Práctica 6. Funciones

Objetivos de la práctica:

- Adquirir la capacidad de programar en lenguaje C usando funciones.
- Conocer qué procedimiento se sigue para la declaración y uso de funciones.

1. Conceptos fundamentales

En las prácticas anteriores vimos qué etapas teníamos que seguir cuando programamos en lenguaje C:

- En papel, diseñar el algoritmo que resuelve el problema que queremos resolver.
- Traducir el algoritmo a lenguaje C. Para ello usaremos un editor de textos como por ejemplo el *gedit*.
- Compilar el programa para obtener un ejecutable. Vamos a utilizar el compilador *gcc*. El comando a utilizar sería: `$gcc -o ejecutable fuente.c`
- Ejecutar el programa y probar que su funcionamiento es el deseado. Emplearemos el comando `$./ejecutable`

2. 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.

2.1. 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`.

2.2. 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`

2.3. 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
}
```

2.4. 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 nombre 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;
```

```
}
```

2.5. 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);
}
```

3. Ejercicios propuestos

3.1. Ejercicio 1

– Realice un programa `volumen.c` en lenguaje C que calcule volúmenes de figuras geométricas.

- La función `main` presentará un menú para saber si se quiere calcular el volumen de un cono (1) o el volumen de ortoedro (2). En función de la opción elegida, se leerán los datos necesarios y se llamará a la función `volumen_cono` o a la función `volumen_ortoedro`. La función `main` imprimirá el resultado que le devuelva la función.
- La función `volumen_cono` recibe como argumentos el radio de la base y la altura. Devuelve el volumen del cono ($1/3 * PI * radio^2 * altura$)
- La función `volumen_ortoedro` recibe como argumentos dos lados de la base y la altura. Devuelve el volumen del ortoedro ($lado1 * lado2 * altura$)

3.2. Ejercicio 2

Modifica el programa anterior para que el menú incluya una opción 3 "Salir". Se mostrará el menú y mientras la opción elegida no sea la 3 no se saldrá del programa. Es decir, se muestra el menú, si la opción es 1 ó 2 se calcula el volumen, se imprime por pantalla y vuelve a mostrar el menú. Si la opción es distinta de 1, 2 y 3 da un mensaje de error y vuelve a mostrar el menú. Si la opción es 3, sale del programa.

3.3. Ejercicio 3

Realice un programa en C que lea dos números y compruebe que el primero es menor que el segundo. Si no es así dará un mensaje de error. Si los números son correctos, llamará a una función que se le pasan los dos números e imprime todos los números pares desde el mayor hasta el menor por pantalla. Por ejemplo si se lee el 3 y el 7, ha de imprimir 6, 4. Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.

3.4. Ejercicio 4

Realice un programa en C que lea dos números y compruebe que son positivos. Si no lo son debe dar un mensaje de error. Si son positivos, debe llamar a una función que dados dos números, base y exponente, calcula la base elevada al exponente. La función devuelve el resultado y el main lo imprime por pantalla. Compílalo con el compilador `gcc`. Después, ejecútalo para comprobar que funciona correctamente.