Data Structures Lab Manual

Lab 1: Abstract Classes, Templates, and ArrayList Implementation

Prepared for: Students of Data Structures

Department of Computer Science  
Fast School of Computing

# Objectives

- To understand the concept of Abstract Classes in C++.  
- To learn how to use Templates for generic programming.  
- To implement a List using an Array.  
- To use the implemented ArrayList class for storing Person information.

# Lab Outcomes

After completing this lab, students will be able to:  
1. Define and use abstract classes.  
2. Apply templates for type-independent code.  
3. Implement an Array-based List using OOP principles.  
4. Store and manipulate user-defined objects (Person) in a custom ArrayList.

# Theory Recap

## 1. Abstract Classes

A class that cannot be instantiated directly. Contains at least one pure virtual function. Provides a base interface for derived classes.

Example:  
class Shape {  
public:  
 virtual void draw() = 0; // pure virtual function  
};

## 2. Templates

Allow writing generic classes and functions. Enable code reusability with different data types.

Example:  
template <typename T>  
T add(T a, T b) {  
 return a + b;  
}

## 3. List Implementation

We define an abstract parent class List that contains the interface. We create a child class ArrayList that implements the List using an array.

# Lab Task

## Define List (Abstract Class)

template <typename T>  
class List {  
public:  
 virtual void insert(T element) = 0;  
 virtual void remove(int index) = 0;  
 virtual T get(int index) = 0;  
 virtual int size() = 0;  
 virtual void display() = 0;  
 virtual ~List() {}  
};

## Create Person Class

#include <string>  
class Person {  
public:  
 string name;  
 string gender;  
 int age;  
  
 Person(string n = "", string g = "", int a = 0) {  
 name = n;  
 gender = g;  
 age = a;  
 }  
  
 friend ostream& operator<<(ostream& os, const Person& p) {  
 os << "Name: " << p.name << ", Gender: " << p.gender << ", Age: " << p.age;  
 return os;  
 }  
};

## Step 4: Test in main() Function

int main() {  
 ArrayList<Person> people;  
  
 people.insert(Person("Ali", "Male", 20));  
 people.insert(Person("Sara", "Female", 22));  
 people.insert(Person("Ahmed", "Male", 19));  
  
 cout << "All Persons:" << endl;  
 people.display();  
  
 cout << "\nRemoving index 1 (Sara)...\n";  
 people.remove(1);  
 people.display();  
  
 cout << "\nAccessing index 0: " << people.get(0) << endl;  
  
 return 0;  
}

# Student Tasks

1. Implement the ArrayList with the above structure.  
2. Add more methods such as:  
 - bool isEmpty()  
 - bool contains(T element)  
 - clear() (remove all elements)  
3. Modify the program to allow user input for adding Person objects into the ArrayList.

# Submission Guidelines

- Submit your .cpp file with proper comments.  
- Make sure your program compiles and runs successfully.  
- Write a short explanation of how templates and abstract classes helped in your implementation.