Cuestionario

Funciones

1. ¿Qué es una función? ¿Se requiere la utilización de funciones al escribir un programa en C?
2. Citar tres ventajas de la utilización de funciones.
3. ¿Qué se entiende por una llamada a una función? ¿Desde qué partes de un programa se puede llamar a una función?
4. ¿Qué son los argumentos? ¿Cuál es su propósito? ¿Qué otro término se utiliza a veces en lugar de argumento?
5. ¿Cuál es la finalidad de la instrucción return?
6. ¿Cuáles son las dos principales componentes de una definición de función?
7. ¿Cómo se escribe la primera línea de una definición de función? ¿Cuál es el propósito de cada elemento o cada grupo de elementos?
8. ¿Qué son los argumentos formales? ¿Qué son los argumentos reales? ¿Cuál es la relación entre ambos tipos de argumentos?
9. Citar otros términos que se utilizan en lugar de argumento formal y argumento real.
10. ¿Pueden coincidir los nombres de los argumentos formales dentro de una función con los nombres de otras variables definidas fuera de la función? Explicarlo.
11. ¿Pueden coincidir los nombres de los argumentos formales dentro de una función con los nombres de otras variables definidas dentro de la función? Explicarlo y comparar la respuesta con la última pregunta
12. Citar las reglas relacionadas con el uso de la instrucción return. ¿Se pueden incluir varias expresiones en una instrucCión return? ¿Se pueden incluir varias instrucciones return en una función?
13. ¿Qué relación debe existir entre el tipo de datos que aparece al comienzo de la primera línea de la definición de una función y el valor devuelto por la instrucción return?
14. ¿Por qué se puede incluir una instrucción return en una función que no devuelve ningún valor?
15. ¿Cuál es la finalidad de la palabra reservada va id? ¿Dónde se utiliza esta palabra reservada?
16. Citar las reglas relacionadas con la llamada a funciones. ¿Qué relación debe existir entre los argumentos reales y los formales correspondientes en la definición de la función? ¿Están sujetos a las mismas restricciones los argumentos reales que los formales?
17. ¿Se puede llamar a una función desde más de un lugar en un programa?
18. ¿Qué son los prototipos de funciones? ¿Cuál es su propósito? ¿Dónde se colocan normalmente los prototipos de funciones en un programa?

Desarrollo

1. Una función es un segmento de programa que realiza determinadas tareas bien definidas. Todo programa en C consta de una o más funciones (ver sección 1.5). Una de estas funciones tiene que llamarse main. La ejecución del programa siempre comenzará por las instrucciones contenidas en main.

2. Ventajas:

- Permite también al programador construir una biblioteca a medida de rutinas de uso frecuente o de rutinas que se ocupen del manejo de elementos dependientes del sistema.
- Favorece la portabilidad, ya que se pueden escribir programas sin prestar atención a las características dependientes del sistema.
- Se puede añadir la correspondiente función de biblioteca al programa durante el proceso de compilación.

3. Se entiende como un argumento para tener acceso a una función. Los argumentos que aparecen en la llamada a la función se denominan argumentos reales, en contraste con los argumentos formales que aparecen en la primera línea de la definición de la función. (También se llaman simplemente argumentos, o parámetros reales.)

4. Los argumentos se denominan argumentos formales, ya que representan los nombres de los elementos que se transfieren a la función desde la parte del programa que hace la llamada también se **llaman parámetros o parámetrosformales**.

5. La instrucción return simplemente devuelve el control al punto del programa desde donde se llamó a la función, sin ninguna transferencia de información.
6. Una definición de función puede incluir varias instrucciones return, conteniendo cada una de ellas una expresión distinta. Las funciones que incluyen varias bifurcaciones suelen requerir varias instrucciones return. Esta función utiliza la instrucción **i f - e 1 s e** en lugar del operador condicional.
7.
8. En una llamada normal a una función, habrá un argumento real por cada argumento formal. Los argumentos reales pueden ser constantes, variables simples, o expresiones más complejas. No obstante, cada argumento real debe ser del mismo tipo de datos que el argumento formal correspondiente. Recordar que el valor de cada argumento real es transferido a la función y asignado al correspondiente argumento formal.
   Los argumentos reales deben corresponderse con los argumentos formales de la definición de la función; es decir, el número de argumentos reales debe ser el mismo que el número de argumentos formales y cada argumento real debe ser del mismo tipo de datos que el correspondiente argumento formal.
9. También se llaman simplemente argumentos, o parámetros reales.
10. Los argumentos reales pueden ser distintos de una llamada a otra. En todo caso, dentro de cada llamada a una función los argumentos reales deben corresponderse con los argumentos formales de la definición de la función; es decir, el número de argumentos reales debe ser el mismo que el número de argumentos formales y cada argumento real debe ser del mismo tipo de datos que el correspondiente argumento formal.

11.

12. La instrucción return puede faltar en la definición de una función, aunque esto se considera generalmente como una mala práctica de programación. Si una función alcanza el final del bloque sin encontrarse una instrucción return, se devuelve el control al punto de llamada sin devolverse ninguna información. Se recomienda en estos casos una instrucción return vacía (sin expresión), para hacer más clara la lógica de la función y hacer más cómodas las modificaciones futuras de la función.

13. Si una función alcanza el final del bloque sin encontrarse una instrucción return, se devuelve el control al punto de llamada sin devolverse ninguna información.

14.

15.

16. En todo caso, dentro de cada llamada a una función los argumentos reales deben corresponderse con los argumentos formales de la definición de la función; es decir, el número de argumentos reales debe ser el mismo que el número de argumentos formales y cada argumento real debe ser del mismo tipo de datos que el correspondiente argumento formal

17. Puede haber diversas llamadas a la misma función desde varios lugares de un programa. Los argumentos reales pueden ser distintos de una llamada a otra.

18. El prototipo de una función es una línea similar a la primera de su declaración: tipo del resultado, seguido del nombre de la función y de la lista de tipos de datos de los parámetros separados por comas y rodeados por paréntesis. Toda función que se invoca debe ir precedida o de su definición o de su prototipo. La definición y el prototipo de la función pueden estar presentes en el mismo fichero.