

Programación básica

Alma González
Octubre 2021



Funciones

- Una función es un bloque de código que realiza una tarea específica. Ejemplo: calcular el cuadrado de un número:
 - El argumento de la función será el número 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`, `math.h`, etc. Ej. `printf`, `scanf`, `sqrt`, `calloc`, `malloc`, `exp`, `cos`, `log`, etc.... Y aquellas definidas por el usuario.
- Dependiendo de la complejidad y requerimientos de nuestro programa podemos definir tantas funciones como sea necesario.

Definición de funciones

- 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 tipo: void funcion().** Funciones sin argumentos de entrada y sin argumentos de salida. Solo realizan acciones.
 - **Funciones tipo: int funcion() , float funcion(), double funcion(), etc.** Funciones sin argumentos de entrada, pero con argumentos de salida. El tipo de función corresponde al tipo del argumento de salida.
 - **Funciones tipo: void funcion(tipo argumento)** Funciones con argumentos de entrada pero sin argumentos de salida, donde tipo corresponde al tipo del argumento de entrada, i.e. int, float, double, etc...
 - **Funciones tipo: tipo funcion(tipo argumento).** Funciones con argumentos de entrada y salida: donde “**tipo**” corresponde al **tipo** del argumento de entrada y salida según corresponda, i.e. **int, float, double, etc...**

Estructura de código con funciones definidas por el usuario

```
#include <stdio.h>

//Declaración de la función. nombre_funcion es un nombre asignado 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 )
{
    ... ..
}
|
```

Who can see what you share here?

```
#include <stdio.h>

void functionName()
{
    ... ..
}

int main()
{
    ... ..
    functionName();
    ... ..
}
```

Ejemplo: Función calculo del cuadrado de un número.

**Función tipo
void sin
argumentos
de salida**

```
#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 void con argumentos de entrada y salida

Los argumentos de entrada/salida pueden ser variables “solos” o apuntadores (no arreglos)

En el caso de argumentos apuntadores, estos debe estar declarados en la función también.

```
#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;
}
```

Funciones con manejo de arreglos (apuntadores)

```
#include <stdio.h>
#include <stdlib.h>

void cuadrado_(float *x_ptr, float *x2_ptr);

float *cuadrado(float *x_ptr);

int main(void){
    int i;
    float x[4];
    float x2[4];
    float *x2;
    printf("Introduce 4 numeros\n");
    for (i=0; i<4; i++){
        scanf("%f",&x[i]);
    }
    cuadrado_(x,x2_);
    for (i=0; i<4; i++){
        printf("El cuadrado de %f es %f\n",x[i],x2[i]);
    }

    printf("Segunda funcion\n");
    x2=cuadrado(x);
    for (i=0; i<4; i++){
        printf("El cuadrado de %f es %f\n",x[i],x2[i]);
    }
}

void cuadrado_(float *x_ptr, float *x2_ptr){
    int j;
    for (j=0; j<4;j++){
        x2_ptr[j]=x_ptr[j]*x_ptr[j];
    }
}

float *cuadrado(float *x_ptr){
    int j;
    float *x2_ptr=NULL;

    x2_ptr=malloc(4*sizeof(float));

    for (j=0; j<4;j++){
        x2_ptr[j]=x_ptr[j]*x_ptr[j];
    }
    return x2_ptr;
}
```

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.
- Utiliza ejemplos vistos en clases pasadas, para reescribirlos definiendo funciones. Puede ser un ejercicio que use 4 funciones diferentes, 1 de cada tipo. O bien varios ejercicios donde se usen funciones de diferentes tipos.
 - 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.

Las funciones algunas deben operar con variables independientes y otras con arreglos.