



Lab Manual

**OBJECT ORIENTED PROGRAMMING**

**Semester : Fall 2023**

**Program : BS**

**Course Title and Name : CSC 213**

**Credits : 1**

**Faculty : Mehar Khan Niazi**

**Student Name :**

**Student ID :**

**Total Marks : 100**

**Obtained Marks :**

**Submitted Date :**

**TABLE OF CONTENTS**

|  |  |  |
| --- | --- | --- |
| **Lab No.** | **Experiment Title** | **Date** |
| **1** | **Transition from C to C++**  **●** Understanding the shift from C to C++  ● Setting up the development environment (IDE,  compiler)  ● Basic syntax and structure transition  ● Hands-on experience with cin, cout, std::  Namespace, variables, loops, and conditions |  |
| **2** | **Overview of programming paradigms/ Introduction to Classes and Objects**  ● Importance of OOP, Why OOP, Procedural  Programing vs OOP  ● Basic syntax and structure of a class in C++  ● Introduction to classes, objects  ● Access Specifiers in Classes |  |
| **3** | **Functions and Constructors**   * Introduction to Constructors * Default Constructors * Passing and Returning Objects to/from Functions * Private Functions and Variable Scopes |  |
| **4** | **Overloading**   * Method Overloading * Constructor Overloading (Parameterized Constructors) * Introduction to Copy Constructors |  |
| **5** | **Classes and Objects Continued**   * Working with Arrays of Objects * Destructors * Constant Member Functions & Objects |  |

|  |  |  |
| --- | --- | --- |
| **6** | **Inheritance**   * Creating Derived and Base Classes * Derived Class Constructors * Understanding the Protected Specifier * Function Overriding * Scope Resolution with Overridden Functions |  |
| **7** | **Level of Inheritance**   * Single Inheritance * Multilevel Inheritance * Multiple Inheritance |  |
| **8.** | **Mid-Term Break** |  |

**Week 7 Lab: Level of Inheritance**

**Lab Objective:**

The aim of this lab session is to introduce students levels of inheritance in Object-Oriented Programming (OOP) using C++. The objectives are:

* Understand the concepts of single, multilevel, and multiple inheritance and perceive their real-world applications.
* Recognize the distinctions between different inheritance patterns.
* Design class hierarchies rooted in inheritance patterns.
* Gain hands-on experience through practical examples reflecting various levels of inheritance.

**Tools/Software Requirement:**

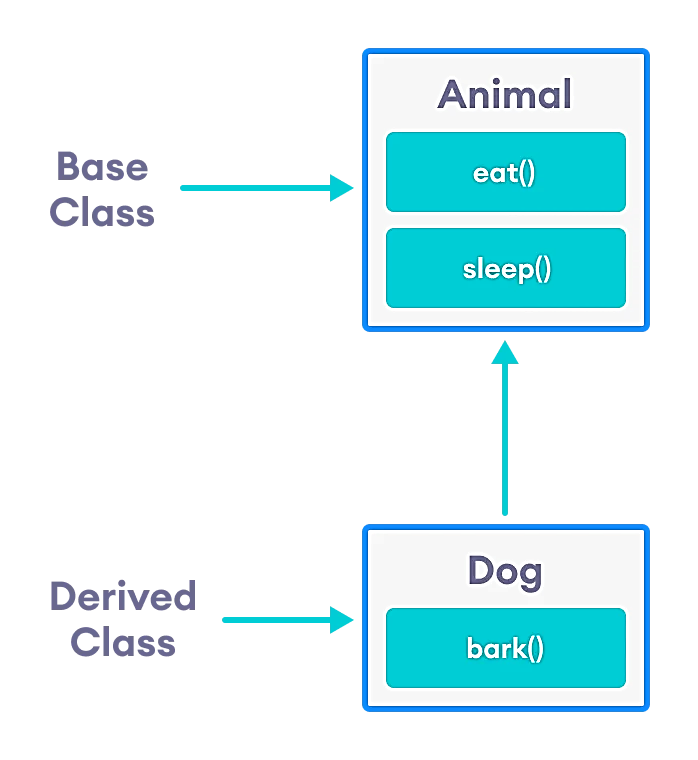
* Dev-C++ / Online C++ Compiler

**Theory:**

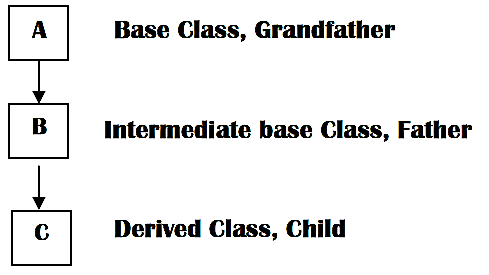
**Inheritance:**

Inheritance provides a mechanism in OOP to create a new class using properties and behaviors (i.e., fields and methods) from an existing class. The main advantage of inheritance is for code reuse. The existing class is termed the base/parent class, and the new class is called the derived/child class.

**1. Single Inheritance:**

In single inheritance, a class inherits from a single base class. This restricts a derived class to inherit features from one parent.

**2. Multilevel Inheritance:**

In this type of inheritance, a derived class is created from another derived class. This forms a chain of classes, where a class inherits from a class that is also derived from another class.

**3. Multiple Inheritance:**

Multiple inheritance allows a derived class to inherit properties and behaviors from more than one base class. However, it can lead to complications, like the Diamond Problem. A diagram of a structure

Description automatically generatedA diagram of a family

Description automatically generated

**Code Examples:**

**Example 1: Single Inheritance**

#include <iostream>

using namespace std;

// Base class

class Animal {

public:

void eat() {

cout << "Can eat!" << endl; }};

// Derived class

class Dog : public Animal {

public:

void bark() {

cout << "Can bark!" << endl; }};

int main() {

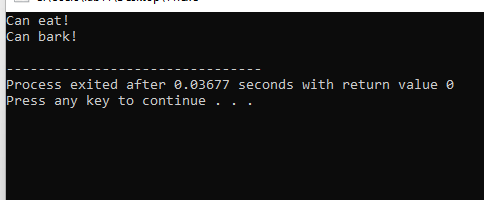
Dog myDog;

myDog.eat();

myDog.bark();

return 0; }

**Output:**

****

**Explanation:**

**Example 2: Multilevel Inheritance**

#include <iostream>

using namespace std;

// Base class

class Person {

public:

void walk() {

cout << "I can walk." << endl;

}

};

// First derived class

class Student : public Person {

public:

void study() {

cout << "I can study." << endl;

}

};

// Second derived class

class HighSchoolStudent : public Student {

public:

void playSports() {

cout << "I can play sports." << endl;

}

};

int main() {

HighSchoolStudent zia;

zia.walk();

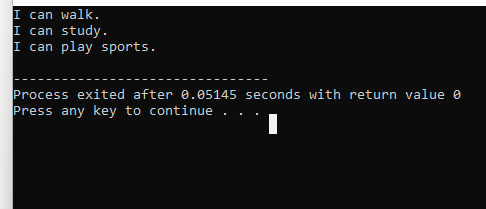
zia.study();

zia.playSports();

return 0;

}

**Output:**

****

**Explanation:**

**Example 3: Multilevel Inheritance**

#include <iostream>

using namespace std;

// Base class

class Vehicle {

public:

void move() {

cout << "Vehicle can move." << endl;

}

};

// First derived class

class Car : public Vehicle {

public:

void hasFourWheels() {

cout << "Cars have four wheels." << endl;

}

};

// Second derived class

class Sedan : public Car {

public:

void comfort() {

cout << "Sedans offer comfort." << endl;

}

};

int main() {

Sedan mySedan;

mySedan.move();

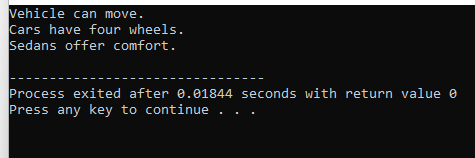
mySedan.hasFourWheels();

mySedan.comfort();

return 0;

}

**Output:**

****

**Explanation:**

This program shows the example of multilevel inheritance. This output is displayed when the program is excecuted.

**Example 4: Multiple Inheritance**

#include <iostream>

using namespace std;

// First base class

class Teacher {

public:

void teach() {

cout << "I can teach." << endl;

}

};

// Second base class

class Researcher {

public:

void research() {

cout << "I can research." << endl;

}

};

// Derived class inheriting from both Teacher and Researcher

class Professor : public Teacher, public Researcher {};

int main() {

Professor drMateen;

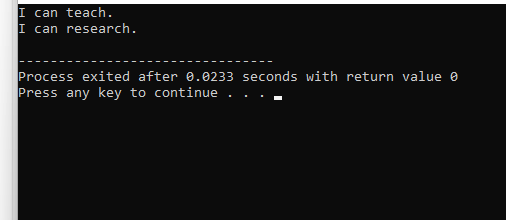
drMateen.teach();

drMateen.research();

return 0;

}

**Output:**

****

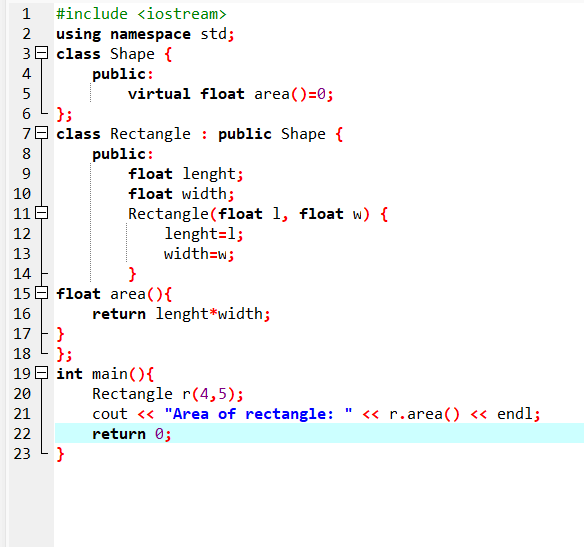
**Explanation:**

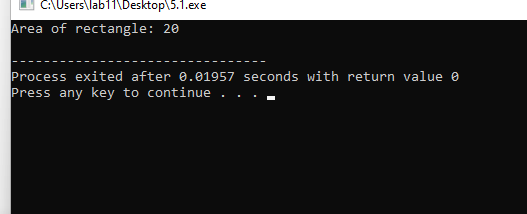
**Lab 7 Tasks:**

**Requirements (Code, Output and Reason/Explanation)**

**Task 1: Single Inheritance:**

Design a base class Shape and a derived class Rectangle. The base class should have a method to calculate area, and the derived class should implement it specifically for a rectangle.





**Task 2: Multiple Inheritance:**

Create two base classes, `Athlete` and `Student`. Design a derived class `StudentAthlete` that inherits from both base classes. Show how methods from both parent classes can be used by the derived class.

**Task 3: Multilevel Inheritance:**

Design a class `Fruit`, derive a class `Citrus` from it, and further derive classes `Orange` and `Lemon` from `Citrus`.

**Task 4: Protected Members and Inheritance:**

Create a base class with protected members. Derive a class from this base and showcase how protected members of the base class are accessible in the derived class.

**Task 5: Multiple Inheritance:**

Create two base classes, Writer and Painter. Design a derived class AuthorArtist that inherits from both base classes. Demonstrate the usage of methods from both parent classes in the derived class.

**Task 6:** Code the Below Picture and Demonstrate the usage of methods from both parent classes in the derived class.A diagram of a family

Description automatically generated