Silicon Misiones

Misiones, República Argentina

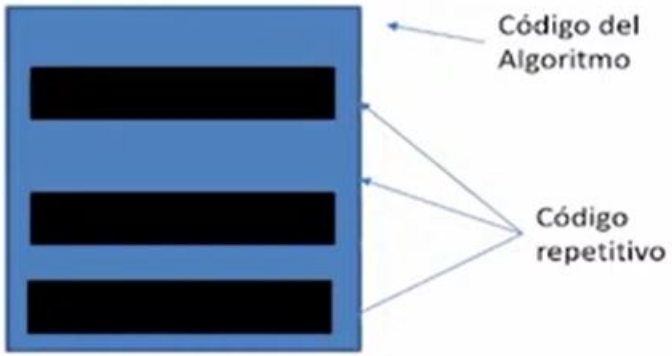
INTRODUCCIÓN A LA PROGRAMACIÓN

FUNCIONES

CLASE 10

CONCEPTO DE FUNCIÓN

- En ocasiones nos encontraremos con tareas que tenemos que repetir varias veces en distintos puntos de nuestro programa.



- Si tecleamos varias veces el mismo fragmento de programa no sólo tardaremos más en escribir, sino que el programa final resultará menos legible, será más fácil que cometamos algún error alguna de las veces que volvemos a teclear el fragmento repetitivo, o que decidamos hacer una modificación y olvidemos hacerla en alguno de los fragmentos.

FUNCIONES

- Por eso, conviene evitar que nuestro programa contenga código repetitivo. Una de las formas de evitarlo es usar "subrutinas", una posibilidad que la mayoría de lenguajes de programación permiten, y que en ocasiones recibe el nombre de "procedimientos" o de "funciones" (existe algún matiz que hace que esas palabras no sean realmente sinónimas y que comentaremos más adelante).

- El concepto de Funciones es aplicable también a los conocidos como Sub Programas, Procedimientos, Store Procedures, Web Services, etc

QUÉ ES UNA FUNCIÓN?

- **Función:** Crea sub-procesos/algoritmos o funciones aparte, que pueden ser llamados en cualquier momento sin tener que hacer la secuencia de acciones en el proceso/algoritmo principal.

Se debe llamar el comando con la palabra 'Funcion' de primero, seguido del nombre de la función:

```
Funcion nombreFuncion
```

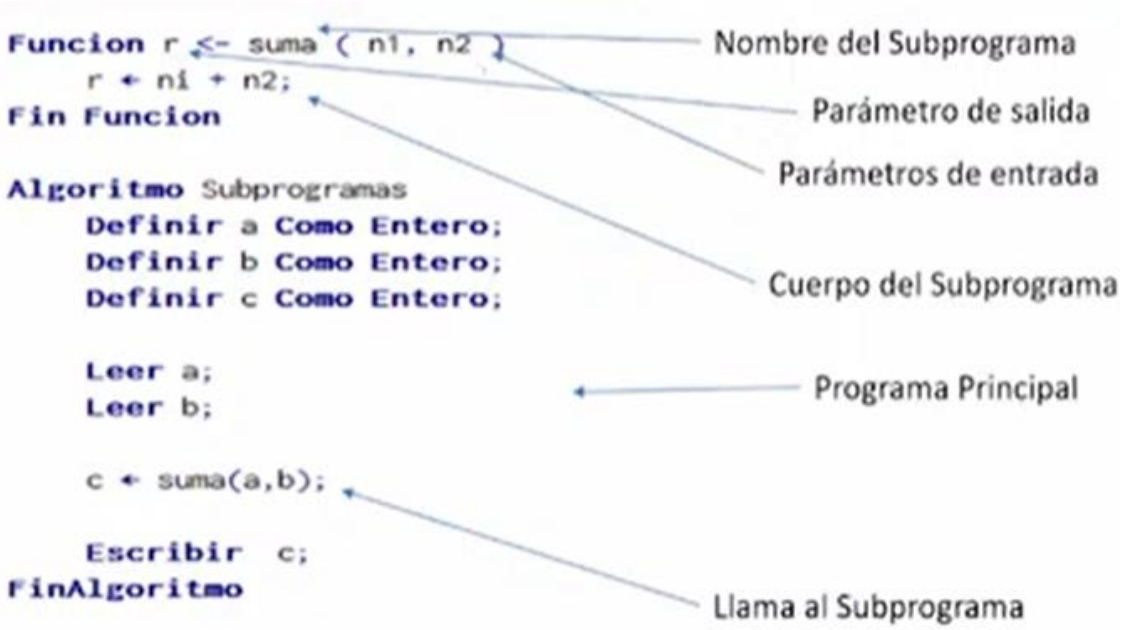
Cabe mencionar que si la función traerá un valor de vuelta (es decir devolverá un valor), este recibe uno o más argumentos y además requerirá una variable, de esta manera utilizamos el argumento '**POR VALOR**', ejemplo:

```
Funcion variableFuncion <- nombreFuncion(argumentos)
```

Podemos usar el argumento '**POR REFERENCIA**' con esto indicamos que el valor del argumento será utilizado como variable de la función y este mismo será modificado, al ser así no seria necesario escribir la variable de la función pero si al lado del argumento escribir 'por referencia':

```
Funcion nombreFuncion(argumentos por referencia)
```

- Ejemplo. Suma de dos valores

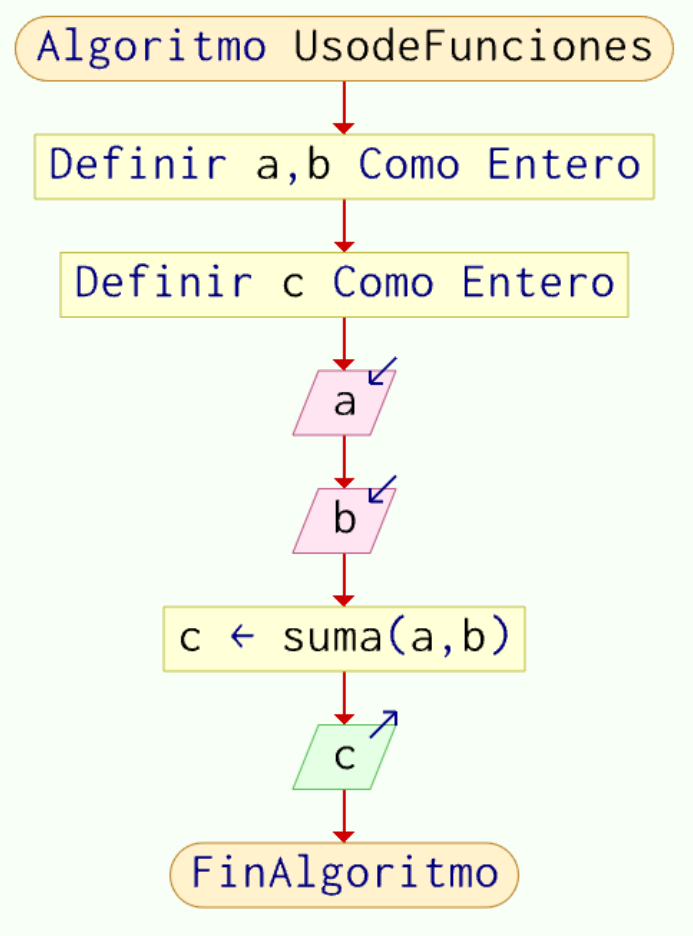


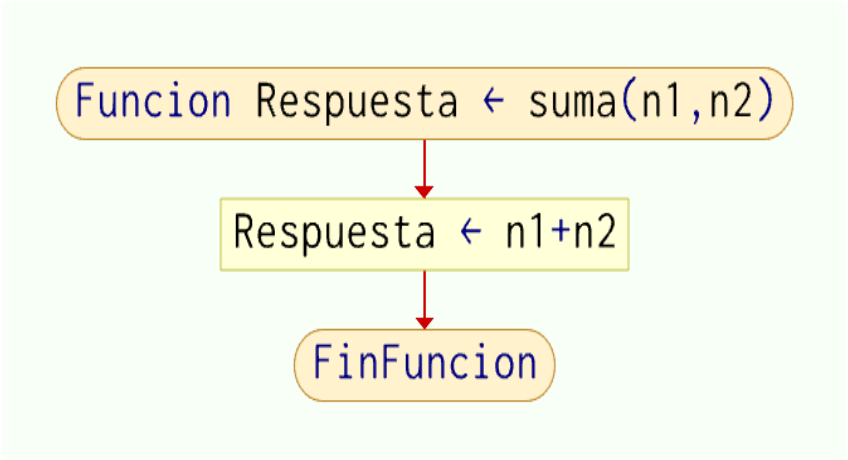
- Ejemplo. Suma de dos valores. Hacer prueba de escritorio.

```
Funcion Respuesta ← suma(n1,n2)
    Respuesta ← n1 + n2;
FinFuncion

Algoritmo UsodeFunciones
    Definir a, b Como Entero
    Definir c Como Entero
    Leer a
    Leer b

    c = suma(a,b);
    Escribir c;

FinAlgoritmo
```

a	b	c	n1	n2	r
5	10				

- Diagrama de Flujo del ejemplo.





FUNCIONES

- Vamos a empezar por crear una "subrutina que escriba 20 guiones, que podríamos utilizar para subrayar textos. Un programa completo que escriba tres textos y los subraye:

```
Algoritmo SubProcesos01
    Escribir " Primer ejemplo"
    Para x <- 1 Hasta 20 Hacer
        Escribir Sin Saltar "-"
    FinPara
    Escribir ""

    Escribir " Segundo ejemplo"
    Para x <- 1 Hasta 20 Hacer
        Escribir Sin Saltar "-"
    FinPara
    Escribir ""

    Escribir " Tercer ejemplo"
    Para x <- 1 Hasta 20 Hacer
        Escribir Sin Saltar "-"
    FinPara
    Escribir ""
FinAlgoritmo
```

FUNCIONES

- Muy repetitivo.
- Sería un poco más elegante si lo reescribimos así:

- Mucho más legible, pero todavía no está tan bien como debería: siempre estamos escribiendo 20 guiones, aunque el texto sea más largo o más corto.

```
Algoritmo SubProcesos02
    Escribir " Primer ejemplo"
    Subrayar

    Escribir " Segundo ejemplo"
    Subrayar

    Escribir " Tercer ejemplo"
    Subrayar
FinAlgoritmo

Subproceso Subrayar
    Para x <- 1 Hasta 20 Hacer
        Escribir Sin Saltar "-"
    FinPara
    Escribir ""
FinSubproceso
```

FUNCIONES

- En la mayoría de lenguajes de programación se puede indicar detalles adicionales ("parámetros") para que se puedan utilizar desde dentro de esa subrutina. Por ejemplo, en nuestro caso podríamos indicarle qué texto queremos escribir y qué longitud queremos que tenga la secuencia de guiones:
- Sería un poco más elegante si lo reescribimos así:

```
Algoritmo SubProcesos03
    EscribirSubrayado(" Primer ejemplo", 16)
    EscribirSubrayado(" Segundo ejemplo", 17)
    EscribirSubrayado(" Tercer ejemplo", 16)
FinAlgoritmo

Subproceso EscribirSubrayado(texto, cantidad)
    Escribir texto
    Para x <- 1 Hasta cantidad Hacer
        Escribir Sin Saltar "-"
    FinPara
    Escribir ""
FinSubproceso
```

FUNCIONES

- Existen funciones predefinidas para manejo de cadenas de texto; una de ellas es "Longitud", que nos indica la cantidad de letras que tiene un texto, de modo que nuestro programa se podría simplificar así:

```
Algoritmo SubProcesos04
    EscribirSubrayado("Primer ejemplo")
    EscribirSubrayado("Segundo ejemplo")
    EscribirSubrayado("Tercer ejemplo")
FinAlgoritmo

Subproceso EscribirSubrayado(texto)
    Escribir texto
    Para x <- 1 Hasta Longitud(texto) Hacer
        Escribir Sin Saltar "-"
    FinPara
    Escribir ""
FinSubproceso
```

PRÁCTICA

- Crear una función que calcule si un número es primo o no (lo vamos a hacer de la forma más simple pero también de la menos eficiente) aplicar la definición, probando a dividir entre todos los números que hay entre 1 y n; si hemos encontrado dos divisores -o menos, para el número uno-, entonces el número es primo:

```
SubAlgoritmo resultado <- Primo ( num )
    cantidadDivisores <- 0
    Para cont <- 1 Hasta num Hacer
        Si num % cont = 0 Entonces
            cantidadDivisores <- cantidadDivisores + 1
        FinSi
    FinPara
    Si cantidadDivisores <= 2 Entonces
        resultado <- verdadero
    Sino
        resultado <- falso
    FinSi
FinSubAlgoritmo

Algoritmo PrimosDel1Al30
    Para n <- 1 hasta 30
        si Primo(n) Entonces
            Imprimir n
        FinSi
    FinPara
FinAlgoritmo
```

MUCHAS GRACIAS!!

Sigamos practicando…