Funciones en Lenguaje C

La modularización, es una técnica usada por los programadores para hacer sus códigos más cortos, ya que consiste en reducir un gran problema complejo, en pequeños problemitas más sencillos, concentrándose en la solución por separado, de cada uno de ellos. En C, se conocen como funciones aquellos trozos de códigos utilizados para dividir un programa con el objetivo que, cada bloque realice una tarea determinada. En las funciones juegan un papel muy importe las variables, ya que como se ha dicho estas pueden ser locales o globales.

* Variables Globales: Estas se crean durante toda la ejecución del programa, y son globales, ya que pueden ser llamadas, leídas, modificadas, etc; desde cualquier función. Se definen antes del main().
* Variables Locales: Estas, pueden ser utilizadas únicamente en la función que hayan sido declaradas.

La sintaxis de una función es la siguiente:

*Tipo\_de\_datos nombre\_de\_la\_funcion(tipo y nombre de argumentos)*

{

acciones

}

Donde:

* tipo\_de\_retorno: es el tipo del valor devuelto por la función, o, en caso de que la función no devuelva valor alguno, la palabra reservada void.
* nombre\_de\_la\_función: es el nombre o identificador asignado a la función.
* lista\_de\_parámetros: es la lista de declaración de los parámetros que son pasados a la función. Éstos se separan por comas. Debemos tener en cuenta que pueden existir funciones que no utilicen parámetros.
* cuerpo\_de\_la\_función: está compuesto por un conjunto de sentencias que llevan a cabo la tarea específica para la cual ha sido creada la función.
* return expresión: mediante la palabra reservada return, se devuelve el valor de la función.

## Ejemplo

Crear un programa para calcular el precio de un producto basándose en el precio base del mismo y el impuesto aplicable. A continuación mostramos el código fuente de dicho programa:

*#include <stdio.h>*

*float precio(float base, float impuesto); /\* declaración \*/*

*main()*

*{*

*float importe = 2.5;*

*float tasa = 0.07;*

*printf("El precio a pagar es: %.2f\n", precio(importe, tasa));*

*return 0;*

*}*

*float precio(float base, float impuesto) /\* definición \*/*

*{*

*float calculo;*

*calculo = base + (base \* impuesto);*

*return calculo;*

*}*

El ejemplo anterior se compone de dos funciones, la función requerida main y la función creada por el usuario precio, que calcula el precio de un producto tomando como parámetros su precio base y el impuesto aplicable. La función precio calcula el precio de un producto sumándole el impuesto correspondiente al precio base y devuelve el

valor calculado mediante la sentencia return. Por otra parte, en la función main declaramos dos variables de tipo float que contienen el precio base del producto y el impuesto aplicable. La siguiente sentencia dentro de la función main es la llamada a la función de biblioteca printf, que recibe como pará-

metro una llamada a la función precio, que devuelve un valor de tipo float. De esta

manera, la función printf imprime por la salida estándar el valor devuelto por la función precio. Es importante tener en cuenta que las variables importe y tasa (argumentos) dentro de la función main tienen una correspondencia con las variables base e impuesto (parámetros) dentro de la función precio respectivamente.

En el ejemplo anterior, justo antes de la función main, hemos declarado la función precio. La intención es que la función main sea capaz de reconocerla. Sin embargo, la definición de dicha función aparece después de la función main. Las definiciones de función pueden aparecer en cualquier orden dentro de uno o más ficheros fuentes.

## Declaración de funciones

Antes de empezar a utilizar una función debemos declararla. La declaración de una función se conoce también como prototipo de la función. En el prototipo de una función se tienen que especificar los parámetros de la función, así como el tipo de dato que devuelve.

Los prototipos de las funciones que se utilizan en un programa se incluyen generalmente en la cabecera del programa y presentan la siguiente sintaxis:

*tipo\_de\_retorno nombre\_de\_la\_función(lista\_de\_parámetros);*

En el prototipo de una función no se especifican las sentencias que forman parte de la misma, sino sus características. Por ejemplo:

*int cubo(int numero);*

En este caso se declara la función cubo que recibe como parámetro una variable de tipo entero (numero) y devuelve un valor del mismo tipo. En ningún momento estamos especificando qué se va a hacer con la variable numero, sólo declaramos las características de la función cubo. Cabe señalar que el nombre de los parámetros es opcional y se utiliza para mejorar la comprensión del código fuente. De esta manera, el prototipo de la función cubo podría

expresarse de la siguiente manera:

*int cubo(int);*

Los prototipos de las funciones son utilizados por el compilador para verificar que se accede a la función de la manera adecuada con respecto al número y tipo de parámetros, y al tipo de valor de retorno de la misma.

Las funciones de biblioteca se declaran en lo que se conocen como ficheros de cabecera o ficheros .h (del inglés headers, cabeceras). Cuando deseamos utilizar alguna de las funciones de biblioteca, debemos especificar el fichero .h en que se encuentra declarada la función, al inicio de nuestro programa. Por ejemplo, si deseamos utilizar la función printf en nuestro programa, debemos incluir el fichero stdio.h que contiene el prototipo de esta función.

Tras declarar una función, el siguiente paso es implementarla. Generalmente, este paso se conoce como definición. Es precisamente en la definición de una función donde se especifican las instrucciones que forman parte de la misma y que se utilizan para llevar a cabo la tarea específica de la función. La definición de una función consta de dos partes, el encabezado y el cuerpo de la función. En el encabezado de la función, al igual que en el prototipo de la misma, se tienen que especificar los parámetros de la función, si los utiliza y el tipo de datos que devuelve, mientras que el cuerpo se compone de las instrucciones necesarias para realizar la tarea para la cual se crea la función. La sintaxis de la definición de una función es la siguiente:

*tipo\_de\_retorno nombre\_de\_la\_función(lista\_de\_parámetros)*

*{*

*sentencias;*

*}*

El tipo\_de\_retorno representa el tipo de dato del valor que devuelve la función. Este tipo debe ser uno de los tipos simples de C, un puntero a un tipo de C o bien un tipo struct. De forma predeterminada, se considera que toda función devuelve un tipo entero (int). En otras palabras, si en la declaración o en la definición de una función no se

especifica el tipo\_de\_retorno, el compilador asume que devuelve un valor de tipo int. El nombre\_de\_la\_función representa el nombre que se le asigna a la función. Se recomienda que el nombre de la función esté relacionado con la tarea que lleva a cabo. En caso de que la función utilice parámetros, éstos deben estar listados entre paréntesis a continuación del nombre de la función, especificando el tipo de dato y el nombre de cada parámetro. En caso de que una función no utilice parámetros, se pueden dejar los paréntesis vacíos o incluir la palabra void, que indica que la función no utiliza parámetros. Después del encabezado de la función, debe aparecer, delimitado por llaves ({ y }), el cuerpo de la función compuesto por las sentencias que llevan a cabo la tarea específica de la función.

Ejemplo:

#include <stdio.h>

int cubo(int base);

main()

{

int numero;

for(numero=1; numero<=5; numero++)

{

printf("El cubo del número %d es %d\n", numero, cubo(numero));

}

return 0;

}

int cubo(int base)

{

int potencia;

potencia = base \* base \* base;

return potencia;

}

Devolución de valores

Una función en C sólo puede devolver un valor. Para devolver dicho valor, se utiliza la

palabra reservada return cuya sintaxis es la siguiente: return <expresión>; Donde <expresión> puede ser cualquier tipo de dato salvo un array o una función. Además, el valor de la expresión debe coincidir con el tipo de dato declarado en el prototipo de la función. Por otro lado, existe la posibilidad de devolver múltiples valores mediante la utilización de punteros o estructuras. Dentro de una función pueden existir varios return dado que el programa devolverá el control a la sentencia que ha llamado a la función en cuanto encuentre la primera sentencia return. Si no existen return, la ejecución de la función continúa hasta la llave del final del cuerpo de la función (}). Hay que tener en cuenta que existen funciones que no devuelven ningún valor. El tipo de dato devuelto por estas funciones puede ser void, considerado como un tipo especial de dato. En estos casos, la sentencia return se puede escribir como return o se puede omitir directamente. Por ejemplo:

*void imprime\_cabecera();*

*{*

*printf("esta función sólo imprime esta línea");*

*return;*

*}*

*equivale a:*

*void imprime\_cabecera();*

*{*

*printf("esta función sólo imprime esta línea");*

*}*

Para que una función realice la tarea para la cual fue creada, debemos acceder o llamar a la misma. Cuando se llama a una función dentro de una expresión, el control del programa se pasa a ésta y sólo regresa a la siguiente expresión de la que ha realizado la llamada cuando encuentra una instrucción return o, en su defecto, la llave de cierre al final de la función. Generalmente, se suele llamar a las funciones desde la función main, lo que no implica que dentro de una función se pueda acceder a otra función. Cuando queremos acceder a una función, debemos hacerlo mediante su nombre seguido de la lista de argumentos que utiliza dicha función encerrados entre paréntesis. En caso de que la función a la que se quiere acceder no utilice argumentos, se deben colocar los paréntesis vacíos. Cualquier expresión puede contener una llamada a una función. Esta llamada puede ser parte de una expresión simple, como una asignación, o puede ser uno de los operandos de una expresión más compleja. Por ejemplo:

*a = cubo(2);*

*calculo = b + c / cubo(3);*