C++ Programming Practice Questions

100 Comprehensive Programming Problems
Difficulty Levels: Beginner to Advanced
Topics Covered: Basic C++, OOP, Pointers, Memory Management, Templates, STL, File Handling, Advanced Features

---

1. Basic C++ Programs (Questions 1-25)

1.1. Basic Input/Output and Operations

Question 1: Write a C++ program that takes two numbers as input and displays their sum, difference, product, and quotient with proper exception handling for division by zero.

Question 2: Create a program that reads a character and displays whether it's a vowel, consonant, digit, or special character.

Question 3: Implement a program that converts temperature between Celsius and Fahrenheit with menu-driven approach.

Question 4: Write a program to swap two numbers using a third variable and without using a third variable.

Question 5: Create a program that calculates simple and compound interest with user inputs.

Question 6: Write a program to find the largest among three numbers using conditional statements.

Question 7: Implement a program that checks if a year is a leap year or not.

Question 8: Create a calculator using switch-case that handles basic arithmetic operations.

Question 9: Write a program that determines the type of triangle based on sides (equilateral, isosceles, scalene).

Question 10: Implement a program that calculates electricity bill based on units consumed with slab rates.

1.2. Looping Structures

Question 11: Write a program to print all prime numbers between 1 and 100.

Question 12: Create a program to find the sum of all even numbers between 1 and n.

Question 13: Implement a program to print various patterns (pyramid, diamond, number patterns).

Question 14: Write a program to find the GCD and LCM of two numbers using Euclidean algorithm.

Question 15: Create a program that reverses a number and checks if it's a palindrome.

Question 16: Implement a program to find all Armstrong numbers between 1 and 1000.

Question 17: Write a program to calculate the sum of series: 1 + 1/2 + 1/3 + ... + 1/n.

Question 18: Create a program to convert decimal to binary, octal, and hexadecimal.

Question 19: Implement a program to find the factorial of a number using recursion.

Question 20: Write a program to generate Fibonacci series up to n terms using both iteration and recursion.

Question 21: Create a program to find the sum of digits of a number until it becomes single digit.

Question 22: Implement a program to print multiplication table for a given number.

Question 23: Write a program to find all perfect numbers in a given range.

Question 24: Create a program to calculate the sum of squares of first n natural numbers.

Question 25: Implement a program to check if a string is palindrome or not.

---

2. Functions and Arrays (Questions 26-40)

2.1. Functions

Question 26: Write a function that checks if a number is prime and use it to find all prime numbers in a range.

Question 27: Create a function to calculate power of a number using recursion.

Question 28: Implement a function that returns the reverse of a number.

Question 29: Write a function with default arguments and demonstrate its usage.

Question 30: Create a recursive function to find the sum of digits of a number.

Question 31: Implement function overloading for area calculation (circle, rectangle, triangle).

Question 32: Write a program demonstrating pass by value, pass by reference, and pass by pointer.

Question 33: Create inline functions for simple mathematical operations.

Question 34: Implement a function that returns multiple values using references.

Question 35: Write a program with function templates for generic programming.

2.2. Arrays and Strings

Question 36: Write a program to find the largest and smallest element in an array.

Question 37: Implement a program to sort an array using bubble sort algorithm.

Question 38: Create a program to search an element in an array using linear and binary search.

Question 39: Write a program to merge two sorted arrays into one sorted array.

Question 40: Implement a program to find the frequency of each element in an array.

Question 41: Create a program to rotate an array by k positions.

Question 42: Write a program to find the second largest element in an array.

Question 43: Implement a program to remove duplicates from a sorted array.

Question 44: Create a program to find the union and intersection of two arrays.

Question 45: Write a program to implement matrix addition, subtraction, and multiplication.

---

3. Pointers and Memory Management (Questions 46-55)

Question 46: Write a program to demonstrate pointer arithmetic with arrays.

Question 47: Implement dynamic memory allocation for 1D and 2D arrays.

Question 48: Create a program that uses pointers to swap two numbers.

Question 49: Write a function that returns a pointer to the maximum element in an array.

Question 50: Implement a linked list with basic operations (insert, delete, display).

Question 51: Create a program that demonstrates the use of this pointer.

Question 52: Write a program to manage student records using dynamic memory allocation.

Question 53: Implement smart pointers (unique_ptr, shared_ptr) for resource management.

Question 54: Create a program that demonstrates deep copy vs shallow copy.

Question 55: Write a program to detect memory leaks using appropriate techniques.

---

4. Object-Oriented Programming (Questions 56-75)

4.1. Classes and Objects

Question 56: Create a Student class with attributes like name, roll number, marks, and methods to calculate grade and display information.

Question 57: Implement a Bank Account class with deposit, withdraw, and balance inquiry functionality.

Question 58: Write a Circle class with methods to calculate area, circumference, and diameter.

Question 59: Create a Rectangle class with methods to calculate area, perimeter, and diagonal.

Question 60: Implement a Complex Number class with overloaded operators for arithmetic operations.

4.2. Constructors and Destructors

Question 61: Create a Book class with appropriate constructors (default, parameterized, copy) and destructor.

Question 62: Implement a Date class with validation for day, month, year and methods to compare dates.

Question 63: Write a Time class with overloaded operators for addition and subtraction of time.

Question 64: Create a String class that mimics basic string operations.

Question 65: Implement a Stack class using arrays with push, pop, and peek operations.

4.3. Inheritance

Question 66: Create a base class Shape with derived classes Circle, Rectangle, Triangle implementing area calculation.

Question 67: Implement a Person class with Student and Teacher as derived classes.

Question 68: Write a Vehicle class with Car and Bike as derived classes, each with specific attributes.

Question 69: Create a base class Employee with derived classes FullTime and PartTime with different salary calculations.

Question 70: Implement multilevel inheritance: Person → Student → GraduateStudent.

4.4. Polymorphism

Question 71: Create abstract base class Animal with derived classes Dog, Cat implementing virtual functions.

Question 72: Implement function overloading for a Calculator class.

Question 73: Write a program demonstrating runtime polymorphism with virtual functions.

Question 74: Create a base class with pure virtual functions and multiple derived classes.

Question 75: Implement operator overloading for a Vector class.

---

5. File Handling (Questions 76-85)

Question 76: Write a program to create a file and write student information to it.

Question 77: Implement a program to read from a file and display its contents.

Question 78: Create a program to copy contents from one file to another.

Question 79: Write a program to count words, lines, and characters in a file.

Question 80: Implement a student record system using file handling (add, search, delete records).

Question 81: Create a program to merge two files into a third file.

Question 82: Write a program to find and replace specific text in a file.

Question 83: Implement a program that handles binary files for storing objects.

Question 84: Create a program to maintain inventory using file handling.

Question 85: Write a program that demonstrates random access in files.

---

6. Templates and STL (Questions 86-95)

6.1. Templates

Question 86: Write a function template to find the maximum of two values of any type.

Question 87: Create a class template for a Stack data structure.

Question 88: Implement a function template for bubble sort.

Question 89: Write a class template for a Pair that can hold two values of different types.

Question 90: Create a template function to swap two values.

6.2. Standard Template Library

Question 91: Write a program to demonstrate vector operations.

Question 92: Implement a program using map to count word frequency in a text.

Question 93: Create a program that uses STL algorithms for sorting and searching.

Question 94: Write a program to demonstrate the use of set and multiset.

Question 95: Implement a priority queue for job scheduling.

---

7. Exception Handling (Questions 96-100)

Question 96: Implement exception handling for division by zero and array index out of bounds.

Question 97: Create a program demonstrating custom exception classes.

Question 98: Write a program with multiple catch blocks and exception specification.

Question 99: Implement a program that demonstrates stack unwinding.

Question 100: Create a complete banking system with all OOP concepts, file handling, templates, STL, and exception handling.