Functions in C - Module - 2

Function Definition and Calling in C

Function Definition

A **function definition** contains the actual implementation of the function. It specifies:

- **What the function does**
- **How it processes input (parameters)**
- **What value (if any) it returns**

Syntax of Function Definition:

```c
return_type function_name(parameter_list) {
    // Body of the function
    // Statements
    return value;  // Optional for non-void functions
}
```

- **return_type**: The type of value the function returns. Use void if the function doesn't return anything.
- **function_name**: Name of the function.
- **parameter_list**: List of variables passed into the function. Each parameter has a data type and a variable name.
- **return value**: The result the function sends back to the caller. Only used if the return_type is not void.

Example:

```c
int add(int a, int b) {  // Function definition
    return a + b;        // Returns the sum of a and b
}
```

Function Call

A **function call** is used to invoke a function and execute the code inside its body. When a function is called, the control of the program jumps to the function definition, executes it, and then returns to the point after the call.

Introduction to 'C' Language

Syntax of Function Call:

```c
function_name(arguments);
```

- **function_name:** The name of the function being called.
- **arguments:** Values or variables passed to the function. These must match the parameter list in the definition in both **number** and **type**.

Example:

```c
int result = add(5, 10);  // Calls the 'add' function with arguments 5 and 10
```

User-Defined Functions vs. Standard (Library) Functions in C

In C programming, functions help organize and reuse code effectively. Functions fall into two main categories: user-defined functions and standard (library) functions.

1. User-Defined Functions

These are functions created by the programmer to perform specific tasks that may not be covered by standard library functions.

Characteristics:

- Written by the user to meet specific needs.
- Can have any name (except reserved keywords).
- Perform tasks such as calculations, data manipulation, or specific operations.

Why Use User-Defined Functions?

- They make your code modular and easier to maintain.
- You can reuse them in multiple parts of the program.
- User-defined functions in C make programs easier to understand by breaking down complicated operations into smaller, named sections.
- This "abstraction" means that instead of seeing all the detailed steps at once, you can just call a function by name, like calculateInterest() or sortArray(), and know what it does without needing to see how it works every time. This makes your code cleaner and easier to read.

2. Standard (Library) Functions

Introduction to 'C' Language

These are predefined functions provided by the C standard library to perform common tasks, such as input/output operations, mathematical calculations, and string manipulations.

Examples:

- printf(): Outputs text to the screen.
- scanf(): Takes input from the user.
- sqrt(): Computes the square root of a number (requires #include <math.h>).
- strlen(): Calculates the length of a string (requires #include <string.h>).

Why Use Standard Functions?

- They save time and effort since you don't have to write common functionalities from scratch.
- They are optimized for performance and thoroughly tested.
- Available across all C compilers and platforms.

Solved Example: Multiply Two Numbers using Functions

```c
#include <stdio.h>

// **Function Definition**
int multiply(int x, int y) {
    return x * y;  // **Returns the product of x and y**
}

int main() {
    int a = 4, b = 6;
    // **Function Call**
    int product = multiply(a, b);
    printf("The product of %d and %d is %d\n", a, b, product);
    return 0;
}
```

Introduction to 'C' Language

Assignment

Ques 1: Write a program using a function to check if a number is odd or even.
Ques 2: Write a program to calculate the factorial of a number using a function and a loop.
Ques 3: Create a program that checks if a number is prime using a function.
Ques 4: Create a function to print the multiplication table of a given number up to a certain range.

- Function name: **printTable**
- Inputs: An integer num and range
- Output: Print the multiplication table.

Ques 5: Create a function to check if a given number is a palindrome.

- Function name: **isPalindrome**
- Input: An integer
- Output: Print whether the number is a palindrome or not.

Ques 6: Write a C program that calculates the area and perimeter of a rectangle based on user inputs for length and width.
Ques 7: Write a C program that calculates the volume and surface area of a cube based on user input for the side length.