1. Introduction to C++

1. What are the key differences between Procedural Programming and Object-Oriented Programming (OOP)?

Procedural Programming (POP) is a programming approach that focuses mainly on functions and procedures to perform operations.

In POP, the program is divided into small functions, and these functions operate on shared global data.

The main concept of POP is procedure-oriented, which means it concentrates more on the sequence of actions to be performed.

Languages like C follow the POP approach.

In POP, data is less secure because it can be accessed globally by multiple functions.

It follows a top-down approach where the program is broken into smaller tasks step by step.

On the other hand, Object-Oriented Programming (OOP) focuses on objects and classes rather than just functions.

OOP combines data and functions into a single unit called a class.

It follows a bottom-up approach, where small objects are created first and then combined to form a complete program.

OOP provides better security through data hiding and encapsulation.

OOP supports important features such as inheritance, polymorphism, abstraction, and encapsulation.

These features are not available in POP.

OOP is more suitable for large and complex software development.

It improves code reusability and maintainability.

In OOP, real-world entities can be represented easily using objects.

Therefore, OOP is considered more advanced and powerful compared to POP.

2. List and explain the main advantages of OOP over POP.

Object-Oriented Programming provides many advantages over Procedural Programming.

One of the main advantages is data security, which is achieved using encapsulation.

Encapsulation ensures that data is hidden and can only be accessed through specific methods.

OOP allows code reusability through inheritance, which means one class can inherit properties from another class.

This reduces code duplication and saves time.

OOP also supports polymorphism, which allows functions to behave differently in different situations.

Abstraction helps in hiding complex implementation details and showing only essential features.

OOP makes the program structure more organized and systematic.

It improves readability and understanding of code.

OOP is easier to maintain and modify because changes in one class do not affect other parts of the program.

It is highly suitable for large-scale software development projects.

Debugging becomes easier because errors can be traced within specific objects.

OOP promotes modular programming.

It increases flexibility and scalability of programs.

It allows real-world modeling more effectively.

Overall, OOP improves program efficiency, security, and management compared to POP.

3. Explain the steps involved in setting up a C++ development environment.

To start programming in C++, it is necessary to set up a proper development environment.

The first step is to install a C++ compiler such as Dev C++, CodeBlocks, or Turbo C++.

A compiler is required to convert C++ source code into machine code.

After installation, open the Integrated Development Environment (IDE).

The IDE provides tools to write, compile, and run programs easily.

Next, create a new project or source file in the IDE.

Save the file with a .cpp extension, which indicates a C++ file.

After writing the program code, click on the compile option to check for syntax errors.

If errors are found, correct them and recompile the program.

Once compilation is successful, run the program to see the output.

The output will appear in the console window.

It is important to ensure that the compiler path is correctly configured.

Also, make sure that necessary header files like iostream are included.

Proper installation and configuration help avoid technical issues.

A well-configured environment makes coding faster and easier.

Therefore, setting up the development environment is the first important step in learning C++.

4. What are the main input/output operations in C++? Provide examples.

Input and output operations are essential parts of any C++ program.

In C++, input and output operations are performed using cin and cout.

These are defined in the iostream header file.

The cout statement is used to display output on the screen.

It uses the insertion operator (<<) to send data to the output stream.

The cin statement is used to take input from the user.

It uses the extraction operator (>>) to receive data from the keyboard.

For example, cout << "Hello"; displays text on the screen.

Similarly, cin >> num; allows the user to enter a number.

These operations make the program interactive.

Multiple values can be taken using cin by separating variables with >>.

Formatting output is also possible using endl for a new line.

Input and output operations are simple and easy to use in C++.

They help in communicating between the user and the program.

Without input and output, a program cannot interact with users.

Therefore, understanding cin and cout is very important in C++ programming.