UNIDAD 5: FUNCIONES

RESUMEN

• La mejor manera de desarrollar y dar mantenimiento a un programa grande es dividiéndolo en varios módulos de programa más pequeños, cada uno de los cuales es más manejable que el programa original. En C, los módulos se escriben como funciones.

• Una función se invoca mediante una llamada a dicha función. La llamada a la función menciona a la función por su nombre y proporciona información (como argumentos) que la función necesita para realizar su tarea.

• El propósito del ocultamiento de información es que las funciones tengan acceso sólo a la información que necesitan para completar sus tareas. Ésta es una manera de implementar el principio del menor privilegio; uno de los principios más importantes de la buena ingeniería de software.

• Por lo general, las funciones se invocan en un programa, escribiendo el nombre de la función seguido por un paréntesis izquierdo, seguido por el argumento de la función (o una lista de argumentos separada por comas) y por el paréntesis derecho.

• El tipo de dato double es un tipo de dato de punto flotante como float. Una variable de tipo double puede almacenar un valor mucho más grande en magnitud y precisión que un número de tipo float.

• Cada argumento de función puede ser una constante, una variable, o una expresión.

• Auna variable local se le conoce sólo en la definición de la función. Las otras funciones no están autorizadas para conocer el nombre de las variables locales de dichas funciones, y tampoco están autorizadas para conocer los detalles de implementación de cualquier otra función.

• El formato general para la definición de una función es:

```
tipo-valor-retorno nombre-función( lista-parámetros )
{
        cuerpo-función
}
```

• El tipo-valor-retorno especifica el tipo de valor devuelto por la función a la que se invoca. Si una función no devuelve valor, el tipo-valor-retorno se declara como void. El nombre-función es cualquier identificador válido. La lista-parámetros es una lista separada por comas que contiene las definiciones de las variables que se pasarán a la función. Si una función no recibe valor alguno, lista-parámetros se declara como void. El cuerpo-función es un conjunto de definiciones e instrucciones que constituyen la función.

• Los argumentos que se pasan a una función deben coincidir en número, tipo y orden con los parámetros en la definición de la función.

• Cuando un programa encuentra una llamada a una función, el control se transfiere desde el punto de invocación hasta la función que fue invocada, las instrucciones de la función invocada se ejecutan y el control regresa a la invocación de la función.

• Una función invocada puede devolver el control al punto de invocación de tres maneras. Si la función no devuelve valor alguno, el control se devuelve cuando ésta alcanza la llave derecha que indica el fin de la función, o por medio de la ejecución de la instrucción: return;

• Si la función devuelve un valor, la instrucción: return expresión; devuelve el valor de la expresión.

• El prototipo de una función declara el tipo de retorno de la función y declara el número, los tipos, y el orden de los parámetros que la función espera recibir.

• Los prototipos de las funciones permiten al compilador verificar que las funciones se invocan de manera correcta.

• El compilador ignora los nombres de variables mencionadas en el prototipo de la función.

• Cada biblioteca estándar tiene un encabezado correspondiente, el cual contiene los prototipos de todas las funciones de la biblioteca, así como las definiciones de las distintas constantes simbólicas necesarias para dichas funciones.

• Los programadores pueden crear e incluir sus propios encabezados.

• Cuando un argumento se pasa por valor, se crea una copia del valor de la variable y dicha copia se pasa a la función invocada. Los cambios que se hagan a esta copia dentro de la función no afectan el valor de la variable original.

• En C, todas las llamadas se hacen por valor.

• La función rand genera un entero entre 0 y RAND_MAX, el cual se define en C para ser al menos 32767.

• Los prototipos de las funciones rand y srand se encuentran en <stdlib.h>.

• Los valores producidos por rand se pueden escalar y modificar para crear valores dentro de un rango específico.

• Para randomizar un programa, utilice la función srand de la biblioteca estándar de C.

• Por lo general, la llamada a la función srand se inserta en un programa, una vez que éste se depuró totalmente. Durante la depuración, es mejor omitir srand. Esto garantiza la repetición de valores, lo cual es esencial para asegurarse de que las correcciones al programa de generación de números aleatorios funcionan adecuadamente.

• Para crear números aleatorios sin tener que introducir una semilla cada vez, utilizamos srand(time(NULL)). La función time devuelve el número de segundos que han pasado desde que inició el día. El prototipo de la función se localiza en el encabezado <time.h>.
• La ecuación general para escalar y modificar un número aleatorio es: n = a + rand() % b;
• donde a es el valor de cambio (es decir, el primer número del rango deseado de enteros consecutivos), y b es el factor de escalamiento (es decir, la longitud del rango de los enteros consecutivos).
• Una enumeración, introducida mediante la palabra reservada enum, es un conjunto de enteros constantes representado mediante identificadores. Los valores en un enum comienzan con 0 y se incrementan en 1. También es posible asignar un valor entero a cada identificador en un enum. Los identificadores de una enumeración deben ser únicos, pero los valores pueden ser duplicados.
• En un programa, cada identificador tiene los atributos clase de almacenamiento, duración del almacenamiento, alcance y vinculación.
• C proporciona cuatro clases de almacenamiento indicadas mediante los especificadores de clase de almacenamiento: auto, register, extern y static.
• La duración de almacenamiento de un identificador se refiere al tiempo de existencia de un identificador en memoria.
• El alcance de un identificador se refiere al lugar en el que se puede hacer referencia a un identificador dentro del programa.
• La vinculación de un identificador determina, para un programa con múltiples archivos fuente, si un identificador es reconocido sólo en el archivo fuente actual o en cualquier archivo fuente mediante las declaraciones apropiadas.
• Las variables con duración automática de almacenamiento se crean cuando se entra al bloque en el que fueron definidas; éstas existen mientras el bloque se encuentra activo y se destruyen cuando se abandona el bloque. Por lo general, las variables locales de una función tienen una duración automática de almacenamiento.
• El especificador de clase de almacenamiento register puede colocarse antes de la declaración de una variable automática para sugerir al compilador que mantenga la variable en uno de los registros de alta velocidad del hardware de la computadora. El compilador podría ignorar las declaraciones register. La palabra reservada register solamente se puede utilizar con variables de duración automática de almacenamiento.
• Las palabras reservadas extern y static se utilizan para declarar identificadores para las variables y las funciones de duración estática de almacenamiento.
• Las variables con duración estática de almacenamiento se asignan y se inicializan una vez que comienza la ejecución del programa.
• Existen dos tipos de identificadores con duración estática de almacenamiento: identificadores externos (tales como variables globales y nombres de función) y variables locales declaradas con el identificador de clase de almacenamiento static.
• Las variables globales se crean al colocar las definiciones fuera de cualquier definición de función. Las variables globales retienen sus valores a través de la ejecución del programa.
• Las variables locales declaradas como static retienen su valor a lo largo de las llamadas a la función en la que se definieron.
• Todas las variables numéricas de duración estática de almacenamiento se inicializan en cero si el programador no las inicializa
explícitamente.
• Los cuatro tipos de alcance para un identificador son: alcance de función, alcance de archivo, alcance de bloque y alcance de prototipo de función.
• Las etiquetas son los únicos identificadores con alcance de función. Las etiquetas se pueden utilizar en cualquier parte de la función en la que aparecen, pero no se puede hacer referencia a ellas fuera del cuerpo de la función.
• Un identificador declarado fuera de cualquier función tiene alcance de archivo. Dicho identificador es “conocido” en todas las funciones desde el punto en el que se declara el identificador y hasta el punto en el que termina el archivo.
• Los identificadores que se definen dentro de un bloque tienen alcance de bloque. El alcance de bloque termina al alcanzar la llave derecha de terminación de bloque (}).
• Las variables definidas al principio de una función tienen alcance de bloque, así como los parámetros de ésta, los cuales son considerados como variables locales por la función.
• Cualquier bloque puede contener definiciones de variables. Cuando los bloques están anidados, y un identificador en el bloque externo tiene el mismo nombre que un identificador en el bloque interno, el identificador en el bloque externo se mantiene “oculto” hasta que el bloque interno termina.
• Los únicos identificadores con alcance de prototipo de función son los que se utilizan en la lista de parámetros de un prototipo de función. Los identificadores utilizados en el prototipo de una función pueden reutilizarse en cualquier parte del programa sin crear ambigüedades.

• Una función recursiva es una función que se invoca a sí misma de manera directa o indirecta.
• Si una función recursiva se invoca mediante un caso base, la función simplemente devuelve un resultado. Si la función se invoca a sí misma mediante un problema más complejo, la función divide el problema en dos partes conceptuales. Una pieza que la función sabe cómo resolver y una versión más sencilla del problema original. Debido a que este nuevo problema es similar al problema original, la función lanza una llamada recursiva para trabajar con el problema más sencillo.
• Para que la recursividad termine, cada vez que la función recursiva se invoca a sí misma por medio de un problema más sencillo que el problema original, la secuencia de problemas cada vez más sencillos debe converger en el caso base. Cuando la función reconoce el caso base, devuelve el resultado a la función llamada previamente, y se origina una secuencia de resultados hacia arriba hasta que la llamada a la función original devuelve el resultado final.
• El estándar de ANSI no especifica el orden en el que se evalúan los operandos de la mayoría de los operadores (incluso +). De los muchos operadores de C, el estándar especifica el orden de evaluación de los operandos de los operadores &&, ||, el operador coma (,) y ?:. Los primeros tres son operadores binarios cuyos dos operandos se evalúan de izquierda a derecha. El último operador es el único operador ternario de C. Su operando más a la izquierda se evalúa primero; si dicho operando da como resultado un número diferente de cero, el operando de en medio se evalúa a continuación y el último operando se ignora; si el operando más a la izquierda da como resultado cero, a continuación se evalúa el tercer operando y el operador que se encuentra en el centro se ignora.
• Tanto la iteración como la recursividad se basan en una estructura de control: la iteración utiliza una estructura de repetición; y la recursividad utiliza una estructura de selección.
• Tanto la iteración como la recursividad involucran la repetición: la iteración utiliza de manera explícita una estructura de repetición;
la recursividad logra la repetición a través de llamadas repetidas a una función.
• La iteración y la recursividad involucran una prueba de terminación: la iteración termina cuando falla la condición de continuación de ciclo; la recursividad termina cuando se reconoce el caso base.
• La iteración y la recursividad pueden repetirse indefinidamente: en el caso de la iteración ocurre un ciclo infinito si la condición de continuación de ciclo nunca se hace falsa; la recursividad infinita ocurre si el paso recursivo no reduce el problema de manera que converja en el caso base.
• La recursividad invoca de manera repetida al mecanismo, y por consecuencia se presenta

EJERCICIOS DE AUTOEVALUACIÓN

5.1 Responda cada una de las siguientes preguntas:

a) A un módulo de programa en C, se le llama **Función**.

b) Una función se invoca mediante una **llamada a función**..

c) A una variable que sólo se conoce dentro de la función en la que se definió se le llama **Variable local.**

d) La instrucción **return** dentro de una función se utiliza para pasar el valor de una expresión hacia la función que la invoca.

e) La palabra reservada **void** se utiliza dentro de una función para indicar que ésta no devuelve valor alguno, o para indicar que la función no contiene parámetros.

f) El **Alcance** de un identificador se refiere a la porción del programa en la que se puede utilizar dicho identificador.

g) Las tres formas de devolver el control desde la función invocada hasta la función que llama son **return; o return expresion; o al encontrar la llave derecha de fin de función.**

h) Un **Prototipo de función** permite al compilador verificar el número, tipo y orden de los argumentos que se pasan a una función.

i) La función **rand** se utiliza para producir números aleatorios.

j) La función **srand** se utiliza para establecer la semilla de los números aleatorios para randomizar un programa.

k) Los especificadores de clase de almacenamiento son: **auto, register, extern y static.**

l) Se asume que las variables declaradas dentro de un bloque, o en la lista de parámetros de una función, tienen una clase de almacenamiento **auto**, a menos que se especifique lo contrario.

m) El especificador de clase de almacenamiento **register** es una recomendación al compilador para que almacene una variable en uno de los registros de la computadora.

n) Una variable definida fuera de cualquier bloque o función es una variable **externa,global.**

o) Para que una variable local de una función retenga su valor entre las llamadas a la misma, la variable se debe declarar con el especificador de clase de almacenamiento **static**.

p) Los cuatro posibles alcances de un dentificador son **alcance de función, alcance de archivo, alcance de bloque y alcance de prototipo de función.**

q) Una función que se invoca a sí misma de manera directa o indirecta es una función **Recursividad.**
r) Por lo general, una función recursiva tiene dos componentes: uno que proporciona un medio para que termine la recursividad a través de la evaluación de un caso **Base**, y otro que expresa el problema como una llamada recursiva a un problema ligeramente más sencillo que el de la llamada original.

5.2 Para el siguiente programa, establezca el alcance (si es alcance de función, de archivo, de bloque o de prototipo de función) de cada uno de los siguientes elementos.
a) La variable x en main. **Alcance de bloque.**
b) La variable y en cubo. **Alcance de bloque.**
c) La función cubo. **Alcance de archivo.**
d) La función main. **Alcance de archivo.**
e) El prototipo de la función para cubo. **Alcance de archivo.**
f) El identificador y en el prototipo de la función cubo. **Alcance de prototipo de función.**

5.4 Indique el encabezado para cada una de las siguientes funciones.
a) La función hipotenusa que toma dos argumentos de punto flotante de precisión doble, lado1 y lado2, y devuelve un resultado de punto flotante de precisión doble.
double hipotenusa(double lado1, double lado2)
b) La función elMenor que toma tres enteros, x, y, z, y devuelve un entero.
int elMenor(int x, int y, int z)
c) La función instrucciones que no recibe argumentos y no devuelve valor alguno. [Nota: Por lo general, dichas funciones se utilizan para desplegar instrucciones para el usuario.]
void instrucciones(void)
d) La función intAfloat que toma un argumento entero, numero, y devuelve un resultado en punto flotante.
float intAfloat (int numero)

Responda cada una de las siguientes preguntas.
a) ¿Qué significa elegir números de manera “aleatoria”?
Elegir números de manera "aleatoria" significa seleccionar números sin seguir un patrón predecible o secuencia establecida. Los números aleatorios se escogen al azar, sin un orden específico, lo que imita la incertidumbre y variabilidad presentes en eventos del mundo real.
b) ¿Por qué la función rand es tan útil para simular juegos de azar?
La función rand es útil para simular juegos de azar porque genera números aparentemente aleatorios en un rango específico, permitiendo simular resultados aleatorios en un programa. Esto es esencial para crear la sensación de azar y variabilidad en simulaciones de juegos y otras aplicaciones que involucran incertidumbre.
c) ¿Por qué randomiza un programa por medio de srand? ¿Bajo qué circunstancias es recomendable no randomizar?
Randomizar un programa mediante srand es útil para generar secuencias de números pseudoaleatorios diferentes en cada ejecución, lo que es fundamental para simular juegos de azar o eventos aleatorios. Sin embargo, durante la depuración, es recomendable omitir srand para que los resultados sean repetibles y facilitar la corrección de errores.
d) ¿Por qué a menudo es necesario escalar y/o modificar los valores producidos por rand?
A menudo es necesario escalar y/o modificar los valores producidos por rand porque los valores generados por esta función están en un rango específico (generalmente de 0 a RAND_MAX). Escalar y modificar permite ajustar estos valores para que se ajusten al rango deseado y cumplan con las necesidades específicas de la aplicación, como generar números en un rango personalizado o con cierta distribución.
e) ¿Por qué la simulación computarizada de situaciones reales es una técnica muy útil?
La simulación computarizada de situaciones reales es útil porque permite analizar y comprender eventos y fenómenos complejos en un entorno controlado y reproducible. Las simulaciones pueden proporcionar información valiosa, realizar experimentos sin riesgos, explorar diferentes escenarios y tomar decisiones informadas. Esto es especialmente útil cuando los eventos reales son costosos, peligrosos o difíciles de replicar en la vida real.

Escriba instrucciones que asignen enteros de manera aleatoria a la variable n en los siguientes rangos:

a) $1 \leq n \leq 2$	n = rand() % 2 + 1;
b) $1 \leq n \leq 100$	n = rand() % 100 + 1;
c) $0 \leq n \leq 9$	n = rand() % 10;
d) $1000 \leq n \leq 1112$	n = rand() % 113 + 1000;
e) $-1 \leq n \leq 1$	n = rand() % 3 - 1;
f) $-3 \leq n \leq 11$	n = rand() % 15 - 3;

TERMINOLOGIA

Abstracción: El proceso de simplificar la representación de algo para hacerlo más manejable y comprensible. En programación, se refiere a ocultar los detalles internos de una función o módulo y proporcionar solo la funcionalidad esencial.

Alcance: El ámbito o contexto en el que un identificador (como una variable o una función) es válido y puede ser referenciado.

Alcance de archivo: Alcance de un identificador declarado fuera de cualquier función, donde el identificador es conocido en todas las funciones del archivo.

Alcance de bloque: Alcance de un identificador declarado dentro de un bloque de código, donde el identificador es válido solo dentro de ese bloque.

Alcance de función: Alcance de una etiqueta, donde la etiqueta se puede utilizar en cualquier parte de la función en la que aparece, pero no fuera de ella.

Alcance de prototipo de función: Alcance de los identificadores utilizados en la lista de parámetros de un prototipo de función, que pueden reutilizarse en cualquier parte del programa sin crear ambigüedades.

Almacenamiento automático: Clase de almacenamiento para variables locales que se crean cuando se entra en el bloque en el que se definen y se destruyen al abandonar ese bloque.

Argumento en una llamada a función: Los valores proporcionados a una función cuando se invoca para realizar su tarea.

Biblioteca estándar de C: Conjunto de funciones y macros predefinidas en el lenguaje C que proporcionan utilidades comunes y funcionalidades esenciales.

Bloque: Una sección de código delimitada por llaves {} en la que se agrupan declaraciones y expresiones. También puede referirse a un bloque de código en una función.

Caso base en la recursividad: El punto de terminación de una función recursiva, donde la función deja de llamarse a sí misma y devuelve un resultado.

Clases de almacenamiento: Etiquetas que especifican cómo se almacenan y acceden a las variables y funciones en C. Las clases incluyen auto, register, extern y static.

Coerción de argumentos: Conversión implícita de tipos de datos en una llamada a función para que coincidan con los parámetros esperados.

Compilador optimizado: Un compilador que realiza optimizaciones en el código fuente para mejorar el rendimiento y la eficiencia del programa compilado.

Copia de un valor: Cuando se pasa un valor a una función por valor, se crea una copia del valor original, y cualquier cambio realizado dentro de la función no afectará al valor original.

Definición de una función: La implementación real de una función que define cómo se realiza una tarea específica.

Divide y vencerás: Un enfoque de resolución de problemas que implica dividir un problema en subproblemas más pequeños y resolverlos individualmente.

Duración automática de almacenamiento: La duración de una variable local con almacenamiento automático, que existe mientras el bloque en el que se declara está activo y se destruye al salir del bloque.

Duración de almacenamiento: El período de tiempo durante el cual una variable existe en la memoria.

Efectos colaterales: Cambios realizados por una función en variables u objetos fuera de su ámbito.

Encabezado: Una sección de código que proporciona información sobre funciones, variables y constantes definidas en un archivo de código fuente.

Encabezados de la biblioteca estándar: Archivos de encabezado que contienen prototipos de funciones y definiciones de constantes para las funciones de la biblioteca estándar de C.

Enum (enumeración): Un tipo de datos definido por el usuario que consiste en un conjunto de valores enteros constantes representados por identificadores.

Escalamiento: El proceso de ajustar los valores generados por funciones como rand para que estén dentro de un rango específico.

Especificador de clase de almacenamiento: Palabras clave como auto, register, extern y static que definen cómo se almacenan y acceden a las variables.

Especificador de conversión %s: Utilizado en la función printf para formatear y mostrar cadenas de caracteres.

Expresiones mixtas: Expresiones que involucran diferentes tipos de datos, lo que puede resultar en promoción y conversión automática de tipos.

Función: Un bloque de código que realiza una tarea específica y puede ser invocado desde otras partes del programa.

Función definida por el programador: Una función creada por el programador para realizar una tarea específica que no está predefinida en el lenguaje.

Función factorial: Una función matemática que calcula el producto de todos los enteros positivos desde 1 hasta un número dado.

Función invocada: Una función que se llama desde otra función para realizar una tarea específica.

Función que llama: La función que invoca a otra función para realizar una tarea específica.
Función recursiva: Una función que se invoca a sí misma directa o indirectamente para resolver problemas mediante la división en subproblemas más pequeños.
Funciones matemáticas de la biblioteca: Funciones predefinidas en la biblioteca matemática estándar de C para realizar operaciones matemáticas comunes.
Generación de números aleatorios: Proceso de generación de secuencias de números aparentemente aleatorios utilizando algoritmos.
Ingeniería de software: Disciplina que se ocupa del diseño, desarrollo y mantenimiento de software de calidad utilizando principios y prácticas eficientes.
Invocar a una función: Llamar a una función para que realice una tarea específica.
Iteración: Repetición controlada de una serie de instrucciones mediante ciclos o bucles.
Jerarquía de promoción: Reglas para la conversión automática de tipos de datos en expresiones, donde los tipos de menor rango se promocionan a tipos de mayor rango.
Lista de parámetros: Los valores que se pasan a una función cuando se la invoca.
Llamada a una función: Acción de invocar una función para que realice una tarea específica.
Llamada por referencia: Pasar la dirección de memoria de una variable a una función para que pueda modificar directamente su valor.
Llamada por valor: Pasar el valor de una variable a una función para que la función trabaje con una copia del valor original.
Llamada recursiva: Invocar a una función desde sí misma para resolver un problema.
Llamar a una función: Invocar a una función para que realice una tarea específica.
Modificación: Cambios realizados a una variable o estructura de datos.
Números pseudoaleatorios: Secuencias de números generados por algoritmos que imitan características de aleatoriedad.
Ocultamiento de información: Limitar el acceso a ciertos detalles internos de una función o módulo, permitiendo que solo se acceda a lo esencial.
Principio del menor privilegio: Un principio de ingeniería de software que dicta que una función o módulo debe tener acceso solo a la información necesaria para realizar su tarea.
Prototipo de función: Declaración anticipada de una función que incluye su nombre, tipo de valor de retorno y lista de parámetros.
rand: Una función de la biblioteca estándar de C que genera números pseudoaleatorios en el rango de 0 a RAND_MAX.
RAND_MAX: El valor máximo que puede generar la función rand, definido en la biblioteca estándar.
Randomizar: Hacer que un programa genere valores aleatorios utilizando la función rand.
Recursividad: Un enfoque de resolución de problemas en el que una función se invoca a sí misma para resolver problemas más pequeños y similares.
return: Una declaración que finaliza una función y puede devolver un valor a la función que la invocó.
Simulación: Proceso de imitar el comportamiento de sistemas del mundo real utilizando un programa de computadora para observar y analizar su funcionamiento.
srand: Una función de la biblioteca estándar de C que inicializa la semilla para generar números pseudoaleatorios con la función rand.
time: Una función de la biblioteca estándar de C que devuelve el tiempo actual en segundos desde una fecha de referencia.
Tipo del valor de retorno: El tipo de dato que una función devuelve como resultado de su ejecución.
unsigned: Un modificador de tipo de datos que se aplica a enteros para indicar que solo se utilizan valores positivos o cero.
Variable automática: Una variable local con almacenamiento automático que se crea cuando se entra en el bloque donde se define y se destruye al salir del bloque.
Variable global: Una variable declarada fuera de cualquier función, que se puede acceder desde cualquier parte del programa.
Variable local: Una variable declarada dentro de una función o bloque, que solo es válida y accesible dentro de ese ámbito.
Variable static: Una variable con duración estática de almacenamiento que retiene su valor entre llamadas a la función en la que se define.
Vinculación: La relación entre identificadores y sus definiciones en diferentes partes de un programa, como archivos fuente múltiples.
void: Un tipo de datos que indica la ausencia de valor o que una función no devuelve ningún valor.