4. Functions and Scope

1. What is a function in C++? Explain the concept of function declaration, definition, and calling.

A function in C++ is a block of code that performs a specific task.

Functions help in dividing a large program into smaller and manageable parts.

They improve readability and reusability of code.

Instead of writing the same code multiple times, we can call a function whenever needed.

A function consists of three main parts: declaration, definition, and calling.

Function declaration tells the compiler about the function name, return type, and parameters before it is used.

It is also known as function prototype.

Function definition contains the actual body of the function where the logic is written.

Function calling means executing the function from the main function or another function.

When a function is called, control transfers to the function body.

After execution, control returns back to the calling function.

Functions can return values or may not return anything (void).

They can also accept parameters to process input values.

Functions help in modular programming.

They make debugging easier because errors can be found in specific functions.

Proper use of functions makes programs structured and organized.

Therefore, functions are very important in C++ programming.

2. What is the scope of variables in C++? Differentiate between local and global scope.

Scope of a variable defines the area in a program where the variable can be accessed.

In C++, variables can have local scope or global scope.

A local variable is declared inside a function or block.

It can only be accessed within that function or block.

Once the function ends, the local variable is destroyed.

Local variables improve security because they are not accessible outside the function.

A global variable is declared outside all functions.

It can be accessed by all functions in the program.

Global variables remain in memory throughout the execution of the program.

However, excessive use of global variables is not recommended.

It may lead to confusion and errors in large programs.

Local variables help in modular programming.

Global variables make data sharing easier among functions.

Proper scope management improves program efficiency.

Understanding scope prevents variable conflicts.

It also avoids accidental modification of data.

Therefore, scope plays a very important role in C++ programming.

3. Explain recursion in C++ with an example.

Recursion is a process in which a function calls itself.

It is used to solve problems that can be divided into smaller sub-problems.

In recursion, a function keeps calling itself until a base condition is met.

The base condition is very important to stop infinite execution.

Without a base condition, the program may crash.

Recursion is commonly used for mathematical calculations like factorial and Fibonacci series.

It simplifies complex problems into smaller ones.

Each recursive call creates a new memory block in the stack.

After reaching the base case, the function starts returning values.

Recursion makes code shorter and cleaner.

However, it may use more memory compared to loops.

It is important to write recursion carefully.

Improper recursion may lead to stack overflow.

Recursion improves logical thinking skills.

It is widely used in data structures like trees and graphs.

Therefore, recursion is a powerful concept in C++ programming.

4. What are function prototypes in C++? Why are they used?

A function prototype is a declaration of a function before its actual definition.

It informs the compiler about the function name, return type, and parameters.

Function prototypes are usually written before the main function.

They allow functions to be defined after the main function.

Without a prototype, the compiler may generate an error.

Prototypes help in proper type checking.

They ensure that correct arguments are passed to the function.

If incorrect parameters are passed, the compiler shows an error.

Function prototypes improve program organization.

They make large programs easier to manage.

They support modular programming.

Prototypes also improve readability of code.

They are especially useful in multi-file programs.

Using prototypes prevents logical errors.

They make the program structured and systematic.

Therefore, function prototypes are important in C++ programming.