C++ (Foundation of Programming)

Important Note for Students:
This list of questions and answers is like a helpful guide for your upcoming interview. It's designed to give you an idea of what to expect and help you get ready.

But remember:

1. **Variety of Questions:** The same questions can be asked in many different ways, so don't just memorise the answers. Try to understand the concept behind each one.
2. **Expect Surprises:** There might be questions during your interview that are not on this list. It's always good to be prepared for a few surprises.
3. **Use This as a Starting Point:** Think of this material as a starting point. It shows the kind of questions you might encounter, but it's always good to study beyond this list during your course.

OOP

1. What is Object-Oriented Programming (OOP)?
Object-Oriented Programming is a programming paradigm that organizes data and behaviour into reusable structures called objects.

2. What are the four main principles of OOP?
The four main principles of OOP are encapsulation, inheritance, polymorphism, and abstraction.

3. What is encapsulation in C++?
Encapsulation is the process of combining data and functions into a single unit called a class. It helps in hiding the implementation details and provides data protection.

4. What is a class in C++?
A class is a user-defined data type that encapsulates data and functions into a single entity. It serves as a blueprint for creating objects.

5. What is inheritance in C++?
Inheritance is a mechanism that allows a class to inherit properties and behaviors from another class. It promotes code reuse and supports the concept of hierarchical relationships.

6. What is polymorphism in C++?
Polymorphism is the ability of an object to take on many forms. In C++, it can be achieved through function overloading and virtual functions.

7. What is function overloading in C++?
Function overloading is a feature that allows multiple functions with the same name but different parameters. The compiler determines the appropriate function to call based on the arguments provided.

8. What is abstraction in C++?
Abstraction is the process of simplifying complex systems by breaking them down into smaller, more manageable parts. It involves hiding unnecessary details and exposing only the essential features.

9. What is a constructor in C++?
A constructor is a special member function of a class that is automatically called when an object of that class is created. It is used to initialize the object's data members.

10. What is the difference between a class and an object in C++?
A class is a blueprint or template for creating objects, while an object is an instance of a class. In other words, a class defines the properties and behaviors, whereas an object represents a specific instance of those properties and behaviors.

Encapsulation

1. Why is encapsulation important in C++?
Encapsulation helps in creating more modular and maintainable code. It provides data protection and allows for better control over the behavior of the class.

2. How does encapsulation achieve data hiding in C++?
Encapsulation achieves data hiding by making the data members of a class private, which means they can only be accessed and modified through the public member functions of the class.

3. What is the role of access specifiers in encapsulation?
Access specifiers (public, private, and protected) in C++ determine the visibility and accessibility of the class members. They control how the data and functions of a class can be accessed by other parts of the program.

4. What is the static keyword in C++?
The static keyword in C++ is used to define class-level or static members. Static members belong to the class itself rather than to individual objects of the class.

5. What is a static data member in C++?
A static data member is a class-level variable that is shared by all objects of the class. It is declared using the static keyword and exists independently of any particular object.

6. How is a static data member different from an instance (non-static) data member?
Unlike an instance data member, a static data member is associated with the class itself rather than with individual objects. There is only one copy of a static data member shared among all objects.

7. What is a static member function in C++?
A static member function is a function that belongs to the class rather than to any particular object. It can be called using the class name without the need for an object.

8. Can a static member function access non-static data members of a class?
No, a static member function cannot directly access non-static data members because it does not have access to any specific object's data. It can only access other static members and global variables.

9. What is the benefit of using the static keyword in C++?
The static keyword provides several benefits, such as creating class-level variables, allowing access to class-specific functions without an object, and controlling the scope and lifetime of variables and functions within a program.

Inheritance

1. What is the base class and derived class in inheritance?
The base class is the class from which properties and behaviors are inherited, while the derived class is the class that inherits those properties and behaviors.

2. How is inheritance denoted in C++?
In C++, inheritance is denoted by using the colon (:) followed by the access specifier and the name of the base class after the derived class declaration.

3. What are the different types of inheritance in C++?
C++ supports multiple types of inheritance, including single inheritance, multiple inheritance, multilevel inheritance, and hierarchical inheritance.

4. What is single inheritance in C++?
Single inheritance is a type of inheritance where a derived class inherits properties and behaviors from a single base class.

5. What is multiple inheritance in C++?
Multiple inheritance is a type of inheritance where a derived class can inherit properties and behaviors from multiple base classes.

6. What is multilevel inheritance in C++?
Multilevel inheritance is a type of inheritance where a derived class is derived from another derived class, creating a chain of inheritance.

7. What is hierarchical inheritance in C++?
Hierarchical inheritance is a type of inheritance where multiple derived classes inherit properties and behaviors from a single base class.

8. What is the access specifier used in inheritance?
In C++, the access specifiers (public, private, and protected) are used to determine the accessibility of the inherited members in the derived class.

9. How does inheritance promote code reuse in C++?
Inheritance allows the derived class to inherit the properties and behaviors of the base class, reducing the need to rewrite code. It promotes code reuse and helps in creating more modular and maintainable programs.

Polymorphism & Data abstraction

1. What is compile-time (static) polymorphism in C++?
Compile-time polymorphism is achieved through function overloading and operator overloading, where the appropriate function or operator is determined during compile-time based on the arguments or operands.

2. What is runtime (dynamic) polymorphism in C++?
Runtime polymorphism is achieved through virtual functions and function overriding. The function to be executed is determined at runtime based on the actual object type rather than the declared type.

3. What is a virtual function in C++?
A virtual function is a member function of a base class that is declared using the virtual keyword. It is intended to be overridden in derived classes and allows for runtime polymorphism.

4. What is function overriding in C++?
Function overriding is the process of providing a different implementation of a virtual function in a derived class. It allows the derived class to provide its own specific behavior while preserving the interface of the base class.

5. What is data abstraction in C++?
Data abstraction is a concept that focuses on hiding unnecessary implementation details and exposing only essential information to the outside world. It is achieved through the use of classes and access specifiers.

6. How is data abstraction implemented in C++?
Data abstraction is implemented by defining a class that encapsulates the data and provides public interfaces (member functions) to interact with the data. The private data members are hidden from direct access.

7. What is the benefit of using data abstraction in C++?
Data abstraction helps in achieving better modularity and code maintenance. It allows for better control over data access and modification, preventing unwanted changes and ensuring data integrity.

9. How are polymorphism and data abstraction related in C++?
Polymorphism and data abstraction are closely related concepts in C++. Polymorphism allows objects to be treated uniformly, regardless of their specific types, which enhances data abstraction by providing a high level of abstraction and code reusability.

Constructor & Exception handling

1. What is a constructor in C++?
A constructor is a special member function of a class that is automatically called when an object of that class is created. It is used to initialize the object's data members.

2. What is the purpose of a constructor in C++?
The purpose of a constructor is to initialize the state of an object. It allows for the proper allocation of resources, setting default values, and performing any necessary setup tasks.

3. Can a constructor have a return type in C++?
No, a constructor does not have a return type. It is implicitly invoked when an object is created and does not return a value explicitly.

4. What is a destructor in C++?
A destructor is a special member function of a class that is automatically called when an object goes out of scope or is explicitly destroyed. It is used to release resources and perform cleanup tasks.

5. What is the purpose of a destructor in C++?
The purpose of a destructor is to free resources and clean up any memory allocated by the object. It is used to ensure proper cleanup and prevent memory leaks.

6. Can a destructor have parameters in C++?
No, a destructor does not take any parameters. It is invoked automatically and does not require explicit calling from the user.

7. What is exception handling in C++?
Exception handling is a mechanism in C++ that allows for handling and managing runtime errors or exceptional situations that may occur during program execution. It helps in preventing program crashes and allows for graceful error handling.

8. What are the keywords used in exception handling in C++?
The keywords used in exception handling in C++ are try, catch, and throw. The try block is used to enclose the code that may potentially throw an exception, catch block is used to handle and respond to exceptions, and throw is used to explicitly raise an exception.

9. How does exception handling work in C++?
When an exception occurs, the program transfers control to the nearest catch block that can handle the exception type. If no suitable catch block is found, the program terminates. Exception handling allows for catching and handling exceptions, enabling error recovery or graceful termination of the program.

10. Can we define custom exception classes in C++?
Yes, in C++, it is possible to define custom exception classes by inheriting from the standard exception class or any of its derived classes. Custom exception classes can provide more specific information about the exceptional situation and allow for more precise exception handling.