Easy Exercises (1-5)

Exercise 1: Basic Class Creation

Create a class `Rectangle` with private data members `length` and `width`. Include public member functions to set and get these values.

Exercise 2: Default Constructor

Write a class `Student` with private members `name` (string) and `age` (int). Create a default constructor that initializes name to "Unknown" and age to 0.

Exercise 3: Parameterized Constructor

Modify the `Student` class to include a parameterized constructor that accepts name and age as parameters.

Exercise 4: Access Specifier Practice

Create a class `BankAccount` with private member `balance` and public member functions `deposit()` and `getBalance()`. Ensure balance cannot be directly accessed from outside the class.

Exercise 5: Constructor Overloading

Create a class `Circle` with a private member `radius`. Implement three constructors:

- Default constructor (radius = 0)
- Parameterized constructor accepting radius

Intermediate Exercises (6-8)

Exercise 6: Initialization List

Create a class `Employee` with const member `employeeID` and reference member `department`. Use member initialization list in the constructor to initialize these members along with regular members `name` and `salary`.

Exercise 7: Multiple Constructors with Validation

Design a class `Date` with private members day, month, year. Implement:

- Default constructor (sets to current date)
- Parameterized constructor with validation (e.g., month 1-12, appropriate days per month)
- Copy constructor

Include a display function that shows date in DD/MM/YYYY format.

Exercise 8: Private Constructor Pattern

Create a class `DatabaseConnection` that ensures only one instance can be created (singleton pattern basics). Use a private constructor and a public static method `getInstance()` to control object creation.

Difficult Exercises (9-10)

Exercise 9: Complex Constructor Chain

Design a class hierarchy: `Person` (base) and `Employee` (derived).

- Person has private members: name, age, and ID (const)
- Employee has private members: salary, department, and employee code
- Implement all types of constructors in both classes
- Use initialization lists properly
- Demonstrate constructor chaining from derived to base class
- Include a static member to auto-generate IDs

Exercise 10: Resource Management with RAII

Create a class `SmartArray` that:

- Manages a dynamic integer array (private pointer member)
- Implements parameterized constructor (accepts size)
- Implements copy constructor with deep copy
- Implements destructor to prevent memory leaks
- Has private member functions for memory allocation/deallocation
- Includes public methods: `setValue()` , `getValue()` , `getSize()`
- Bonus: Implement move constructor (C++11)