

Programación en C: Funciones - II

1ºDAM IoT

Funciones

Para declarar una función en C seguimos una estructura como la siguiente:

```
tipo_valor_retorno identificador_funcion (tipoDato parámetro1, tipoDato parámetro2, tipoDato parámetroN)  
{  
    instrucción 1;  
    instrucción 2;  
    ...  
    instrucción n;  
    return tipo_valor_retorno;  
}
```

```
int suma (int num1, int num2){  
    return num1+num2;  
}
```

Es importante que una función sea declarada y/o implementada antes de usarse y lo haremos fuera de la función main.

Dentro de una función podemos declarar variables, usar condicionales, bucles, etc. Todo lo que hemos visto hasta ahora, incluso llamar a otra función o llamarse a sí misma.

Nos vamos a encontrar con funciones también que no devuelven nada o que no requieren de ningún parámetro.

Funciones que no devuelven
nada

Funciones que no devuelven nada

Las funciones que no devuelven nada normalmente se llaman procedimientos. En C se siguen llamando funciones y se dice que devuelven **void**. Pueden tener o no parámetros. En este ejemplo vemos una función sin parámetros.

```
void identificador_funcion ()  
{  
    instrucción 1;  
    instrucción 2;  
    ...  
    instrucción n;  
}
```

Si no tienen parámetros se escriben los paréntesis vacíos ().

Si la función no devuelve nada no tiene return.

Funciones que no devuelven nada

Ejemplo de una función que imprime unos mensajes de bienvenida.

```
#include <stdio.h>
#include <string.h>
void presentacion()
{
    printf("Hola usuario\n");
    printf("Comienza nuestro programa\n");
}
int main()
{
    presentacion();
    char palabra[31];
    printf("Ingrese una palabra:");
    fflush(stdin);
    gets(palabra);
    int cant = strlen(palabra);
    printf("La palabra %s tiene %i letras\n", palabra, cant);
    return 0;
}
```

La función no tiene parámetros

La función debemos implementarla antes de usarla.

No tiene return porque con void indicamos que no devuelve nada.

Llamada a la función.

Funciones que no devuelven nada

La función siempre debe declararse antes de usarla. Podemos poner la declaración al principio e inicializarla más adelante.

```
#include <stdio.h>
#include <string.h>
void presentacion();
int main()
{
    presentacion();
    char palabra[31];
    printf("Ingrese una palabra:");
    fflush(stdin);
    gets(palabra);
    int cant = strlen(palabra);
    printf("La palabra %s tiene %i letras\n", palabra, cant);
    return 0;
}
void presentacion()
{
    printf("Hola usuario\n");
    printf("Comienza nuestro programa\n");
}
```

Declaración de la función

Llamada a la función

Inicialización de la función.

Funciones que no devuelven nada

Ejercicio: Crea una aplicación calculadora. Con suma, resta y raíz cuadrada. Muestra al usuario un menú con las opciones del programa: 1.Sumar, 2.Restar, 3. Raíz cuadrada, 4. Salir. El programa no termina hasta que el usuario selecciona la opción 4. Para mostrar el menú utiliza una función llamada muestraMenu().

Funciones que no devuelven nada

Solución:

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

void muestraMenu(){
    printf("Introduzca una opción\n");
    printf("1. Sumar\n");
    printf("2. Restar\n");
    printf("3. Raíz\n");
    printf("4. Salir\n");
}
```

Declaración e implementación de la función.

No devuelve nada, sólo imprime mensajes.

No tiene parámetros.

Funciones que no devuelven nada

```
int main(){
    float numero1, numero2;
    int opcion = 0;
    do{
        muestraMenu();
        fflush(stdin);
        scanf("%d", &opcion);
        switch (opcion)
        {
            case 1:
                printf("Introduce número 1\n");
                fflush(stdin);
                scanf("%f", &numero1);
                printf("Introduce número 2\n");
                fflush(stdin);
                scanf("%f", &numero2);
                printf("Resultado suma: %f \n", numero1+numero2);
                break;
            case 2:
                printf("Introduce número 1\n");
                fflush(stdin);
                scanf("%f", &numero1);
                printf("Introduce número 2\n");
                fflush(stdin);
                scanf("%f", &numero2);
                printf("Resultado resta: %f \n", numero1-numero2);
                break;
```

Continuación de la solución:

```
        break;
        case 3:
            printf("Introduce número 1\n");
            fflush(stdin);
            scanf("%f", &numero1);
            printf("Resultado raíz cuadrada: %f \n", sqrt(numero1));
            break;
        case 4:
            printf("Adiós\n");
            break;
        default:
            printf("Opción incorrecta\n");
            break;
    }
    while(opcion!=4);
    return 0 ;
}
```

Funciones que no devuelven nada

Aunque una función no devuelva nada se le pueden pasar parámetros para que opere con ellos. Los parámetros irán separados por comas y se indicará el tipo de dato de cada uno de ellos.

void identificador_funcion (tipo_dato par1, tipo_dato par2, ... tipo_dato parn)

{

instrucción1;

instrucción2;

...

instrucciónn;

}

Funciones que no devuelven nada

Ejemplo: Función que dados dos números enteros informa de cual es mayor.

Declaración e implementación de la función.
Tiene dos parámetros enteros.
No devuelve nada, void.

Podemos llamar a la función en nuestra programa tantas veces como queramos.

Los parámetros con los que llamamos a la función son copiados en num1 y num2 que son variables de la función.

```
#include <stdio.h>

void imprimeMayor (int num1, int num2){
    if (num1>num2){
        printf("El mayor es %d", num1);
    }else if (num1<num2){
        printf("El mayor es %d", num2);
    }else{
        //Imprimo uno de los dos sólo
        printf("Los dos son iguales %d", num1);
    }
}

int main(){
    int numero1, numero2;
    printf("Introduce número 1\n");
    fflush(stdin);
    scanf("%d", &numero1);
    printf("Introduce número 2\n");
    fflush(stdin);
    scanf("%d", &numero2);
    //numero1 se copia en num1
    //numero2 se copia en num2
    imprimeMayor (numero1, numero2);
    //podría pasarle directamente dos valores 5 y 2
    //5 se copia en num1
    //2 se copia en num2
    imprimeMayor (5, 2);
    return 0 ;
}
```

Funciones que no devuelven nada

Ejercicio:

Crea un programa que solicite al usuario lo que gana por hora y las horas que ha trabajado. El programa calcula el sueldo del usuario = sueldo por hora * horas trabajadas.

Utiliza una función que imprima un mensaje de bienvenida al usuario.

Utiliza otra función para calcular el sueldo e imprimirlo por pantalla. Esta función recibe como parámetros las cantidades introducidas por el usuario (sueldo por hora y horas trabajadas). No devuelve nada, simplemente imprime el resultado por pantalla.

Funciones que no devuelven nada

Solución:

Podemos declarar todas las funciones que queramos en un programa.

Cada parámetro puede ser de distinto tipo, en este caso float e int. Cuando llamemos a la función debemos hacerlo en el mismo orden.

Llamada a la función “bienvenido”

Llamada a la función “calcularSueldo”

```
#include <stdio.h>

void bienvenido(){
    printf("Bienvenido Usuario\n");
    printf("Vamos a calcular tu sueldo\n");
}

void calcularSueldo (float s, int h){
    float sueldo = s * h;
    printf("El sueldo total a pagar es %.2f\n", sueldo);
}

int main(){
    float sueldoHora;
    int horas;
    bienvenido();
    printf("Introduce tu sueldo a la hora\n");
    fflush(stdin);
    scanf("%f", &sueldoHora);
    printf("Introduce el número de horas\n");
    fflush(stdin);
    scanf("%d", &horas);
    calcularSueldo(sueldoHora, horas);
    return 0 ;
}
```

Funciones que no devuelven nada

Ejercicio: Modifica el programa calculadora que hiciste anteriormente. Añade una nueva opción para la división. La división la hará una función que recibirá por parámetros los dos números leídos, comprobará si el segundo es 0 e imprimirá un error, en caso contrario imprimirá por pantalla el valor de la división.

Funciones que no devuelven nada

Solución:

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

void muestraMenu(){
    printf("Introduzca una opción\n");
    printf("1. Sumar\n");
    printf("2. Restar\n");
    printf("3. Raíz\n");
    printf("4. División\n");
    printf("5. Salir\n");
}

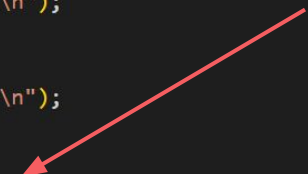
void division(float num1, float num2){
    if (num2==0){
        printf("Error: División por cero\n");
    }else{
        printf("Resultado división %f\n", num1/num2);
    }
}
```

Funciones que no devuelven nada

Solución:

```
int main(){
    float numero1, numero2;
    int opcion = 0;
    do{
        muestraMenu();
        fflush(stdin);
        scanf("%d", &opcion);
        switch (opcion)
        {
            case 1:
                printf("Introduce número 1\n");
                fflush(stdin);
                scanf("%f", &numero1);
                printf("Introduce número 2\n");
                fflush(stdin);
                scanf("%f", &numero2);
                printf("Resultado suma: %f \n", numero1+numero2);
                break;
            case 2:
                printf("Introduce número 1\n");
                fflush(stdin);
                scanf("%f", &numero1);
                printf("Introduce número 2\n");
                fflush(stdin);
                scanf("%f", &numero2);
                printf("Resultado resta: %f \n", numero1-numero2);
                break;
```

```
            case 3:
                printf("Introduce número 1\n");
                fflush(stdin);
                scanf("%f", &numero1);
                printf("Resultado raíz cuadrada: %f \n", sqrt(numero1));
                break;
            case 4:
                printf("Introduce número 1\n");
                fflush(stdin);
                scanf("%f", &numero1);
                printf("Introduce número 2\n");
                fflush(stdin);
                scanf("%f", &numero2);
                division(numero1, numero2);
                break;
            case 5:
                printf("Adiós\n");
                break;
            default:
                printf("Opción incorrecta\n");
                break;
        }
    }while(opcion!=5);
    return 0 ;
}
```



Alcance/ámbito de las variables

Alcance/ámbito de las variables

En C distinguimos entre variables locales y globales:

- **Variables locales:**
 - Todas las variables declaradas en las definiciones de función son variables locales , son conocidas sólo en la función en la cual están definidas (ninguna otra función tiene acceso a ellas).
 - Las variables locales comienzan su existencia cuando la función es llamada y desaparecen cuando la función termina su ejecución, por esta razón se conocen como **variables automáticas**.
- **Variables globales:**
 - Pueden ser accedidas por cualquier función.
 - Son las declaradas fuera del main.

Alcance/ámbito de las variables

```
#include <stdio.h>
void bienvenido(){
    printf("Bienvenido Usuario\n");
    printf("Vamos a calcular tu sueldo\n");
}
void calcularSueldo (float s, int h){
    float sueldo = s * h;
    printf("El sueldo total a pagar es %0.2f\n", sueldo);
}
int main(){
    float sueldoHora;
    int horas;
    bienvenido();
    printf("Introduce tu sueldo a la hora\n");
    fflush(stdin);
    scanf("%f", &sueldoHora);
    printf("Introduce el número de horas\n");
    fflush(stdin);
    scanf("%d", &horas);
    calcularSueldo(sueldoHora, horas);
    return 0 ;
}
```

s, h y sueldo son variables locales de calcularSueldo. Mueren cuando termina la función. Es decir, desde fuera de la función no podemos operar con ellas porque no existen.

Recuerda que main es una función, así que las variables sueldoHora y horas son variables locales de main(). Desde las funciones bienvenido y/o calcularSueldo no podemos operar con ellas.

Alcance/ámbito de las variables

```
#include <stdio.h>

void bienvenido(){
    printf("Bienvenido Usuario\n");
    printf("Vamos a calcular tu sueldo\n");
}

void calcularSueldo (float sueldoHora, int horas){
    float sueldo = sueldoHora * horas;
    printf("El sueldo total a pagar es %0.2f\n", sueldo);
}

int main(){
    float sueldoHora;
    int horas;
    bienvenido();
    printf("Introduce tu sueldo a la hora\n");
    fflush(stdin);
    scanf("%f", &sueldoHora);
    printf("Introduce el número de horas\n");
    fflush(stdin);
    scanf("%d", &horas);
    calcularSueldo(sueldoHora, horas);
    return 0 ;
}
```

Observa este caso. Las variables de la función `calculaSueldo` se llaman igual que las variables de la función `main`. Pero son variables distintas.

Cuando llamamos a `calculaSueldo` el valor de `sueldoHora` del `main` se copia en la variable `sueldoHora` de `calculaSueldo`. Idem para `horas`.

Alcance de las variables

```
#include <stdio.h>
//Variable global
//Todas las funciones la ven
float sueldo;
void bienvenido(){
    printf("Bienvenido Usuario\n");
    printf("Vamos a calcular tu sueldo\n");
}
void calcularSueldo (float s, int h){
    sueldo = s * h; //Puedo acceder a sueldo
    printf("El sueldo total a pagar es %0.2f\n", sueldo);
}
int main(){
    float sueldoHora;
    int horas;
    bienvenido();
    printf("Introduce tu sueldo a la hora\n");
    fflush(stdin);
    scanf("%f", &sueldoHora);
    printf("Introduce el número de horas\n");
    fflush(stdin);
    scanf("%d", &horas);
    calcularSueldo(sueldoHora, horas);
    printf("Imprimo sueldo desde el main %0.2f\n", sueldo);
    return 0 ;
}
```

Ahora sueldo es una **variable global**. Está declarada fuera de las funciones. Todas las funciones pueden acceder a ella.

La función calcularSueldo puede acceder a ella y modificar su valor.

En la función main() podemos verla y ver el valor modificado de sueldo.

Alcance de las variables

Ejercicio: ¿Qué imprimen todos los printf?

```
#include <stdio.h>
float sueldo = 5; //Variable global
void bienvenido(){
    printf("Bienvenido Usuario\n");
    printf("Vamos a calcular tu sueldo\n");
}
void calcularSueldo (float sueldoHora, int horas){
    float sueldo = sueldoHora * horas;
    printf("El sueldo total a pagar es %0.2f\n", sueldo);
    sueldoHora = 10;
    horas = 10;
}
int main(){
    float sueldoHora = 5;
    int horas = 5;
    calcularSueldo(sueldoHora, horas);
    printf("sueldoHora %f\n", sueldoHora); //??
    printf("horas %d\n", horas); //??
    printf("sueldo %f\n", sueldo); //??
    return 0 ;
}
```

Alcance de las variables

Solución: ¿Qué imprimen todos los printf?

```
El sueldo total a pagar es 25.00
sueldoHora 5.000000
horas 5
sueldo 5.000000
```

```
#include <stdio.h>
float sueldo = 5; //Variable global
void bienvenido(){
    printf("Bienvenido Usuario\n");
    printf("Vamos a calcular tu sueldo\n");
}
void calcularSueldo (float sueldoHora, int horas){
    float sueldo = sueldoHora * horas;
    printf("El sueldo total a pagar es %0.2f\n", sueldo);
    sueldoHora = 10;
    horas = 10;
}
int main(){
    float sueldoHora = 5;
    int horas = 5;
    calcularSueldo(sueldoHora, horas);
    printf("sueldoHora %f\n", sueldoHora); //??
    printf("horas %d\n", horas); //??
    printf("sueldo %f\n", sueldo); //??
    return 0 ;
}
```

Funciones que devuelven un valor

Funciones que devuelven un valor

Las funciones que devuelven un valor deben indicar el tipo de dato que devuelven y deben tener al menos una instrucción llamada **return** que devuelve un dato del mismo tipo que hemos indicado.

***tipo_valor_retorno** identificador_funcion (<parametros>)*

{

instrucción1;

instrucción2;

...

instrucciónn;

return tipo_valor_retorno;

}

El dato devuelto debe coincidir con el tipo que hemos indicado.



Nota: La función puede tener más de un return pero debemos asegurarnos que desde cualquier camino se llegue a un return para devolver el valor.

Funciones que devuelven un valor

```
#include<stdio.h>

int calcularSuperficie(int lado)
{
    int superficie=lado*lado;
    return superficie;
}

int main()
{
    int valor;
    int sup;
    printf("Ingrese el valor del lado del cuadrado:");
    scanf("%i",&valor);
    sup=calcularSuperficie(valor);
    printf("La superficie del cuadrado es %i",sup);
    return 0;
}
```

La función calcularSuperficie devuelve un entero resultado de multiplicar lado * lado.

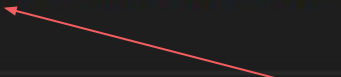
Asignamos el valor que devuelve la función a una variable del mismo tipo.
También podríamos haber llamado directamente a la función calcularSuperficie desde el printf.

Funciones que devuelven un valor

```
#include <stdio.h>

int calcularSuperficie(int lado)
{
    int superficie = lado * lado;
    return superficie;
}

int main()
{
    int valor;
    printf("Ingrese el valor del lado del cuadrado:");
    scanf("%i", &valor);
    printf("La superficie del cuadrado es %i", calcularSuperficie(valor));
    return 0;
}
```



Podemos imprimir directamente lo que devuelve la función sin utilizar una variable para recoger el valor devuelto.

Funciones que devuelven un valor

Ejercicio: Solicita al usuario dos números enteros. Crea una función que dados dos números devuelva el mayor de ellos. Es el mismo ejercicio que hiciste con una función de tipo void. Ahora no imprimas nada en la función, devuelve el número e impríme el número en el main.

Funciones que devuelven un valor

Solución 1:

```
#include <stdio.h>
int imprimeMayor (int num1, int num2){
    int mayor;
    if (num1>num2){
        mayor = num1;
    }else if (num1<num2){
        mayor = num2;
    }else{
        //Guardo uno de los dos
        mayor = num1;
    }
    //Devuelvo el valor de la variable mayor
    return mayor;
}
int main(){
    int numero1, numero2;
    printf("Introduce número 1\n");
    fflush(stdin);
    scanf("%d", &numero1);
    printf("Introduce número 2\n");
    fflush(stdin);
    scanf("%d", &numero2);

    int numMayor = imprimeMayor (numero1, numero2);
    printf("Mayor: %d\n", numMayor);

    printf("Mayor: %d\n", imprimeMayor(5, 2));
    return 0 ;
}
```

Declaramos una variable mayor dónde guardaremos el valor mayor y finalmente la devolveremos con return. Observa que es de tipo entero, igual que el valor que devuelve la función.

Guardo el valor devuelto de la función en una variable y la imprimo.

En este ejemplo, ves que como parámetro le puedo pasar valores directamente e incluso imprimir directamente el valor devuelto por la función.

Funciones que devuelven un valor

Solución 2:

```
int imprimeMayor (int num1, int num2){  
    if (num1>num2){  
        return num1;  
    }else if (num1<num2){  
        return num2;  
    }  
    return num1;  
}
```

Observa que en esta implementación de `calculaMayor` todos los caminos están cubiertos. Es decir, siempre llegamos a un `return`. En el momento que se llega a un `return` se sale de la función y no se continúa.

Funciones que devuelven un valor

Solución 3:

```
int imprimeMayor (int num1, int num2){  
    if (num1>num2){  
        return num1;  
    }  
    //Si llego aquí son iguales o num2 mayor  
    return num2;  
}
```

Podemos simplificar aún más.



Funciones que devuelven un valor

Ejercicio: ¿El siguiente ejemplo es correcto?

```
int esIgual (int a, int b){  
    if (a==b){  
        printf("a es menor que b");  
        return 1;  
    }  
    if(a>b){  
        printf("a es mayor que b");  
        return 0;  
    }  
}
```


Funciones que devuelven un valor

Solución: No, porque si $a < b$ no tiene return.

```
int esIgual (int a, int b){  
    if (a==b){  
        printf("a es menor que b");  
        return 1;  
    }  
    if(a>b){  
        printf("a es mayor que b");  
        return 0;  
    }  
}
```