PROGRAMACIÓN BÁSICA

PROF. ALMA GONZÁLEZ

FUNCIONES

- Una función es un bloque de código que realiza una tarea especifica. Ejemplo: calcular el cuadrado de un numero:
    - El argumento de la función será el numero del que se va a calcular el cuadrado.
    - La salida de la función es el resultado de operar para obtener el cuadrado de la función.
- Las funciones nos permiten dividir nuestro programa, que puede ser complicado y extenso, en pequeños bloques: funciones.
- En C hay dos tipos de funciones: las definidas en las librerías de C, como stdlib.h, stdio.h, etc. Ej. printf, scanf, sqrt,calloc,malloc, etc.... Y aquellas definidas por el usuario.
- Dependiendo de la complejidad y requerimientos de nuestro programa podemos definir tantas funciones como sea necesario.

FUNCIONES

- La estructura de un programa que incluye funciones es:

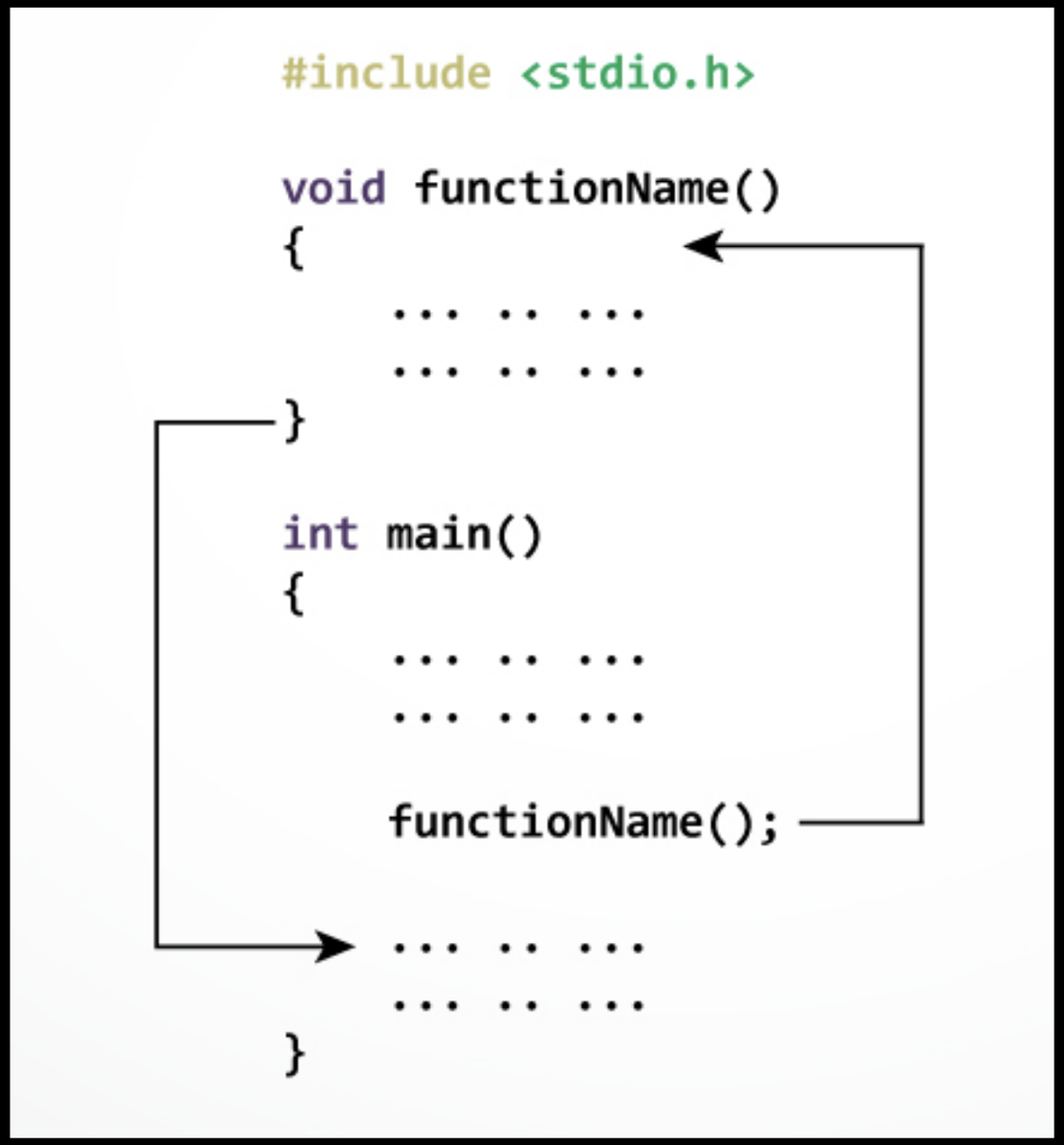
```
#include <stdio.h>

//Declaración de la función. nombre_funcion es un
nombre asiganado a la función y debe ser único y
exclusivo para dicha función.

float nombre_funcion(declaracion  argumentos);
int main()
{
    ... .. ...
    ... .. ...

    nombre_funcion(argumentos);    //Uso de la función

    ... .. ...
    ... .. ...
}
//Definición de las acciones que realiza la función
float nombre_funcion(declaración del tipo de
argumentos )
{
    ... .. ...
    ... .. ...
}
```



IMPORTANTE

- Debemos definir el tipo de argumentos de entrada que necesita la función para ejecutarse. Ej. float, int, double, etc.
- Dentro de la función debemos definir las variables que se puedan requerir para llevar a cabo las tareas. Estas variables se definen localmente es decir solo pueden usarse dentro de la función donde se definieron.
- Diferentes tipos de funciones:
  - Funciones sin argumentos de entrada y sin argumentos de salida. Solo realizan acciones. Funciones tipo: void funcion().
  - Funciones sin argumentos de entrada, pero con argumentos de salida. Funciones tipo: int funcion() , float funcion(), double funcion(), etc... El tipo corresponde al tipo del argumento de salida.
  - Funciones con argumentos de entrada pero sin argumentos de salida: Funciones tipo: void funcion(tipo argumento), donde tipo corresponde al tipo del argumento de entrada, i.e. int, float, double, etc...
  - Funciones con argumentos de entrada y salida: Funciones tipo: tipo funcion(tipo argumento), donde "tipo" corresponde al tipo del argumento de entrada y salida según corresponda, i.e. int, float, double, etc...

EJEMPLO: FUNCIÓN QUE CALCULA EL CUADRADO DE UN NUMERO X^2

```
#include <stdio.h>
//Declaración de la función.
nombre_funcion es un nombre
asiganado a la función y debe ser
único y exclusivo para dicha función.

float nombre_funcion(argumentos);
int main()
{
.....
   nombre_funcion(argumentos);
//Uso de la función
... ... ...
}
//Definición de las acciones que
realiza la función
float nombre_funcion(declaración del
tipo de  argumentos )
{   ... ... ...
```

```
#INCLUDE <STDIO.H>

VOID CUADRADO();

INT MAIN(){
      CUADRADO();
      RETURN 0;
}

VOID CUADRADO( ){
      FLOAT X,X2;
      PRINTF("INTRODUCE UN NUMERO\N");
      SCANF("%F",&X);
      X2=X*X;
            PRINTF("EL CUADRADO DE %F ES
%F\N",X,X2);
      }
```

COMPARACIÓN ENTRE TIPOS DE FUNCIONES

Función tipo void .

```
#include <stdio.h>
void cuadrado();
int main(){
        cuadrado();
        return 0;
}
void cuadrado( ){
       float x,x2;
       printf("Introduce un
       numero\n");
      scanf("%f",&x);
      x2=x*x;
       printf("El cuadrado de %f es
%f\n",x,x2);
        }
```

Función tipo float con argumentos de entrada y salida.

```
#include <stdio.h>
float cuadrado(float h);
int main(){
        float x,x2;
        printf("Introduce un numero\n");
        scanf("%f",&x);
        x2=cuadrado(x);
        printf("El cuadrado de %f es
%f\n",x,x2);
        return 0;
}
float cuadrado(float h){
        return h*h;
        }
```

EJERCICIO.

- Repetir el ejemplo de la función cuadrado para los dos tipos que faltan:
    - con argumentos de entrada pero sin argumentos de salida.
    - con argumentos de salida, pero sin argumentos de entrada.
- Realizar un programa que use 4 funciones: 1 de cada tipo:
    - sin argumentos de entrada y/o salida.
    - con argumentos de entrada pero sin argumentos de salida.
    - con argumentos de salida, pero sin argumentos de entrada.
    - con argumentos de entrada y salida.