# Práctica 4. Funciones

## Objetivos

- Adquirir la capacidad de diseñar algoritmos para la resolución de problemas básicos.
- Conocer cómo se traduce un algoritmo escrito en pseudocódigo en un lenguaje de programación.
- Adquirir la capacidad de programar en lenguaje C usando funciones.
- Conocer qué procedimiento se sigue para la declaración y uso de funciones.

## 1. Las funciones

Las funciones son fragmentos independientes de código fuente diseñados para realizar una tarea específica.

Sus ventajas son:

- Facilita la programación: permite descomponer un programa en bloques.
- Mejora la legibilidad del código.
- Permite reutilizar código.
- etc

En C todo son funciones.

### 1.1. Función main

La función main es la función principal del programa. Cuando ejecutamos un programa con `$./ejecutable` la primera instrucción que se ejecuta es la primera instrucción de la función `main`.

## 1.2. Sintaxis

La manera de definir una función es la siguiente:

```
tipo_datos_resultado nombrefuncion(tipo_datos_argumento nombreargumento1)
{

  //Aquí van las instrucciones de la función
  //Todas acaban en ;

  //La última instrucción es un return
}
```

- El nombre de la función es el que queramos, sin espacios, sin acentos, etc.
- Los argumentos son los datos que se le pasan a la función. Van entre paréntesis. Para cada argumento hay que indicar el tipo de datos y el nombre. Se separan por comas. Ejemplo: `(int base, int altura)`
- `tipo_datos_resultado` es el tipo de datos del resultado que devuelve la función. Se pone delante del nombre de la función. Si no devuelve nada se indica `void`, si devuelve un entero, `int`, etc.
- Las instrucciones de la función van entre llaves.
- Para devolver un resultado se usa la instrucción `return`

## 1.3. Estructura de un programa con funciones

Usando funciones, la estructura de un programa tiene la declaración de prototipos, antes del main, y la definición de las funciones después del main. Los prototipos tienen la misma línea que la definición pero acaba en punto y coma. Por ejemplo

```
tipo_datos_resultado nombrefuncion(tipo_datos_argumento nombreargumento1);
```

Un esquema sería la siguiente:

```
//Librerías que usa el programa C
//SINTAXIS: #include <nombrelibrería>
...
//Definición de constantes, etc
//SINTAXIS: #define NOMBRECONSTANTE <valor_constante>
...
```

```
   //Declaración de prototipos de funciones
   tipo_datos_resultado nombrefuncion1(tipo_datos_argumento nombreargumento1);
   tipo_datos_resultado2 nombrefuncion2(tipo_datos_argumento1 nombreargumento1,
tipo_datos_argumento2 nombreargumento2);


   //Función principal del programa
   int main ()
   {

     //Aquí van las instrucciones de nuestro programa.
     //Todas acaban en ;

     return 0;
   }

   //Resto de funciones del programa
   tipo_datos_resultado nombrefuncion1(tipo_datos_argumento nombreargumento1)
   {

     //Aquí van las instrucciones de la función
     //Todas acaban en ;

     //La última instrucción es un return
   }

   tipo_datos_resultado2 nombrefuncion2(tipo_datos_argumento1 nombreargumento1,
tipo_datos_argumento2 nombreargumento2)
   {

     //Aquí van las instrucciones de la función
     //Todas acaban en ;

     //La última instrucción es un return
   }
```

## 1.4. Llamadas a funciones

Una vez que hemos declarado los prototipos y definido las funciones, podemos llamarlas desde cualquier parte del código. Para ello, se indica el nom-

bre de la función y entre paréntesis los valores/variables que le pasamos como argumentos. Podemos tener varios casos, que no tengan argumentos, que tengan uno, que tengan más de uno. Además, hay funciones que no devuelven nada (`void`) y otras que sí. Las que devuelven algo necesitan almacenar ese valor en una variable (o bien se evalúa en una condición. Por ejemplo `if (funcion()>0){...}`).

Por ejemplo:

```
int main ()
{

  int valor, a, b;
  //Función sin argumentos que devuelve un int
  valor=funcion1();
  //Función con dos argumentos enteros que devuelve un int
  valor=funcion1(1, 5);
  //Función con dos argumentos enteros que devuelve un int
  valor=funcion1(a, b);
  //Función con dos argumentos enteros que no devuelve nada
  funcion1(a, b);
  //Función sin argumentos que no devuelve nada
  funcion1();
  return 0;
}
```

## 1.5. Ejemplo

El programa siguiente tiene una función que calcula el área de un triángulo.

```
#include <stdio.h>
float areaTriangulo(float base, float altura);
int main () {
  float b, a, area;
  printf("Introduzca la base del triángulo");
  scanf("%f", &b);
  printf("Introduzca la altura del triángulo");
  scanf("%f", &a);
  //Llamada a la función. Deja el resultado en area
  area=areaTriangulo(b,a);
  printf("El área es%f", area);
  return 0;
}
```

```
float areaTriangulo(float base, float altura){
  float resultado;
  resultado=base*altura/2;
  return (resultado);
}
```

# 2. Ejercicios propuestos

## 2.1. Ejercicio 1

Escribe en un archivo con extensión `.c` un programa `Calcula_nota` cuya función `main` lea un valor entre 0 y 10 se lo pase a función `calcula_nota` que según el rango, imprima suspenso, aprobado, notable, sobresaliente.

## 2.2. Ejercicio 2

Escribe en un archivo con extensión `.c` un programa `suma_n_numeros` cuya función `main` lea un valor entero positivo, compruebe que es positivo y si no da error hasta que sea positivo. Luego se lo pasará a una función `suma_n_primeros_nums` que recibe ese número y devuelve la suma de los n primeros números enteros.

## 2.3. Ejercicio 3

Escribe en un archivo con extensión `.c` un programa `potencia` cuya función `main` lea dos valores enteros positivos (base y exponente), comprueba que sean positivos y si no da error hasta que lo sean. Luego se lo pasará a una función `potencia` que recibe esos números y devuelve el resultado de elevar la base al exponente.

## 2.4. Ejercicio 4

Realice un programa `volumen.c` en lenguaje C que calcule volúmenes de figuras geométricas. Para ello la función `main` llamará a una función `menu` que presenta un menú para saber si se quiere calcular el volumen de un cono (1) o el volumen de ortoedro (2) o salir (3). Si la opción no es 1 ó 2 ó 3 da un mensaje de error y los vuelve a pedir hasta que sea un valor correcto. Cuando sea correcto, la función `menu` devuelve ese resultado. La función `main` según el valor que recibe de la función `menu`, leerá los valores correspondientes para

calcular el volumen del cono o del ortoedro. Después, con esos valores, llamará a la función `volumen_cono` o `volumen_ortoedro` que dados esos valores que recibe como argumentos, devuelve el volumen del cono o del ortoedro, respectivamente.

## 2.5. Ejercicio 5

Realice un programa en C que la función `main` lea dos números y compruebe que el primero es menor que el segundo. Si no es así dará un mensaje de error y los volverá a leer. Si los números son correctos, llamará a una función `imprime_pares_intervalo` que imprimirá todos los números pares desde el mayor hasta el menor por pantalla (que los habrá recibido como argumentos). Por ejemplo si se lee el 3 y el 7, ha de imprimir 6, 4.

## 2.6. Ejercicio 6

Realice un programa en C que la función `main` lea un entero `n` que llame a la función `factorial` y devuelve el factorial del número que se le pasa como argumento. La función `main` imprime el resultado por pantalla.