# Funciones y bibliotecas.

***Programación I – Laboratorio I.***
***Tecnicatura Superior en Programación.***
***UTN-FRA***

**Autores:** *Pablo Gil*
*Hector Farina*

***Revisores:*** *Ing. Ernesto Gigliotti*
*Lic. Mauricio Dávila*

*Versión : 1*

# Índice de contenido

# 1 Funciones y bibliotecas

## 1.1 Diseño de una función

El diseño de la función es una tarea que debe llevar un tiempo importante para su análisis ya que de acuerdo a la forma en la que se piensa nos puede servir para un solo programa o para muchas aplicaciones.
Para el diseño de la función debemos tener en cuenta 3 puntos:

- **Determinación de la tarea a realizar:** No tener en claro este punto significa no saber que hacer, hasta dónde llegar y menos aún que codigo escribir. Se necesita tener muy en claro qué es lo que va a hacer la función.

- **Determinación de los parámetros formales:** Cuando ya tenemos en claro que es lo que hay que hacer se debe determinar cómo obtener los datos para realizar la tarea que se debe hacer. Por ejemplo, si la función lee algo del teclado no es necesario pasarle datos con lo cual el parámetro formal será VOID. En cambio si la función se dedica a realizar alguna búsqueda podemos pensar 2 alternativas:
  - El dato a buscar se lo paso a la función
  - El dato a buscar lo lee la función

  De acuerdo al origen de los datos los parámetros formales cambian en cantidad y tipo. La respuesta a cuál de las opciones a elegir la da el sentido común y el análisis de cual de las 2 formas es mas genérica. Entonces en este momento se decide de acuerdo al origen de los datos cuales van a ser los parámetros formales de la función.

- **Determinación del valor a retornar:** Finalmente resta definir cual será el tipo de dato que retorna la función. Al igual que en el punto anterior el valor a retornar va íntimamente relacionado con la tarea que realice la función.

## 1.2 Recursividad

Recursividad es el proceso de definir algo en términos de sí mismo. Básicamente un problema podrá ser resuelto en forma recursiva si la solución se puede expresar en términos de sí misma, para obtener la solución deberá resolverse el mismo problema sobre un conjunto de datos de entrada menor.
Se dice que una función es recursiva cuando se llama a sí misma. El lenguaje C soporta funciones recursivas. El caso más común para explicar recursividad es el cálculo del factorial. Si bien encaramos el cálculo del factorial utilizando funciones recursivas, es necesario hacer notar que se puede hacer el mismo programa sin usar recursividad.

Si recordamos la definición de factorial

$$n! = \begin{cases} 1 & \text{si } n=0 \\ n * (n-1)! & \text{si } n>0 \end{cases}$$

Si queremos calcular el factorial de 4 tendremos

$4! = 4 . 3 . 2 . 1 = 24$

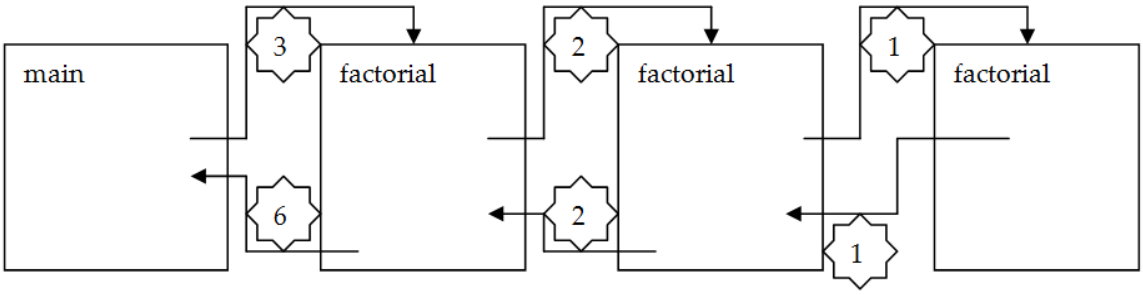
Realicemos el programa que calcula el factorial

```
#include <stdio.h>

int factorial(int );

void main(void)
{
	int valor,result;

	printf("\nIngrese numero:");
	scanf("%d",&valor);
	result=factorial(valor);
	printf("\nEl factorial de %d es %d",valor,result);
}

int factorial(int n)
{
	int resp;
	if(n==1)
		return 1;
	resp=n* factorial(n-1);
	return (resp);
}
```

En la función se ve claramente que se vuelve a llamar a sí misma.

Hay que tener muy presente que cada vez que la función se llama a sí misma, se está utilizando memoria en la pila del programa , que en algún momento se debe liberar. Es decir que en algún momento la función debe comenzar a retornar, en nuestro ejemplo se comienza a retornar cuando el valor de n es 1.

Veamos gráficamente el ejemplo del factorial para entender el funcionamiento.




Suponemos que se desea calcular el factorial de 3. En el dibujo se representa el desarrollo del programa. Junto con cada una de las flechas se encuentra una estrella que en su interior contiene el valor que se le pasa a la función o el valor de retorno.

La función factorial es llamada 3 veces con los valores 3 , 2 y 1 respectivamente. En la última llamada debido a la línea if (n==1) se produce el retorno de la función. Todas las funciones recursivas deben tener una condición que permita el comienzo del retorno, si esto no ocurre se puede producir un desbordamiento de la pila.

## 1.3 Biblioteca de funciones propias

Todas aquellas funciones que en algún momento se desarrollaron y creemos que estan lo suficientemente depuradas y son aptas para usarlas en distintos programas las podemos incluir en una biblioteca propia.

La idea de las funciones, recordando la introducción a este tema, es poder usarlas en muchos programas independizándonos del código que ejecutan. De acuerdo a esto cada vez que se use una función no se debe volver a escribir el código, solamente se la llama.
Para poder aplicar en la práctica esta idea lo que se hace es generar un nuevo archivo en el compilador de C en el cuál solo se escribirán todas aquellas funciones desarrolladas en programas anteriores, en este archivo no figura *main*.

Un ejemplo podría ser un archivo **funciones.c**:

```
/* Funcion usada para verificar el ingreso de caracteres.Chequea para que
   solo sean admitidas las letras S o N.
   Se ingresa void y devuelve un entero sin signo
   0 si se ingreso la N
   1 si se ingreso la S
*/
unsigned int verifica(void)
{
	char letra;
	printf("\nIngrese opcion...S/N?");
	letra=toupper(getche());
	while(!((letra=='S')||(letra=='N')))
	{
		printf("\nHa ingresado opci¢n no valida...Reintente el ingreso(S/N)... ");
		letra=toupper(getche());
	}
	return (letra=='S');
}
```

Luego escribimos un archivo de cabecera con el prototipo de las funciones escritas en el archivo funciones.c, llamado funciones.h con el siguiente contenido:

```
unsigned int verifica(void);
```

Se observa que en ninguno de los dos archivos, aparece la función *main*. Es simplemente escribir un programa que no tiene *main* y en el cual se escriben todas las funciones que el programador ha desarrollado. Este archivo solamente se compila y se verifica que no contenga errores de compilación, se supone que no los debe tener ya que se han copiado las funciones de programas en los cuales estaban funcionando.

Para poder incluir estas funciones en un programa solo se necesita agregar en el programa un *include* como el siguiente:

```
#include <stdio.h>
#include "funciones.h"
```