Data Structures and Algorithms — Lab 2

# Topic

In this lab, we will revise the concepts of Abstract classes, Pure virtual functions, and Templates.

# Objective

* Explore the concepts of Abstract classes
* Understand the concept of pure virtual function.
* Revision of concepts related to OOP.

# Outcomes

1. A thorough understanding of abstract classes in Object-Oriented Programming (OOP) will be achieved. The concept of abstract classes as blueprints for derived classes will be explored, and their role in creating organized and maintainable class hierarchies will be grasped.
2. The concept and purpose of pure virtual functions will be understood. The way pure virtual functions enforce the implementation of specific functions in derived classes and facilitate polymorphism will be learned, and the implementation of these functions will be practiced.
3. Concepts related to Object-Oriented Programming, including encapsulation, inheritance, polymorphism, and abstraction, will be revised. A deeper understanding of how to design and implement class hierarchies using advanced OOP techniques will be gained.

# Content

The following sections will be covered during this lab session.

1. **An abstract class** is a class that is designed to be specifically used as a base class. An abstract class contains at least one pure virtual function. You declare a pure virtual function by using a pure specifier ( = 0 ) in the declaration of a virtual member function in the class declaration**.**
2. **A pure virtual function** is a function that must be overridden in a derived class and need not be defined
3. **Selection Sort** will cover implementing the algorithm by repeatedly finding the smallest element in the unsorted part of the array and swapping it with the current element. Students will write, compile, and execute the code, ensuring the program sorts the array correctly.
4. **Linear Search** will involve writing code to search for a specific value by iterating through each element in the array sequentially. Students will compile and execute the code, testing it with different inputs to ensure correctness.
5. **Binary Search** will guide students through implementing the algorithm on sorted arrays, using an iterative method to divide the search space by half at each step. Students will write, compile, and test the code with edge cases such as single-element arrays or missing values.

# Tasks instructions

Students are required to complete the following tasks during lab time.

Create a private repository on their GitHub accounts. The name of the repository should be: **Lab-2-DSA**. All files should be uploaded to this repository; this includes the header and cpp files. Please note that the name of class-related files should be the same as the class name. We can name the main file Task\_1.cpp or Task.cpp if there is only 1 task. We recommend you work in .h files for classes only since we must work on templates.

## Task 1

Create an abstract class Shape with a pure virtual function area(). Derive two classes, Circle and Rectangle, from Shape. Implement the area() function for both derived classes.

**Requirements:**

* The Circle class should have a private attribute radius and a constructor to initialize it.
* The Rectangle class should have private attributes: length and width. The class should have a constructor to initialize the attributes.
* Write a main function to create objects of both classes and display their areas.

## Task 2

Design an abstract class Employee with a pure virtual function calculateSalary(). Derive two classes: FullTimeEmployee and PartTimeEmployee, from Employee.

**Requirements:**

* FullTimeEmployee should have a fixed salary, while PartTimeEmployee should be paid based on hours worked and an hourly rate.
* Implement the calculateSalary() method in both classes.
* Write a main function that creates one full-time and one part-time employee, sets their details, and prints their salaries.

## Task 3

For this task, use the code implemented in Lab 1 for linear and binary search using templates.

Analyze the main function below and create appropriate classes using the concepts of pure virtual functions, inheritance and abstract classes:

int main() {

**// Create book objects**

    Book book1("The Catcher in the Rye", "J.D. Salinger", 277);

    Book book2("To Kill a Mockingbird", "Harper Lee", 324);

**// Create newspaper objects**

    Newspaper newspaper1("Washington Post", "2024-10-13", "Morning Edition");

    Newspaper newspaper2("The Times", "2024-10-12", "Weekend Edition");

**// Create a library object**

    Library library;

**// Add books and newspapers to the library**

    library.addBook(book1);

    library.addBook(book2);

    library.addNewspaper(newspaper1);

    library.addNewspaper(newspaper2);

**// Display the entire collection**

    cout << "Before Sorting:\n";

    library.displayCollection();

**// Sort books by pages and newspapers by edition**

    library.sortBooksByPages();

    library.sortNewspapersByEdition();

    cout << "\nAfter Sorting:\n";

    library.displayCollection();

**// Search for a book by title**

    Book\* foundBook = library.searchBookByTitle("The Catcher in the Rye");

    if (foundBook) {

        cout << "\nFound Book:\n";

        foundBook->display();

    } else {

        cout << "\nBook not found.\n";

    }

**// Search for a newspaper by name**

    Newspaper\* foundNewspaper = library.searchNewspaperByName("The Times");

    if (foundNewspaper) {

        cout << "\nFound Newspaper:\n";

        foundNewspaper->display();

    } else {

        cout << "\nNewspaper not found.\n";

    }

    return 0;

}