Lab 11 – Abstract Class

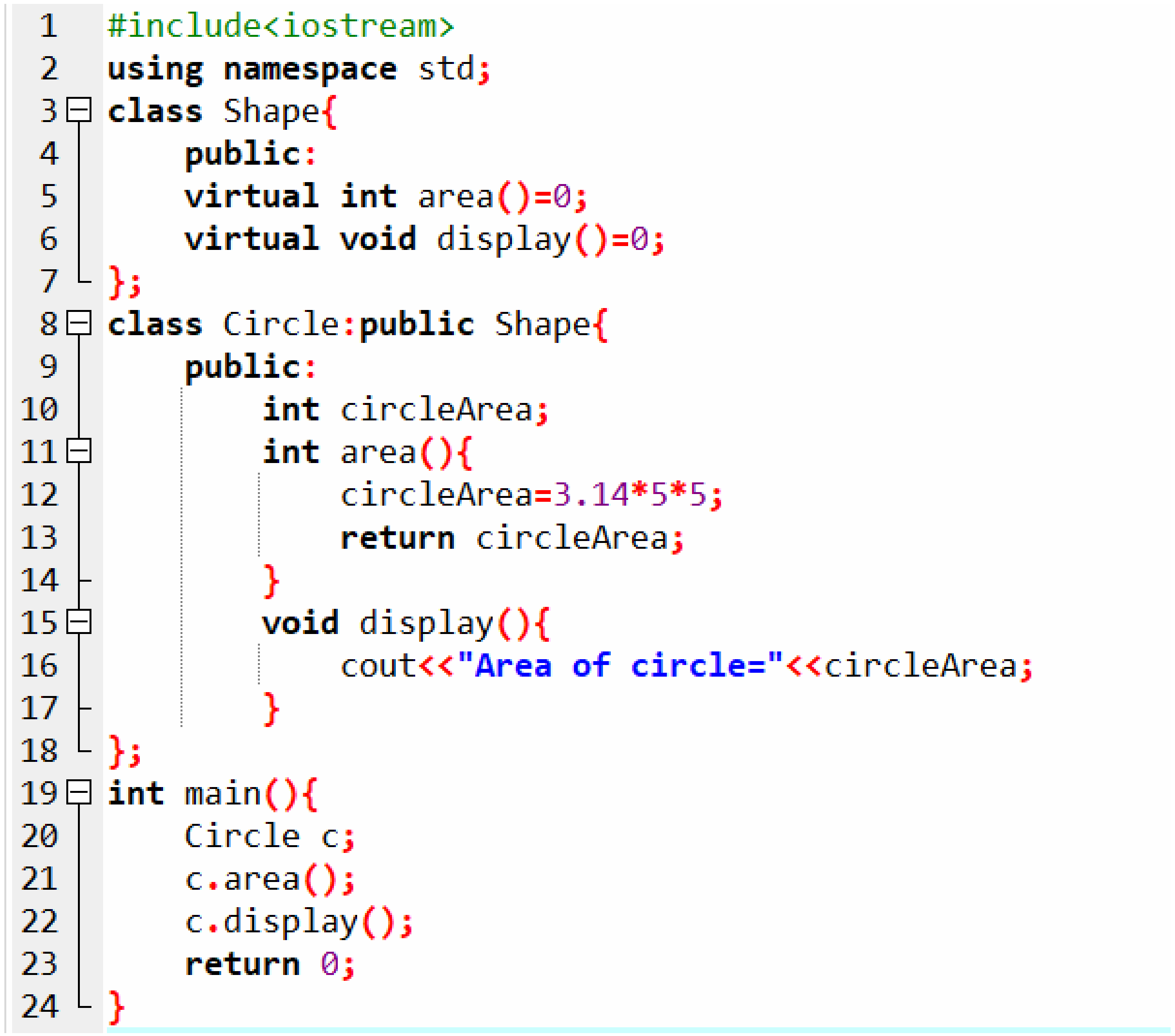
Lab Outcomes:

- **Understanding Data Encapsulation and Access Modifiers**: Develop an understanding of how C++ implements encapsulation through access modifiers (`public`, `private`, and `protected`). Learn how these modifiers control access to class members and ensure data integrity by limiting access to internal states.
- **Implementing Getter and Setter Functions**: Learn how to define and implement getter and setter functions to safely access and modify private data members of a class. Understand the importance of these functions in providing controlled access to the class's internal state while maintaining encapsulation.
- **Applying Inheritance and Polymorphism**: Gain hands-on experience in implementing inheritance and polymorphism in C++, allowing for the creation of hierarchical class structures. Learn how base and derived classes interact, and how to override base class methods in derived classes for dynamic behavior.
- **Understanding Virtual Functions and Dynamic Binding**: Develop a clear understanding of virtual functions in C++ and how they enable dynamic binding. Learn how virtual functions allow for polymorphic behavior, enabling the proper method to be called on objects of derived classes at runtime.
- **Designing and Using Abstract Classes for Generalization**: Learn how to design and implement abstract classes in C++ to represent general concepts, requiring derived classes to provide specific implementations. Understand the role of pure virtual functions in abstract classes and how they enable the use of polymorphism and generalized programming.

Abstract classes act as expressions of general concepts from which more specific classes can be derived. You can't create an object of an abstract class type. However, you can use pointers and references to abstract class types. You create an abstract class by declaring at least one pure virtual member function.

A **pure virtual function** (or abstract function) in C++ is a virtual function for which we can have an implementation, But we must override that function in the derived class, otherwise, the derived class will also become an abstract class. A pure virtual function is declared by assigning 0 in the declaration.

- In C++, a = 0 at the end of a virtual function declaration indicates that the function is **pure virtual**, making the class an **abstract class**.
- It signals that the derived classes must provide an implementation for this function.



```
#include<iostream>
using namespace std;
class Shape{
	public:
	virtual int area()=0;
	virtual void display()=0;
};
class Circle:public Shape{
	public:
		int circleArea;
		int area(){
			circleArea=3.14*5*5;
			return circleArea;
		}
		void display(){
			cout<<"Area of circle="<<circleArea;
		}
};
int main(){
	Circle c;
	c.area();
	c.display();
	return 0;
}
```

```
#include <iostream>

using namespace std;

class Student {

private:

    int age;    // private data member

public:

    // Setter function

    void setAge(int a) {

        if (a > 0)        // validation (controlled access)

            age = a;
        else
            age = 0;
    }
    // Getter function
    int getAge() {
        return age;
    }
};
int main() {
    Student s;
    s.setAge(20);               // setting value using setter
    cout << "Age: " << s.getAge(); // accessing value using getter
    return 0;
}
```

Task:

You're tasked with developing a student management system for a university. The system should be able to handle various tasks related to student information, such as registration, course enrollment, and GPA calculation. To ensure proper design and implementation, you decide to use abstract classes and polymorphism.

Design and implement a C++ program to fulfill the following requirements:

1. Define an abstract class named **Student** with the following characteristics:
    - It should have pure virtual functions for displaying student information (**virtual void display() = 0;**) and calculating the GPA (**virtual float calculateGPA() = 0;**).
    - The class should contain data members to store basic information about a student, such as name, ID, and age.
2. Implement two derived classes: **UndergraduateStudent** and **GraduateStudent**, each representing a specific type of student. Each class should inherit from the **Student** class and provide implementations for the **display()** and **calculateGPA()** functions:
    - For the **UndergraduateStudent** class:
        - Implement the **calculateGPA()** function to calculate the GPA based on the grades obtained in undergraduate courses.

        - Implement the **display()** function to display the details of the undergraduate student, including their name, ID, age, and GPA.
    - For the **GraduateStudent** class:
        - Implement the **calculateGPA()** function to calculate the GPA based on the grades obtained in graduate courses.
        - Implement the **display()** function to display the details of the graduate student, including their name, ID, age, and GPA.
3. Write a main program to test your implementation:
    - Create objects of each student class (UndergraduateStudent, GraduateStudent).
    - Prompt the user to input information for each student, including name, ID, age, and grades.
    - Calculate and display the GPA of each student using polymorphism.
    - Ensure that the program compiles without errors and produces correct results for different types of students.

Lab #11 Marks distribution

	ER1	**ER6**	**ER8**
Task	3points	3 points	4 points

Lab # 11 Rubric Evaluation Guideline:

#	**Qualities & Criteria**	**0 < Poor <=0.5**	**0.5 < Satisfactory <= 1.5**	**1.5 < Excellent <=2**
ER1	**Task Completion**	Minimal or no program functionality was achieved.	Some tasks were completed, but the program has errors or incomplete functionalities.	All tasks were completed, and the program runs without errors.
#	**Qualities & Criteria**	**0 < Poor <=0.5**	**0.5 < Satisfactory <= 1.5**	**1.5 < Excellent <=2**

ER6	Program Output	Output is inaccurate or poorly presented.	Output is mostly accurate but may lack labels, captions, or formatting.	Output is clear, accurate, and well presented with labels, captions, and proper formatting.
#	**Qualities & Criteria**	**0 < Poor <= 1.5**	**1.5< Satisfactory <= 3**	**3< Excellent <= 4**
ER8	Question & Answer	Answers some questions but not confidently or based on lab task knowledge.	Answers most questions confidently and based on lab task knowledge.	Answers all questions confidently and demonstrates a deep understanding of the given lab task.