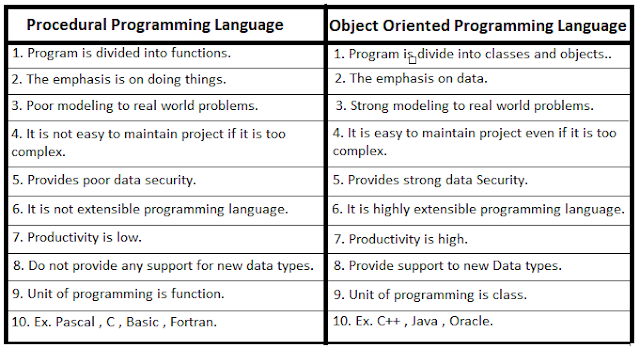
**1. Introduction to C++**

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



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

* Security*: -*OOP is more secure than POP because it supports access control and data hiding.
* Code Maintenance:-OOP is easier to maintain than POP, especially for large, complex projects.
* Code Reusability*:-*OOP encourages the creation of reusable modules, which can be used across projects. This reduces redundant coding and speeds up development.
* Modularity:-OOP breaks down complex code into smaller, more manageable chunks, making it easier to maintain and update.

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

To set up a C++ development environment, you need a text editor and a compiler. Here are some steps you can take:

* Install a C++ compiler: You can download Microsoft's Visual Studio from the Visual Studio website. Visual Studio is a full IDE that includes an editor, debugger, and build systems.
* Create a new file: Open your IDE or text editor and create a new file.
* Write the code: Write your C++ program.
* Compile the program: Use a command line to compile the program.
* Run the program: Use a command line to run the program

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

* The two instances **cout in C++** and **cin in C++** of iostream class are used very often for printing outputs and taking inputs respectively. These two are the most basic methods of taking input and printing output in C++. To use cin and cout in C++ one must include the header file iostream in the program.

*2. Variables, Data Types, and Operators*

1*.What are the different data types available in C++? Explain with examples.*

**Primitive Data Types**: These data types are built-in or predefined data types and can be used directly by the user to declare variables. example: int, char, float, bool, etc. Primitive data types available in C++ are:

* Integer
* Character
* Boolean
* Double
* Float

**Derived Data Types:** Derived Data types  are derived from the primitive or built-in data types are referred to as Derived Data Types. These can be of four types namely:

* Array
* String
* Class

2. *Explain the difference between implicit and explicit type conversion in C++.*

* Implicit type conversion is done automatically by the compiler, while explicit type conversion is done manually by the programmer. Explicit typecasting in C++ can be done in two ways - by using the assignment operator or the cast operator.

*3.What are the different types of operators in C++? Provide examples of each.*

* **Arithmetic operators**
* These operators calculate values or variables in C++ programs. Examples include:
  + **Division**: / Divides the operand in the numerator by the operand in the denominator. For example, x / y.
  + **Modulus**: % Gives the remainder for an integer division. For example, x % y.
  + **Increment**: ++ Increases the integer value by one. For example, ++x.
  + **Decrement** -- Decreases the integer value by one. For example, --x.
* **Assignment operators**
* These operators assign values to variables. Examples include:
  + **Addition assignment**: += Adds an expression to a variable. For example, variable += expression.
  + **Subtraction assignment** -= Subtracts an expression from a variable. For example, variable -= expression.
  + **Multiplication assignment**: \*= Multiplies an expression to a variable. For example, variable \*= expression.
  + **Division assignment**: /= Divides an expression to a variable. For example, variable /= expression.
* **Logical operators**
* These operators determine whether a statement is true or false. Examples include:
  + **Logical and**: && Returns true if both statements are true.
  + **Logical or** || Returns true if one of the statements is true.
  + **Logical not**: ! Reverses the result, returning false if the result is true.

4. *Explain the purpose and use of constants and literals in C++.*

* Constants are used to represent fixed values, while literals are used to specify actual data values directly in the code:
* **Constants**: Fixed values that can't be modified after they are defined. Constants can be used in arithmetic and logical expressions, passed as arguments to functions, and returned as results from functions. Constants are declared using the const keyword followed by a data type.
* **Literals**: Fixed values that are directly written in the code. Literals are used to specify actual data values directly in the code. Literals have a specific syntax based on their type. For example, float literals can be represented in decimal or exponential format

*3. Control Flow Statements*

*1.What are conditional statements in C++? Explain the* if-else *and* switch *statements.*

Conditional statements in C++ are used to control the flow of a program by performing different actions based on the result of a condition. Here are some conditional statements in C++:

* **If**: Executes a block of code if a specified condition is true
* **If-else**: Executes one block of code if the condition is true and another block if it's false
* **Else-if**: Specifies a new condition to test if the first condition is false
* **Switch**: Executes one of many alternative blocks of code based on the value of a variable
* **If-else**: The most commonly used decision-making statement in C++. The condition can only have two values, true or false.
* **Switch**: The expression in a switch statement must result in a constant value. The condition can be an integral type or a class type that can be converted to an integral type.

*2.What is the difference between* for*,* while*, and* do-while *loops in C++?*

The main difference between for, while, and do-while loops in C++ is when and how the condition is checked:

* **For loop**: Used when you know how many iterations are needed.
* **While loop**: Repeats as long as a condition is true. The condition is checked before executing the loop body.
* **Do-while loop**: Guarantees at least one execution of the loop body, even if the condition is false. The condition is checked after executing the loop body.

*3.How are* break *and* continue *statements used in loops? Provide examples.*

Break and continue statements are used in loops to control the flow of execution and make code more efficient. Here's how they work:

* **Break statement**
* Terminates a loop or switch statement completely. For example, in C, you can use a break statement to jump out of a for loop when a certain condition is met.
* **Continue statement**
* Terminates the current iteration of a loop and continues with the next iteration. For example, in Python, you can use a continue statement to skip over certain values in a list without exiting the loop
* **Break statement**: for(int i = 0; i < 10; i++) { if(i == 5) break; }
* **Continue statement**: for(int i = 0; i < 10; i++) { if(i == 5) continue; }

*4.Explain nested control structures with an example.*

* A nested control structure is a control statement that's placed inside another control statement. For example, you can nest an If...Then...Else block within a For...Next loop. Here are some examples of control structures:
* **If and else**: Tests a condition and acts on it
* **For**: Executes a loop a fixed number of times
* **While**: Executes a loop while a condition is true
* **Repeat**: Executes an infinite loop
* **Break**: Breaks the execution of a loop
* **Next**: Skips an iteration of a loop

*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, and is an important part of code reusability and readability:

* **Declaration**: Establishes the function's name, return type, and parameters.
* **Definition**: Includes the body of the function, which is the code that will be executed.
* **Calling**: A function executes only when it is called.

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

* The scope of a variable in C++ is the part of a program where the variable can be used, and is determined when the variable is declared. The two main types of variable scope in C++ are local and global:
* **Local scope**: A variable declared within a function block can only be accessed within that function block. Local variables only exist in RAM while the function is active, and disappear when the function ends.
* **Global scope**: A variable declared outside of any function block can be accessed from anywhere in the program. Global variables are valid from the time they are declared until the end of the program.

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

Recursion in C++ is a technique that involves a function calling itself repeatedly until a base case is reached. The base case is the simplest example that the function can solve without recursion. Here are some examples of recursion in C++:

* **Factorial**: This function uses recursion to perform repetitive calls to itself. The recursive condition is n\*factorial(n-1). For example, if n is 5, then 5 will be multiplied by factorial(5-1), then 4 will be passed to the function. This will continue until n becomes 1.
* **Adding a range of numbers**: This example uses recursion to break down the complicated task of adding a range of numbers into the simpler task of adding two numbers. The function sum(int k) adds parameter k to the sum of all numbers smaller than k. When k becomes 0, the function returns 0.
* **Fibonacci numbers**: This program uses recursion to determine the nth Fibonacci number. It accepts the input n and returns 0 or 1 depending on whether n is 0 or 1. Otherwise, it adds the two Fibonacci numbers before calculating the Fibonacci number recursively.
* Recursion is a useful technique for breaking down complicated problems into simpler ones. It's also great for working with tree and graph structures.

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

* A function prototype in C++ is a declaration that provides information about a function before the actual function definition. It's used to inform the compiler about the function's existence and details so that it can validate and integrate function call statements.

*5.Arrays and Strings*

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

* In C++, an array is a collection of data items of the same type that are accessed by a common name.
* A one-dimensional array stores a single list of various elements having a similar data type. A two-dimensional array stores an array of various arrays, or a list of various lists, or an array of various one-dimensional arrays. It represents multiple data items in the form of a list.

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

In C++, arrays are initialized by following their declaration with a list of initial values enclosed in braces:

* **Syntax**: Data\_type Array\_Name [size] = {initializer}
* **Example**: int a[2][3] = {0,0,0,1,1,1}

Here are some examples of one-dimensional (1D) and two-dimensional (2D) arrays in C++:

* **1D array**

int numbers[5] is a 1D array of 5 elements. A 1D array stores a list of elements with a similar data type.

* **2D array**

int numbers[12][5][6] is an array of 12 elements, each of which is a 2D array. A 2D array stores an array of various arrays, or a list of various lists, or an array of various 1D arrays.

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

* Here are some string operations and functions in C++:
* **std::string class**: This class represents strings and is one of the fundamental datatypes in C++. It comes with many built-in functions.
* **String definition**: You can define a string in C++ using the string keyword or C-style strings.
* **String input**: The getline() function reads a string from an input stream. The stringstream class takes multiple strings as input at once.
* **String conversion**: The to\_string function converts an integer to a string.
* **String replacement**: The replace() function replaces a part of a string with another string.
* **String insertion**: The insert() function inserts a string at a specified index.
* **String removal**: The erase() function removes characters from a string.
* **String comparison**: The compare() function compares two strings.

6. Introduction to Object-Oriented Programming

1. *Explain the key concepts of Object-Oriented Programming (OOP).*

Object-oriented programming (OOP) is a programming model that uses objects to solve problems. The key concepts of OOP are:

* **Encapsulation**: The idea that an object contains all important information, but only exposes a select amount.
* **Abstraction**: The idea that objects only reveal the internal mechanisms that are relevant to other objects, hiding unnecessary implementation code.
* **Inheritance**: The idea that classes can reuse code and properties from other classes.
* **Polymorphism**: The idea that objects can share behaviors and take on more than one form.
* **Objects**: A group of interrelated variables and functions that provide a clear structure for a program.
* **Classes**: A way to bundle attributes together.
* **Methods**: Functions defined within a class that determine the behavior of an object instance.
* **Instantiation**: The process of creating an object from a class.

2. *What are classes and objects in C++? Provide an example****.***

In C++, classes are user-defined data types that serve as a blueprint for creating objects. Objects are instances of a class, and they have access to the class's methods and variables. Here's an example of a class and object in C++:

* **Class**: A class is a blueprint for creating objects. For example, a class called "Car" can define characteristics like brand, model, year, and color, and functions like ignition, acceleration, and braking.
* **Object**: An object is an instance of a class. For example, each individual car with its specific brand, model, and year is an object of the Car class.

3. *What is inheritance in C++? Explain with an example.*

Inheritance in C++ is a feature that allows a class to inherit properties and methods from another class, without modifying the original class. This is a key feature of Object-Oriented Programming (OOP).

Class shape{

Public:

Int l,b;

Void input(){ cout<<"input the length:"<<endl;

cin>>length;

cout<<"input the breadth:"<<endl;

cin>>breadth;}

};

Class Rectangle: public shape{

public:

float area()

{

return length\*breadth;

}

};

int main()

{

rectangle obj;

obj.input();

cout<<"Area of rectangle:"<<obj.area();

}

4. *What is encapsulation in C++? How is it achieved in classes?*

Encapsulation in C++ is a technique that groups related data and functions into a class, and hides the data and functions from outside access. This is achieved by declaring class variables as private, which means they can only be accessed within the class.

In C++, we can bundle data members and functions that operate together inside a single class. For example,

Class Rectangle{

Public:

Int l,b;

Int getArea(){

Return l\*b;

}