EJEMPLOS DE PROGRAMACIÓN DE FUNCIONES EN C:

1.- Realizar una función llamada **par**, que toma **un número entero como parámetro**, y **devuelve 1 si es par o devuelve 0 si es impar**. NOTA: Para saber si un número entero es par, al dividirlo entre 2 su resto debe ser 0.

```
/* mpar.c: Permite probar la función par. */
#include<stdio.h>
int par(int);       // o    int par(int numero);
int main()
{
    int numero, resultado;
    printf("Introduzca un número:\n");
    scanf("%i",&numero);

    resultado=par(numero);

    if (resultado==1)
        printf("Es par.\n");
    else
        printf("Es impar.\n");

    return(0);
}

/* Función par: Devuelve un valor indicando si un
número entero es par o no. */
int par(int numero)
{
    if((numero%2)==0)
        return(1);
    else
        return(0);
}
```

- Si la función **se llama par**, tendrá el prototipo con la siguiente estructura:

  <tipo> **par**(<parámetros>);

- Si nos dicen que **toma un parámetro que es un número entero**, quiere decir que es de tipo int el único parámetro que tiene, además deberemos inventarnos un nombre para esa variable que va entre paréntesis, por ejemplo numero, así tendrá el prototipo:

  <tipo> par(**int numero**);

- Si nos dicen que **devuelve 1 o 0**, en función de si es par o no, quiere decir que el tipo del dato que devuelve es el tipo del 1 o el 0, es decir **devuelve un número entero**, por tanto de tipo int, quedando finalmente así el prototipo de la función:

  int par(int numero);

- Si nos dicen que la función **devuelve** algo, quiere decir que en algún momento debe devolver con una función return un valor, por tanto, si debe devolver 0 o 1, en algún sitio debe haber un **return(0);** o un **return(1);**

- Si nos dicen que la función **devuelve** algo, ese algo que devolverá deberá recogerse desde la función main en una variable del mismo tipo que el valor devuelto. En nuestro caso, como par devuelve un número entero (0 o 1) deberemos declarar en la función main una variable entera en la que luego se almacenará el valor devuelto por par.

```
int main()
{
    int resultado;
    …
    resultado=par(numero);
    …
```

*NOTA: Lo lógico si se programa una función que detecta si un número es par, es que **devuelva un SÍ** si es par **o un NO** si no es par, **el problema es que el lenguaje C no tiene este tipo de valores**; así que **se suelen devolver un 1 como SÍ y un 0 como NO**, en cualquier función que tenga que responder SÍ o NO a algo (por ejemplo, una función que diga si un número es negativo, devolverá 1 si sí lo es y un 0 si no lo es).*

2.- Realizar una función llamada **media2**, que toma **dos números reales como parámetros**, y **devuelve un número real** que es **la media de los dos números pasados como parámetros**. NOTA: Para calcular la media de dos números, se suman, y ese resultado se divide entre 2.

```
/* mmedia2.c: Permite probar la función media2. */
#include<stdio.h>
float media2(float,float);
//o float media2(float n1, float n2);

int main()
{
    float n1, n2, resultado;
    printf("Introduzca un número real:\n");
    scanf("%f",&n1);
    printf("Introduzca otro número real:\n");
    scanf("%f",&n2);

    resultado=media2(n1,n2);
    printf("La media es: %f.\n",resultado);

    return(0);
}

// Función media2: Devuelve la media de 2 números.
float media2(float n1, float n2)
{
    float resultado;
    resultado=(n1+n2)/2;
    return(resultado);
}
```

3.- Realizar una función llamada **media3**, que toma **tres números reales como parámetros**, y **no devuelve nada.** Esa función debe calcular **la media de los tres números pasados como parámetros y mostrar con un mensaje cuál es la media calculada**. NOTA: Para calcular la media de tres números, se suman, y ese resultado se divide entre 3.

```
/* mmedia3.c: Permite probar la función media3. */
#include<stdio.h>
void media3(float,float,float);
//o void media3(float n1,float n2,float n3);

int main()
{
    float n1, n2, n3;
    printf("Introduzca un número real:\n");
    scanf("%f",&n1);
    printf("Introduzca otro número real:\n");
    scanf("%f",&n2);
    printf("Introduzca otro número real:\n");
    scanf("%f",&n3);

    media3(n1,n2,n3);

    return(0);
}

// Función media3: Informa de la media de 3 números.
void media3(float n1, float n2, float n3)
{
    float resultado;
    resultado=(n1+n2+n3)/3;
    printf("La media es:%f\n",resultado);
}
```

- Si nos dicen que la función **no devuelve nada**, quiere decir que la función es de tipo **void:**

void <nombrefunción> (<parámetros>);

y que **nunca debe aparecer una función return** dentro de ella devolviendo un valor; además, como no devuelve ningún valor a la función main, **main no debe declarar ninguna variable resultado donde almacenar lo devuelto por la función media3, porque media3 no devuelve nada** (no tiene return).

4.- Realizar una función llamada **saludo**, que **no tiene parámetros**, y **no devuelve nada.** Esa función debe **mostrar un mensaje en pantalla: "Hola"**.

```
/* msaludo.c: Permite probar la función saludo. */
#include<stdio.h>
void saludo(void);
//o void saludo();

int main()
{
    saludo();

    return(0);
}

// Función saludo: Muestra "Hola".
void saludo()
{
    printf("Hola");
}
```

5.- Realizar una función llamada **saludo2**, que **toma una cadena de hasta 10 caracteres como parámetro**, y **no devuelve nada.** Esa función debe **mostrar en pantalla la cadena que recibe como parámetro**.

```
/* msaludo2.c: Permite probar la función saludo2. */
#include<stdio.h>
void saludo2(char cadena[11]);

int main()
{
    char mensaje[11]="Hola";
    saludo2(mensaje);

    return(0);
}

// Función saludo2: Muestra la cadena recibida.
void saludo2(char cadena[11])
{
    printf("%s",cadena);
}
```

6.- Realizar una función llamada **negativo**, que **toma un número entero como parámetro**, y **devuelve 1 si es negativo o 0 si no lo es.**

NOTA: Un número es negativo si es menor que 0.

```
// mnegativo.c: Permite probar la función negativo.
#include<stdio.h>
int negativo(int numero);

int main()
{
    int n,resultado;

    printf("Introduzca un número:\n");
    scanf("%i",&n);

    resultado=negativo(n);

    if (resultado==1)
        printf("Es un número negativo.\n)";
    else
        printf("No es negativo.\n");

    return(0);
}

/* Función negativo: Devuelve 1 si es negativo el número
entero pasado como parámetro, o 0 si no lo es. */
int negativo(int numero)
{
    int res;
    if (numero<0)
        res=1;
    else
        res=0;
    return(res);
}
```

7.- Realizar una función llamada **ultima**, que **toma una cadena de hasta 10 caracteres como parámetro**, y **devuelve el último carácter.**

Esa función debe **devolver el último carácter si no es vacía (es decir, si tiene caracteres);** si es vacía (“”) debe **devolver un carácter terminador (‘\0’) para indicar que era vacía**.

```
/* multima.c: Permite probar la función ultima. */

#include<stdio.h>
#include<string.h>

char ultima(char cadena[11]);

int main()
{
    char cadena[11], ultimocaracter;

    printf(“Introduzca una cadena:\n”);
    gets(cadena);

    ultimocaracter=ultima(cadena);

    if (ultimocaracter==’\0’)
        printf(“Error: cadena vacía.\n”);
    else
        printf(“El último es: %c\n”,ultimocaracter);

    return(0);
}

/* Función ultima: Devuelve el último carácter de la cadena
recibida. */

char ultima(char cadena[11])
{
    int longitud;
    char caracter ;

    longitud=strlen(cadena);
    if (longitud==0)
        caracter=‘\0’;
    else
        caracter=cadena[longitud-1];

    return(caracter);
}
```

8.- Realizar una función llamada **strlen2**, que **toma una cadena de hasta 1000 caracteres como parámetro**, y **devuelve un número entero** que es **el número de caracteres que tiene.**

NOTA: Debe recorrerse la cadena (vector de caracteres) desde la posición 0 hasta que se encuentre el carácter terminador (‘\0’). El número de caracteres que tiene coincide con la posición donde está el terminador.

```
/* mstrlen2.c: Permite probar la función strlen2. */
#include<stdio.h>
int strlen2(char cadena[1001]);
int main()
{
    char cadena[1001];
    int longitud;

    printf(“Introduzca una cadena:\n”);
    gets(cadena);
    longitud=strlen2(cadena);
    printf(“La longitud es: %i\n”,longitud);

    return(0);
}
/* Función strlen2: Devuelve el número de caracteres de la
cadena recibida como parámetro. */
int strlen2(char cadena[1001])
{
    int x=0;

    while(cadena[x]!=’\0’)
    {
            x++;
    }

    return(x);
}
```

9.- Realizar dos funciones: una llamada **pedir**, que **no toma parámetros**, y **devuelve un número entero;** y otra llamada **triple,** que **toma un número entero como parámetro** y devuelve un número entero.

La función pedir debe **pedir por teclado un número entero,** y **devolverlo**.

La función triple, debe **calcular el triple del número que recibe como parámetro** y **devolver el resultado.**

```
/* m2funciones.c: Permite probar las funciones pedir y triple
juntas. */

#include<stdio.h>

int pedir();
int triple(int);

int main()
{
    int numero, total;

    numero=pedir();

    total =triple(numero);

    printf("El triple de %i es: %i\n",numero, total);

    return(0);
}

// Función pedir: Pide y devuelve un número entero.
int pedir()
{
    int n;
    printf("Introduzca un número entero:\n");
    scanf("%i",&n);
    return(n);
}

// Función triple: devuelve el triple del parámetro.
int triple(int num)
{
    int n;
    n=3*num;
    return(n);
}
```

10.- Realizar una función llamada **ceros**, que **toma como parámetro una matriz de 3x4 de números enteros** y **no devuelve nada**.

Debe rellenar con ceros la matriz de 3x4 que recibe como parámetro.

11.- Realizar una función llamada **mostrar**, que **toma como parámetro una matriz de 3x4 de números enteros** y **no devuelve nada**.

Debe mostrar el contenido de las celdas de la matriz en pantalla.

```
/* mmatriz.c: Permite probar las funciones ceros y mostrar juntas.
*/

#include<stdio.h>

void ceros(int matriz[3][4]);
void mostrar(int matriz[3][4]);

int main()
{
    int matriz[3][4];

    ceros(matriz);
    mostrar(matriz);

    return(0);
}

// Función ceros: Pone las celdas a cero.
void ceros(int matriz[3][4])
{
    int fila, columna;
    for(fila=0;fila<=2;fila++)
        for(columna=0;columna<=3;columna++)
            matriz[fila][columna]=0;
}

// Función mostrar: Muestra la matriz.
void mostrar(int matriz[3][4])
{
    int fila, columna;
    for(fila=0;fila<=2;fila++)
    {
        for(columna=0;columna<=3;columna++)
        {
            printf("%i ", matriz[fila][columna]);
        }
        printf("\n");
    }

}
```