C++ COMPLETE SHORT NOTES (Proper Definitions)

■ C++ Tutorial Section

• C++ Intro → C++ is a general-purpose, high-level, object-oriented programming language developed by Bjarne Stroustrup in 1979 as an extension of C, designed to support both procedural and object-oriented programming.

• C++ Get Started → Getting started with C++ requires installing a compiler (e.g., GCC, MSVC), writing code in a .cpp file, and compiling it into an executable program.

• C++ Syntax → Syntax refers to the set of rules and structure for writing valid C++ programs (e.g., main() function, {} braces, ; semicolons).

• C++ Output → Output in C++ is performed using the cout object (from) with the insertion operator <<.

• C++ Comments → Comments are non-executable text in code used for explanation, written with // (single-line) or /*...*/ (multi-line).

• C++ Variables → Variables are named memory locations that store data values which can be modified during program execution.

• C++ Constants → Constants are fixed values declared using const or #define that cannot be changed after initialization.

• C++ User Input → Input is taken from the user using the cin object (from) with the extraction operator >>.

• C++ Data Types → Data types define the kind of data a variable can hold, such as int, float, char, string, bool, double.

• C++ Type Casting → Type casting is the conversion of one data type into another, either implicitly or explicitly.

• C++ Operators → Operators are symbols that perform operations on operands, such as arithmetic (+,-,*), relational (==,<,>), logical (&&,||), and increment/decrement (++,--).

• C++ Strings → Strings are sequences of characters stored and manipulated using the string class from .

• C++ Math → C++ provides predefined mathematical functions in , such as sqrt(), pow(), and round().

• C++ Booleans → A Boolean is a logical data type that can hold only two values: true or false.

• C++ If...Else → The if...else statement allows conditional execution of code blocks based on a logical condition.

• C++ Switch → The switch statement is a multi-branch decision structure that executes one case out of many possible options.

• C++ While Loop → A while loop repeats a block of code as long as its condition evaluates to true.

• C++ Do...While Loop → A do...while loop executes the block at least once, then continues while the condition is true.

• C++ For Loop → A for loop is a count-controlled loop consisting of initialization, condition, and update expressions.

• C++ Break/Continue → break immediately terminates a loop, while continue skips the current iteration and moves to the next.

• C++ Arrays → Arrays are collections of elements of the same type stored in contiguous memory locations.

• C++ Multidimensional Arrays → Multidimensional arrays are arrays of arrays (e.g., 2D arrays for tables or matrices).

• C++ Structures → Structures are user-defined data types that group variables of different types under one name.

• C++ Enums → Enums are user-defined types consisting of named integer constants.

• C++ References → A reference is an alias for an existing variable, created using the & symbol.

• C++ Pointers → Pointers are variables that store the memory address of another variable.

• C++ Null Pointer → A null pointer (nullptr) is a pointer that does not point to any valid memory location.

• C++ Pointer Arithmetic → Pointer arithmetic allows performing operations (++ , -- , +, -) on addresses stored in pointers.

• C++ Memory Management → Memory can be dynamically allocated and deallocated using new and delete.

• C++ Preprocessor Directives → Preprocessor directives (e.g., #include, #define) are instructions processed before compilation.

■ C++ Functions Section

• Functions → A function is a self-contained block of code designed to perform a specific task and can be reused by calling it with its name.

• Function Parameters → Parameters are variables passed into a function to provide input data for processing.

• Return Values → A return value is the result that a function sends back to the part of the program that called it, using the return statement.

• Default Arguments → Default arguments are predefined values assigned to parameters, used when no explicit value is passed.

• Inline Functions → An inline function is one where the compiler substitutes the function body directly at the call location, avoiding a function call overhead.

• Function Overloading → Function overloading is the ability to define multiple functions with the same name but different parameter lists.

• Scope → Scope defines the region in a program where a variable or function is accessible (local, global, block).

• Static Variables → A static variable retains its value across multiple function calls instead of being reinitialized each time.

• Recursion → Recursion is the process where a function calls itself directly or indirectly until a base condition is met.

• Lambda → A lambda is a small, anonymous function written in-line using the syntax []{}.

■ C++ Classes Section (OOP)

• OOP → Object-Oriented Programming is a programming paradigm based on the concept of objects that contain both data and methods.

• Classes/Objects → A class is a blueprint for creating objects, while an object is a specific instance of a class.

• Class Methods → Methods are functions defined inside a class that operate on the class's data members.

• Constructors → A constructor is a special function automatically invoked when an object is created to initialize its data members.

• Destructors → A destructor is a special function automatically invoked when an object is destroyed to release resources.

• Access Specifiers → Access specifiers (public, private, protected) control the accessibility of class members.

• Encapsulation → Encapsulation is the bundling of data and methods within a class, restricting direct access to some components.

• Friend Functions → A friend function is a non-member function that has permission to access private/protected members of a class.

• Static Members → Static members are class-level variables or functions shared by all objects of the class.

• Inheritance → Inheritance is the process of creating new classes (derived) from existing ones (base) to promote code reusability.

• Polymorphism → Polymorphism means "many forms", allowing the same function name to behave differently depending on context.

- Compile-time Polymorphism → Achieved via function overloading and operator overloading.
- Run-time Polymorphism → Achieved via virtual functions and function overriding.

• Abstraction → Abstraction is the concept of hiding implementation details and showing only essential features to the user.

• Operator Overloading → Operator overloading allows redefining operators (like +, -, ==) for user-defined types.

• Virtual Functions → A virtual function is a base class function declared with virtual that can be overridden in derived classes for dynamic dispatch.

• Pure Virtual Functions → A pure virtual function has no body (=0) and makes a class abstract, meaning it cannot be instantiated.

• Templates → Templates allow writing generic code that works with any data type.

• Namespaces in Classes → Namespaces can be used inside classes to group identifiers and prevent name conflicts.

• Files → File handling in C++ uses to read and write data to files.

• Date/Time → Date and time functionalities are provided by the library.

■ C++ Errors Section

• Errors → Errors are problems in code that prevent successful compilation or execution.

• Types of Errors →

- Syntax Errors → Violations of language rules (e.g., missing semicolon).
- Logical Errors → Program runs but produces incorrect results.
- Runtime Errors → Errors that occur during program execution (e.g., divide by zero).

• Debugging → Debugging is the process of identifying and correcting errors in code.

• Exceptions → Exceptions are runtime anomalies handled using try, catch, and throw blocks.

• Input Validation → Input validation ensures that user-provided data is correct and within expected limits.

• Assertions → Assertions are statements that test assumptions in code, using the assert() macro.

■ C++ Data Structures (STL)

• Vectors → Vectors are dynamic arrays that can grow or shrink in size at runtime.

• List → A list is a doubly linked list that allows fast insertion and deletion.

• Stacks → A stack is a Last-In-First-Out (LIFO) data structure where insertion and deletion happen at the top.

• Queues → A queue is a First-In-First-Out (FIFO) structure where insertion happens at the rear and deletion at the front.

• Priority Queue → A queue where elements are dequeued based on priority instead of insertion order.

• Deque → A double-ended queue allows insertion and deletion at both ends.

• Sets → A set is a collection of unique, sorted elements.

• Unordered Set → An unordered set stores unique elements in no particular order.

• Maps → A map is a key-value container where keys are unique and sorted.

• Unordered Map → An unordered map is a hash-based key-value container with no ordering.

• Multimap/Multiset → These containers allow duplicate keys or elements.

• Iterators → Iterators are pointers-like objects used to traverse containers.

• Algorithms → STL algorithms are predefined functions such as sort(), find(), count(), reverse().

• Pairs & Tuples → Pairs store two values, tuples store multiple values in one object.

■ C++ Namespaces

• Namespaces → A namespace is a declarative region that provides a scope to identifiers (variables, functions, classes) to avoid name conflicts.

• Using Directive → The using namespace directive allows accessing namespace members without explicit qualification.

■ C++ How To

• Add Two Numbers → A basic program that reads two numbers with cin and prints their sum with cout.

• Random Numbers → Random numbers are generated using rand() and seeded with srand().

• Sleep/Delay → The program can be paused using functions like sleep() or this_thread::sleep_for().

■ C++ Reference

• Documentation of Keywords, Libraries, and Functions → The C++ reference is an official guide to all reserved keywords, functions, and standard libraries.

• Standard Library (STL) → STL provides containers, algorithms, iterators, and utilities to simplify programming.

• Header Files → Header files (e.g., , ,) contain function and class declarations that must be included in programs.
