CPP INTERVIEW QUESTIONS BANK (BASIC LEVEL)

ILLINDA CHENNAKESAV

1. What is C++?

C++ is a general-purpose programming language that supports procedural, object-oriented, and generic programming. It was developed by Bjarne Stroustrup.

2. What are the key features of C++?

- Object-Oriented Programming
- Operator Overloading
- Inheritance
- Polymorphism
- Encapsulation
- Abstraction
- Templates

3. What is an object in C++?

An object is an instance of a class. It can hold both data (attributes) and methods (functions).

4. What is a class in C++?

A class is a blueprint for creating objects. It encapsulates data for the object and methods to manipulate that data.

5. Explain what a constructor is.

A constructor is a special type of member function of a class that is executed whenever an object of that class is instantiated.

6. What is the difference between a constructor and a destructor?

- **Constructor:** Initializes an object when it's created.
- **Destructor:** Cleans up when an object is destroyed.

7. What are access specifiers in C++?

Access specifiers define the accessibility of class members. Types include:

- **public:** Accessible from outside the class.
- **private:** Accessible only from within the class.
- **protected:** Accessible in derived classes.

8. What is inheritance?

Inheritance allows one class to inherit attributes and methods from another class, promoting code reusability.

9. What is polymorphism?

Polymorphism allows objects of different classes to respond to the same function call in different ways. It can be achieved using function overloading or overriding.

10. Explain function overloading.

Function overloading is when multiple functions with the same name but different parameters are defined.

11. What is operator overloading?

Operator overloading allows C++ operators to be redefined and used with user-defined types.

12. What is a virtual function?

A virtual function is a member function in the base class that can be overridden in a derived class. It enables dynamic (run-time) polymorphism.

13. What is an abstract class?

An abstract class contains at least one pure virtual function. It cannot be instantiated directly and is intended to be a base class.

14. What is a template?

Templates allow functions or classes to operate with generic types, enabling code reuse for any data type.

15. What is the use of the `this` pointer in C++?

`this` is a pointer that refers to the current instance of the class.

16. What is the difference between `new` and `malloc()`?

- **new:** Allocates memory and calls the constructor.
- **malloc():** Allocates memory but does not call the constructor.

17. What is a reference variable?

A reference variable provides an alias for an existing variable.

18. What is the difference between `==` and `=` operators?

- `=` is the assignment operator.
- `==` is the equality operator.

19. What is the difference between a shallow copy and a deep copy?

- **Shallow Copy:** Copies the memory address.

- **Deep Copy:** Copies the actual values stored in the objects.

20. What is the difference between a stack and a heap?

- **Stack:** Memory is managed automatically, and variables are removed once the function call ends.
- **Heap:** Memory is managed manually, and it remains until explicitly freed.

21. What is a pointer?

A pointer is a variable that holds the memory address of another variable.

22. What is NULL pointer in C++?

A NULL pointer is a pointer that points to nothing (address 0).

23. What is `delete` in C++?

`delete` is used to deallocate memory that was allocated using `new`.

24. What is the function of `static` keyword?

The `static` keyword makes variables or functions maintain their value across multiple calls.

25. What is `const` keyword?

The `const` keyword defines a variable whose value cannot be changed after initialization.

26. Explain the concept of "constructor overloading."

Constructor overloading allows a class to have more than one constructor with different parameter lists.

27. What is a friend function in C++?

A friend function can access the private and protected members of a class, though it is not a member of the class.

28. What are function pointers in C++?

A function pointer is a pointer that points to the address of a function.

29. What is `namespace` in C++?

A `namespace` is a declarative region that provides scope to identifiers (names of types, functions, variables) to avoid naming conflicts.

30. What is the use of `extern` keyword in C++?

`extern` is used to declare a global variable or function in another file.

31. What are the types of polymorphism in C++?

1. Compile-time (static) polymorphism: Achieved through function and operator overloading.
2. Run-time (dynamic) polymorphism: Achieved through inheritance and virtual functions.

32. What is a copy constructor?

A copy constructor is used to create a new object as a copy of an existing object.

33. What is the difference between a class and a struct in C++?

In a class, members are private by default, while in a struct, they are public by default.

34. Explain the term "recursion."

Recursion is a process where a function calls itself directly or indirectly.

35. What is the use of `inline` function in C++?

An `inline` function is expanded in line when it is called, which may improve performance by reducing the overhead of a function call.

36. What is a `vtable`?

A `vtable` (Virtual Table) is a mechanism used in C++ to support dynamic (run-time) polymorphism through virtual functions.

37. What is the significance of a virtual destructor?

A virtual destructor ensures that destructors are called in the correct order in a class hierarchy when deleting objects.

38. What is RAII (Resource Acquisition Is Initialization)?

RAII is a programming concept in C++ where resources are acquired and released through object lifetime, often through constructors and destructors.

39. What is a segmentation fault?

A segmentation fault occurs when a program tries to access an invalid memory location.

40. What is multiple inheritance?

Multiple inheritance allows a class to inherit from more than one base class.

41. What is function overriding?

Function overriding occurs when a derived class has a definition for one of the member functions of the base class.

42. What is the `explicit` keyword used for?

The `explicit` keyword prevents the compiler from using implicit constructors for type conversions.

43. What is typecasting in C++?

Typecasting is the conversion of one data type into another. There are two types:

- **Implicit Typecasting:** Automatically performed by the compiler.
- **Explicit Typecasting:** Manually done by the programmer using casting operators like `(int)` or C++-specific `static_cast<>`.

44. What are the types of typecasting in C++?

1. **Static Cast** (`static_cast<>`)
2. **Dynamic Cast** (`dynamic_cast<>`)
3. **Const Cast** (`const_cast<>`)
4. **Reinterpret Cast** (`reinterpret_cast<>`)

45. What is an exception in C++?

An exception is an error-handling mechanism in C++ that handles runtime errors, using `try`, `catch`, and `throw` blocks.

46. What is a try-catch block?

A `try-catch` block is used to handle exceptions. Code that may throw an exception is placed in a `try` block, and the `catch` block catches and handles that exception.

47. What is the difference between `throw` and `throws`?

- **throw:** Used to throw an exception.
- **throws:** C++ does not have `throws`, but `throw` is used to specify that a function might throw an exception.

48. What are the types of inheritance in C++?

1. **Single Inheritance:** A class inherits from a single base class.

2. **Multiple Inheritance:** A class inherits from more than one base class.
3. **Multilevel Inheritance:** A class inherits from a class that is already derived from another class.
4. **Hierarchical Inheritance:** Several classes inherit from a single base class.
5. **Hybrid Inheritance:** A combination of more than one type of inheritance.

49. What is dynamic memory allocation in C++?

Dynamic memory allocation refers to allocating memory during runtime using `new` and deallocating it using `delete`.

50. What is the difference between `delete` and `delete[]`?

- **delete:** Used to free memory allocated for a single object.
- **delete[]:** Used to free memory allocated for an array of objects.

51. What are function templates?

Function templates allow you to create a function that can work with any data type. Syntax:

```
template<typename T>
T add(T a, T b) {
    return a + b;
}
```

52. What are class templates?

Class templates allow classes to work with generic data types.

```
template<typename T>
class Box {
    T value;
    public:
        Box(T val) : value(val) {}
        T getValue() { return value; }
};
```

53. What is STL (Standard Template Library)?

STL is a collection of C++ template classes for data structures and algorithms like `vector`, `list`, `map`, `stack`, etc.

54. What are iterators in C++?

Iterators are objects that point to elements in containers like arrays, lists, or maps. They provide a way to access elements in these data structures sequentially.

55. What is the difference between a `vector` and a `list`?

- **Vector:** A dynamic array that provides random access and contiguous storage.
- **List:** A doubly linked list, which allows easy insertion and deletion but does not provide random access.

56. What is the difference between `map` and `unordered_map`?

- **map:** Elements are stored in a sorted order based on keys.
- **unordered_map:** Elements are stored in an arbitrary order, but it offers faster access due to hashing.

57. What is a `deque` in C++?

A `deque` (double-ended queue) allows insertion and deletion from both ends, providing more flexibility than a vector or a list.

58. What is a `stack` in C++?

A stack is a container that follows the LIFO (Last In, First Out) principle. Elements can only be inserted or removed from the top.

59. What is a `queue` in C++?

A queue is a container that follows the FIFO (First In, First Out) principle. Elements are inserted at the back and removed from the front.

60. What is the difference between `stack` and `queue`?

- **Stack:** Operates in LIFO order.
- **Queue:** Operates in FIFO order.

61. What is the difference between a pointer and a reference?

- **Pointer:** Stores the memory address of a variable. It can be reassigned.
- **Reference:** An alias for an existing variable. It cannot be reassigned after initialization.

62. What is the use of the `volatile` keyword in C++?

The `volatile` keyword tells the compiler not to optimize the variable, as its value might change unexpectedly, often used in multi-threaded programs or with hardware interrupts.

63. What is the difference between `const` and `volatile`?

- **const:** Ensures the variable cannot be changed after initialization.

- **volatile:** Ensures the variable is not optimized away, even if it appears unused in the code.

64. What is a lambda function in C++?

A lambda function is an anonymous function that can capture variables from its surrounding scope. Syntax:

auto lambda = [ ](int x) { return x + 2; };

65. What is a smart pointer in C++?

Smart pointers manage the lifetime of dynamically allocated objects, ensuring proper memory management. Types include:

- **std::unique_ptr**
- **std::shared_ptr**
- **std::weak_ptr**

66. What is the difference between `std::unique_ptr` and `std::shared_ptr`?

- **std::unique_ptr:** Maintains sole ownership of an object.
- **std::shared_ptr:** Allows multiple pointers to share ownership of the same object.

67. What is the function of `std::weak_ptr`?

A `std::weak_ptr` holds a non-owning reference to an object managed by a `std::shared_ptr`. It helps avoid cyclic dependencies.

68. What is a segmentation fault?

A segmentation fault occurs when a program tries to access memory that it is not allowed to. It usually results from accessing a null or uninitialized pointer.

69. What is a dangling pointer?

A dangling pointer refers to memory that has already been freed, yet the pointer still holds the address of the deleted object.

70. What is a memory leak?

A memory leak occurs when dynamically allocated memory is not properly deallocated, leading to wastage of memory resources.

71. What are inline functions?

Inline functions expand the function's code at the point of call, reducing the overhead of a function call.

72. What are friend classes?

A friend class can access private and protected members of another class.

73. What is a mutable keyword in C++?

The `mutable` keyword allows a member of a class to be modified even if it is a part of an object declared as `const`.

74. What is a pure virtual function?

A pure virtual function is a virtual function with no implementation in the base class, forcing derived classes to override it.

75. What is a copy assignment operator?

The copy assignment operator is used to assign one object to another of the same class. It is implemented using the `operator=`.

76. What is the Rule of Three in C++?

If a class defines one (or more) of the following:

1. Destructor
2. Copy constructor
3. Copy assignment operator

Then it likely needs to define all three.

77. What is the Rule of Five in C++?

The Rule of Five extends the Rule of Three and includes:

1. Move constructor
2. Move assignment operator

78. What is the difference between shallow copy and deep copy?

- **Shallow Copy:** Copies only the memory address (pointer) of the object, leading to multiple objects pointing to the same data.
- **Deep Copy:** Copies the actual object data, allocating separate memory for the new object.

79. What is the difference between early binding and late binding?

- **Early Binding:** Function call is resolved at compile-time (e.g., function overloading).

- **Late Binding:** Function call is resolved at run-time using virtual functions (polymorphism).

80. What is a function pointer in C++?

A function pointer stores the address of a function. It can be used to call functions indirectly.

```
void (*funcPtr)(int);
funcPtr = &someFunction;
(*funcPtr)(5);
```

81. What is the use of `assert()` in C++?

`assert()` is used to perform debugging checks. If the expression inside `assert()` evaluates to false, the program terminates with an error message.

82. What are the different types of comments in C++?

- **Single-line comment:** `//`
- **Multi-line comment:** `/* ... */`

83. What is the `auto` keyword in C++?

The `auto` keyword allows the compiler to automatically deduce the type of the variable from its initializer.

```
auto x = 10; // x is an int
```

84. What is the `decltype` keyword?

`decltype` is used to query the type of an expression at compile-time. It is useful when deducing return types of functions.

85. What is the use of `std::move`?

`std::move` is used to cast an object to an rvalue reference, enabling move semantics (transferring ownership of resources instead of copying).

86. What is move semantics in C++?

Move semantics is a feature that allows the transfer of resources from one object to another, eliminating unnecessary copying. It is implemented using move constructors and move assignment operators.

87. What is the difference between `std::move` and `std::forward`?

- **std::move:** Casts an object to an rvalue reference, enabling move semantics.

- **std::forward:** Used to forward arguments while preserving their value category (lvalue or rvalue).

88. What is an lvalue and an rvalue in C++?

- **lvalue:** Refers to an object that has a persistent address (e.g., variables).
- **rvalue:** Refers to a temporary value that does not have a persistent address (e.g., result of expressions).

89. What is a virtual destructor in C++?

A virtual destructor ensures that the destructor of a derived class is called when a base class pointer pointing to a derived class object is deleted.

90. What is the difference between a regular and a virtual function?

- **Regular Function:** Resolved at compile-time, based on the type of the pointer/reference.
- **Virtual Function:** Resolved at run-time using the vtable, enabling polymorphism.

91. What is a friend class?

A friend class can access the private and protected members of another class. It is useful when two or more classes need to work closely together.

92. What is a pure virtual function?

A pure virtual function is a function that must be overridden in derived classes. It is defined as:

virtual void someFunction() = 0;

93. What is a virtual table (vtable)?

A vtable is a lookup table used by the compiler to resolve function calls in the case of polymorphism. Each class with virtual functions has its own vtable.

94. What is function overriding?

Function overriding allows a derived class to provide a specific implementation for a function that is already defined in its base class.

95. What is a virtual base class?

In multiple inheritance, a virtual base class ensures that the base class is shared among all derived classes, avoiding the "diamond problem."

96. What is a functor in C++?

A functor (function object) is an object that can be treated like a function. It is created by overloading the `operator()`.

struct Functor {

void operator()(int x) { std::cout << x; }

};

97. What are manipulators in C++?

Manipulators are functions used to format output. Common examples include:

- `std::endl`: Inserts a newline character and flushes the output buffer.
- `std::setw()`: Sets the field width for output.
- `std::fixed`: Forces decimal numbers to be printed with a fixed-point notation.

98. What is the difference between `cin.get()` and `cin`?

- **`cin`:** Reads input until the first whitespace is encountered.
- **`cin.get()`:** Reads a single character including whitespaces like newline and spaces.

99. What is the difference between `std::endl` and `\n`?

- **`std::endl`:** Inserts a newline and flushes the output stream.
- **`\n`:** Only inserts a newline without flushing the output.

100. What is the difference between exit() and return in C++?

- **exit():** Terminates the program immediately, regardless of the point in the code.
- **return:** Exits a function and returns control to the calling function.