Estructuras Secuenciales

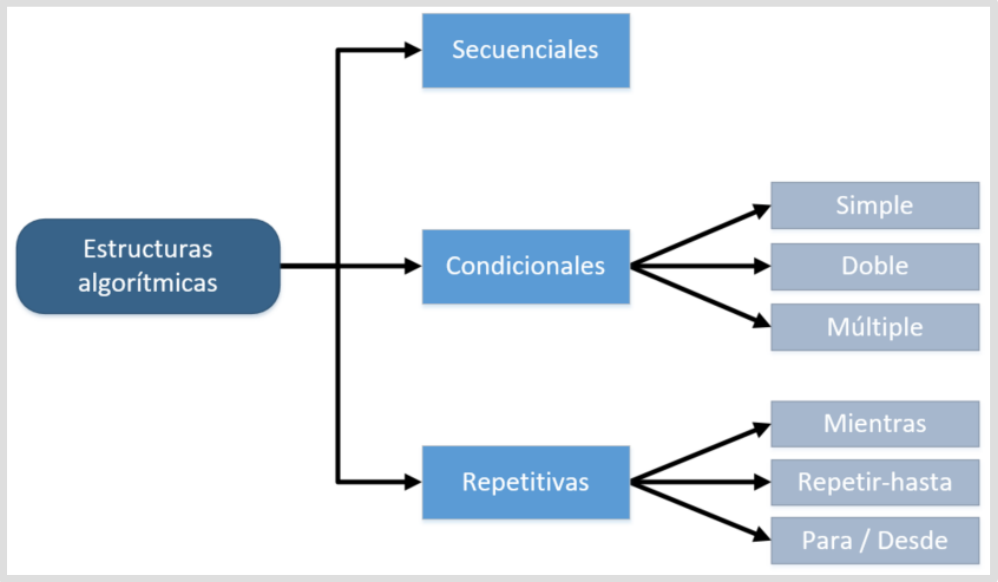
Programación Lógica

UNIVERSIDAD SIGLO 21 | MIEMBRO DE LA RED ILUMNO

1. Estructuras secuenciales

El diseño de un algoritmo debe ser fácil de comprender para facilitar su posterior codificación. La programación estructurada es un paradigma que tiene por propósito escribir programas en forma clara y precisa utilizando únicamente tres estructuras de programación: secuenciales, condicionales y repetitivas. El teorema del programa estructurado indica que todo diseño de algoritmo puede ser implementado utilizando estas estructuras de programación.

Figura 1: Estructuras algorítmicas



Fuente: elaboración propia.

Las estructuras secuenciales son aquellas en las que las acciones se ejecutan una a continuación de la otra, en secuencia. Por ejemplo:

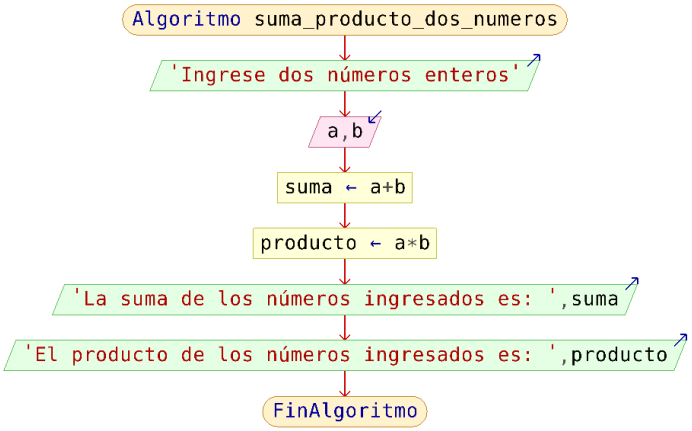
leer(A)
leer(B)
suma ← *A + B*
mostrar(suma)

Las acciones anteriormente indicadas se ejecutan cuando la anterior ha finalizado. Es decir, la acción *leer(B)* se ejecutará luego de que finalice la acción *leer(A)*, la acción *suma* ← *A + B* se ejecutará una vez finalizada la acción *leer(B),* y así sucesivamente.

Ejercicios de aplicación

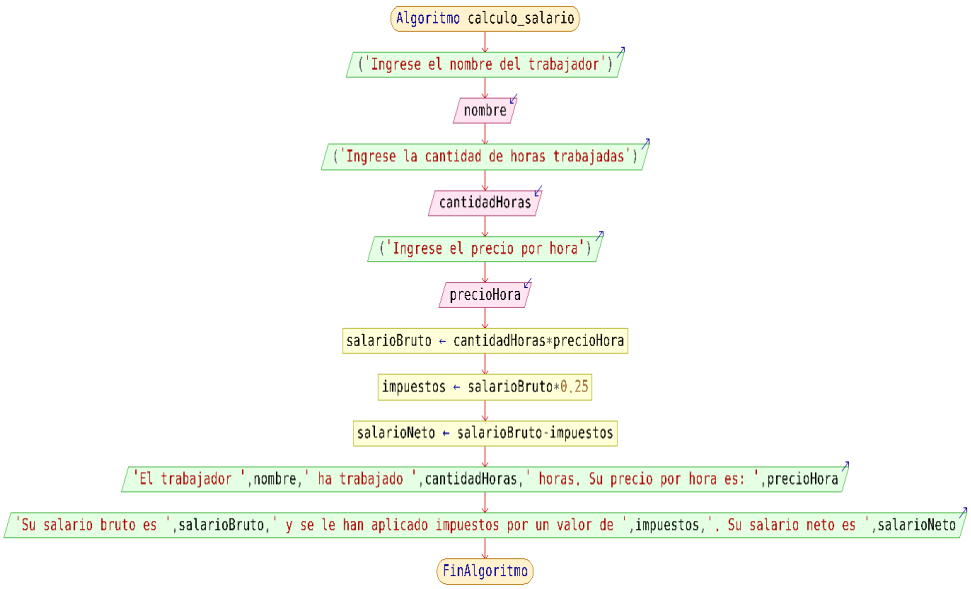
1) Implementar un algoritmo que permita calcular y mostrar por pantalla la suma y el producto de dos números enteros ingresados por teclado.

```
algoritmo suma_producto_dos_numeros
var
    entero: a, b, suma, producto
inicio
    mostrar("Ingrese dos números enteros")
    leer(a, b)
    suma ← a +b
    producto ← a * b
    mostrar("La suma de los números ingresados es: ", suma)
    mostrar("El producto de los números ingresados es: ", producto)
fin
```



2) Implementar un algoritmo que permita calcular el salario neto y bruto de un trabajador conociendo los siguientes datos que se ingresan por teclado: nombre, número de horas trabajadas, precio por hora del trabajador. El salario bruto está determinado por la multiplicación del número de horas trabajadas y el precio de cada hora. Para obtener el salario neto, se debe deducir el 25 % del salario bruto.

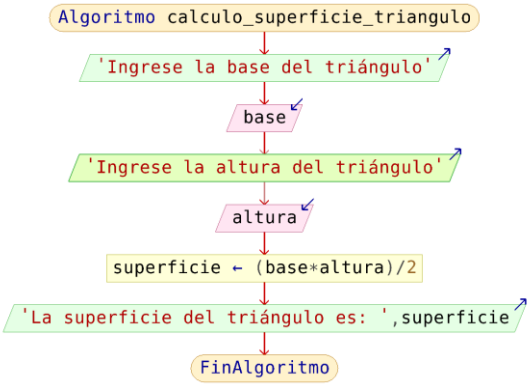
algoritmo calculo_salario
var
entero: cantidadhoras
real: precioHora, salarioBruto, salarioNeto, impuestos
cadena: nombre
inicio
mostrar("Ingrese el nombre del trabajador")
leer(nombre)
mostrar("Ingrese la cantidad de horas trabajadas")
leer(cantidadHoras)
mostrar("Ingrese el precio por hora")
leer(precioHora)
*salarioBruto ← cantidadHoras * precioHora*
*impuestos ← salarioBruto * 0,25*
salarioNeto ← salarioBruto - impuestos
mostrar("El trabajador ", nombre, " ha trabajado ", cantidadHoras, " horas. Su precio por hora es: ", precioHora)
mostrar("Su salario bruto es " , salarioBruto, " y se le han aplicado impuestos por un valor de ", impuestos, ". Su salario neto es ", salarioNeto)
fin



3) Implementar un algoritmo que permita calcular y mostrar la superficie de un triángulo teniendo en cuenta que las dimensiones de su altura y base se ingresan por teclado.

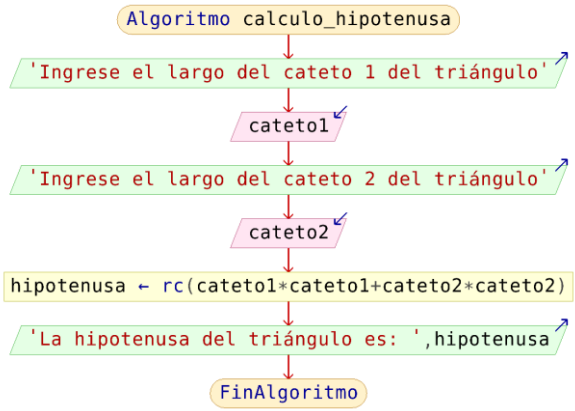
algoritmo calculo_superficie_triangulo
var
entero: base, altura
real: superficie

inicio
mostrar(“Ingrese la base del triángulo”)
leer(base)
mostrar(“Ingrese la altura del triángulo”)
leer(altura)
*superficie ← (base * altura) / 2*
mostrar(“La superficie del triángulo es: ”, superficie)
fin



4) Implementar un algoritmo que permita calcular la hipotenusa de un triángulo rectángulo teniendo en cuenta que se ingresan por teclado los valores de sus catetos.

algoritmo calculo_hipotenusa
var
real: cateto1, cateto2, hipotenusa
inicio
mostrar("Ingrese el largo del cateto 1 del triángulo")
leer(cateto1)
mostrar("Ingrese el largo del cateto 2 del triángulo")
leer(cateto2)
*hipotenusa ← rc(cateto1*cateto1 + cateto2*cateto2) // se utiliza una función para // calcular la raíz cuadrada*
mostrar("La hipotenusa del triángulo es: ", hipotenusa)
fin


Algoritmo calculo_hipotenusa
'Ingrese el largo del cateto 1 del triángulo'
cateto1
'Ingrese el largo del cateto 2 del triángulo'
cateto2
hipotenusa ← rc(cateto1*cateto1+cateto2*cateto2)
'La hipotenusa del triángulo es: ',hipotenusa
FinAlgoritmo