunicarse. Esto nos facilita la legibilidad e interpretación de los programas.

5. Existen varios tipos de lenguajes de programación de alto nivel. Algunos fueron creados para facilitar la resolución de cierto tipo de problemas, aunque predominan los de aplicación general. A su vez, estos lenguajes suelen responder a ciertos modelos o paradigmas de diseño de soluciones.

15. Ejercicios

[estos no se presentan, luego publicaremos las respuestas]
Realice las etapas para la resolución de problemas hasta la de codificación inclusive (algoritmo) para los siguientes problemas. Utilice un lenguaje coloquial para describir los pasos del algoritmo:

- Ejercicio 1.

  Calcular el volumen de un cilindro conociendo su altura y el radio de la base.

  Nota: Volumen cilindro = Superficie de la Base x Altura.

- Ejercicio 2.

  Un usuario desea conocer cuánto debe pagar por el consumo de energía eléctrica realizado en el último período. Se conocen el costo del KW sin impuestos, la lectura actual del medidor y la lectura del período anterior. Además en concepto de impuestos los usuarios abonan un 22 % sobre el total correspondiente al consumo.

- Ejercicio 3.

  Se conocen 3 datos numéricos. Determinar e informar el mayor.

Consigna para el Primer Foro de Debates
Participación obligatoria con 1 aporte al menos.

- Investigue algunos lenguajes de programación de alto nivel, elija 1 (uno) y mencione sus características, para qué tipo de problemas fueron creados, ventajas y desventajas.