Informática

Teoría sobre algoritmia

Definición 1 (Algoritmo) *Un algoritmo es una lista de instrucciones que realizan una descripción precisa y paso a paso de un proceso que garantiza la resolución de cualquier problema que pertenezca a un tipo determinado y que termina luego de un número finito de pasos.*

Características de un algoritmo:

- Precisión
- Eficacia
- Finalización garantizada

Definición 2 (Problema algorítmico) *Se llama problema algorítmico a aquel que puede resolverse a través de un algoritmo.*

Ejemplos:

Problema Algorítmico	Algoritmo
Preparar un pollo frito	*Receta para preparar un pollo frito*
Cambiar la rueda de un auto	*Instrucciones para cambiar la rueda de un auto*
Hacer funcionar un electrodoméstico	*Manual de uso del electrodoméstico*

Escribir todos los números positivos que existen no es un problema algorítmico porque no termina nunca.

Ventajas por las cuales es preferible usar computadoras para resolver algunos problemas:

- Velocidad
- Manejo de gran cantidad de datos
- Son menos proclives a cometer errores

Consideremos el siguiente **problema:** *Resolver un problema con la computadora*

¿Puede proponer un algoritmo para resolver dicho problema?

Algoritmo asociado:

1er paso: *Entender el problema*

2do paso: *Determinar cómo podemos resolver ese problema*

3er paso: *Determinar cómo puede resolverlo la computadora*

4to paso: *Transmitir esa información a la computadora*

5to paso: *Dejar que la computadora actúe*

Si observamos bien el algoritmo propuesto, el 4to paso habla de *"transmitir la información a la computadora"*. Este paso es lo que se conoce con el nombre de **"CODIFICACIÓN"** y para ello se utilizan los llamados **"Lenguajes de Programación"** o **"Lenguajes Algorítmicos"**.

Características de los Lenguajes de Programación:

- Tienen un vocabulario y una sintaxis muy limitada
- El vocabulario contiene sólo aquellos tipos de acciones básicas que puede entender la computadora.
  Ejemplos: Mostrar datos, Leer datos, Asignar un valor a una variable, Realizar una operación aritmética, etc.
- La sintaxis es muy rígida y no permite variaciones de estilo

Existen diversos paradigmas de programación, es decir, distintos estilos de programación. En este curso nos vamos a concentrar en la llamada:

"PROGRAMACIÓN ESTRUCTURADA"

En este paradigma, los algoritmos se construyen utilizando ciertas estructuras básicas:

- **Estructura Secuencial**
- **Estructura Selectiva**
- **Estructuras Repetitivas**

Veamos en detalle cada una de estas estructuras:

Estructura Secuencial:
Es una secuencia de acciones que se realizan en un orden dado que no puede ser alterado.

Ejemplo:

$A \leftarrow 10$
Mostrar(A)

Estructura Selectiva:
Consta de:

- *una condición lógica cuyo valor lógico se chequea*
- *una serie de acciones que se realizan o no según la condición*

Ejemplo:

`Si` $(A < 10)$ `entonces` Mostrar ("El valor es menor a 10")

La forma general es la siguiente:

`Si` CONDICIÓN `entonces` INSTRUCCIONES CUANDO LA CONDICIÓN ES VERDADERA

`sino` INSTRUCCIONES CUANDO LA CONDICIÓN ES FALSA

Estructuras Repetitivas:

a) `Para` VARIABLE DE CONTROL `=` VALOR INICIAL `a` VALOR FINAL `hacer`, `con incremento` VALOR INCREMENTO

INSTRUCCIONES A EJECUTAR PARA CADA VALOR DE LA VARIABLE DE CONTROL

`Fin_Para`

Observación: El valor final debe ser mayor o igual al valor inicial. De lo contrario las instrucciones no se ejecutan

Ejemplo:

```
Para i = 1 a 5 hacer, con incremento 1
    Mostrar('El valor de i es: ', i)
Fin_Para
```

Estas instrucciones mostrarán los siguientes mensajes al usuario:

El valor de i es: 1
El valor de i es: 2
El valor de i es: 3
El valor de i es: 4
El valor de i es: 5

b) `Mientras` CONDICIÓN `hacer`

INSTRUCCIONES A EJECUTAR CADA VEZ QUE LA CONDICIÓN ES VERDADERA

`Fin_Mientras`

Observación: Puede ocurrir que las instrucciones no se ejecuten nunca. Esto ocurre cuando se chequea la condición por primera vez y esta es falsa.

c) `Repetir hacer`
INSTRUCCIONES A EJECUTAR CADA VEZ
ANTES DE CHEQUEAR LA CONDICIÓN
`hasta que` CONDICIÓN

Observación: Las instrucciones se ejecutan al menos una vez.

Veamos cómo podemos escribir un algoritmo. Como dijimos anteriormente, para que una computadora sea capaz de leer, interpretar y ejecutar un algoritmo debemos codificarlo en un lenguaje "comprensible" por las máquinas. Estos lenguajes se denominan **"Lenguajes de Programación"**. A continuación veremos un lenguaje que se utiliza para describir algoritmos, que si bien no es un lenguaje de programación, su sintaxis es muy cercana. Este lenguaje que utilizaremos se denomina **"Pseudo-código"** y los algoritmos escritos en este lenguaje artificial se denominan **"Pseudo-algoritmos"**.

Un algoritmo descripto en este lenguaje comienza con la palabra clave `Algoritmo` seguido del nombre del Algoritmo. Luego, aparece la palabra `var` indicando que en dicho lugar deben declararse, es decir, identificarse todas las variables que se utilizarán en el mismo junto con la clase de datos que pueden guardar. Finalmente, se pueden observar las palabras `Comienzo` y `Fin` que limitan el cuerpo del algoritmo, es decir, encierran todas las instrucciones que lo componen.

Resumiendo, un algoritmo descripto en pseudo-código tendrá para nosotros la siguiente forma:

```
Algoritmo Nombre del Algoritmo
var
   ...
Comienzo
   ...
   ...
   ...
Fin
```

Ejemplo 1:

Considere el siguiente **problema algorítmico:** *"Calcular el promedio de 3 notas"*

Lo primero que debemos hacer es pensar cómo resolveríamos nosotros el problema. Es decir, dar el algoritmo como una descripción paso a paso en lenguaje natural.

Algoritmo (en lenguaje natural)**:**

1er paso: Obtener las 3 notas

2do paso: Sumar las 3 notas

3er paso: Dividir la suma por la cantidad de notas, o sea, dividir por 3

4to paso: Mostrar el resultado

Una vez que sabemos cómo resolver el problema, debemos decirle a la computadora cómo resolverlo. En este caso, debemos traducir el algoritmo del lenguaje natural al lenguaje de programación deseado. Nosotros por el momento trabajaremos en pseudo-código.

Algoritmo (en pseudo-código)**:**

```
Algoritmo PROMEDIO1
var
    N1, N2, N3, S, P: números
Comienzo
    Mostrar('Ingrese la 1er nota:')
    Leer(N1)
    Mostrar('Ingrese la 2da nota:')
    Leer(N2)
    Mostrar('Ingrese la 3er nota:')
    Leer(N3)
    S ← N1 + N2 + N3
    P ← S/3
    Mostrar('El promedio es: ', P)
Fin
```

Ejemplo 2:

Considere ahora, una variación del ejemplo anterior, o sea, el siguiente **problema algorítmico:** *"Calcular el promedio de N notas"*

Algoritmo (en lenguaje natural)**:**

1er paso: Obtener el valor N

2do paso: Obtener las N notas

3er paso: Sumar las N notas

4to paso: Dividir la suma por N

5to paso: Mostrar el resultado

Una vez que sabemos cómo resolver el problema, debemos describir el pseudo-algoritmo.

Algoritmo (en pseudo-código)**:**

```
Algoritmo PROMEDIO2
var
    N, NOTA, i, S, P: números
Comienzo
    Mostrar('Cuántas notas desea utilizar para
calcular el promedio?')
    Leer(N)
    S ← 0
    Para i = 1 a N hacer, con incremento 1
        Mostrar('Ingrese una nota: ')
        Leer(NOTA)
        S ← S + NOTA
    Fin_Para
    P ← S/N
    Mostrar('El promedio es: ', P)
Fin
```

Una vez que contamos con el algoritmo descripto en pseudo-código, debemos probar si funciona. Como el pseudo-código no es comprensible directamente por una computadora, debemos probarlo nosotros mismos. Es por esto que, antes de continuar, vamos a introducir un concepto más, el concepto de **"Traza de un algoritmo"**.

Definición 3 (Traza de un Algoritmo) *La traza de un algoritmo se puede definir como la ejecución manual de forma secuencial de las sentencias que lo componen. Así, una traza de un algoritmo muestra los valores que van adoptando las variables a medida que se va ejecutando el mismo.*

Supongamos ahora que queremos calcular la traza del algoritmo "PROMEDIO2" usando como entradas las notas 3, 7 y 8.

Mostramos la traza de un algoritmo usando una tabla, donde cada columna está asociada a una variable. Mientras vamos avanzando en la ejecución paso a paso del algoritmo, las variables van tomando distintos valores. Mostramos a continuación los distintos valores que van asumiendo las variables del algoritmo "PROMEDIO2", o sea, su traza para dichas entradas:

N	**NOTA**	**i**	**S**	**P**
3	3	1	0	6
	7	2	3	
	8	3	10	
		4	18	

En la traza, podemos observar que el resultado es 6, que coincide con el promedio de las notas consideradas. Por lo tanto, (al menos para los datos propuestos) el algoritmo parece funcionar.

Vamos ahora a proponer dos algoritmos equivalentes a PROMEDIO2 que reemplazan la estructura de repetición **Para** por las estructuras de repetición **Mientras** y **Repetir**.

El algoritmo PROMEDIO3 es equivalente a PROMEDIO2 y utiliza la estructura Mientras:

```
Algoritmo PROMEDIO3
var
    N, NOTA, i, S, P: números
Comienzo
    Mostrar('Cuántas notas desea utilizar para
calcular el promedio?')
    Leer(N)
    i ← 1
    S ← 0
    Mientras (i ≤ N) hacer
        Mostrar('Ingrese una nota: ')
        Leer(NOTA)
        S ← S + NOTA
        i ← i + 1
    Fin_Mientras
    P ← S/N
    Mostrar('El promedio es: ', P)
Fin
```

El algoritmo PROMEDIO4 es equivalente a PROMEDIO2 y utiliza la estructura Repetir:

```
Algoritmo PROMEDIO4
var
    N, NOTA, i, S, P: números
Comienzo
    Mostrar('Cuántas notas desea utilizar para
calcular el promedio?')
    Leer(N)
    i ← 1
    S ← 0
    Repetir
        Mostrar('Ingrese una nota: ')
        Leer(NOTA)
        S ← S + NOTA
        i ← i + 1
    hasta que (i > N)
    P ← S/N
    Mostrar('El promedio es: ', P)
Fin
```

Dos algoritmos son equivalentes cuando al ejecutarse con las mismas entradas producen resultados similares. Calcule las trazas de los algoritmos PROMEDIO3 y PROMEDIO4 con las notas 3, 7 y 8 para verificar la equivalencia mencionada anteriormente.

Otra manera de describir algoritmos es mediante la utilización de **"Diagramas de Flujo"**. Ahora bien, ¿Qué es un Diagrama de flujo?

Definición 4 (Diagrama de Flujo) *Un diagrama de flujo es una descripción gráfica de un algoritmo. Está conformado por figuras conectadas por flechas.*

Para ejecutar un algoritmo representado por un diagrama de flujo se comienza por la figura que representa el **Inicio** y se siguen las flechas, de figura a figura, ejecutándose las instrucciones indicadas por cada figura hasta llegar a la figura que representa el **Fin** del algoritmo. El tipo de figura indica el tipo de instrucción que representa.

Existen numerosas herramientas que nos ayudan a diseñar y ejecutar diagramas de flujo. Una de dichas herramientas es **DFD** (denominada así por la traducción al inglés de diagrama de flujo de datos: "Data Flow Diagram").

Todos los algoritmos propuestos en este apunte pueden ser representados por un diagrama de flujo en DFD y ejecutados automáticamente usando una computadora. Para ver las equivalencias entre las estructuras básicas vistas hasta ahora y las figuras (objetos) disponibles en DFD, consulte el Manual de Referencia de DFD.