PAPER-IX: OBJECT ORIENTED PROGRAMMING USING C++

COMPLETE QUESTION BANK BY UNIT (20 Short, 15 Medium, 10 Long, 5 Comprehensive)

Unit I: Principles of OOP, C++ Fundamentals & Functions

1 Mark Questions (Short Answer / Objective)

Q1. What is the basic philosophy behind the Object-Oriented Programming (OOP) paradigm?

Q2. List any four basic concepts of Object-Oriented Programming.

Q3. Define an 'Object' in the context of OOP.

Q4. What is a 'Class' in C++?

Q5. Define the concept of Data Encapsulation.

Q6. What do you understand by Data Abstraction?

Q7. What is Inheritance in OOP?

Q8. Define Polymorphism with a single sentence description.

Q9. State any two major benefits of utilizing the OOP paradigm over procedural styles.

Q10. Name any two standard applications of Object-Oriented Programming systems.

Q11. How does C++ differ from standard C regarding programming paradigm support?

Q12. What is a token in C++? Give two examples.

Q13. Name the four categories of built-in data types supported natively in C++.

Q14. What is the purpose of the scope resolution operator (::) in C++?

Q15. Explain the usage of the memory management operator 'new'.

Q16. What is the function of the 'delete' operator in C++?

Q17. Define reference variables in C++ with their basic notation rules.

Q18. What is an inline function in C++?

Q19. Under what conditions might the C++ compiler ignore an 'inline' request function prefix?

Q20. What are default arguments in a C++ function parameter list?

2 Mark Questions

Q1. Differentiate between dynamic binding and message passing in OOP systems.

Q2. How do tokens, identifiers, and keywords differ from each other in C++?

Q3. Explain the difference between reference variables and standard pointer variables in C++.

Q4. State the rules and default values associated with initializing default arguments in a function.

Q5. What are const arguments in C++ functions? Why are they used?

Q6. Explain the working principle and performance benefits of inline functions.

Q7. What is a friend function? State its main defining characteristic regarding class access.

Q8. Differentiate between the structure layout of a C program and a C++ program structure.

Q9. Explain how the operators 'cin' and 'cout' work with stream extraction and insertion operators.

Q10. What are structural control statements? List the three categories available in C++.

Q11. Explain type compatibility and dynamic casting basics in C++ expressions.

Q12. What is an abstract data type (ADT)? How is it realized in C++?

Q13. Why is the friend function considered a bridge between two independent classes?

Q14. List four object-oriented programming languages other than C++.

Q15. What happens when a friend function is declared inside the private section of a class?

5 Mark Questions

Q1. Discuss the structural benefits and real-world applications of Object-Oriented Programming.

Q2. Explain the basic concepts of OOP: Encapsulation, Abstraction, Inheritance, and Polymorphism with examples.

Q3. Compare C and C++ across five distinct functional domains including memory management and paradigm shift.

Q4. Explain the structure of a standard C++ program with a complete code snippet detailing headers, namespace, and main structures.

Q5. Describe the distinct argument passing mechanisms in C++ (Pass by Value, Pass by Reference, Pass by Pointer) with snippets.

Q6. What are inline functions? Discuss their advantages, limitations, and situations where inline expansion fails.

Q7. Explain the concept of a Friend Function. Write a C++ program showcasing a friend function tracking private elements of a class.

Q8. Elaborate on the data types and data qualifiers supported by C++ along with their system memory bounds.

Q9. Discuss the operations and priorities of operators in C++, highlighting memory management and scope resolution operators.

Q10. Write a C++ program to demonstrate function execution using default arguments and const qualifiers inside parameters.

8 Mark Questions

Q1. Provide a comprehensive analysis of the core characteristics of Object-Oriented Programming, comparing them directly against structural procedural programming paradigms with concrete illustrations.

Q2. Design and write a complete, modular C++ program to find the maximum of two numbers belonging to two different classes using a common friend function shared across both classes.

Q3. Write an essay detailing the execution mechanics of function calls in C++. Contrast standard function call overheads with inline expansion pipelines, supported by code and memory diagrams.

Q4. Construct a fully operational C++ program that implements a custom banking environment using expressions, tokens, and control structures to manage deposit, withdrawal, and balance inquiries safely.

Q5. Explain the various argument passing techniques in C++. Write a single C++ program that swaps two integer variables using all three methods, explaining the stack allocation variations of each.

Unit II: Classes, Objects, Constructors & Destructors

1 Mark Questions (Short Answer / Objective)

Q1. What is the default access specifier for members inside a C++ class?

Q2. How do you define a member function outside its class declaration block?

Q3. What is a nested member function in C++?

Q4. Can private member functions be accessed directly by an object instance?

Q5. How is memory allocated for class member variables vs member functions?

Q6. What is a static data member in a C++ class definition?

Q7. How do you initialize a static data member outside its class block?

Q8. What is a static member function? What elements can it access?

Q9. Define an 'array of objects' using a short declaration syntax example.

Q10. Can an object instance be passed as a function argument by value?

Q11. What is a constructor in C++?

Q12. State the return type specification required for a C++ constructor function.

Q13. What is a parameterized constructor?

Q14. Can a class contain multiple constructors? What is this technique called?

Q15. Define a copy constructor.

Q16. What is a dynamic constructor?

Q17. What is dynamic initialization of objects?

Q18. What is a destructor? Write its naming convention format indicator.

Q19. Can a destructor accept arguments or parameters?

Q20. When does a destructor automatically execute during execution lifecycles?

2 Mark Questions

Q1. Differentiate between private and public access specifiers within classes.

Q2. Explain the memory allocation behavior when multiple object instances of a class are instantiated.

Q3. What are the special characteristics of static data members in C++?

Q4. Explain why static member functions can only access static data or static functions.

Q5. Show how objects can be passed as arguments and returned from a function.

Q6. What is a constructor with default arguments? Provide a brief code syntax example.

Q7. Differentiate between copy constructor and assignment operator execution states.

Q8. Explain the concept and necessity of a destructor function in resource recycling.

Q9. What is the role of an implicit default constructor generated by the compiler?

Q10. How do arrays within a class differ fundamentally from arrays of objects?

Q11. Explain the syntax and usage of making an outside function inline within classes.

Q12. What is a private member function? Write a scenario where it is required.

Q13. How does dynamic initialization of objects allow runtime variables to shape constructors?

Q14. Explain dynamic constructors and how they link with runtime memory allocations.

Q15. What happens if an object goes out of scope? Trace the destruction progression sequence.

5 Mark Questions

Q1. Explain how member functions of a class are defined both inside and outside the class boundaries, providing clear code segments.

Q2. Discuss the characteristics and use cases of static data members and static member functions with a working class script.

Q3. Explain the concept of an Array of Objects. Write a C++ program that tracks an array of 5 Employee objects (ID, Name, Salary).

Q4. What are constructors? Detail the structural rules for declaring constructors and explain constructor overloading.

Q5. Explain the Copy Constructor with its explicit syntax, detailing scenarios where it is implicitly invoked by the runtime.

Q6. What is dynamic initialization of objects? Provide a complete C++ program demonstrating this concept using runtime user inputs.

Q7. Discuss Destructors in C++. Detail their properties, execution order, and structural formatting inside class definitions.

Q8. Write a C++ program that illustrates passing object structures as function parameters via both call-by-value and call-by-reference.

Q9. Explain the implementation details of arrays within a class structure, creating a processing loop example for a 'Student' mark sheet.

Q10. Discuss how private member functions can be leveraged within nested member function calls to hide auxiliary operations.

8 Mark Questions

Q1. Design a C++ class named 'Matrix' that handles a dynamic 2D array inside. Implement default constructors, parameterized constructors, dynamic constructors, and a destructor to prevent memory leaks.

Q2. Write a comprehensive program managing an inventory tracking layout. Utilize static data members to track total items, static member functions for summary reporting, and arrays of objects for storage.

Q3. Explain the lifecycle of objects in C++. Construct a program tracking execution pathways across global, local, static, and dynamically allocated objects, logging constructor and destructor text strings.

Q4. Elaborate on the types of constructors supported by C++. Write a program for a 'Complex' number class that implements four distinct variations of constructors, enabling versatile object generation forms.

Q5. Create a program featuring a class 'Account'. Showcase structural member access limits, nesting operations, passing objects to external evaluation units, and explicit object return strategies.

Unit III: Inheritance & Polymorphism

1 Mark Questions (Short Answer / Objective)

Q1. What is the core objective of inheritance in object-oriented programming?

Q2. Name the class whose properties are inherited by a subclass.

Q3. What are the three visibility modes available during class derivation?

Q4. Define single inheritance.

Q5. What is multiple inheritance?

Q6. How does hierarchical inheritance differ from multilevel inheritance?

Q7. What problem is solved by using virtual base classes in hybrid networks?

Q8. Define an abstract class.

Q9. What is a pure virtual function?

Q10. What is a member class within the context of class containment?

Q11. Define polymorphism in C++.

Q12. What is compile-time polymorphism?

Q13. Name the operator or mechanism responsible for runtime polymorphism.

Q14. What does the 'this' pointer point to within an executing member function?

Q15. What is a pointer to a derived class variable?

Q16. Define function overloading.

Q17. What is operator overloading?

Q18. Name two operators in C++ that cannot be overloaded under any rules.

Q19. What keyword must be prefixed to a base class function to enable overriding?

Q20. What is a virtual destructor, and why is it declared?

2 Mark Questions

Q1. Differentiate between inheritance and containment (nesting of classes).

Q2. Explain the impact of the 'protected' visibility specifier during derivation loops.

Q3. What is the ambiguity problem in multiple inheritance? Show the resolution syntax.

Q4. Explain how constructors are executed in a multi-level inheritance chain hierarchy.

Q5. What makes a class abstract? Can we instantiate an abstract class?

Q6. Explain the concept of pointers to objects with a brief code assignment example.

Q7. Describe the properties and automatic availability of the 'this' pointer.

Q8. Differentiate between function overloading and function overriding.

Q9. What is a pure virtual function? Show its syntax of initialization specification.

Q10. Explain how compile-time polymorphism differs from runtime polymorphism mechanisms.

Q11. What are the restrictions on operator overloading in C++?

Q12. Show how a base class pointer can point to a derived class object instance.

Q13. Explain virtual base classes and their diamond-problem mitigations.

Q14. What is the syntax for overloading the unary minus operator inside a class?

Q15. How does a friend function assist in overloading binary operators differently than a member function?

5 Mark Questions

Q1. Explain the five types of inheritance supported by C++ with clear structural structural charts and minimal class declarations.

Q2. Discuss the diamond problem in hybrid inheritance. Show how virtual base classes resolve this problem with a program example.

Q3. Explain abstract classes and pure virtual functions, creating a code framework for a 'Shape' base with 'Circle' derivatives.

Q4. What is operator overloading? Write a C++ program to overload the binary '+' operator to add two 'Distance' objects (Feet, Inches).

Q5. Discuss the 'this' pointer in C++. Explain its hidden usage, mechanics, and write a program returning calling object instances.

Q6. Explain the mechanics of Virtual Functions in achieving runtime polymorphism, providing an executable program tracking behavior.

Q7. Describe function overloading in C++. Explain how the compiler resolves overloaded functions based on argument signatures.

Q8. Write a C++ program demonstrating constructors and destructors execution order across multiple and multilevel inheritance setups.

Q9. Explain the concepts of member classes and nesting of classes (containment) with a detailed code implementation structure.

Q10. Write a C++ program to overload unary increment (++) and decrement (--) operators using both member functions and friend setups.

8 Mark Questions

Q1. Design an academic administration database structure using hierarchical and multilevel inheritance models. Implement a 'Person' base tracking names, branching down into 'Student' and 'Faculty' subclasses with dynamic reports.

Q2. Explain runtime polymorphism in deep structural detail. Write an application utilizing base class pointers managing an array of different geometric shape objects, calling virtual methods to print calculated areas dynamically.

Q3. Develop a complete 'String' handling class in C++ by overloading operators: '+' for concatenation, '==' for equality checks, '=' for safe deep copying, and '<<' / '>>' streams for direct input/output manipulations.

Q4. Provide an analytical breakdown of visibility modes (`public`, `protected`, `private`) in inheritance. Create a three-tier inheritance testing matrix program to prove member reachability configurations.

Q5. Write a program showcasing an automated payroll tracking layout. Combine abstract classes, virtual base structures, virtual destructors, and pointer-to-object arrays to calculate salaries across distinct employment variants.

Unit IV: Console I/O, Manipulators & File Management

1 Mark Questions (Short Answer / Objective)

Q1. What is a stream in the context of C++ input and output systems?

Q2. Name the root base class of the C++ standard stream hierarchy layout.

Q3. Which stream class handles standard input operations?

Q4. Name two unformatted console I/O member functions.

Q5. What is the purpose of the 'getline()' function in stream processing?

Q6. Define a manipulator in C++.

Q7. Which header file must be included to use parameterized manipulators?

Q8. What does the 'setw()' manipulator achieve?

Q9. Name the stream class used specifically for file output operations.

Q10. Which member function is explicitly called to close an open file stream?

Q11. What does the file mode 'ios::app' signify?

Q12. What is the default mode when opening a file with an 'ifstream' object?

Q13. Which function detects if the file pointer has reached the end-of-file?

Q14. What is the purpose of the 'seekg()' member function?

Q15. What does 'tellp()' return during file writing operations?

Q16. Differentiate between 'seekp' and 'seekg' function tasks.

Q17. Name two functions used for unformatted sequential binary file input/output.

Q18. What is a command-line argument?

Q19. What does the first argument parameter 'argc' indicate in main functions?

Q20. What is the role of the 'bad()' function in file error checking routines?

2 Mark Questions

Q1. Differentiate between formatted and unformatted console input/output operations.

Q2. Explain how the ios class width() and precision() formatting functions operate.

Q3. What is the difference between manipulating streams via ios flags versus using manipulators?

Q4. Show the code statement required to open a file named 'data.txt' for both reading and writing simultaneously.

Q5. Explain the role of file stream classes: ifstream, ofstream, and fstream.

Q6. What are file pointers? Name the two types of file pointers managed inside C++ streams.

Q7. Explain the differences between ios::ate and ios::app file opening modes.

Q8. Describe the basic approach for detecting runtime file tracking errors using the fail() function.

Q9. What information does the 'argv' parameter matrix pack during program startups?

Q10. Write a snippet showing how to read an array of objects from a binary file using read().

Q11. Explain the usage of the setfill() manipulator with a short structural example.

Q12. What happens if you open an existing file in standard ios::out mode without flags?

Q13. Explain how random access is achieved in files using relative offset parameters.

Q14. Describe the difference between text mode and binary mode file operations in C++.

Q15. How do command-line arguments enhance the modularity of file processing programs?

5 Mark Questions

Q1. Explain the C++ Stream Class Hierarchy layout, detailing the relationships between ios, istream, ostream, and iostream.

Q2. Discuss formatted console I/O using ios member functions (width, precision, fill, setf, unsetf) with a formatting code script.

Q3. What are manipulators? Explain both non-parameterized and parameterized manipulators (endl, setw, setprecision, setfill) with code.

Q4. Write a C++ program that opens a text file, counts the total number of characters, words, and lines present inside, and outputs the count.

Q5. Explain file opening modes in C++. Provide a descriptive grid tracking eight distinct file tracking mode combinations.

Q6. Describe sequential file handling in C++. Write a program using put() and get() to copy a binary asset structure.

Q7. Discuss random access file operations. Explain the syntax and behavior of seekg, seekp, tellg, and tellp with positional diagrams.

Q8. Write a C++ program that demonstrates writing and reading structured records (objects) to a binary file using write() and read().

Q9. Explain error handling routines during file tracking operations. Detail the roles of eof(), fail(), bad(), and good().

Q10. Write a C++ program that accepts two file names as command-line arguments and appends the content of the first file onto the second.

8 Mark Questions

Q1. Design a robust, menu-driven C++ program managing a 'Student' registry database file via binary file operations. Implement functionality to insert new records, view all records, search by roll number, and modify existing entries via random access file pointers.

Q2. Write an essay with supporting program layouts explaining the internal structure of C++ Console I/O systems. Contrast traditional unformatted stream character processing with formatted fields and custom manipulators.

Q3. Develop a comprehensive file duplication engine that operates via command-line options. The utility must validate arguments, handle file absence states gracefully with stream state flags, track read markers, and provide precise operational integrity logs.

Q4. Explain file pointer manipulations comprehensively. Write a diagnostic program that writes a sequence of objects to an fstream block, reads them backward using relative seeking offsets, logs stream indices, and handles structural integrity exceptions.

Q5. Create a program featuring a low-level log file management class. The module must leverage streams to enforce file protection rules, handle simultaneous multi-mode flags, generate formatted ledger columns via manipulators, and track device write anomalies.