Essential Guide to OOP Concepts and C++ Exam Strategies

These topics typically appear in **Section B** of the C-DAC C-CAT exam and interview rounds for software roles.I. Core OOP Concepts (Notes & Definitions)

- **Encapsulation:**
  - **Definition:** It is the mechanism of binding data and code together and keeping them safe from outside interference. It is often described as "Information Hiding and Data Binding" 1.
  - **Purpose:** It prevents the program from becoming interdependent and complicated 2.
- **Inheritance:**
  - **Definition:** The capability to share and extend the functionality of an existing class. It creates an **"Is-A"** relationship (e.g., A Car *is a* Vehicle) 3.
  - **Benefit:** Its primary purpose is **Reusability** 4. It allows making general classes into more specific classes 5.
  - **UML Representation:** Represented by a line with an arrowhead pointing toward the parent (base) class 6.
- **Polymorphism:**
  - **Definition:** The ability of a message or function to be displayed in more than one form.
  - **Types:**
    - **Compile-time:** Method Overloading and Operator Overloading.
    - **Run-time:** Virtual Functions 7.
- **Abstraction:**
  - **Definition:** Hiding essential details that do not contribute to the essential characteristics of an object 4.
  - **Implementation:** Achieved through Abstract classes and Interfaces 8.

II. Important C++ OOP Points (Exam Specifics)

1. Constructors and Destructors

- **Usage:** Constructors are used to initialize objects and allocate memory for them 4.
- **Default Constructor:** If a user does not provide a constructor, the compiler will provide a default constructor. However, if *any* constructor is defined by the user, the default constructor is **not** provided by the compiler 4.
- **Destructors:** Used to deallocate memory. They cannot be overloaded.

2. Relationships Between Classes

- **Association:** A relationship identified between two object entities 3.
- **Aggregation:** Represents a "Has-A" relationship (e.g., A Car *has an* Engine). It is known as a "Part-of" relationship 3.
- **Composition:** A strong form of aggregation where the child object's lifecycle is dependent on the parent. An object containing another object is called Composition 9.

3. Virtual Functions & Abstract Classes

- **Virtual Functions:** A function declared in a base class that is redefined (overridden) by a derived class.
- **Runtime Polymorphism:** This is achieved *only* when a virtual function is accessed through a **pointer** to the base class 10.
- **Pure Virtual Function:** A function with no definition in the base class (e.g., `virtual void show() = 0;`).
- **Abstract Class:** A class containing at least one pure virtual function.
- **Constraint:** You **cannot create instances** (objects) of an abstract class 11.
- **Pointers:** However, you *can* create pointers to an abstract class 10.

4. Friend Functions

- **Definition:** A function that is not a member of a class but has access to its private and protected members.
- **Demerits:** It breaks encapsulation concepts 12.
- **Exam Trick:** Certain operators cannot be overloaded using friend functions. Specifically, the assignment operator (`=`), function call (`()`), subscript (`[]`), and arrow (`->`) must be member functions. The exam papers specifically note that the `=` operator cannot be overloaded through friend functions 10.

III. Tricks and "Traps" for C-CAT Section B

- **Output Tracing Trap:** Be careful with questions involving `sizeof`.
  - *Example:* `sizeof` a character constant in C is 4 bytes (treated as `int`), but in C++ it is 1 byte 13.
  - *Pointer Arithmetic:* `ptr + 1` increments the address by `sizeof(type)`. If `ptr` is an `int*` (4 bytes), `ptr+1` adds 4 to the address, not 1 14.
- **Operator Precedence:**
  - Unary operators (like `++`, `--`, `!`) generally have higher precedence than binary operators. The postfix `++` has higher precedence than the prefix `*` (dereference) 15.
- **Static Variables:**
  - Static variables retain their value between function calls. If a function increments a static variable and is called twice, the variable will not re-initialize to zero; it will continue from the last incremented value 1.
- **Identify the Object:**
  - **State:** The properties/attributes of an object.
  - **Behavior:** The methods/functions of an object.
  - **Identity:** The property that distinguishes one object from all others 11.

IV. Sample Exam Questions (from Sources)

- **Question:** Which feature allows a small change in user requirements without requiring large changes to the system?
  - **Answer: Modularity** (though sometimes associated with

Abstraction/Encapsulation depending on context options) 16.

- **Question:** A derived class inherits what from the base class?
  - **Answer:** It inherits data members and member functions, but **not** constructors and destructors 4.
- **Question:** Shallow copy is defined as?
  - **Answer:** Member-wise copying of objects 17.
- **Question:** What is the output of `printf("%d", sizeof('A'));` in C?
  - **Answer: 4** (In C, character constants are integers) 13.
- **Question:** If x is a Boolean variable, `x + x = ?`
  - **Answer:** `x` (Idempotent Law) 18.

V. Syllabus for Preparation

To cover the "OOPS" section for C-CAT effectively, focus on these topics listed in the syllabus 19:

1. **Structure vs. Class:** Difference in default access specifiers (Public in Struct, Private in Class).
2. **Constructors/Destructors:** Default, Parameterized, Copy Constructor.
3. **Operator Overloading:** Unary and Binary.
4. **Inheritance:** Single, Multiple, Multilevel, Hybrid, and Diamond Problem (Virtual Inheritance).
5. **Polymorphism:** Virtual Tables (vptr/vtable logic is a common advanced interview topic).
6. **Templates:** Function and Class templates.
7. **Exception Handling:** `try`, `catch`, `throw`.