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

FUNDAMENTOS DE PROGRAMACIÓN

Asignatura correspondiente al plan de estudios de la carrera de Ingeniería Informática

UNIDAD 1

RESOLUCION DE PROBLEMAS

Ing. Horacio Loyarte ®

Facultad de Ingeniería y Ciencias Hídricas
FICH
UNL

UNIDAD 1

Resolución de problemas

Resumen de Conceptos

Introducción

El objetivo de este tema es introducir al lector en una metodología para abordar la resolución de problemas mediante el empleo de algoritmos, los cuales permitirán -más adelante- construir programas que pueden ser ejecutados por una computadora.

Como primer medida se empleará el concepto de algoritmo en forma general usando un lenguaje natural para plantear pasos o acciones, y considerando que el ejecutor de tales acciones puede ser una persona. Luego en la próxima unidad, se formalizará un lenguaje algorítmico planteando reglas sintácticas similares a las de los lenguajes de programación de alto nivel que pueden ser interpretados por una computadora.

El planteo sistemático de un método para la resolución de problemas es una buena práctica para hallar la solución de problemas mediante algoritmos computacionales.

Etapas para la Resolución de Problemas

En el proceso de resolución de problemas se utilizará la siguiente metodología, consistente en atacar el problema en cuestión por etapas. Esta serie de etapas es la siguiente:

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

Se analizarán 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.

a. Definición del Problema

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

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.

En el ejemplo, de puede observar que su enunciado es claro y plantea toda la información necesaria para encarar la resolución del problema.
Solo se podría agregar que existe la posibilidad de que no existan raíces (soluciones) en el campo real, para lo cual se debería proponer una respuesta que contemple ese caso.

b. 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.

En problema ejemplo:

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

c. Elección de un Modelo

Con frecuencia se debe proponer o aplicar un modelo para lograr sistematizar la búsqueda de la solución. Esto es importante, porque de otro modo, se dejaría librado a la habilidad del *solucionador* la resolución del problema. Y lo que se propone es una metodología sistemática que esté más allá del arte o habilidad de una persona.

Como se estudiará más adelante existen ciertos modelos o paradigmas a partir de los cuales se comienza a bosquejar la solución computacional de un problema.

Se estudiarán modelos o paradigmas al introducirnos en la formalización de algoritmos computacionales.

d. Diseño de la solución

En base al modelo o paradigma elegido, se debe plantear el diseño de la solución. Esta etapa está muy ligada al modelo en cuestión: existen técnicas de diseño para cada modelo. Se irán desarrollando estas técnicas con cada paradigma.

En el ejemplo la solución es casi inmediata (problema casi trivial), no requiere un diseño previo. Pues con aplicar la fórmula resolvente se obtienen las soluciones.

e. Codificación

Esta etapa consiste en describir los pasos que deben ejecutarse para resolver una instancia del problema. Es común utilizar aquí un lenguaje algorítmico. Si se elige para codificar un lenguaje interpretable por una computadora (lenguaje de programación) se escribe un *programa.*

```
Comienzo
 Conocer los coeficientes a,b,c
 Calcular discriminante d=b2-4ac
 Calcular r1= (−b+√d)/(2a)
 Calcular r2= (−b−√d)/(2a)
 Informar r1 y r2
Fin
```

f. 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. En otras palabras, aquí se verifica el funcionamiento de la solución propuesta y se detectan los errores que se presenten. La presencia de errores implica pasar a la etapa de depuración.

En el ejemplo, si se efectúa la prueba con la ecuación x^2-3x+2=0 los datos serán: 1,-3,2
Se puede ver que el discriminante d es 1 y las raíces serán r1=l y r2=2.

Pero si se prueba el algoritmo con la ecuación x^2-2x+5=0, se observa que el discriminante es –16 y no se puede aplicar la fórmula resolvente pues no existe solución real para $\sqrt{-16}$.

g. 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.

En el ejemplo inicial se puede observar que la codificación propuesta no funciona en ciertos casos. Se puede mejorarla del siguiente modo:

```
Comienzo
 Conocer los coeficientes a,b,c
 Calcular discriminante d=b2-4ac
 Si d<0 informar "No existen raíces reales"
    sino calcular r1= (–b+√d) /(2a)
         calcular r2= (–b–√d) /(2a)
         informar r1 y r2
Fin
```

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

Acerca de la Elección del Modelo y el Diseño

En la tercer etapa para la resolución de problemas, se propone la elección de un modelo sobre el cual plantear la solución del problema. En la asignatura se trabajará con el modelo que en algorítmica computacional se conoce como programación modular y estructurada. Más adelante se explicará con más detalle este modelo o paradigma.

Por ahora se menciona que dentro de este modelo, la siguiente etapa: el diseño, consiste en plantear *un plan* o *esquema general* de las tareas que deben realizarse para llegar a la solución.

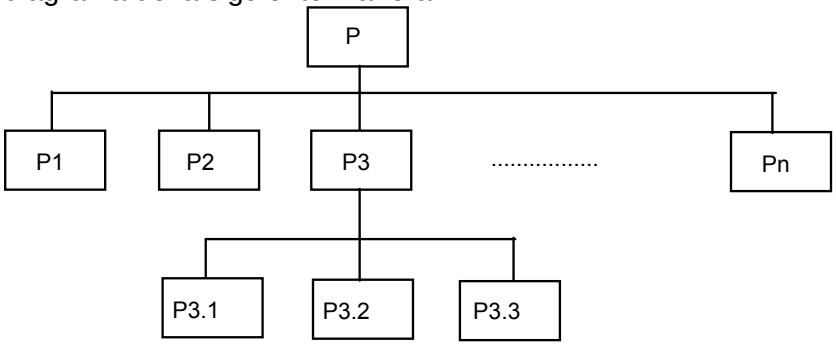
Este esquema general de tareas debe definir QUE HACER. Este conjunto de tareas a realizar recibe el nombre de *división modular*

¿ En qué consiste la división modular ?

La división modular es una estrategia para atacar la complejidad del problema. 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 ello pueda ser comprendido en su totalidad.

Esto, permitirá plantear 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 trivial. Una vez que todos los subproblemas han sido resueltos, podremos afirmar que el problema original ha sido resuelto.

Esquemáticamente esta división modular puede representarse mediante un diagrama de la siguiente manera:



Este proceso de descomposición de un problema en una serie de subproblemas más simples, partiendo de la formulación completa inicial se llama diseño descendente. A menudo se refiere al mismo con otros nombres como desarrollo o diseño top-down, método de refinamiento sucesivos o diseño compuesto.

El diseño de una estrategia y su posterior refinamiento conforman un proceso que requiere de un gran dosis de creatividad y es a su vez, una de las etapas más importantes que encauzará la solución final del problema.

Planteada la división modular indicando que tareas hacer, debemos especificar una lista detallada de como hacerlas, llegando así a definir una solución paso-a-paso del problema. Esta solución así expresada recibe el nombre de algoritmo.

Importante: La división modular tiene sentido en problemas de cierta complejidad y tamaño. En problemas pequeños cuya solución se obtiene en forma casi inmediata o trivial no es necesario su planteo.

Algoritmo

Es común encontrar el término de algoritmo como sinónimo de procedimiento, técnica o método. Pero expresaremos su significado más específicamente.

Definición

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

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.

PRECISION: 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, existen ciertos problemas cuya solución algorítmica no requiere 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, se puede 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. Se puede afirmar que un algoritmo es eficaz si resuelve el problema con un mínimo de recursos y en el menor tiempo posible.

Obsérvese el siguiente caso a resolver mediante el planteo de un algoritmo.

Ejemplo

Problema: Ver una película en DVD.

Elementos (datos): reproductor de DVD, televisor, control remoto del resproductor.

Método y división modular: En problemas pequeños con solución casi trivial no es necesario seleccionar un método ni realizar una división del problema.

Algoritmo:

Comienzo
Encender el televisor.
Pulsar el botón del canal para Video.

Tomar la caja con el DVD.
Sacar el DVD.
Si el DVD tiene la etiqueta hacia abajo
    Darlo vuelta.
Colocar el DVD en la video reproductora.
Seleccionar opción Inicio de película en la interfaz inicial del video.
Mientras continúe la película
    Verla

Fin

El problema del planteo de este tipo de algoritmos, es que el ejecutante debe estar *entrenado* para reconocer y realizar las acciones descriptas. Hay personas que entienden perfectamente lo que es "Seleccionar opción Inicio de película en la interfaz inicial del video", mientras otras no pueden ejecutar tal acción pues desconocen su significado. Los algoritmos computacionales tienen la ventaja de proponer un conjunto de acciones, que cualquier computadora puede ejecutar.

Ejemplo

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.

Condiciones: área triángulo = base * altura /2

Algoritmo

Comienzo

Multiplicar 10 por 5 (Base por altura)

Dividir el resultado del producto anterior 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:

Algoritmo

Comienzo

Conocer base y altura del triángulo.

Multiplicar base por altura.

Dividir el producto anterior por 2.

Informar el valor del cociente.

Fin

Para resolver los problemas siguientes utilice las etapas de resolución de problemas hasta el algoritmo inclusive. Plantee los algoritmos en forma coloquial, del mismo modo que lo haría al indicarle instrucciones a una persona.

Conceptos de Interés

A continuación se definen algunos de los elementos mencionados en las etapas de resolución de problemas.

Procesador o Ejecutante

Es la entidad capaz de comprender un enunciado y ejecutar las tareas descriptas en un algoritmo. El ejecutante de un algoritmo puede ser una persona, o una computadora.

Ambiente

El ejecutante, para poder resolver un problema a través de un algoritmo, requiere contar con los elementos y recursos adecuados. Ese conjunto de recursos para resolver un determinado trabajo se define como *ambiente* del problema.

Acciones y Primitivas

Vimos que el planteo de la solución de un problema, implica la ejecución de un conjunto de pasos que conforman lo que definimos como *algoritmo.* Cada uno de esos pasos se llama ***acción***. La acción es un evento que modifica el ambiente de un algoritmo.

A su vez, decimos que una acción es *primitiva*, si el ejecutante, para llevarla a cabo no requiere de ningún tipo de información adicional.

La acción *no-primitiva* implica que para ser resuelta, el ejecutante requiere que sea descompuesta en acciones primitivas.

Síntesis

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

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).

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.

Se puede 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 se está en presencia de un algoritmo no computacional.

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

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*).

Al ejecutar un algoritmo estamos realizando la prueba del mismo (6ta etapa). Si se detectan errores se deben corregir o depurar (7ma etapa). Por último es importante documentar la solución propuesta (8va etapa).

Actividades

Ejercicios

Resuelva los siguientes problemas aplicando las etapas de resolución estudiadas hasta la codificación (algoritmo) inclusive.

Ejercicio 1.1. Hallar el área de un triángulo rectángulo, conociendo sus tres lados.

Ejercicio 1.2. Del total de estudiantes ingresantes a cierta Universidad se conocen como datos: Nombre y Apellido, Edad, Sexo y Altura. Se desea calcular e informar el promedio de edad de los ingresantes, el promedio de altura, la cantidad de varones y la cantidad de mujeres.

Ejercicio 1.3. Preparar un café instantáneo. Proponga UD. las condiciones y elementos que dispone.

Ejercicio 1.4. 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 1.5. Cambiar el neumático pinchado de un automóvil.

Cuestionario

1.1. Al escribir un programa ¿qué etapa de resolución de problemas se está llevando a cabo?

1.2. Dentro de las características deseables en un algoritmo se habla de finitud. ¿Qué caso puede proponer como ejemplo donde un algoritmo no cumpla con esta característica (algoritmo infinito)?

1.3. ¿Qué inconvenientes puede observar en el planteo de algoritmos usando nuestro lenguaje natural?

1.4. ¿Qué diferencias existen entre acción algorítmica y el de acción primitiva?