: 1. Differences between procedural and object-oriented languages

Answer: Procedural languages focus on procedures or functions that perform computations, while object-oriented languages focus on objects that contain both data and methods.

Here are four differences:

Focus: Procedural languages are centered around functions, while object-oriented languages are centered around objects.

Data Security: Data is less secure in procedural languages as it can be accessed by any function. In object-oriented languages, data is more secure due to the concept of data encapsulation.

Approach: Procedural languages follow a top-down approach, breaking a problem down into smaller functions. Object-oriented languages follow a bottom-up approach, building a solution from smaller, self-contained objects.

Inheritance: Procedural languages do not support inheritance, while object-oriented languages do.

: 2. Structure of a C++ program

Answer: The basic structure of a C++ program includes preprocessor directives, a main function, and a body of code.

A typical C++ program structure is as follows:

C++

```
#include <iostream> // Preprocessor directive

using namespace std; // A using declaration


int main() { // Main function

  // Body of the program

  cout << "Hello, World!"; // Statement

  return 0; // Return statement
```

}

[15-10-2025 19:33] : 3. Defining variables in C++

Answer: Variables in C++ are defined by specifying the data type followed by the variable name.

A variable is a named storage location that can hold a value.

Syntax: data_type variable_name;

Example: int age;

Initialization: You can also initialize a variable when you define it. Example: int age = 30;

[15-10-2025 19:33] : 4. Differences between while and do..while loops

Answer: The main difference is that a while loop checks the condition before executing the loop body, while a do..while loop executes the loop body at least once before checking the condition.

Here are two differences:

Condition Check: The while loop is an entry-controlled loop, meaning the condition is checked at the beginning. The do..while loop is an exit-controlled loop, meaning the condition is checked at the end.

Execution: A while loop may not execute at all if the condition is initially false. A do..while loop is guaranteed to execute at least once, regardless of the initial condition.

[15-10-2025 19:33] : 6. C++ access qualifiers

Answer: The two most common access qualifiers in C++ are public and private.

public: Members declared as public are accessible from anywhere in the program. This is the default access level for struct members.

private: Members declared as private are only accessible from within the same class. This is the default access level for class members.

[15-10-2025 19:34] : 7. Inline function

Answer: An inline function is a function that is expanded inline at the point of its call, similar to a macro.

Advantage: It can increase execution speed by avoiding the overhead of a function call (e.g., saving and restoring registers, pushing and popping arguments).

Disadvantage: It can increase the size of the program's executable file, as the function's code is duplicated at every call site.

Example:

C++

```
inline int add(int a, int b) {
  return a + b;
}
```

[15-10-2025 19:34] : 8. Actual and formal arguments

Answer: Formal arguments are the parameters in a function's definition, while actual arguments are the values passed to the function when it is called.

Formal Arguments: These are the variables declared in the function's parameter list. They are placeholders for the values that will be passed to the function.

Actual Arguments: These are the expressions or values specified in the function call. They are the actual data that is passed to the function.

Example:

C++

```
void my_function(int formal_arg1, int formal_arg2) { // formal arguments
  // function body
```

```
}


int main() {
  int actual_arg1 = 5;
  int actual_arg2 = 10;
  my_function(actual_arg1, actual_arg2); // actual arguments
  return 0;
}
```

[15-10-2025 19:35] : 9. Scope rules

Answer: Scope rules determine the visibility and lifetime of an identifier (like a variable or function) within a program.

Local Scope: Variables declared inside a function or a block are only accessible within that function or block. Their lifetime ends when the function or block exits.

Global Scope: Variables declared outside of any function are accessible from anywhere in the program. Their lifetime is the entire duration of the program's execution.

[15-10-2025 19:36] : 10. Predict the output of the following code

Step 1: Analyze the code

The code initializes an integer b to 0 and an integer array c with values from 1 to 9, followed by 0. A for loop iterates from a = 0 to a < 10. In each iteration, the value of c[a] is added to b. After the loop finishes, the final value of b is printed.

Step 2: Trace the loop execution

The loop will execute for a = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

The value of b will be the sum of all elements in the array c.

Step 3: Determine the final output

After the loop, the value of b is 45. The cout << b; statement will print this value to the console.

Answer:

The output of the code will be

.