**LBS SKILL CENTER KAZHAKOOTTAM**

**C++ SHORT NOTES**

**C++ Basics**

* **C++ HOME**: The main page or index where all C++ topics are organized. It serves as the entry point for beginners and experienced programmers alike to explore the language.
* **C++ Intro**: C++ is a high-performance, general-purpose programming language. It supports both procedural and object-oriented programming and is widely used in system/software development, game programming, and real-time applications.
* **C++ Get Started**: To begin coding in C++, you need a compiler like GCC or an IDE like Code::Blocks or Visual Studio. A basic C++ program includes the main() function and uses the #include directive.
* **C++ Syntax**: C++ syntax defines how programs must be written. Statements end with semicolons, code blocks use curly braces {}, and proper indentation helps readability.
* **C++ Output**: You use the cout object with << to print output to the screen. It is part of the iostream library and is essential for user interaction.
* **C++ Comments**: Comments are ignored by the compiler and used to explain code. Single-line comments use //, and multi-line comments use /\* ... \*/.

**📦 Variables & Input**

* **C++ Variables**: Variables store data that your program uses, such as numbers or text. You must declare the type (e.g., int, float, string) before using a variable.
* **C++ User Input**: The cin object, also from iostream, is used to take input from the user. You can read multiple inputs at once and store them in variables.
* **C++ Data Types**: Common data types include int (integer), float (decimal), char (character), bool (true/false), and string (text). Choosing the right data type is important for memory usage and performance.
* **C++ Operators**: Operators perform operations on variables and values. Categories include arithmetic (+, -), comparison (==, !=), and logical (&&, ||).

**🔡 Strings & Numbers**

* **C++ Strings**: Strings are sequences of characters and are part of the string class. You can perform operations like concatenation, finding length, and comparison.
* **C++ Math**: C++ supports basic arithmetic and advanced mathematical functions through the cmath library. Functions like sqrt(), pow(), and abs() are commonly used in calculations.
* **C++ Booleans**: The bool data type represents logical values—true or false. Boolean expressions are essential in conditions and loops.

**🔁 Control Structures**

* **C++ If...Else**: Allows the program to make decisions. If a condition is true, a specific block of code runs; otherwise, the else block runs.
* **C++ Switch**: An alternative to multiple if...else statements. It matches a variable against several case values.
* **C++ While Loop**: Repeats code as long as a condition is true. It checks the condition before each iteration.
* **C++ For Loop**: A compact loop that includes initialization, condition, and update in one line. It’s often used to iterate over arrays or perform repeated tasks.
* **C++ Break/Continue**: break exits the loop immediately, while continue skips the current iteration and continues with the next.

**🗃️ Data Collections**

* **C++ Arrays**: Arrays store multiple elements of the same type in contiguous memory. They are fixed in size and accessed using indices starting at 0.
* **C++ Structures**: Structures (struct) group different data types under one name. Useful for creating complex data models like a Person with name, age, etc.
* **C++ Enums**: Enums define a set of named integer constants. They make code more readable and manageable, especially for fixed categories.

**🧭 Memory and References**

* **C++ References**: A reference is an alias for another variable. It must be initialized when declared and cannot be changed to refer to another variable.
* **C++ Pointers**: Pointers store memory addresses and can be used for dynamic memory allocation. They are powerful but require careful handling to avoid memory leaks or crashes.

**🧩 Functions**

* **C++ Functions**: Functions are blocks of code designed to perform a specific task. They make code modular and reusable.
* **C++ Function Parameters**: Parameters pass information to functions. They can be passed by value (copy) or by reference (address).
* **C++ Function Overloading**: You can create multiple functions with the same name but different parameter lists. It improves code readability and flexibility.
* **C++ Scope**: Scope defines where a variable or function is accessible. Variables can be local (within a function) or global (outside all functions).
* **C++ Recursion**: A technique where a function calls itself to solve a problem. It must have a base condition to avoid infinite calls.

**🏗️ Object-Oriented Programming**

* **C++ Classes**: A class is a user-defined data type that represents a blueprint for objects. It can contain variables (attributes) and functions (methods).
* **C++ OOP**: Object-Oriented Programming is a paradigm based on objects and classes. Key features include encapsulation, inheritance, and polymorphism.
* **C++ Classes/Objects**: Objects are instances of classes. Each object can have different values but the same structure.
* **C++ Class Methods**: Functions defined inside a class that operate on object data. They can be public, private, or protected.
* **C++ Constructors**: Special class functions automatically called when an object is created. Used for initialization.
* **C++ Access Specifiers**: Define the accessibility of class members—public (accessible everywhere), private (accessible only inside class), and protected.
* **C++ Encapsulation**: Hiding the internal details of objects and exposing only necessary parts. Achieved using access specifiers.
* **C++ Inheritance**: Allows one class to inherit the properties of another. Supports code reuse and hierarchy.
* **C++ Polymorphism**: Means "many forms"—functions or methods can behave differently depending on the object or data type.

**🧪 Advanced Topics**

* **C++ Templates**: Allow functions and classes to work with any data type. Useful in generic programming.
* **C++ Files**: File handling is done using ifstream (read), ofstream (write), and fstream (both). Enables reading/writing data to files.
* **C++ Exceptions**: Provides a way to handle runtime errors using try, catch, and throw blocks. Helps make code robust.
* **C++ Date**: C++ provides date and time handling through libraries like <ctime> and <chrono>. Used for logging, scheduling, and measuring time.