



# FUNCIONES O MÉTODOS

Algoritmia y Programación

# CONTENIDO

- Funciones/métodos
  - Definición
  - Partes de una función
  - Ejemplos
- Ámbito de variables
  - Variables Locales
  - Variables Globales

# DEFINICIÓN

Una función es una porción de código (sub-algoritmo) que forma parte de un algoritmo principal, el cual se encarga de resolver una tarea específica.

# DEFINICIÓN

Una función es una porción de código (sub-algoritmo) que forma parte de un algoritmo principal, el cual se encarga de resolver una tarea específica.

Divide y vencerás: Es una estrategia para resolver problemas grandes dividiéndolo en problemas más pequeños



# DEFINICIÓN

Los métodos reciben datos del programa principal, realizan operaciones y le devuelven los resultados.

Es como una oficina: El programa principal es el jefe, que da instrucciones a sus subordinados (métodos),

ellos realizan una tarea, y cuando terminan le devuelven el resultado y control al jefe



# VENTAJAS FUNCIONES

DE USAR

- Los problemas pequeños son más fáciles de entender, de desarrollar y de mantener (localizar errores).
  
- Se evita código innecesario, pues los métodos se escriben una sola vez, y pueden ser utilizados (llamados) desde diferentes partes del programa, las veces que sea necesario

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos ( $a,b$ ) y determine cual de los 2 es mayor y muestre su hipotenusa.

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** ??

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** determinar mayor de 2 números

calcular hipotenusa

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** determinarMayor(a, b :entero) :

a y b son datos necesarios para calcular el mayor de los dos números

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** determinarMayor(a, b :float) :

    m: float

**SI** (a>b )

        m= a

**SINO**

        m=b

    retornar m

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** calcularHipotenusa(a, b :float) :

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** calcularHipotenusa(a, b :entero) :

a y b son datos necesarios para calcular la hipotenusa

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** calcularHipotenusa(a, b :entero) :

    h: real

$$h=\sqrt{a^2+b^2}$$

    retornar h

# PROBLEMA

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa.

## 1. Análisis del problema

**Entrada:** a, b

**Salidas :** hipotenusa, mayor

**Proceso:** calcularHipotenusa(a, b :entero) :

h: real

$$h = \sqrt{a^2 + b^2}$$

retornar h

¿Cómo se escribe un  
método en Ruby?

# LLAMADO DE MÉTODOS

- Desarrolle un programa que lea los datos de 2 catetos (a,b) y determine cual de los 2 es mayor y muestre su hipotenusa

2. Diseñar el algoritmo y escribirlo en pseudocódigo

**Inicio**

**a,b,mayor : entero**  
**hipotenusa : real**

**a = leer ("digite el valor de a")**  
**b = leer ("digite el valor de b")**

**hipotenusa = calcularHipotenusa(a,b)**  
**mayor = determinarMayor(a,b)**

**imprimir(hipotenusa, mayor)**

**Fin**

**La variable hipotenusa va a tomar el valor que le devuelva el método calcularHipotenusa(a,b)**

# FUNCIONES: EJEMPLO

```
def calcularHipotenusa(a, b)
    h = Math.sqrt( (a*a) + (b*b) )
    return h
end

def calcularMayor(a,b)
    if (a > b)
        m = a
    else
        m = b
    end
    return m
end
```

a, b sólo existen dentro de la función  
h sólo existe dentro de la función calcularHipotenusa

a, b sólo existen dentro de calcularMayor  
al igual m

## Parámetros / argumentos

Hay funciones que reciben algunos valores para poder hacer operaciones con ellos. Si son varios se separan con comas .

**Return:** Las funciones *pueden* devolver un valor

# FUNCIONES: EJEMPLO

```
puts "Ingrese el valor del cateto a"  
catetoA = gets.chomp.to_f  
  
puts "Ingrese el valor del cateto b"  
catetoB = gets.chomp.to_f  
  
puts calcularMayor(catetoA, catetoB)
```

Llamando a la función  
a la función  
calcularMayor (catetoA y  
catetoB)

```
puts "Ingrese dos números"  
numeroA = gets.chomp.to_f  
numeroB = gets.chomp.to_f  
mayor = calcularMayor(numeroA, numeroB)  
  
puts "El mayor entre #{numeroA} y #{numeroB} es #{mayor}"
```

Ver 3)Funciones/hipotenusa.rb

# FUNCIONES: EJEMPLO

## 1. Declarando las funciones

```
def suma(x, y)
    return x + y
end

def resta()
    puts "Digite dos números"
    o1 = gets.chomp.to_f
    o2 = gets.chomp.to_f
    resta = o1 - o2
    puts "Resta #{resta}"
end
```

- Las funciones pueden o no *retornar* un valor
- Las funciones pueden o no recibir *parámetros*

# Funciones: Ejemplo

## 1. Declarando las funciones

```
def suma (x,y)
    return x+y
end

def resta()
    o1 = 20
    o2 = 3
    resta = o1-o2
    puts "Resta #{resta}"
end

def multiplicacion(x,y)
    m = x * y;
    puts "x * y = #{m}"
end
```

```
def division(x, y)
    if y == 0
        puts " Error ... "
    else
        divi = x / y
        puts "Division #{divi}"
    end
end
```

# Funciones: Ejemplo

## 2. Llamando a las funciones

```
sum = suma(5,2)
puts ("Suma: #{sum}")

resta()
multiplicacion(20, 14)
division(10, 2)
```

**Ver ejemplo funciones.rb**

# EJERCICIO

- Se desea reforestar un bosque que mide un número  $n$  de hectáreas. Si la superficie del terreno excede a  $1'000.000\text{m}^2$ , entonces se siembra así:

% superficie	Tipo de árbol
70%	Pino
20%	Oyamel
10%	Cedro

Si la superficie del terreno es menor o igual a  $1'000.000\text{m}^2$  entonces se siembra así:

% superficie	Tipo de árbol
50%	Pino
30%	Oyamel
20%	Cedro

Se desea saber el número de pinos, oyameles y cedros que se deben sembrar, si se sabe que en  $10\text{m}^2$  caben 8 pinos; en  $15\text{m}^2$  caben 15 oyameles, y en  $18\text{m}^2$  caben 10 cedros.

# EJEMPLO

## 1. Análisis del problema

**Entrada:** cant\_h

**Salidas :** cant\_pinos, cant\_oyamel, cant\_cedro

**Proceso:** hallarCantidades

# EJEMPLO

## 1. Diseñar el algoritmo y escribirlo en pseudocódigo

### Inicio

cant\_h: **entero**

cant\_h = **leer**("digite la cantidad de hectáreas")

hallarCantidades( cant\_h )

### Fin

# EJEMPLO

## 1. Diseñar el algoritmo y escribirlo en pseudocódigo

```
hallarCantidades(cant_h: entero){
```

```
    cant_pino,cant_oyamel,cant_cedro, hectarea=10000 : entero
```

```
    SI (cant_h *hectarea) > 1000000
```

```
        cant_pino= (((cant_h *hectarea) *0.7)/10)*8
```

```
        cant_oyamel = (((cant_h *hectarea) *0.2)/15)*15
```

```
        cant_cedro= (((cant_h *hectarea) *0.1)/18)*10
```

```
SINO
```

```
        cant_pino= (((cant_h *hectarea) *0.5)/10)*8
```

```
        cant_oyamel = (((cant_h *hectarea) *0.3)/15)*15
```

```
        cant_cedro= (((cant_h *hectarea) *0.2)/18)*10
```

```
    Imprimir (cant_pino, cant_oyamel, cant_cedro);
```

```
}
```

# EJEMPLO1. Declaración de la función

```
def hallarCantidades(cant_h)
    hectarea=10000
    if ( (cant_h *hectarea) > 1000000)
        cant_pino= (((cant_h *hectarea) *0.7)/10)*8
        cant_oyamel = (((cant_h *hectarea) *0.2)/15)*15
        cant_cedro= (((cant_h *hectarea) *0.1)/18)*10
    else
        cant_pino= (((cant_h *hectarea) *0.5)/10)*8
        cant_oyamel = (((cant_h *hectarea) *0.3)/15)*15
        cant_cedro= (((cant_h *hectarea) *0.2)/18)*10
    end
    puts "cantidad de Pinos #{cant_pino} cantidad de Oyamel #{cant_oyamel}
    cantidad de Cedro #{cant_cedro}"
end
```

# EJEMPLO :2. Llamando la función

```
puts "Ingrese el número de hectáreas:"  
cant_h = gets.chomp.to_i  
  
hallarCantidades(cant_h)
```

**Ver ejemplo reforestacion.rb**

# PROBLEMA

- Una casa de cambio vende 3 tipos de divisas diferentes (dólares \$3080, euros \$3469, libras \$4748). La empresa necesita desarrollar un programa que dé 2 opciones a los clientes: 1. Consulta los precios de cada divisa, y 2. Calcular el valor de la conversión del peso colombiano a la divisa.

Desarrolle un algoritmo que dé la solución al problema antes planteado

## 1. Análisis del problema

**Entrada:** opción (1-consulta, 2-calcular), valor\_a\_convertir

**Salidas :** valor\_de\_divisas valor\_convertido

**Proceso:** ??

# PROBLEMA

- Una casa de cambio vende 3 tipos de divisas diferentes (dólares \$3080, euros \$3469, libras \$4748). La empresa necesita desarrollar un programa que dé 2 opciones a los clientes: 1. Consulta los precios de cada divisa, y 2. Calcular el valor de la conversión del peso colombiano a la divisa.

Desarrolle un algoritmo que dé la solución al problema antes planteado

# PROBLEMA

- Una casa de cambio vende 3 tipos de divisas diferentes (dólares \$3080, euros \$ 3469, libras \$4748). La empresa necesita desarrollar un programa que dé 2 opciones a los clientes: 1. Consulta de precios de las divisas, y 2. el valor de la conversión del peso colombiano a la divisa.
- Desarrolle un algoritmo que dé la solución al problema antes planteado

## 1. Análisis del problema

**Entrada:** opción (1-consulta, 2-cambio), tipo\_divisa, valor\_a\_convertir

**Salidas :** valor\_convertido, valor\_de\_divisas

**Proceso:** mostrar\_precios\_divisas

**calcular\_conversión\_divisa**

# ÁMBITO DE VARIABLES

El ámbito de una variable es el contexto (espacio) dentro del programa en donde ella puede ser utilizada y reconocida.

# VARIABLES GLOBALES Y LOCALES

Las variables se clasifican en 2 tipos:

- Variables Locales
  - Variables Globales
- 
- **Variables Locales:** Son aquellas variables que se crean dentro de una función. Estas variables solamente son reconocidas dentro de esa función donde fueron creadas, y su valor se pierde al finalizar la ejecución de la misma.

# VARIABLES GLOBALES Y LOCALES

▪ **Variables Globales:** Son aquellas variables que son reconocidas en todo lugar del programa, estas pueden ser asignadas fuera de cualquier método, en el programa principal o se pueden referenciar dentro de un método en Ruby con la palabra reservada \$.

Ejemplo:

```
def metodo()
    $var = 5
end
```

# VARIABLES GLOBALES Y LOCALES

x,y : entero

c : entero

Unicamente existe dentro  
de la función

Ámbito de c

w,p : real

Ámbito de w,p

Unicamente viven  
dentro de la  
función

Ambito de x,y

# VARIABLES GLOBALES Y LOCALES

```
def calcularHipotenusa(a, b)
    h = Math.sqrt( (a*a) + (b*b) )
    return h
end
```

```
hip = calcularHipotenusa(2,3)
puts "El valor de la hipotenusa es #{hip}"
```

**h es una Variable Local**

```
def calcularHipotenusa(a,b)  
    $h = Math.sqrt(a*a+b*b)  
end
```

Esto es una variable y puedo consultarla en cualquier parte de código

```
calcularHipotenusa(2,3)  
puts "La hipotenusa es #{$h}"
```

# EJEMPLO

- Desarrollemos el ejemplo de la reforestación, usando variables globales y métodos que no reciben parámetros

# EJEMPLO

```
def hallarCantidades()
    hectarea=10000

    if ( ($cant_h *hectarea) > 1000000)
        cant_pino= (((cant_h *hectarea) *0.7)/10)*8
        cant_oyamel = (((cant_h *hectarea) *0.2)/15)*15
        cant_cedro= (((cant_h *hectarea) *0.1)/18)*10

    else
        cant_pino= (((cant_h *hectarea) *0.5)/10)*8
        cant_oyamel = (((cant_h *hectarea) *0.3)/15)*15
        cant_cedro= (((cant_h *hectarea) *0.2)/18)*10

    end

    puts "cantidad de Pinos #{cant_pino} cantidad de Oyamel #{cant_oyamel}
    cantidad de Cedro #{cant_cedro}"
end
```

No se reciben parámetros

# EJEMPLO

```
puts "Ingrese el número de hectáreas:"  
$cant_h = gets.chomp.to_i  
  
hallarCantidades()
```

La variable ***cant\_h*** es *global* porque se definió en el programa principal. Esta variable es *visible* dentro de cualquier función.

**Ver Ejemplo reforestacionGlobales.rb**

# EJERCICIOS

- Usando funciones, desarrollar un programa que dado el peso, la altura y el sexo de un estudiante. Determine la cantidad de vitaminas que debe consumir, con base en los siguientes criterios:

Si es hombre, y su estatura es mayor a 1.60, y su peso es mayor o igual a 75 kilos, su dosis, será: 20% de la estatura y 80% de su peso, si la estatura es menor o igual a 1.60, la dosis será la siguiente: 30% de la estatura y 70% de su peso.

Si es mujer, y su estatura es mayor o igual a 1.55 y su peso es mayor o igual a 65 kilos, su dosis será: 25% de la estatura y 75% de su peso. Si el peso es menor a 65 kilos, será: 35% de la estatura y 65% de su peso.

# EJERCICIOS

- Un comerciante se dedica a la venta de sillas. Vende tres tipos de sillas:

Tipo	Precio
1	\$5.000
2	\$7.000
3	\$10.000

Por cada cinco sillas compradas se obtiene un descuento, de acuerdo a la tabla

Tipo	Descuento
1	3%
2	5%
3	10%

El resto de sillas se cobran a precio normal. Diseñe un programa que lea el tipo de silla y la cantidad a comprar e imprima la cantidad, el precio unitario, el descuento y el precio total, de lo que debe cancelar el cliente por la compra.

**Nota: El almacén sólo vende un tipo de silla a cada cliente.**