**Creación de funciones y procedimientos (subprocesos)**

**Creación de funciones y procedimientos (subprocesos)**

En muchos casos, 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: además el programa final resultará menos legible, será también más fácil que cometamos algún error alguna de las veces en que volvamos a teclear el fragmento repetitivo, o que decidamos hacer una modificación y olvidemos hacerla en alguno de los fragmentos. 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 permite, 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).

PseInt permite definir "subrutinas" (o "funciones") dentro del pseudocódigo, desde la versión del 10 de octubre de 2012. En su caso, se llaman "subprocesos". Veamos un **ejemplo** de su uso:

Vamos a empezar por crear un subproceso (o "subrutina", o "procedimiento") que escriba 20 guiones, que podríamos utilizar para subrayar textos. Un programa completo que escribiera tres textos y los subrayara podría ser:

**Proceso** SubProcesos01

**Definir** x **como** numerico;

**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** "";

**FinProceso**

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

**Proceso** SubProcesos02

**Escribir** " Primer ejemplo";

Subrayar;

**Escribir** " Segundo ejemplo";

Subrayar;

**Escribir** " Tercer ejemplo";

Subrayar;

**FinProceso**

Subproceso Subrayar

**Definir** x **como** numerico;

**Para** x <- 1 **Hasta** 20 **Hacer**

**Escribir** Sin Saltar "-";

**FinPara**

**Escribir** "";

FinSubproceso

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. 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:

**Proceso** SubProcesos03

EscribirSubrayado(" Primer ejemplo", 16);

EscribirSubrayado(" Segundo ejemplo", 17);

EscribirSubrayado(" Tercer ejemplo", 16);

**FinProceso**

Subproceso EscribirSubrayado(texto, cantidad)

**Definir** x **como** numerico;

**Escribir** texto;

**Para** x <- 1 **Hasta** cantidad **Hacer**

**Escribir** Sin Saltar "-";

**FinPara**

**Escribir** "";

FinSubproceso

Eso todavía es **un poco redundante**: en general, querremos escribir tantos guiones como letras tenga el texto, así que no será necesario indicar ese dato. Desde octubre de 2012, PseInt incluye ciertas 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:

**Proceso** SubProcesos04

EscribirSubrayado("Primer ejemplo");

EscribirSubrayado("Segundo ejemplo");

EscribirSubrayado("Tercer ejemplo");

**FinProceso**

Subproceso EscribirSubrayado(texto)

**Definir** x **como** numerico;

**Escribir** texto;

**Para** x <- 1 **Hasta** Longitud(texto) **Hacer**

**Escribir** Sin Saltar "-";

**FinPara**

**Escribir** "";

FinSubproceso

También podemos crear subprocesos que realicen ciertas operaciones aritméticas y **devuelvan** un resultado. Se puede hacer con la misma palabra "subproceso" que hemos empleado hasta ahora, pero muchos lenguajes de programación distinguen entre un "procedimiento" o "subrutina", cuando se da una serie de pasos pero no se devuelve ningún valor (como habíamos hecho hasta ahora), y una "función", cuando sí se devuelve un resultado (como haremos en esta ocasión), así que usaremos esta última nomenclatura. Su desarrollo se parece mucho a lo que hemos hecho hasta ahora, con la diferencia de que escogeremos un nombre para la variable que guardará el resultado, así:

Funcion resultado <- Maximo( n1, n2)

**Definir** resultado **como** numerico;

**Si** n1 > n2 **Entonces**

resultado <- n1;

**Sino**

resultado <- n2;

**FinSi**

FinFuncion

**Proceso** Funciones01

**Escribir** "El máximo de 5 y 12 es:";

**Escribir** Maximo(5,12);

**Escribir** "El máximo de 25 y 12 es:";

**Escribir** Maximo(25,12);

**Finproceso**

Este programa crea una función que calcula cuál es el mayor de los dos números que se le indican como parámetro, y la usa dos veces, para mostrar el máximo de dos valores prefijados.

No sólo podemos devolver valores numéricos; también podemos devolver cadenas (como las funciones que vimos en el apartado 13) o "valores de verdad" (verdadero, falso). Por ejemplo, podemos 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):

SubProceso resultado <- Primo ( num )

cantidadDivisores <- 0

**Para** cont <- 1 **Hasta** num **Hacer**

**Si** num % cont = 0 **Entonces**

cantidadDivisores <- cantidadDivisores + 1

**FinSi**

Fin **Para**

**Si** cantidadDivisores <= 2 **Entonces**

resultado <- verdadero

**Sino**

resultado <- falso

Fin **Si**

Fin SubProceso

**Proceso** PrimosDel1Al30

**Para** n <- 1 **hasta** 30

**Si** Primo(n) **Entonces**

Imprimir n

**FinSi**

**FinPara**

**FinProceso**

**EJERCICIOS**

**1.-** Escribe una función en pseudocódigo que devuelva el área de un círculo dado su radio. Recuerda que para calcular el área, la fórmula es Π \* r2.

**2.-** Escribe una función en pseudocódigo que devuelva el resultado de un número elevado a un exponente dado.

**3.-** Escribe una función en pseudocódigo que devuelva el factorial de un número N (siendo N un número entero positivo).  El factorial de 5 es el resultado de 5\*4\*3\*2\*1.

Haz después un algoritmo principal que pida por teclado dos números y escriba en pantalla el factorial de cada número comprendido entre los dos pedidos.

**4.-** Escribe una función en pseudocódigo llamada ConversionMoneda (…) que permita convertir una cantidad de dinero dada en dólares, libras o yenes a euros. Estas son las equivalencias aproximadas:

* 1 libra=1,22 euros
* 1 dolar=0,75 euros
* 1 yen=0.009 euros

**5.-** Los empleados de una fábrica trabajan en dos turnos, Diurno y Nocturno. Se desea calcular el jornal diario de acuerdo a con las siguientes reglas:

* La tarifa de las horas diurnas es de 10 €
* La tarifa de las horas nocturnas es de 13,5 €
* En caso de ser festivo, la tarifa se incrementa en un 10% en caso de turno diurno y en un 15% para el nocturno.

Escribe una función en pseudocódigo llamada jornal(…) que tome como parámetros el nº de horas, el turno y el tipo de día (“Festivo”, “Laborable”) y nos devuelva el sueldo a cobrar. Escribe también un algoritmo principal que pida el nombre del trabajador, el día de la semana, turno y nº de horas trabajadas, nos escriba el sueldo a cobrar usando la función anterior.

Ten en cuenta, que en la función nos pide el tipo de día  pero en el algoritmo le pedimos al día  es decir, que debemos saber si el dia que introduce el usuario es festivo o no. Utiliza notación flexible.

<https://www.discoduroderoer.es/ejercicios-propuestos-y-resueltos-de-funciones-pseudocodigo/>