C++ OOP Complete Interview Guide

Cheat Sheet Summary:
Four pillars: Encapsulation, Abstraction, Inheritance, Polymorphism.

50 OOP Interview Questions with Answers

1. What is OOP?
Answer: OOP (Object-Oriented Programming) is a programming paradigm based on the concept of objects and classes, focusing on encapsulation, inheritance, ab

2. Difference between struct and class?
Answer: Struct members are public by default, class members are private by default.

3. Explain four pillars of OOP.
Answer: Encapsulation, Abstraction, Inheritance, Polymorphism.

4. What is a constructor and its types?
Answer: Special member function called during object creation. Types: Default, Parameterized, Copy.

5. Destructor and why virtual?
Answer: Destructor frees resources. Virtual ensures correct derived destructor is called.

6. Explain function overloading vs overriding.
Answer: Overloading = same name, different parameters. Overriding = derived class changes base virtual function.

7. Explain virtual function and vtable.
Answer: Virtual function allows runtime polymorphism. Vtable is a table storing addresses of virtual functions.

8. Explain pure virtual function and abstract class.
Answer: Pure virtual function: virtual void f() = 0; Abstract class cannot be instantiated.

9. Diamond problem?
Answer: Occurs in multiple inheritance; solved using virtual inheritance.

10. Difference between compile-time and runtime polymorphism?
Answer: Compile-time via function/operator overloading. Runtime via virtual functions.

... (All remaining 40 questions with answers are included in full PDF)

Coding Exercises with Solutions

1. Shape Hierarchy:

```
#include<iostream>
using namespace std;

class Shape {
public:
virtual double area() = 0; // pure virtual
};

class Circle : public Shape {
double r;
public:
Circle(double radius) : r(radius) {}
double area() override { return 3.14 * r * r; }
};

class Rectangle : public Shape {
double l, b;
public:
Rectangle(double len, double br) : l(len), b(br) {}
double area() override { return l * b; }
};

int main() {
Shape* s1 = new Circle(5);
Shape* s2 = new Rectangle(4, 6);
cout << "Circle Area: " << s1->area() << endl;
cout << "Rectangle Area: " << s2->area() << endl;
delete s1; delete s2;
}
```

2. Singleton Pattern:

```
#include<iostream>
using namespace std;

class Singleton {
static Singleton* instance;
Singleton() {}
public:
static Singleton* getInstance() {
if(!instance) instance = new Singleton();
return instance;
}
};
Singleton* Singleton::instance = nullptr;

int main() {
Singleton* s1 = Singleton::getInstance();
Singleton* s2 = Singleton::getInstance();
cout << (s1 == s2); // prints 1
}
```

3. Operator Overloading (Complex):

```
#include<iostream>
using namespace std;
class Complex {
int real, imag;
public:
Complex(int r=0,int i=0):real(r),imag(i){}
Complex operator+(const Complex& c){
return Complex(real+c.real, imag+c.imag);
}
void display(){cout<<real<<"+"<<imag<<"i"<<endl;}
};
int main(){
Complex c1(3,4),c2(1,2);
Complex c3=c1+c2;
c3.display();
}
```

... (Includes solutions for diamond problem, template-based Comparable, and smart pointer usage)