Óptica Computacional
Introducción a los Algoritmos

CARRERAS DE:

Licenciatura en Óptica y Contactología.
Tecnicatura Universitaria en Óptica.
Departamento de Física - UNS

ALGORITMOS

Un algoritmo es una secuencia de Acciones que la computadora debe Ejecutar a fin de Resolver un Problema

Es posible demostrar que la resolución de cualquier problema puede reducirse, utilizando sólo las siguientes estructuras algorítmicas: Estructura Secuencial, Estructura Condicional y Estructura Repetitiva.

Estructura Secuencial

El programa ejecuta **Acciones** en forma **Secuencial**
Una Después de Otra

<Acción 1> → <Acción 2> → <Acción 3> → … →

La solución del problema consiste en ir resolviendo etapas intermedias, en forma consecutiva, **siguiendo un orden lógico.**

Ejemplo: Supongamos que deseamos sumar dos números, A y B, cuyos valores se ingresan por teclado, e imprimir el resultado en la pantalla.
¿Cuál es la secuencia lógica para llevar a cabo esta tarea?

Acciones a realizar:

Mostrar Resultado
Sumar A y B
Ingresar valor de A
Ingresar valor de B

Si bien estas son las acciones correctas, el orden en el que están planteadas es incorrecto

La máquina no puede mostrar el resultado hasta que no haga la suma, y no puede ejecutar la suma si no tiene los datos de los valores a sumar!!!

Acciones a realizar en el orden correcto:

Ingresar valor de A
Ingresar valor de B
Sumar A y B
Mostrar Resultado

Programa SciLab: ***SUME*** *dos números,* ***A y B,*** *cuyos valores deben ser* ***INGRESADOS POR TECLADO*** *e* ***IMPRIMA*** *el resultado.*

```
+++++++ PROGRAMA PARA SUMAR DOS NÚMEROS A Y B ++++++++++

// Ingresamos los valores de A y de B por teclado

A=input("ingrese el valor de A. A= ");
B=input("ingrese el valor de B. B= ");

// ya tenemos los valores de A y de B, ahora los sumamos

res=A+B;

//en la variable "res" se guarda el resultado
// ahora mostramos el resultado en pantalla

 disp(res,"la suma A y B es. A+B= ");

//++++++++++++++++++++ FIN ++++++++++++++++++++++++++++
```

+++ Otro Ejemplo:+++ *Programa SciLab para* ***DIVIDIR*** *dos números,* ***A/B,*** *cuyos valores deben ser ingresados por teclado y el resultado, impreso en pantalla*

```
// +++++ PROGRAMA PARA SUMAR DOS NÚMEROS A Y B ++++++++++

// Sentencia para ingresar los valores de A y de B por teclado

A=input("ingrese el valor deL DIVIDENDO. A= ");
B=input("ingrese el valor de DIVISOR.  B= ");

// ya tenemos los valores de A y de B, ahora los dividimos

res=A/B;

// ahora mostramos el resultado en pantalla

 disp(res,"la división entre A y B es. A/B= ");

//++++++++++++++++++++ FIN ++++++++++++++++++++++++++++
```

En el ejemplo anterior podemos llegar a tener un problema

Si B = 0 → A/B = inf

A veces es conveniente anticiparse para que esto no ocurra:

La idea es: en caso que → B=0 → No Ejecutar la División → Mostrar Leyenda "*Error, el valor ingresado del Divisor es cero, vuelva a intentarlo*".

ESTRUCTURA CONDICIONAL

ESTRUCTURA CONDICIONAL

```
Si <condición> Entonces
    Acción 1
Sino
    Acción 2
FinSi
```

ALGORITMO

```
Ingresar valor de A;
Ingresar valor de B;
Si B es igual a 0
    Imprimir mensaje de Error;
Sino
    Dividir A por B;
    Imprimir Resultado;
FinSi
```

ESTRUCTURA CONDICIONAL SIMPLE

TOMA DE DECISIÓN

```
Si  <condición>  Entonces
      Acción 1
      Acción 2
      Acción 3
FinSi
```

Un condicional simple permite tomar la decisión de **"Ejecutar una Tarea"** o **"No ejecutar una Tarea"** en base al resultado **"Verdadero"** o **"Falso"** de la **<condición>**

ESTRUCTURA CONDICIONAL SIMPLE

*+++ **Ejemplo** +++ Un almacén hace un 20% de descuento a los clientes cuya compra supere los $5000 ¿cuál será la cantidad que pagará un cliente por su compra?*

¿Qué dato necesitamos? Uno diría que el importe de la compra, sin embargo Si el importe es > $5000 tendrá un descuento y terminará pagando un 20% menos sobre ese valor

Esta acción se realiza solo en un caso específico

```
Leer Importe;
Si  Importe > 5000
      Importe_Nuevo = Importe*0.8
      Importe = Importe_Nuevo;
FinSi
Imprimir Importe
```

+++ Otro Ejemplo +++ Realice un algoritmo para que, dados dos números, A y B, si el primero es mayor que el segundo, los reste.

```
Leer A; Leer B;
Si  A > B
     Imprimir A-B
FinSi
```

ESTRUCTURA CONDICIONAL ANIDADOS

```
Si  <condición 1>
Entonces
      Acción(es)
 Sino
      Si  <condición 2>
      Entonces
          Acción(es)
      Sino
          Acción (es) o Varias Condiciones
      FinSi
FinSi
```

+++ Ejemplo +++ Se carga una planilla de datos con la información del curso de ingreso de Física. En la planilla Figuran los siguientes datos: Nombre del Alumno, Asistencia (Presente o Ausente), y si estuvo presente la Nota del examen. Se desea hacer un programa que lea los datos de la planilla y en caso de que el alumno haya estado presente determine si aprobó o no el examen y que imprima **APROBADO o DESAPROBADO** a la derecha del nombre según sea el caso. Aprueba si la Nota es **Mayor o Igual a 60**, sino Desaprueba

Datos:

Nombre Asistencia Nota

A Ordenar las Ideassss

```
1ro  → Leer el Nombre del Alumno
2do  → Leer Asistencia
3ro  → Acá depende…
       Si Asistencia = Presente
          Leer Nota
          Si Nota > 59
             Imprimir (Nombre, Aprobado)
          Sino
             Imprimir (Nombre, Desaprobado)
          FinSi
       Sino
          Imprimir (Nombre, Ausente)
       FinSi
```

Estructura Repetitiva

En este caso la Solución consiste en:

1- Repetición de una Acción más Sencilla Mientras se cumple una dada condición

2- Repetición de una Acción más Sencilla un Número Finito de veces

Estructura Repetitiva: Mientras(While)

```
Mientras <condición>  Hacer
      Acción(es)
FinMientras
```

La evaluación de la condición se realiza antes de ejecutar cada iteración

Si la condición no se cumple la primera vez, el ciclo no se ejecutará nunca

```
Leer x;
suma = valor_inicial
Mientras x > 0 hacer
      suma = suma + x
EndMientras
Imprimir suma
```

Estructura Repetitiva: Mientras(While)

En este tipo de ciclos hay que asegurarse que éste contenga una sentencia de finalización, caso contrario el programa podría quedar iterando en un ciclo sin fin.
Es la principal causa de que un Programa "Se Cuelgue"

Ejemplo de Algoritmo mal condicionado

```
N=1
Mientras N >0 hacer
        N=N+1
EndMientras
Imprimir N
```

Algoritmo correctamente condicionado

```
N=1
while N>0 do
      N=N+1
      if N==10000 then
            break
      end
      disp(N);
end
```

Estructura Repetitiva

- `while B: <cuerpo>`
- `B` es una expresión booleana.
- También se la llama **guarda**
- `<cuerpo>` puede ser una sola instrucción o un bloque de instrucciones
  - Se repite mientras `B` devuelva `True`
  - Puede ejecutarse cero o más veces
  - Si es una cantidad finita, el programa termina
  - Si se ejecuta al menos una vez, alguna de las ejecuciones del cuerpo tienen que hacer que `B` sea `False`.
  - El estado final de esa ejecución (cuando `B` es `False`) será el estado final del ciclo.
- La semántica que vimos depende de la suposición de que el ciclo termina.
- Es una suposición **muy** fuerte.
- Normalmente, los ciclos son los responsables de que un programa se *cuelgue*.
- Si queremos escribir programas correctos, tenemos que garantizar que todos sus ciclos terminen.

Estructura Repetitiva: Contador (for)

2- Repetición de una Acción más Sencilla un Número Finito de veces

```
Para <contador>
      Acción(es)
FinPara
```

Se usa cuando se conoce de antemano el número de iteraciones (ciclos), o bien éste es un parámetro que queda definido en pasos anteriores del código

Estructura Repetitiva: Contador (for)

+++ Ejemplo +++ Leer 100 números y sumarlos.

Sabemos que el total de sumas es 100, por lo que en vez de hacer la suma de una vez, conviene hacer 100 sumas simples usando un ciclo repetitivo.
En cada ciclo se agrega un término a la suma.
Se parte del valor inicial de suma (puede ser 0), se lee el siguiente valor y se agrega a la suma. Se repite tantas veces cómo sea necesario

```
suma = valor_inicial
Para n=1 hasta 100 hacer
      Leer x;
      suma = suma+x
EndMientras
Imprimir suma
```

valor inicial de suma (puede ser o no ser 0)

1 y 100 son el valor inicial y final del contador

n → Contador: lleva la cuenta del número de iteraciones (del núm. de veces que se ejecutó el ciclo)

Estructura Repetitiva: Contador (for)

+++ Ejemplo +++ Código SciLab para Leer 10 números y sumarlos.

```
x=rand(10,1)*100;
suma=0;
for n=1:10 do
    num=x(n);
    suma=suma+num;
    A=[num suma]
    disp(A," num suma ");
end
disp(x," valores a sumar ")
disp(suma);
```

```
 valores a sumar


  83.104313
  1.221633
  48.844617
  95.498771
  5.8743121
  82.584649
  29.807416
  7.7575968
  58.460923
  75.287136


--> disp(suma);

  488.44137
```

Estructura Repetitiva: Contador (for)

+++ Ejemplo +++ Código SciLab para Leer N números y sumarlos, donde la cantidad N es definida por el usuario. Los valores a sumar son determinados al azar

```
N=input("cantidad de números a sumar")
x=rand(N,1)*10;
suma=0;
for i=1:N do
   num=x(i);
   suma=suma+num;
end
disp(x," valores a sumar ");
disp(suma);
```

Asignación:

- Es la instrucción *fundamental* y que caracteriza a los programas imperativos.
- Es la única instrucción que permite modificar el valor de una variable.
- Los procedimientos y las funciones también modifican el valor de una variable, pero en definitiva lo hacen mediante asignaciones.
- Tiene esta sintaxis:
  `<nombre_de_variable> = <expresión que denota un valor>` la `expresión que denota un valor` puede ser una constante, otra variable, o bien el resultado de una función (no de un procedimiento).

Algunos detalles de la asignación

- La asignación es una operación que, aunque contiene el símbolo $=$, es asimétrica.
- A la izquierda del $=$ tiene que ir el nombre de la variable.
- A la derecha va la expresión que denota un valor.
- ¿Qué pasa si hacemos `3=7`? `SyntaxError: can't assign to literal`, estamos pidiendo cambiar el valor a algo que **no** es una variable.
- Para remarcar la característica asimétrica de la asignación, hay lenguajes imperativos que usan `:=` en vez del símbolo `=`.
- La asignación tiene un único efecto: provocar que la parte izquierda tome el valor de la expresión de la derecha.
- La asignación provoca que en las posiciones de memoria correspondientes a la variable de la izquierda se escriba o almacene la expresión (o valor) de la derecha.
- La asignación no tiene ninguna otra acción adicional.

```
// +++++++++++++++ SEPARANDO DÍGITOS DE UN NÚMERO ++++++++++++++++++
//************************************************************
//Este es un algoritmo para sumar los dígitos de un número dado y averiguar si suma=10
//Si se cumple esta premisa imprimir el número en pantalla (consola) con un texto
////**********************************************************************

x=[100:999]';    //número a evaluar

//***************** Separación de los dígitos ***********************
a=int(x./100);       //selecciono el dígito correspondiente a la centena
b1=(modulo(x,100));  //calculo el resto de dividir el num x por 100
                     // me quedo con las decenas...
b=int(b1./10);       //divido las decenas por 10 y calculo la parte entera

//así selecciona el dígito correspondiente a la decena c=modulo(b1,10);
//el resto de dividir las decenas por 10 es la unidad

//*************** Suma de los Dígitos ****************************
a3=a^3;
b3=b^3;
c3=c^3; //calcula el cubo de cada dígito
suma=a3+b3+c3;          //suma los cubos de los dígitos del número x

//************************CICLO FOR *****************************
//******** para "barrer" el rango de números entre 100 y 999 ******

for n=1:900  //a medida que el "ciclo for" pasa por los n de 1 a 900 se va
            //seleccionando cada elemento de los vectores que "x" y "suma" que
     if suma(n)==x(n) then    //evalúa si la suma de los cubos de cada dígito es igual a x
     disp(x(n))
     end                     //end del ciclo if
end                          //end del ciclo for
```

Ejercicio de la Guía:

Elabore un algoritmo que imprima los números naturales de tres dígitos ($100 \leq x < 1000$) que sean iguales a las sumas de los cubos de sus dígitos

```
x=[100:999]';   ← genero los número naturales de 3 dígitos

a=int(x./100);
b1=int(modulo(x,100));
b=int(b1./10);
c=modulo(b1,10);
```

selecciono cada uno de los dígitos del número

```
a3=a^3; b3=b^3; c3=c^3; //calcula el cubo de cada dígito
suma=a3+b3+c3; //suma los cubos de los dígitos de x

for n=1:900
    if suma(n)==x(n) then //evalúa si la suma de los cubos de cada
        disp(x(n))        //dígito es igual al número x
    end
end
```