5. Arrays and Strings

1. What are arrays in C++? Explain the difference between single-dimensional and multi-dimensional arrays.

An array is a collection of elements of the same data type stored in contiguous memory locations.

Arrays are used to store multiple values under a single name.

Each element in an array is accessed using an index number.

Index numbers start from zero.

A single-dimensional array stores elements in a linear form.

For example, int arr[5]; creates an array of 5 integers.

It is used to store simple lists of data.

A multi-dimensional array stores data in rows and columns.

It is like a table or matrix.

For example, int matrix[2][2]; creates a 2x2 matrix.

Multi-dimensional arrays are used for complex data representation.

Single-dimensional arrays are simpler and easier to use.

Multi-dimensional arrays require nested loops for processing.

Arrays improve memory management.

They reduce the need for multiple variables.

Arrays are widely used in sorting and searching algorithms.

Therefore, arrays are an essential concept in C++ programming.

2. Explain string handling in C++ with examples.

Strings in C++ are used to store text values.

C++ provides two ways to handle strings: character arrays and string class.

Character arrays store strings as a sequence of characters ending with null character '\0'.

The string class is provided in the string header file.

It is easier and safer to use compared to character arrays.

Strings support various operations such as concatenation, length, and comparison.

For example, string name = "Shivani"; stores a string.

The length() function returns the length of the string.

Strings can be combined using the + operator.

Input and output of strings can be done using cin and cout.

The getline() function is used to take full line input.

String handling is important for user interaction.

It is widely used in text processing applications.

Strings are commonly used in forms and databases.

Proper string handling improves program functionality.

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

3. How are arrays initialized in C++? Provide examples of both 1D and 2D arrays.

Arrays can be initialized at the time of declaration.

For example, int arr[3] = {1, 2, 3}; initializes a one-dimensional array.

If the size is not specified, the compiler automatically counts elements.

Example: int arr[] = {4, 5, 6};.

Elements can also be assigned individually.

Multi-dimensional arrays are initialized using nested braces.

For example, int matrix[2][2] = {{1,2},{3,4}};.

Each row is enclosed within braces.

Initialization makes arrays ready for use.

If values are not provided, default values may be assigned.

Proper initialization prevents garbage values.

Arrays can also be initialized using loops.

Initialization improves program reliability.

It ensures correct starting values.

Therefore, understanding array initialization is important.

4. Explain string operations and functions in C++.

C++ provides many built-in string functions.

The length() function returns the number of characters.

The append() function adds one string to another.

The compare() function compares two strings.

The substr() function extracts a part of a string.

The find() function searches for a character or word.

Strings can also be concatenated using + operator.

The getline() function reads a full line of text.

String operations are useful in text editing programs.

They are used in password validation systems.

String functions make programs more interactive.

Proper use of string operations improves efficiency.

They simplify complex text manipulations.

String handling is widely used in software applications.

Therefore, string operations are essential in C++.