1. **C++ Q&A**

**Q-1**

**a) Define Data Abstraction and Data Encapsulation.**

**Data Abstraction** is the concept of providing only essential information to the outside world and hiding the background details. It helps in reducing programming complexity and effort. **Data Encapsulation** is the mechanism of wrapping the data (variables) and code acting on the data (methods) together as a single unit. In encapsulation, the variables of a class will be hidden from other classes and can be accessed only through the methods of their current class.

**b) Explain the uses of scope resolution operator.**

The **scope resolution operator (::)** is used to define the context in which a variable or function resides. It is used to access global variables, define a function outside a class, and access a class's static members.

**c) Write a program which takes two integers and return the smallest of them using if–else statement.**

#include <iostream>

using namespace std;

int main() {

int a, b;

cout << "Enter two integers: ";

cin >> a >> b;

if (a < b)

cout << "The smallest number is: " << a << endl;

else

cout << "The smallest number is: " << b << endl;

return 0;

}

**d) What are Default Arguments?**

**Default Arguments** are values provided in function declarations that are automatically assigned by the compiler if the caller does not provide a value for those arguments.

**e) What are objects? How they are created?**

**Objects** are instances of a class. They are created by defining a variable of the class type. For example:

class MyClass {

// class definition

};

MyClass obj; // obj is an object of MyClass

1. [19:04]

**f) What is destructor?**

A **destructor** is a member function which destructs or deletes an object. It has the same name as the class, preceded by a tilde (~), and it does not take any arguments nor does it return any value.

**g) What is an inheritance? List the types of inheritance.**

**Inheritance** is a feature of OOP that allows a class to inherit properties and behavior (methods) from another class. The types of inheritance are:

* 1. Single Inheritance
  2. Multiple Inheritance
  3. Multilevel Inheritance
  4. Hierarchical Inheritance
  5. Hybrid Inheritance

**h) Write any four rules for writing a constructor.**

* 1. Constructors should have the same name as the class.
  2. Constructors do not have a return type.
  3. Constructors can be overloaded.
  4. Constructors are automatically called when an object is created.

**i) List the memory management operators in C++.**

The memory management operators in C++ are:

* 1. new - Allocates memory.
  2. delete - Deallocates memory.
  3. new[] - Allocates memory for an array.
  4. delete[] - Deallocates memory for an array.

1. [19:04]

**j) List any four features of OOP's.**

* 1. Encapsulation
     + **Encapsulation** is the mechanism of hiding the internal details of an object and only exposing the necessary parts. It helps in protecting the data from unauthorized access and modification.
  2. Abstraction
     + **Abstraction** is the concept of providing only essential information to the outside world while hiding the background details. It simplifies complex systems by modeling classes appropriate to the problem.
  3. Inheritance
     + **Inheritance** is a feature that allows a class to inherit properties and behavior (methods) from another class. It promotes code reusability and establishes a natural hierarchy between classes.
  4. Polymorphism
     + **Polymorphism** is the ability of different objects to respond to the same function call in different ways. It allows one interface to be used for a general class of actions, making it easier to manage and scale the code.

**k) What is Inline function?**

An **Inline function** is a function that is expanded in line when it is called. When the inline function is called, the compiler replaces the function call with the function code itself.

**l) Define Friend function.**

A **Friend function** is a function that is not a member of a class but has access to the class's private and protected members. It is declared using the friend keyword.

1. [19:05]

**Q-2**

**a) Give any four applications of Object Oriented Programming (OOPs).**

* 1. Real-time systems
  2. Simulation and modeling
  3. Object-oriented databases
  4. AI and expert systems

**b) Explain Inline Function with suitable example.**

An **Inline function** is a function that is expanded in line when it is called. Example:

#include <iostream>

using namespace std;

inline int add(int a, int b) {

return a + b;

}

int main() {

cout << "Sum: " << add(5, 3) << endl;

return 0;

}

**c) What is an array of objects? Explain with suitable example code.**

An **array of objects** is a collection of objects stored in contiguous memory locations. Example:

#include <iostream>

using namespace std;

class MyClass {

public:

int data;

void display() {

cout << "Data: " << data << endl;

}

};

int main() {

MyClass objArray[3];

for (int i = 0; i < 3; i++) {

objArray[i].data = i + 1;

objArray[i].display();

}

return 0;

}

**d) What is constructor? Explain types of constructor.**

A **constructor** is a special member function that initializes objects. Types of constructors:

* 1. Default Constructor
  2. Parameterized Constructor
  3. Copy Constructor

**e) Explain Multiple Inheritance with example.**

**Multiple Inheritance** is a feature where a class can inherit from more than one base class. Example:

#include <iostream>

using namespace std;

class Base1 {

public:

void display() {

cout << "Base1 display" << endl;

}

};

class Base2 {

public:

void show() {

cout << "Base2 show" << endl;

}

};

class Derived : public Base1, public Base2 {

};

int main() {

Derived obj;

obj.display();

obj.show();

return 0;

}

1. [19:05]

**f) Write a short note on Virtual Base Class.**

A **Virtual Base Class** is used in multiple inheritance to avoid multiple copies of the base class being inherited by the derived class. It ensures that only one copy of the base class is inherited.

**g) Write a C++ program to demonstrate Hybrid Inheritance.**

#include <iostream>

using namespace std;

class Base {

public:

void display() {

cout << "Base class" << endl;

}

};

class Derived1 : public Base {

};

class Derived2 : public Base {

};

class Hybrid : public Derived1, public Derived2 {

};

int main() {

Hybrid obj;

obj.Derived1::display(); // Resolving ambiguity

return 0;

}

**h) Write a short note on Friend Function.**

A **Friend function** is a function that is not a member of a class but has access to its private and protected members. It is declared using the friend keyword and can be useful for operator overloading.

**i) What is function overloading? Explain with suitable example.**

**Function overloading** is a feature that allows multiple functions with the same name but different parameters. Example:

#include <iostream>

using namespace std;

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

int main() {

cout << "Sum (int): " << add(5, 3) << endl;

cout << "Sum (double): " << add(5.5, 3.3) << endl;

return 0;

}

1. [19:05]

**j) Write a program to calculate area and circumference of a circle using inline function.**

#include <iostream>

using namespace std;

inline double area(double radius) {

return 3.14159 \* radius \* radius;

}

inline double circumference(double radius) {

return 2 \* 3.14159 \* radius;

}

int main() {

double radius;

cout << "Enter radius: ";

cin >> radius;

cout << "Area: " << area(radius) << endl;

cout << "Circumference: " << circumference(radius) << endl;

return 0;

}

**k) What is inheritance? Explain types of inheritance.**

**Inheritance** is a feature of OOP that allows a class to inherit properties and behavior from another class. Types of inheritance:

* 1. Single Inheritance
  2. Multiple Inheritance
  3. Multilevel Inheritance
  4. Hierarchical Inheritance
  5. Hybrid Inheritance

**l) Explain static data members and static member functions with example.**

**Static data members** are class members that are shared among all objects of the class. **Static member functions** can access only static data members. Example:

#include <iostream>

using namespace std;

class MyClass {

public:

static int count;

MyClass() {

count++;

}

static void displayCount() {

cout << "Count: " << count << endl;

}

};

int MyClass::count = 0;

int main() {

MyClass obj1, obj2, obj3;

MyClass::displayCount();

return 0;

}