                  OOPs PYQs

*Authored by Arkajyoti Naskar*

**Q1) In C what is the utility of functions and discuss passing techniques. (5)**

**Ans.** By using functions, we can avoid rewriting the same logic/code again and again in a program. Thus C functions reduce redundancy and increase reusability. Functions allow programmers to develop programs in a modular fashion, breaking down complex programs into smaller, manageable pieces.

Arguments can be passed to a function by two techniques:

i) Pass by value: The values of actual parameters are copied to function’s formal

   parameters and the two types of parameters are stored in different memory

   locations. So any changes made inside functions are not reflected in actual

   parameters of the caller.

ii) Pass by reference: A reference of the actual parameters are passed to the function

    through pointers. Thus the value of the actual parameter pointer (address of the

    actual parameter) is passed, thus both the actual and formal parameters refer to

    the same locations, so any changes made inside the function are actually reflected

    in actual parameters of the caller.

**Q2) Compare Array and Structure. (2)**

**Ans.**

|  |  |
| --- | --- |
| **Array** | **Structure** |
| Elements are of the same type (Homogeneous Data Type) | Elements are not necessarily of the same type (Heterogeneous Data Type) |
| Elements can be accessed through the “[]” operator using their index | Elements can be accessed through the”.” operator using their names |
| The name of an array points to the first element in the array | The name of a structure is not a pointer |
| No keywords are required to declare an array | “struct” keyword is required to declare a structure |

**Q3) Explain Encapsulation, Inheritance and Polymorphism in the context of OOPs. (6)**

**Ans. Encapsulation:** Encapsulation in C++ is defined as the wrapping up of data, information and methods that manipulate them in a single unit. In Object Oriented Programming, Encapsulation is defined as binding together the data and the functions that manipulate them. Encapsulation helps in Data Abstraction and representation of real world objects and concepts in Object Oriented Programming.

**Inheritance:** The capability of a [class](https://www.geeksforgeeks.org/c-classes-and-objects/) to derive properties and characteristics from another class is called Inheritance. The class that derives the properties is known as the “child” or “derived” class while the class from which the properties are derived is known as the “parent” or “base” class. By enabling sharing of attributes and behaviours between parent and child classes, inheritance provides reusability. Inheritance can be done through 3 modes: private, public and protected and can be of 5 types: single, multiple, multilevel, hierarchical, and hybrid.

**Polymorphism:** Polymorphism is the ability of any data to be processed in more than one form. Through polymorphism, the same function can have different implementations across different classes. Polymorphism can be of two types: Compile time polymorphism and Run time polymorphism.

**\*Q4.)**  **State utility of static members in a class .**

 Utility of static members in a class :

     1.     If the value of the variable is not varied from object to object such type of variable is not recommended to be declared as instance variables .We have to declare such variables as the static variable or class level variable using a static modifier. a static variable can be accessed by using a class name or object reference of a class.

2 . As static variable resides in the memory at the time of class loading we can use them from the class name and object reference .if static variable are in the same class then it can be assessed directly

**Q5.) Why do we need virtual destructors ?**

 Virtual destructors in C++ are used to avoid memory leaks especially when our class contains unmanaged code, i.e., contains pointers or object handles to files, databases or other external objects. A destructor can be virtual. Let's understand why it is needed. Let's consider a scenario in which there are two classes, a base class called Base and a derived class called Derived. Now, constructors are invoked in the order of inheritance and destructors are invoked in the reverse order of inheritance. So, when an instance of the derived class is created, the constructor of the Base class would be invoked first, followed by the Derived class constructor. Similarly, the Derived class destructor would be invoked followed by the Base class destructor at the time when the object goes out of the scope or is destroyed explicitly.

**\*\*Q6.) Comment on size of an object in C++.**

 In C++ , the size of an object mainly depends on the data members of the class. size of an object is greater than or equal to sum of size of all non-static data members in that object. Static member doesn’t contribute to the size of an object because static members belong to  the class , not to a particular object . In case the class contains virtual functions , the size of virtual pointers are added to the objects .

**\*\*Q7.) State the uses of friend functions / Why do we need a friend function? (1.5/2)**

Friend Function is a non-member function that can access the private and protected members of a class in which it is declared as friend. It can be declared either in the public or private or protected part of the class.

When there are multiple classes and one class want to work with private data members or member functions of another class without inheriting it, we make that function friend which assures security of the function i.e. we can control which classes can access the function rather than making it public for all. Friend function is widely used for operator overloading also.

**\*Q8.) System provides us a default copy constructor , still we may write our copy constructor . Why ?**

While using the default copy constructor , a problem arises when a class having a pointer is to be copied . In that case ,the copied object’s pointer member and the main object’s pointer member will point to the same memory location . Whenever a change will be made to that pointer in the main object , that will reflect in that copied object also . This incident is called shallow copy . To avoid this , we write our own copy constructer .

**\*Q9.)   Compare function overloading and overriding .**

  FUNCTION OVERLOADING:

                      Function overloading provides multiple definitions of the same functions with same name but with different signatures such as number of parameters , type of parameters etc . Also , return type can be different for different definitions for a particular function . Function overloading is achieved during compile time . Function overloading can be done in global functions as well as in member functions of a class.

  FUNCTION OVERRIDING:

          Function overriding is the redefinition of base class functions in derived classes . Overridden functions should have the same signatures e.g. number of parameters , return type etc. It is achieved during runtime . Function overriding can’t be achieved without a base class-derived class relationship.

**\*\*Q10.) As a programmer how do you achieve runtime polymorphism and how does system**

**Implement support for it ?**

 Runtime Polymorphism can be achieved through a base class type pointer which will be pointing to the object of derived class and virtual functions when the base class and derived class functions have same signature.

     ``` #include<iostream>

using namespace std;

class B {

   public:

      virtual void s() {

         cout<<" In Base \n";

      }

};

class D: public B {

   public:

      void s() {

         cout<<"In Derived \n";

      }

};

int main(void) {

   D d; // An object of class D

   B \*b= &d; // A pointer of type B\* pointing to d

   b->s(); // prints "D::s() called"

   return 0;

}

```

In the above code  Base class type pointer B is pointing towards the object of derived class D .  We overridden the function s which is  present in both base and derived class. If s is being called , the function of the base class would be executed without the **virtual** keyword  , since the pointer is of base class type . But since we’re using the **virtual** keyword the function , the member function of the derived class is executed . This is called runtime polymorphism .

To support polymorphism C++ uses a special form of late binding known as the virtual table. The **virtual table** is a lookup table of functions used to resolve function calls in a dynamic/late binding manner. The virtual table sometimes goes by other names, such as “vtable”, “virtual function table”, “virtual method table”, or “dispatch table”.

First, every class that uses virtual functions (or is derived from a class that uses virtual functions) is given its own virtual table. This table is simply a static array that the compiler sets up at compile time. A virtual table contains one entry for each virtual function that can be called by objects of the class. Each entry in this table is simply a function pointer that points to the most-derived function accessible by that class.

**\*\*Q11.) What is the utility of abstract class?**

An Abstract Class is written when there are some common features shared by all the objects. Suppose in our project if we have some common features then declare that type of properties and methods in a class as abstract and we can extend wherever we want in our Module. That class will be parent to all but we can’t make an object of that function, it works as a blueprint for derived classes to follow, defining the common interface and functionality that the derived classes should have. Abstract classes can provide default implementations for some member functions, allowing derived classes to override only those functions that need different behaviour.

**\*Q12.) Write a short note on Exception Handling?**

One of the advantages of C++ over C is Exception Handling. Exceptions are abnormal conditions that a program encounters during its execution. There are two types of exceptions: a) Synchronous, b) Asynchronous (i.e., exceptions which are beyond the program’s control, such as disc failure, keyboard interrupts etc.). C++ provides the following specialised keywords for this purpose:

try: Represents a block of code that can throw an exception.

catch: Represents a block of code that is executed when a particular exception is thrown.

throw: Used to throw an exception. Also used to list the exceptions that a function throws but doesn’t handle itself.

**\*Q13.) What is the use of function template and class template? (3)**

Function template is used for creating a set of functions of same name to perform same operations with different type of data. This thing can be done using Function overloading but we have to write functions for every data type which is not desirable and can be done easily using function template. Also, Macro can be used but as there no type checking, some problem may arise in runtime.

Class template is like a function template which creates blueprint for creating classes that can handle different data types. When we want to create same class again and again with different data type, class template is used to make the job easier. It is used for creating generic and container classes.

**Q14.) In STL, in C++, what is an adapter, explain with example. (3)**

Adapters are data type from STL that adapt a container to provide specific interface. In C++ STL, an adapter is a class that adapts or modifies the behaviour of an existing container or iterator class to suit a specific purpose. These are designed to work with the existing container and iterator classes in the STL.

Example: -

Stack:- Stack adapters provide a stack-like interface to an existing container class such as a vector or deque. Stack class is implemented using a deque as its underlying container.

**Stack<int>s;**

**s.push(5);**

**s.push(10);**

**cout<<s.top()<<endl; //10;**

**s.pop();**

**cout<<s.top()<<endl;//5;**

In this example in stack, we can only push or pop from the top and can check if it’s empty but cannot perform operation with iterators.

**Q15.) What is the fundamental difference between map and hash map? (2)**

The fundamental difference between map and hash map(unordered\_map), is the way they store and organize data.

Map uses a red-black tree data-structure, so it stores elements in a sorted-order, based on the key value. The insertion, deletion and searching of elements are based on the key value with an average time complexity of O(logn).

On the other hand, hash map stores elements in an unordered fashion using a hash-table. It uses a hash function to map each key value to an index in the hash table. Insertion, deletion and searching is done with an average time complexity of O(1) here.

**Q16.) In C++, compare structure and class.(1.5)**

In C++, structure and class are both user-defined data-types that group together different variables of different data types under a single name i.e. both can encapsulate.

Unlike structure, class provides access control mechanism(access specifiers- private, public, protected) to secure data. Class also has member functions and have some additional functionalities (like abstraction, inheritance) whereas structure only consists of data members.

**\*Q17.) Compare a normal function, macro and Inline function (1.5)**

In C++, a normal function, macro and inline function all are the ways to define a reusable block of code but they have some key differences.

A normal function is defined to avid rewriting the same logic/code again and again in a program. It can be executed when it’s called by its name with required parameters. It is used to reduce redundancy and increase reusability, modularity.

Macro is pre-processor directive that simply replaces a small block of code with some define value or code during runtime. There is no type checking here, so some problem may arise in runtime.

Whereas Inline function is a function (may or may not be executed) that request the compiler to replace invocation stunt of the function by equivalent code.

**Q18.) Comment on the utility of the private member function. (1.5)**

Private member functions in C++ are used to encapsulate implementation details and protect the internal state of a class from external access. When there is function which is common to one or more classes and the common function doesn’t work independently, works as a part of another function, we make that member function private keeping the class interface simple by hiding unnecessary implementation and sensitive operations, so security is checked. However, if a private member function is working with public data members, the security of the object may be compromised, as the implementation details of the private member function can be indirectly manipulated.

**Q19.) Why do we need a virtual base class? (1.5)**

Virtual base classes are used in hybrid inheritance. It prevents multiple copies of a base class form being created when it is inherited by multiple derived class, it only creates one instance of the base class, regardless of how many times the derived class inherits from it, solving the ambiguous problem occurring before.

**Q20.) Compare Function Overloading and function template? (1)**

Function overloading allows the programmer to use the same function name for same or different implementations based on no. of arguments and type of arguments whereas Function template is used for creating a set of functions of same name to perform same operations with different type of data. In Function overloading, we have to write functions for every data type which is not desirable and can be done easily using function template making the code short and more efficient.

**Q21.) What is namespace? What are using declaration and using directive? (4)**

A namespace is declarative region that provides a scope to the identifiers inside it (the names of types, functions, variables etc.), that allows them to be organized and uniquely identified. Namespaces are used to organize code into logical groups and prevent name collisions that can occur especially when code base includes multiple libraries.

A using declaration introduces an unqualified name as a synonym for an entity declared elsewhere. It is a way to bring a specific identifier from a namespace into current scope.

A using directive is a way to bring all of the identifiers from a namespace into the current scope.

**Q22.) Consider a class has a member function that receives an object reference as argument. What will be the prototype for the function so that it can’t modify the contents of the object(s) made available to it? (2)**

void function\_name(const class\_name &object) const;

**Q23.) Discuss the role of throw verb? (2)**

**‘**throw**’** verb is used to signal an exception. When a piece of code encounters an exceptional situation or error, it can use the ‘throw’ verb to throw an exception object that describes the problem.

When a function in a class has a throw clause in its signature, it indicates that the function may throw an exception. This information can be used by the caller of the function to handle the exception properly.

**class MyClass {**

**public:**

**void myFunction() throw(int) {**

**// function implementation**

**if (condition) {**

**throw exception\_value;**

**}**

**}**

**};**

**int main() {**

**MyClass obj;**

**try {**

**obj.myFunction();**

**} catch (int exception\_value) {**

**// handle the exception**

**}**

**return 0;**

**}**

In this example, the myFunction function has a throw clause indicating that it may throw an int exception. The function is called on an instance of MyClass, obj, and is enclosed within a try-catch block to handle the exception. If the condition is met and the throw keyword is executed, the exception will be propagated to the catch block and the exception\_value will be assigned the value of the thrown int.

**Q24.) Minimal support to compare two objects? (2.5)**

Overload > or < any one operator, it will check all conditions.

**Q25.) How does the concept of namespace influence the design of a solution. (2.5)**

A namespace is declarative region that provides a scope to the identifiers inside it (the names of types, functions, variables etc.), that allows them to be organized and uniquely identified. Namespaces are used to organize code into logical groups and prevent name collisions and conflicts that can occur especially when code base includes multiple libraries.

It enable the development of reusable and modular code, can be used to implement a hierarchical organization of functionalities and to indicate the level of abstraction of different functionality.

**Q26.) What happens when an object is declared using a class template? (2.5)**

When an object is declared using a class template with some specific parameters, the compiler uses the template parameters to generate an instance of a new class where all the data members, all the member functions, static members will be created first and the non-static members will be created last. This new class is then used to create the object.

If the class with given template parameters is already created then this time only the non-static members will be created.

**Virtual Function:-** A pure virtual function (or abstract function) in C++ is a virtual function for which we can have implementation, But we must override that function in the derived class, otherwise the derived class will also become abstract class.

A class is abstract if it has at least one pure virtual function.

We can have pointers and references of abstract class type.

If we do not override the pure virtual function in derived class, then derived class also becomes abstract class.

An abstract class can have constructors.