SAVEETHA SCHOOL OF ENGINEERING

SAVEETHA INSTITUTE OF MEDICAL AND TECHNICAL SCIENCES

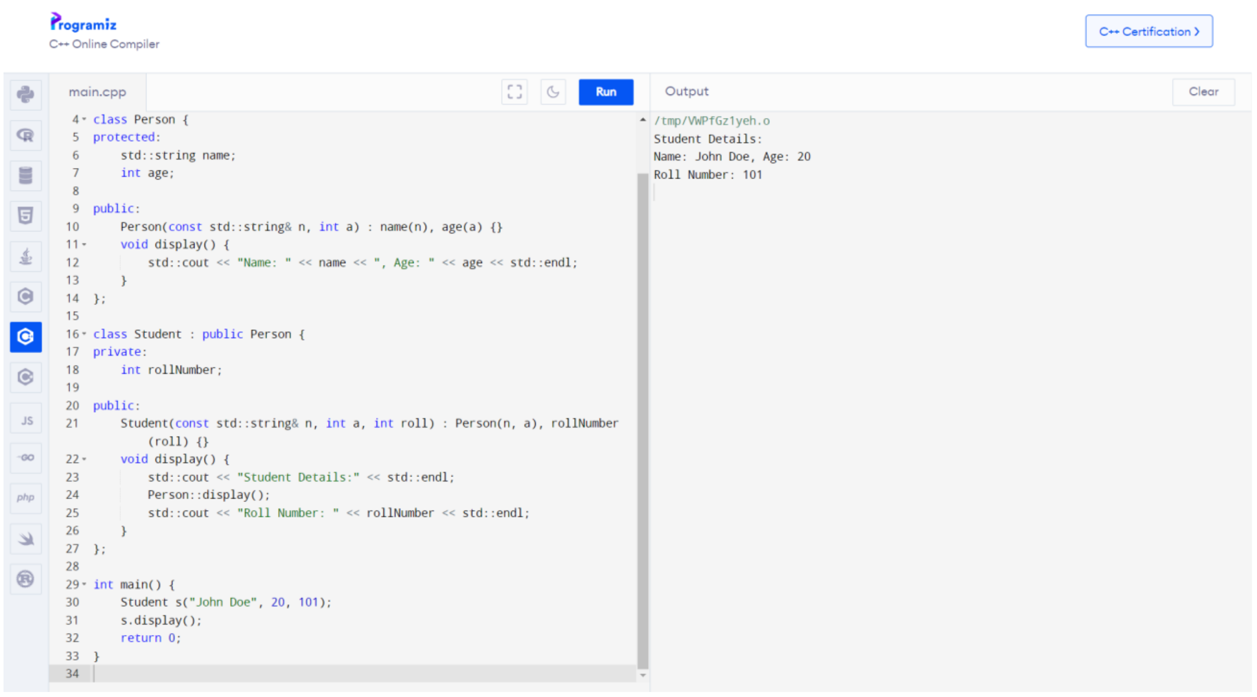
Engineer to Excel

SUB CODE & NAME : DSA01/ Object Oriented Programming with C++

LAB DAY 4/19-03-2024

EASY

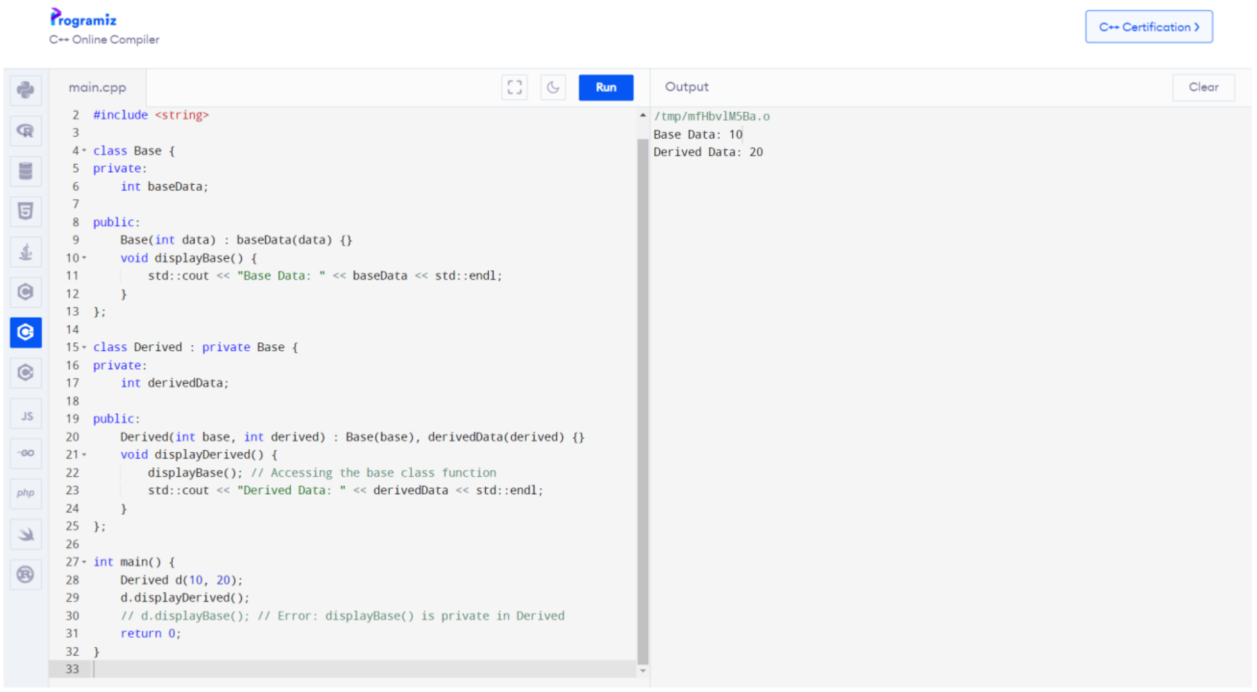
1. Write a C++ program to read and print students using simple inheritance



```cpp
class Person {
protected:
    std::string name;
    int age;

public:
    Person(const std::string& n, int a) : name(n), age(a) {}
    void display() {
        std::cout << "Name: " << name << ", Age: " << age << std::endl;
    }
};

class Student : public Person {
private:
    int rollNumber;

public:
    Student(const std::string& n, int a, int roll) : Person(n, a), rollNumber
        (roll) {}
    void display() {
        std::cout << "Student Details:" << std::endl;
        Person::display();
        std::cout << "Roll Number: " << rollNumber << std::endl;
    }
};

int main() {
    Student s("John Doe", 20, 101);
    s.display();
    return 0;
}
```

```
/tmp/VWPfGz1yeh.o
Student Details:
Name: John Doe, Age: 20
Roll Number: 101
```

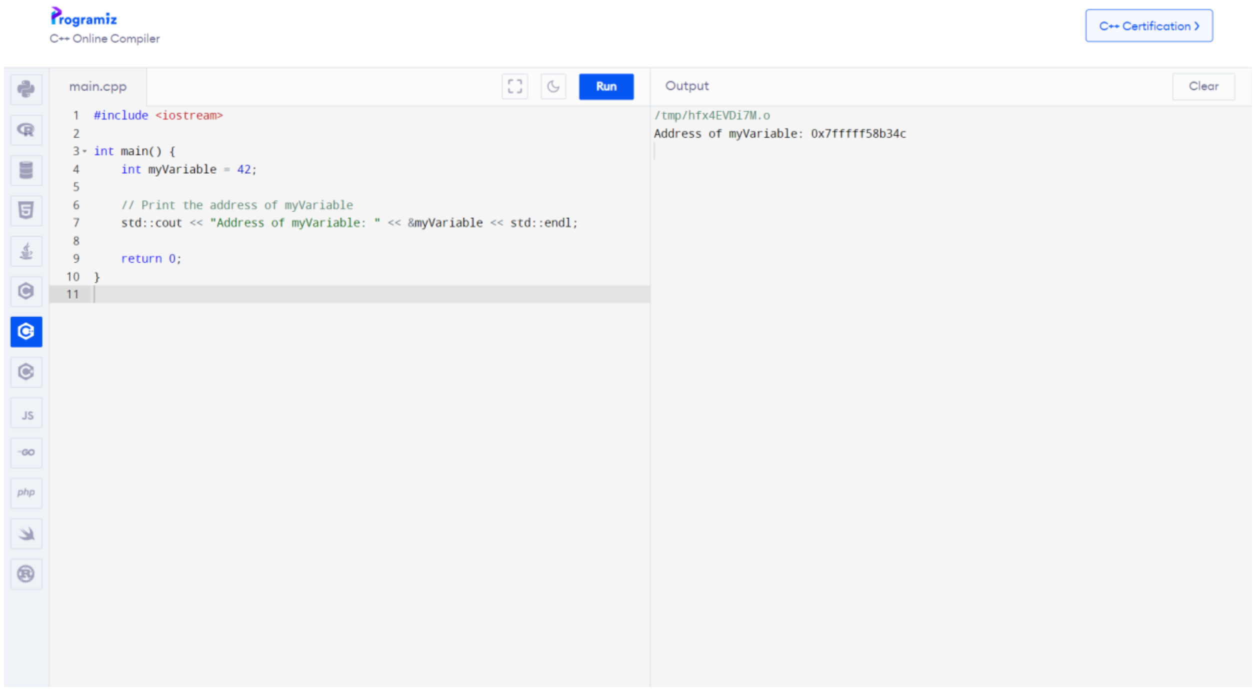
2. Write C++ Program C++ program to demonstrate example of private simple inheritance



```cpp
#include <string>

class Base {
private:
    int baseData;

public:
    Base(int data) : baseData(data) {}
    void displayBase() {
        std::cout << "Base Data: " << baseData << std::endl;
    }
};

class Derived : private Base {
private:
    int derivedData;

public:
    Derived(int base, int derived) : Base(base), derivedData(derived) {}
    void displayDerived() {
        displayBase(); // Accessing the base class function
        std::cout << "Derived Data: " << derivedData << std::endl;
    }
};

int main() {
    Derived d(10, 20);
    d.displayDerived();
    // d.displayBase(); // Error: displayBase() is private in Derived
    return 0;
}
```

```
/tmp/mfHbvlM5Ba.o
Base Data: 10
Derived Data: 20
```

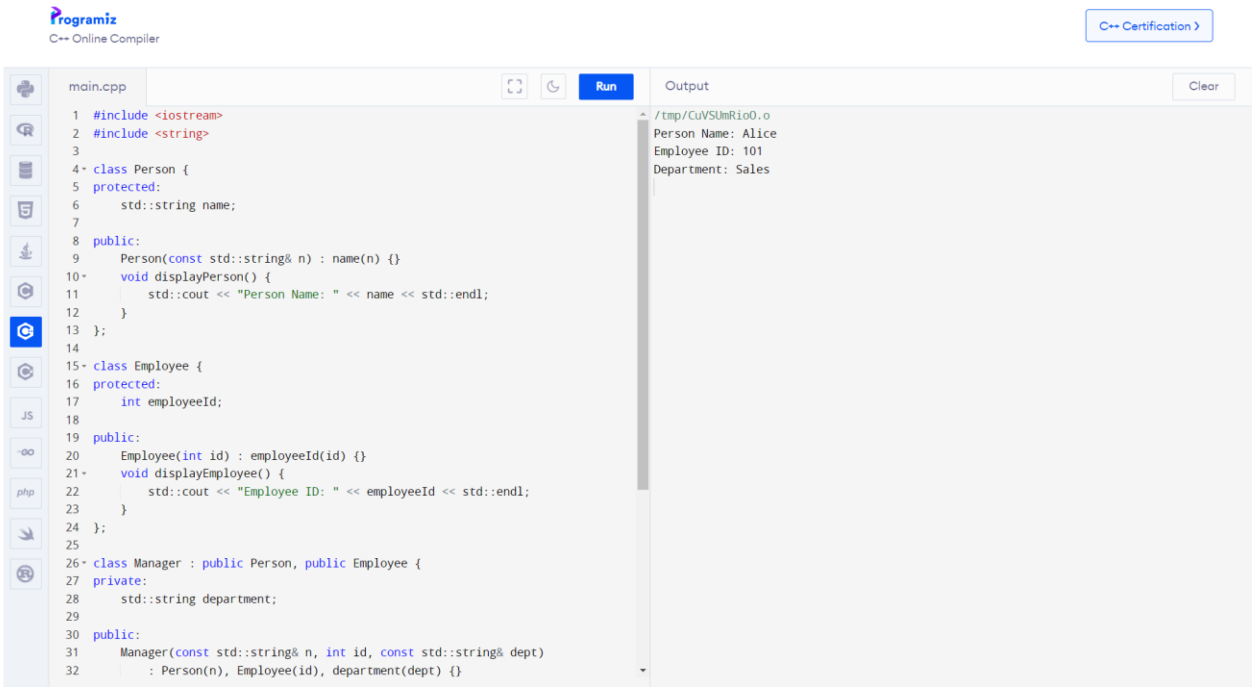
3. Write a program to build a C++ code to print the address of the variable.



```cpp
#include <iostream>

int main() {
    int myVariable = 42;

    // Print the address of myVariable
    std::cout << "Address of myVariable: " << &myVariable << std::endl;

    return 0;
}
```

```
/tmp/hfx4EVDi7M.o
Address of myVariable: 0x7fffff58b34c
```

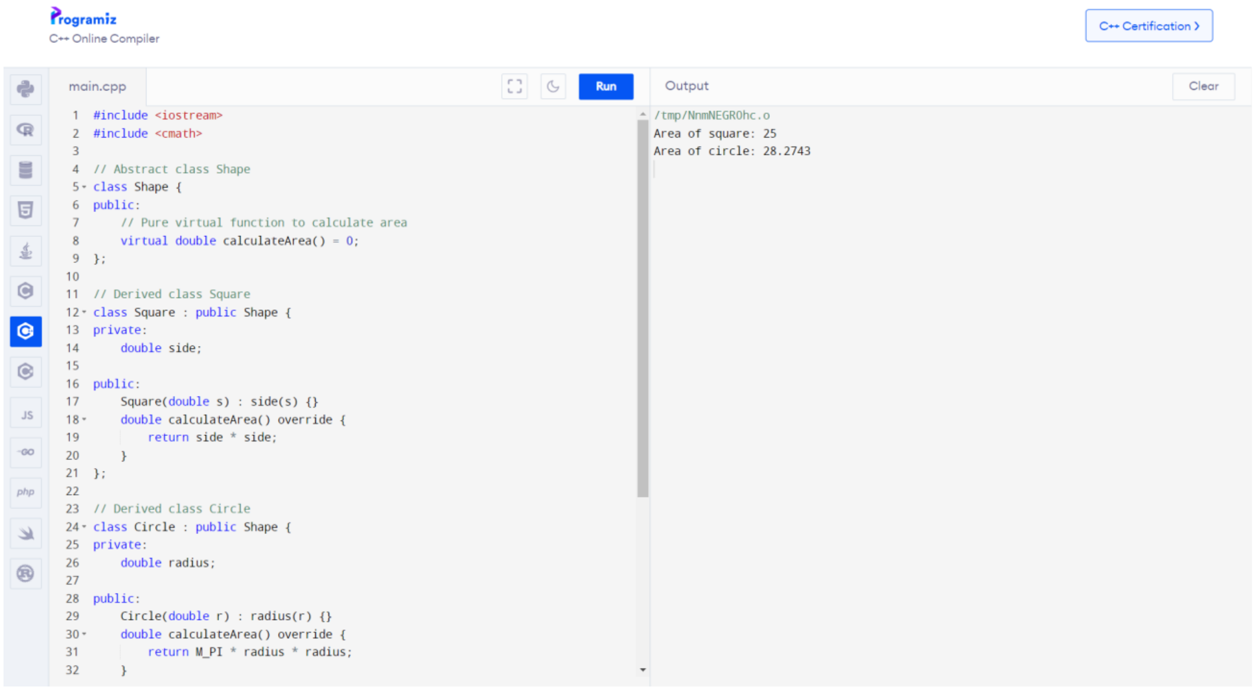
4. Write a program in C++ to Program to demonstrate multiple inheritance



```cpp
#include <iostream>
#include <string>

class Person {
protected:
    std::string name;

public:
    Person(const std::string& n) : name(n) {}
    void displayPerson() {
        std::cout << "Person Name: " << name << std::endl;
    }
};

class Employee {
protected:
    int employeeId;

public:
    Employee(int id) : employeeId(id) {}
    void displayEmployee() {
        std::cout << "Employee ID: " << employeeId << std::endl;
    }
};

class Manager : public Person, public Employee {
private:
    std::string department;

public:
    Manager(const std::string& n, int id, const std::string& dept)
        : Person(n), Employee(id), department(dept) {}
```

```
/tmp/CuVSUmRio0.o
Person Name: Alice
Employee ID: 101
Department: Sales
```

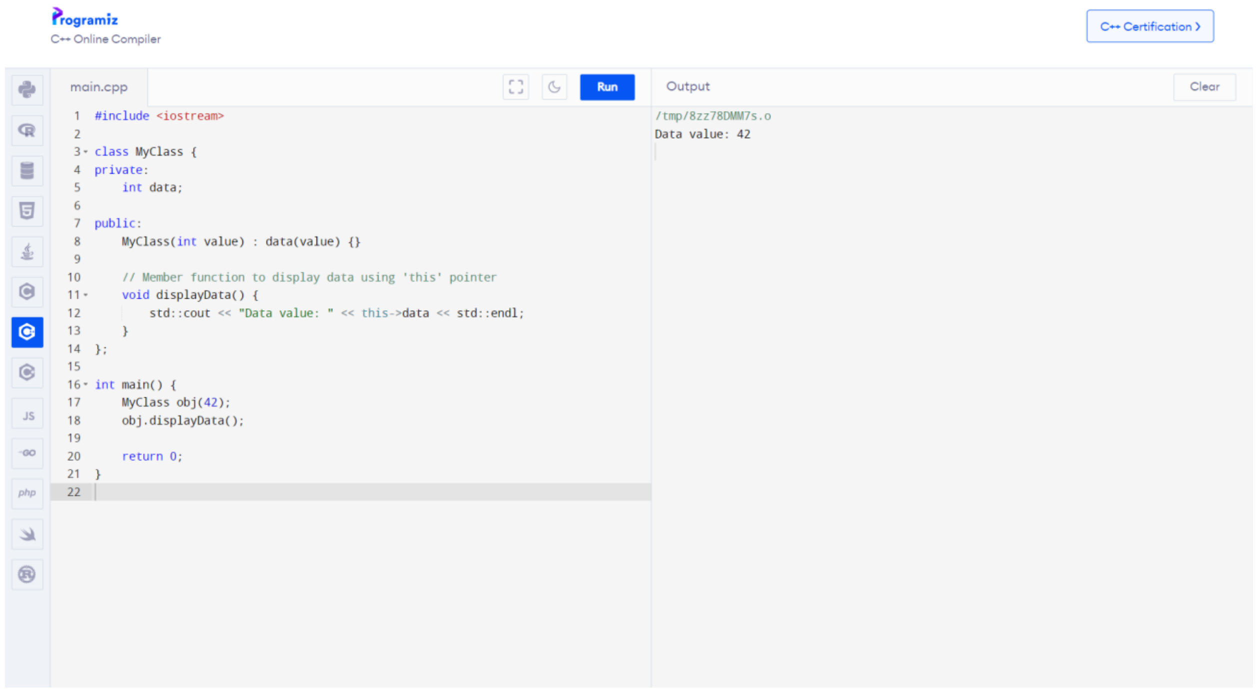
5. Write a C++ code to find area of square and circle using abstract class and pure virtual function



```cpp
#include <iostream>
#include <cmath>

// Abstract class Shape
class Shape {
public:
    // Pure virtual function to calculate area
    virtual double calculateArea() = 0;
};

// Derived class Square
class Square : public Shape {
private:
    double side;

public:
    Square(double s) : side(s) {}
    double calculateArea() override {
        return side * side;
    }
};

// Derived class Circle
class Circle : public Shape {
private:
    double radius;

public:
    Circle(double r) : radius(r) {}
    double calculateArea() override {
        return M_PI * radius * radius;
    }
```

```
/tmp/NnmNEGR0hc.o
Area of square: 25
Area of circle: 28.2743
```

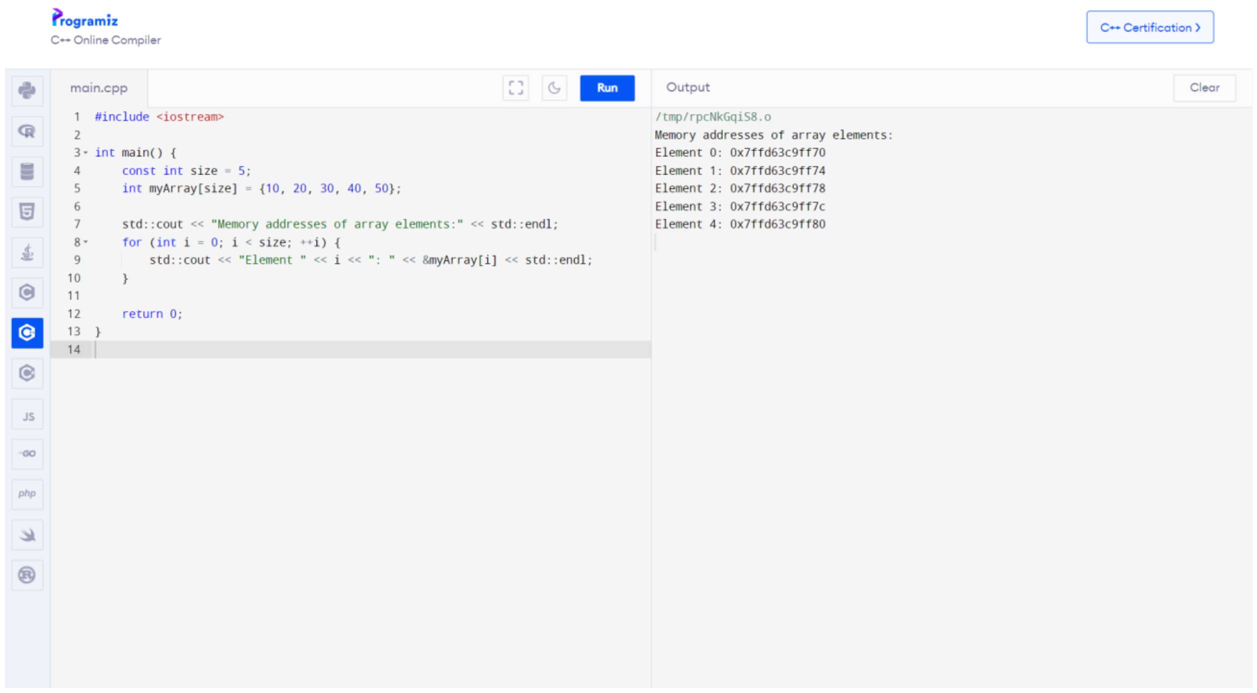
6.Write a C++ Program that illustrates how to use ‘this’ pointer.



```cpp
#include <iostream>

class MyClass {
private:
    int data;

public:
    MyClass(int value) : data(value) {}

    // Member function to display data using 'this' pointer
    void displayData() {
        std::cout << "Data value: " << this->data << std::endl;
    }
};

int main() {
    MyClass obj(42);
    obj.displayData();

    return 0;
}
```

```
/tmp/8zz78DMM7s.o
Data value: 42
```

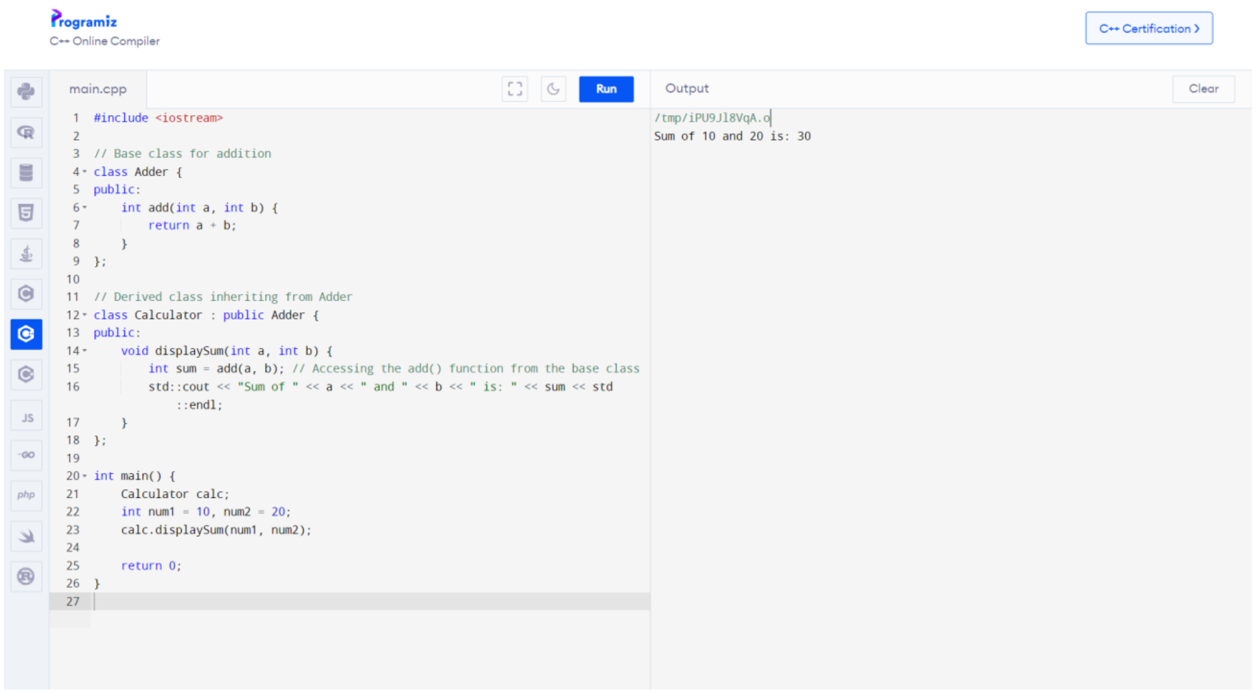
7.Write a C++ Program to display address of each element of an array.



```cpp
#include <iostream>

int main() {
    const int size = 5;
    int myArray[size] = {10, 20, 30, 40, 50};

    std::cout << "Memory addresses of array elements:" << std::endl;
    for (int i = 0; i < size; ++i) {
        std::cout << "Element " << i << ": " << &myArray[i] << std::endl;
    }

    return 0;
}
```

```
/tmp/rpcNkGqiS8.o
Memory addresses of array elements:
Element 0: 0x7ffd63c9ff70
Element 1: 0x7ffd63c9ff74
Element 2: 0x7ffd63c9ff78
Element 3: 0x7ffd63c9ff7c
Element 4: 0x7ffd63c9ff80
```

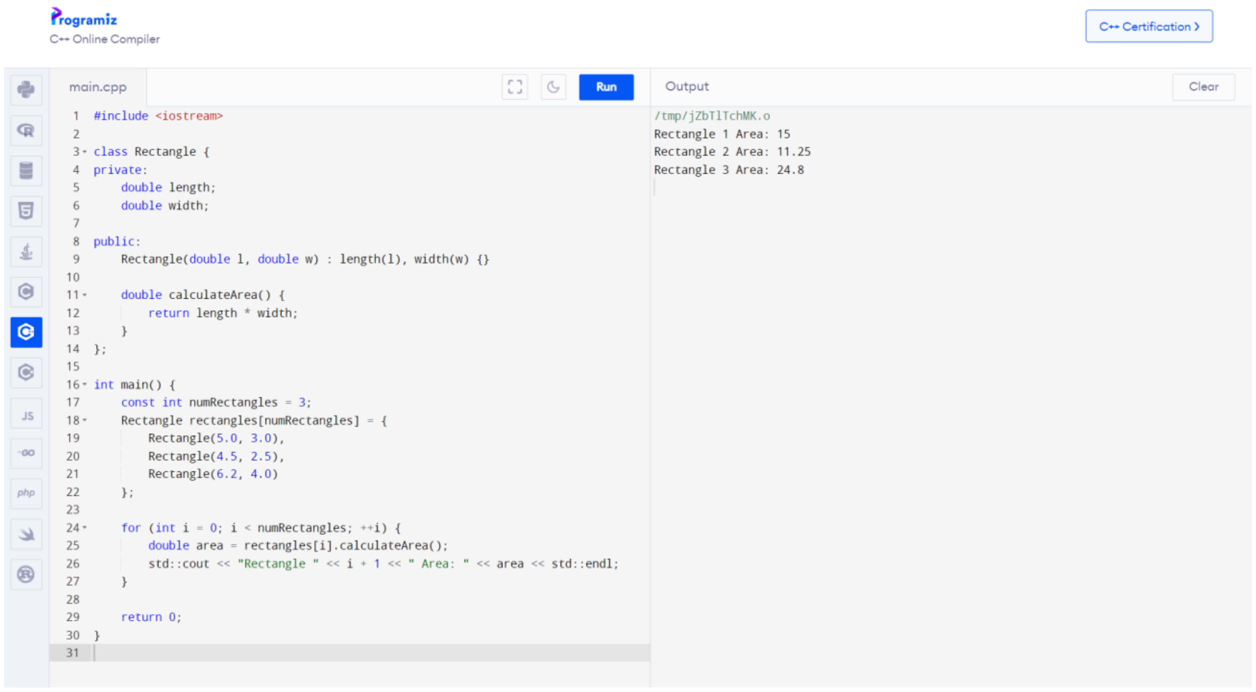
8.Write a C++ program to find the sum of two numbers using the concept of C++ multiple inheritance.



```cpp
#include <iostream>

// Base class for addition
class Adder {
public:
    int add(int a, int b) {
        return a + b;
    }
};

// Derived class inheriting from Adder
class Calculator : public Adder {
public:
    void displaySum(int a, int b) {
        int sum = add(a, b); // Accessing the add() function from the base class
        std::cout << "Sum of " << a << " and " << b << " is: " << sum << std
            ::endl;
    }
};

int main() {
    Calculator calc;
    int num1 = 10, num2 = 20;
    calc.displaySum(num1, num2);

    return 0;
}
```

```
/tmp/iPU9Jl8VqA.o
Sum of 10 and 20 is: 30
```

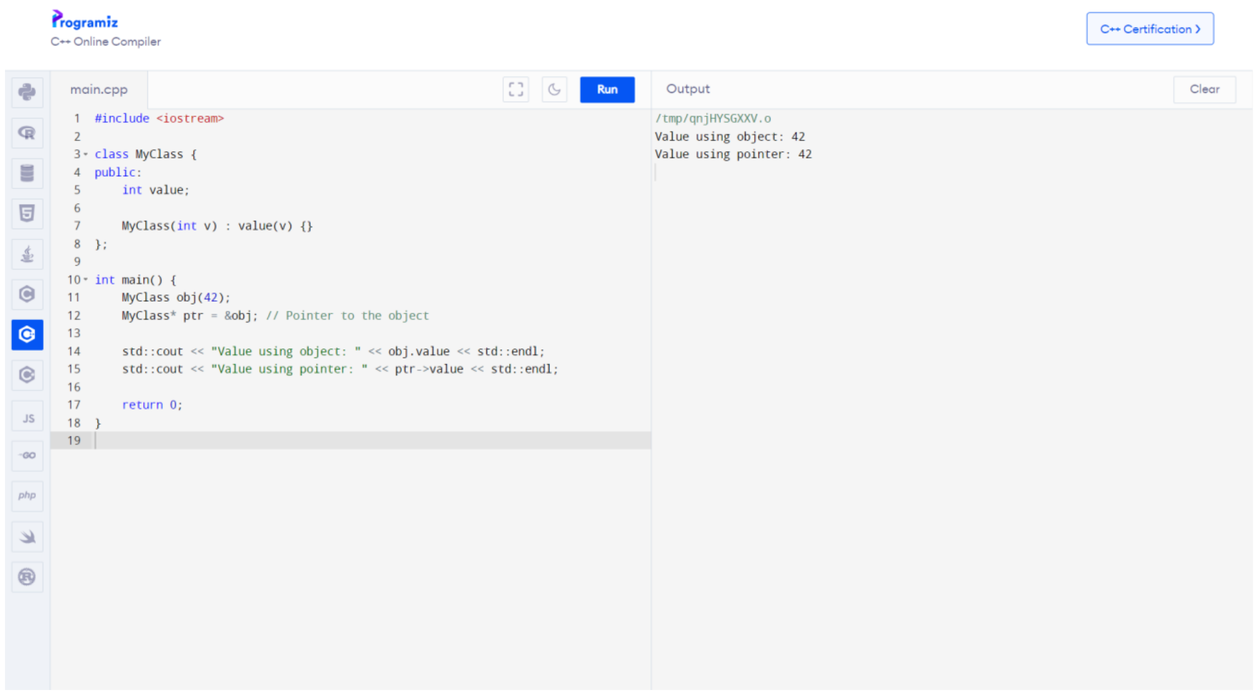
9.Build a C++ code to find the area of a rectangle using the concept of array of objects.



```cpp
#include <iostream>

class Rectangle {
private:
    double length;
    double width;

public:
    Rectangle(double l, double w) : length(l), width(w) {}

    double calculateArea() {
        return length * width;
    }
};

int main() {
    const int numRectangles = 3;
    Rectangle rectangles[numRectangles] = {
        Rectangle(5.0, 3.0),
        Rectangle(4.5, 2.5),
        Rectangle(6.2, 4.0)
    };

    for (int i = 0; i < numRectangles; ++i) {
        double area = rectangles[i].calculateArea();
        std::cout << "Rectangle " << i + 1 << " Area: " << area << std::endl;
    }

    return 0;
}
```

```
/tmp/jZbTlTchMK.o
Rectangle 1 Area: 15
Rectangle 2 Area: 11.25
Rectangle 3 Area: 24.8
```

10.Write a C++ code to print the given values in program using pointer to object

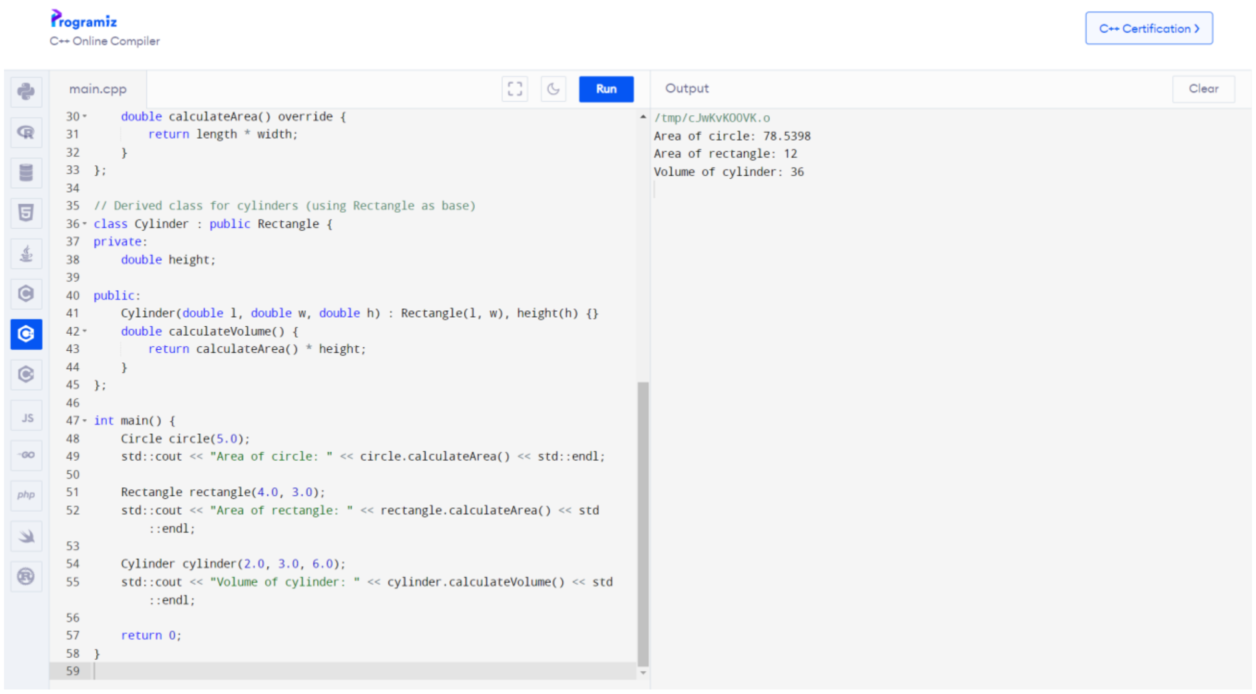


```cpp
#include <iostream>

class MyClass {
public:
    int value;

    MyClass(int v) : value(v) {}
};

int main() {
    MyClass obj(42);
    MyClass* ptr = &obj; // Pointer to the object

    std::cout << "Value using object: " << obj.value << std::endl;
    std::cout << "Value using pointer: " << ptr->value << std::endl;

    return 0;
}
```

```
/tmp/qnjHYSGXXV.o
Value using object: 42
Value using pointer: 42
```

MEDIUM

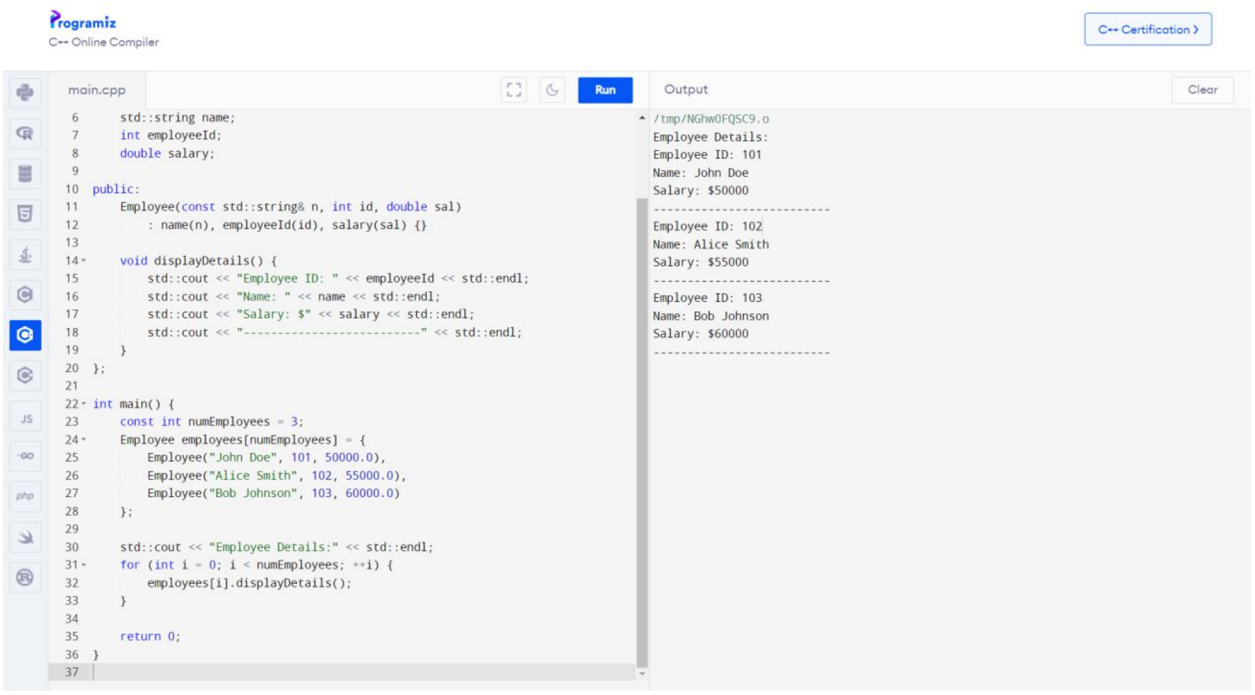
1 Develop a C++ code to find area circle, rectangle and volume of cylinder using the concept of multilevel inheritance.



```cpp
    double calculateArea() override {
        return length * width;
    }
};

// Derived class for cylinders (using Rectangle as base)
class Cylinder : public Rectangle {
private:
    double height;

public:
    Cylinder(double l, double w, double h) : Rectangle(l, w), height(h) {}
    double calculateVolume() {
        return calculateArea() * height;
    }
};

int main() {
    Circle circle(5.0);
    std::cout << "Area of circle: " << circle.calculateArea() << std::endl;

    Rectangle rectangle(4.0, 3.0);
    std::cout << "Area of rectangle: " << rectangle.calculateArea() << std
        ::endl;

    Cylinder cylinder(2.0, 3.0, 6.0);
    std::cout << "Volume of cylinder: " << cylinder.calculateVolume() << std
        ::endl;

    return 0;
}
```

```
/tmp/cJwKvKOOVK.o
Area of circle: 78.5398
Area of rectangle: 12
Volume of cylinder: 36
```

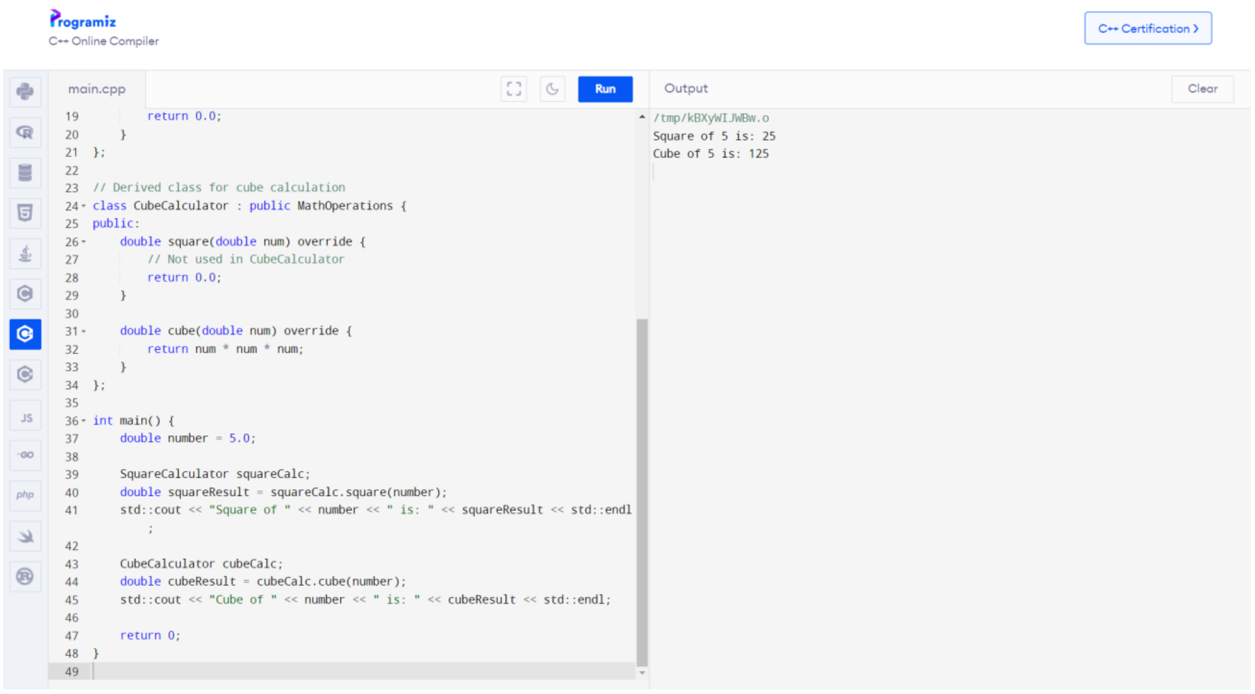
2. Write a C++ program for employee details using an array of objects



```cpp
    std::string name;
    int employeeId;
    double salary;

public:
    Employee(const std::string& n, int id, double sal)
        : name(n), employeeId(id), salary(sal) {}

    void displayDetails() {
        std::cout << "Employee ID: " << employeeId << std::endl;
        std::cout << "Name: " << name << std::endl;
        std::cout << "Salary: $" << salary << std::endl;
        std::cout << "-------------------------" << std::endl;
    }
};

int main() {
    const int numEmployees = 3;
    Employee employees[numEmployees] = {
        Employee("John Doe", 101, 50000.0),
        Employee("Alice Smith", 102, 55000.0),
        Employee("Bob Johnson", 103, 60000.0)
    };

    std::cout << "Employee Details:" << std::endl;
    for (int i = 0; i < numEmployees; ++i) {
        employees[i].displayDetails();
    }

    return 0;
}
```

```
/tmp/NGhwOFQSC9.o
Employee Details:
Employee ID: 101
Name: John Doe
Salary: $50000
-------------------------
Employee ID: 102
Name: Alice Smith
Salary: $55000
-------------------------
Employee ID: 103
Name: Bob Johnson
Salary: $60000
-------------------------
```

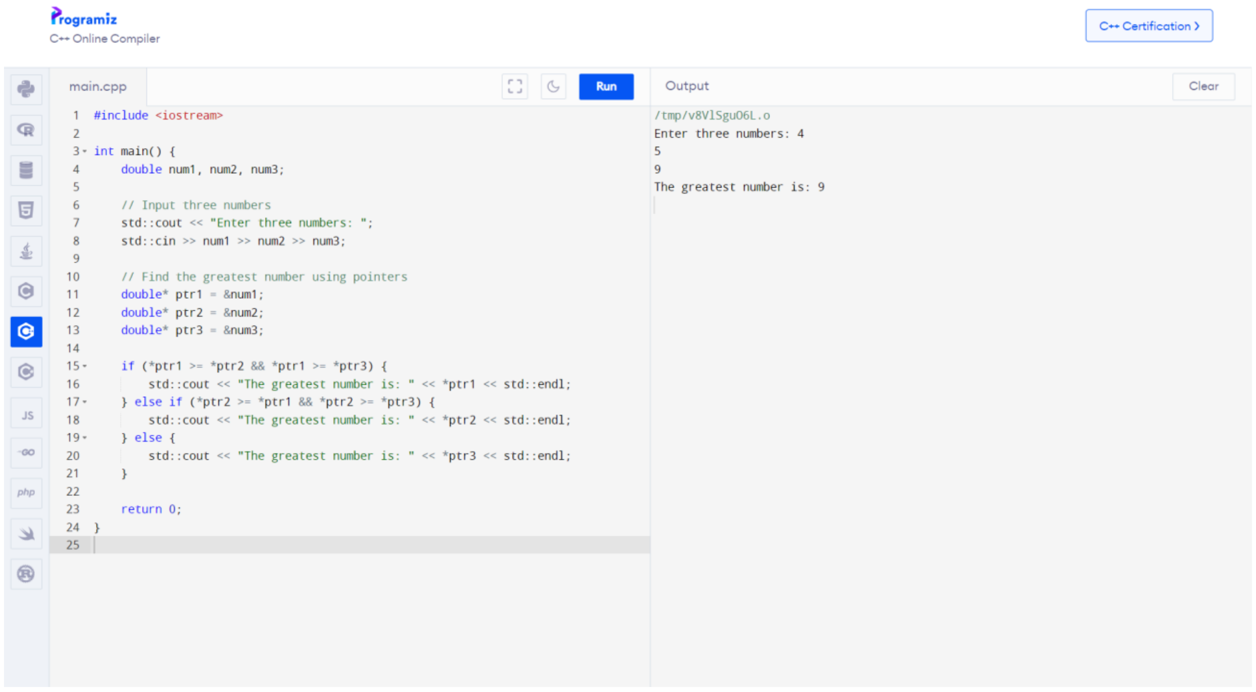
3. Develop a C++ code to find to get square and cube of a number using Hierarchical inheritance



```cpp
        return 0.0;
    }
};

// Derived class for cube calculation
class CubeCalculator : public MathOperations {
public:
    double square(double num) override {
        // Not used in CubeCalculator
        return 0.0;
    }

    double cube(double num) override {
        return num * num * num;
    }
};

int main() {
    double number = 5.0;

    SquareCalculator squareCalc;
    double squareResult = squareCalc.square(number);
    std::cout << "Square of " << number << " is: " << squareResult << std::endl
        ;

    CubeCalculator cubeCalc;
    double cubeResult = cubeCalc.cube(number);
    std::cout << "Cube of " << number << " is: " << cubeResult << std::endl;

    return 0;
}
```

```
/tmp/kBXyWIJWBw.o
Square of 5 is: 25
Cube of 5 is: 125
```

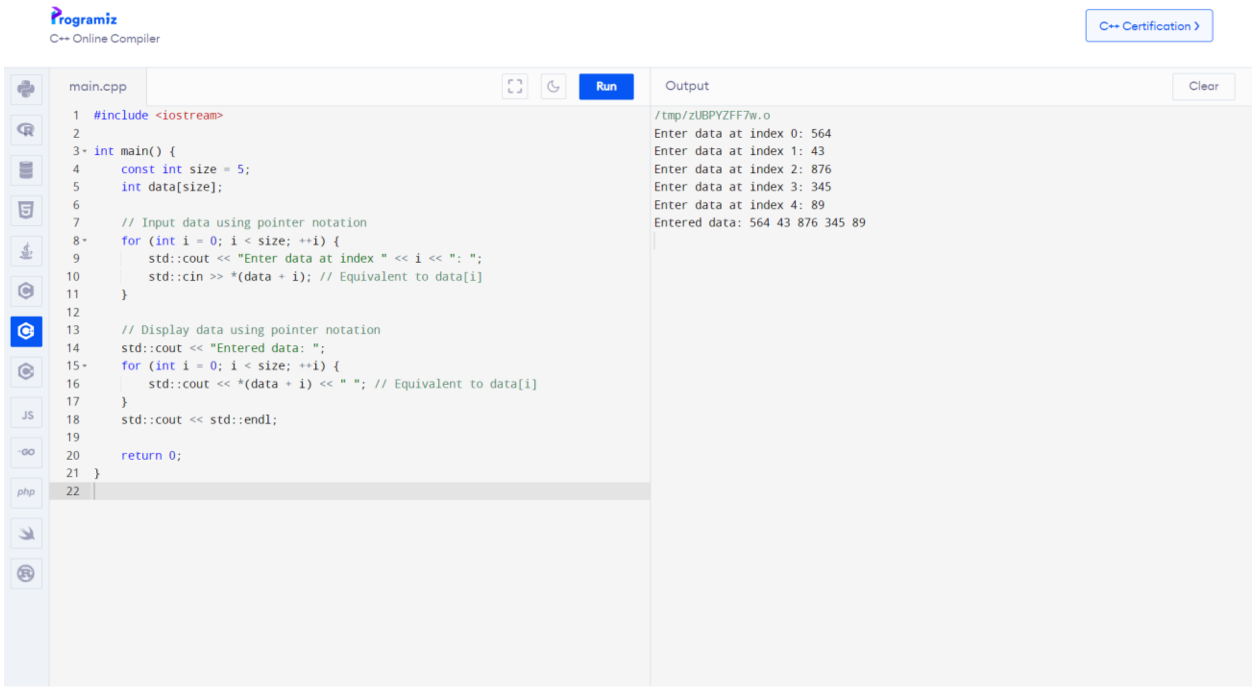
4.Write a C++ Program to find the greatest of three numbers using pointers.



```cpp
#include <iostream>

int main() {
    double num1, num2, num3;

    // Input three numbers
    std::cout << "Enter three numbers: ";
    std::cin >> num1 >> num2 >> num3;

    // Find the greatest number using pointers
    double* ptr1 = &num1;
    double* ptr2 = &num2;
    double* ptr3 = &num3;

    if (*ptr1 >= *ptr2 && *ptr1 >= *ptr3) {
        std::cout << "The greatest number is: " << *ptr1 << std::endl;
    } else if (*ptr2 >= *ptr1 && *ptr2 >= *ptr3) {
        std::cout << "The greatest number is: " << *ptr2 << std::endl;
    } else {
        std::cout << "The greatest number is: " << *ptr3 << std::endl;
    }

    return 0;
}
```

```
/tmp/v8VlSgu06L.o
Enter three numbers: 4
5
9
The greatest number is: 9
```

5. Write a program in C++ to insert and display data entered by using pointer notation.

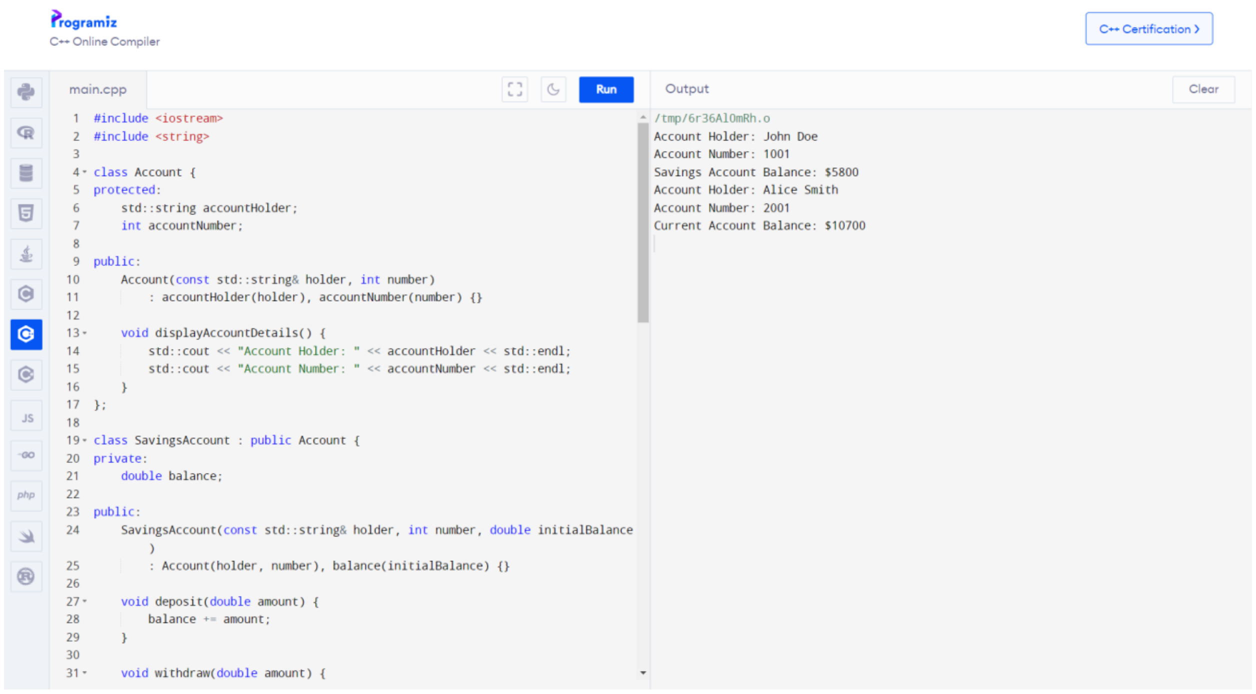


```cpp
#include <iostream>

int main() {
    const int size = 5;
    int data[size];

    // Input data using pointer notation
    for (int i = 0; i < size; ++i) {
        std::cout << "Enter data at index " << i << ": ";
        std::cin >> *(data + i); // Equivalent to data[i]
    }

    // Display data using pointer notation
    std::cout << "Entered data: ";
    for (int i = 0; i < size; ++i) {
        std::cout << *(data + i) << " "; // Equivalent to data[i]
    }
    std::cout << std::endl;

    return 0;
}
```

```
/tmp/zUBPYZFF7w.o
Enter data at index 0: 564
Enter data at index 1: 43
Enter data at index 2: 876
Enter data at index 3: 345
Enter data at index 4: 89
Entered data: 564 43 876 345 89
```

HARD

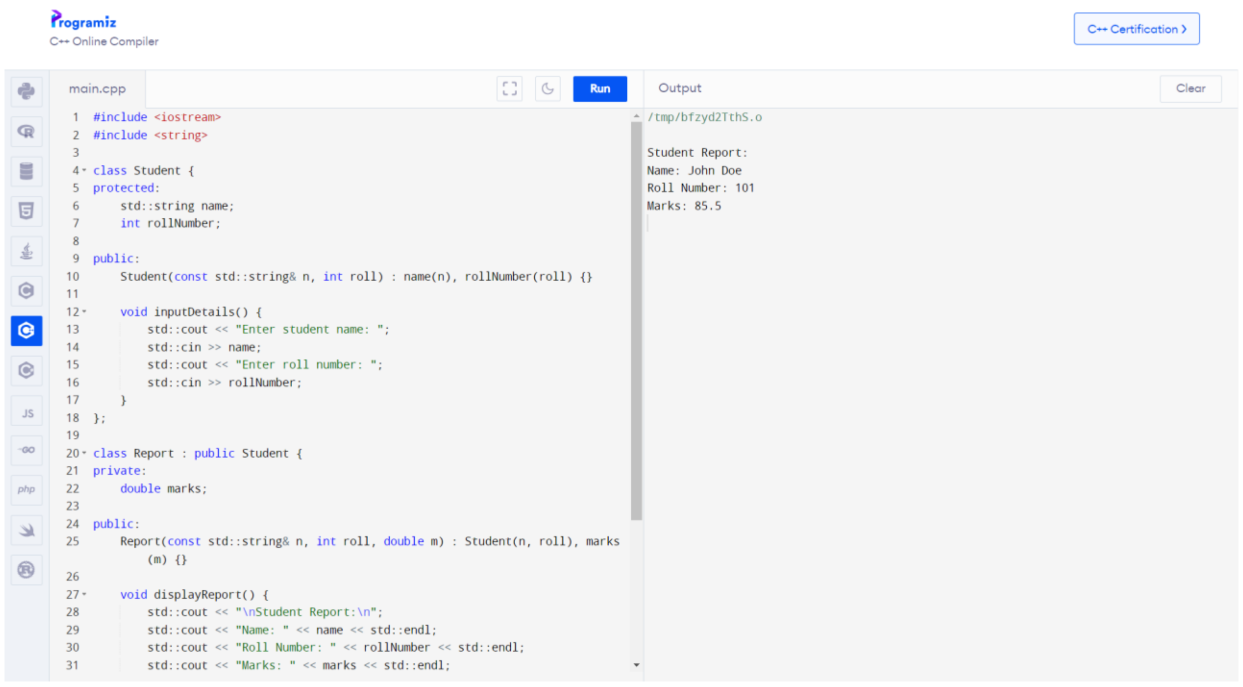
1. Develop a C++ code for a Bank management system using multiple inheritance.



```cpp
#include <iostream>
#include <string>

class Account {
protected:
    std::string accountHolder;
    int accountNumber;

public:
    Account(const std::string& holder, int number)
        : accountHolder(holder), accountNumber(number) {}

    void displayAccountDetails() {
        std::cout << "Account Holder: " << accountHolder << std::endl;
        std::cout << "Account Number: " << accountNumber << std::endl;
    }
};

class SavingsAccount : public Account {
private:
    double balance;

public:
    SavingsAccount(const std::string& holder, int number, double initialBalance
        )
        : Account(holder, number), balance(initialBalance) {}

    void deposit(double amount) {
        balance += amount;
    }

    void withdraw(double amount) {
```

```
/tmp/6r36A1OmRh.o
Account Holder: John Doe
Account Number: 1001
Savings Account Balance: $5800
Account Holder: Alice Smith
Account Number: 2001
Current Account Balance: $10700
```

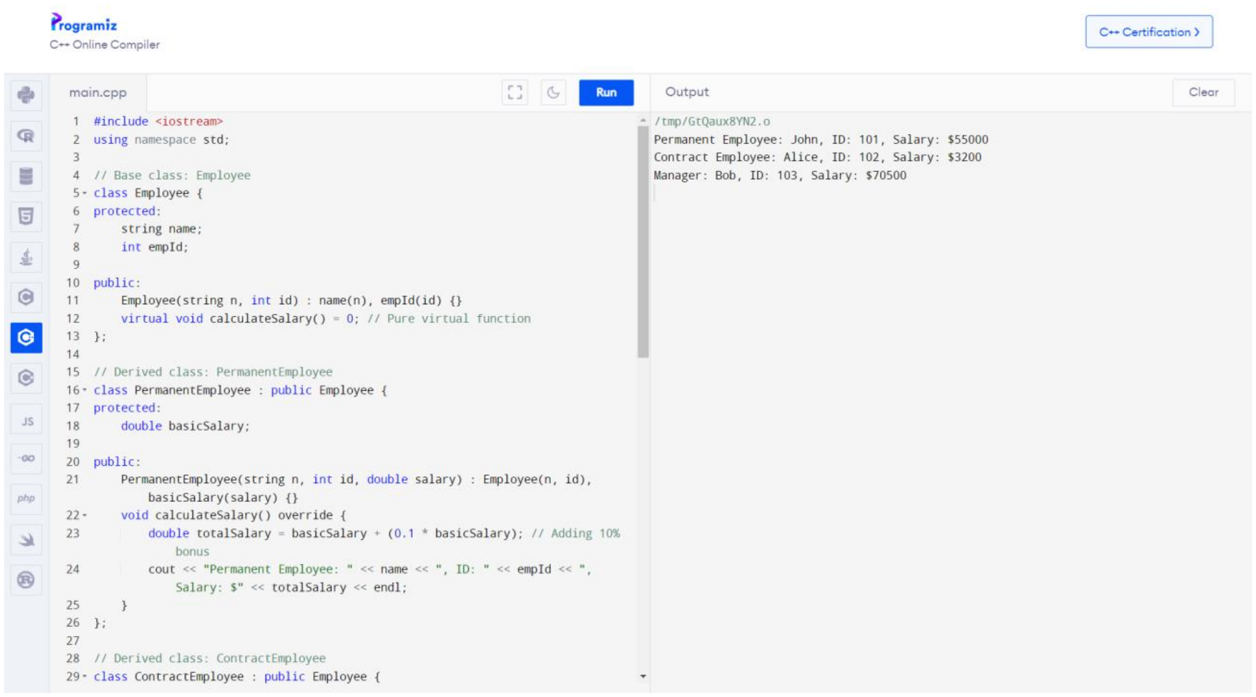
2. Build a C++ program for student report using single inheritance.



```cpp
#include <iostream>
#include <string>

class Student {
protected:
    std::string name;
    int rollNumber;

public:
    Student(const std::string& n, int roll) : name(n), rollNumber(roll) {}

    void inputDetails() {
        std::cout << "Enter student name: ";
        std::cin >> name;
        std::cout << "Enter roll number: ";
        std::cin >> rollNumber;
    }
};

class Report : public Student {
private:
    double marks;

public:
    Report(const std::string& n, int roll, double m) : Student(n, roll), marks
        (m) {}

    void displayReport() {
        std::cout << "\nStudent Report:\n";
        std::cout << "Name: " << name << std::endl;
        std::cout << "Roll Number: " << rollNumber << std::endl;
        std::cout << "Marks: " << marks << std::endl;
```

```
/tmp/bfzyd2TthS.o

Student Report:
Name: John Doe
Roll Number: 101
Marks: 85.5
```

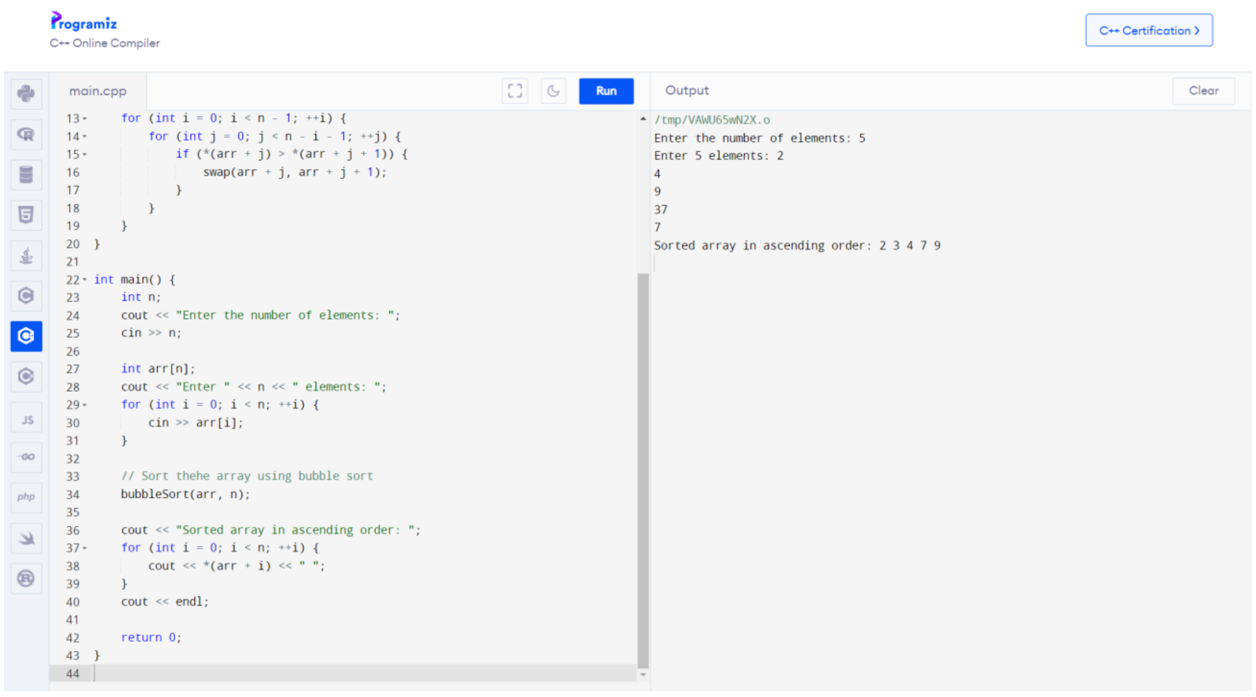
3. Write a C++ code for Employee's salary using hybrid inheritance



```cpp
#include <iostream>
using namespace std;

// Base class: Employee
class Employee {
protected:
    string name;
    int empId;

public:
    Employee(string n, int id) : name(n), empId(id) {}
    virtual void calculateSalary() = 0; // Pure virtual function
};

// Derived class: PermanentEmployee
class PermanentEmployee : public Employee {
protected:
    double basicSalary;

public:
    PermanentEmployee(string n, int id, double salary) : Employee(n, id),
        basicSalary(salary) {}
    void calculateSalary() override {
        double totalSalary = basicSalary + (0.1 * basicSalary); // Adding 10%
            bonus
        cout << "Permanent Employee: " << name << ", ID: " << empId << ",
            Salary: $" << totalSalary << endl;
    }
};

// Derived class: ContractEmployee
class ContractEmployee : public Employee {
```

```
/tmp/GtQaux8YN2.o
Permanent Employee: John, ID: 101, Salary: $55000
Contract Employee: Alice, ID: 102, Salary: $3200
Manager: Bob, ID: 103, Salary: $70500
```

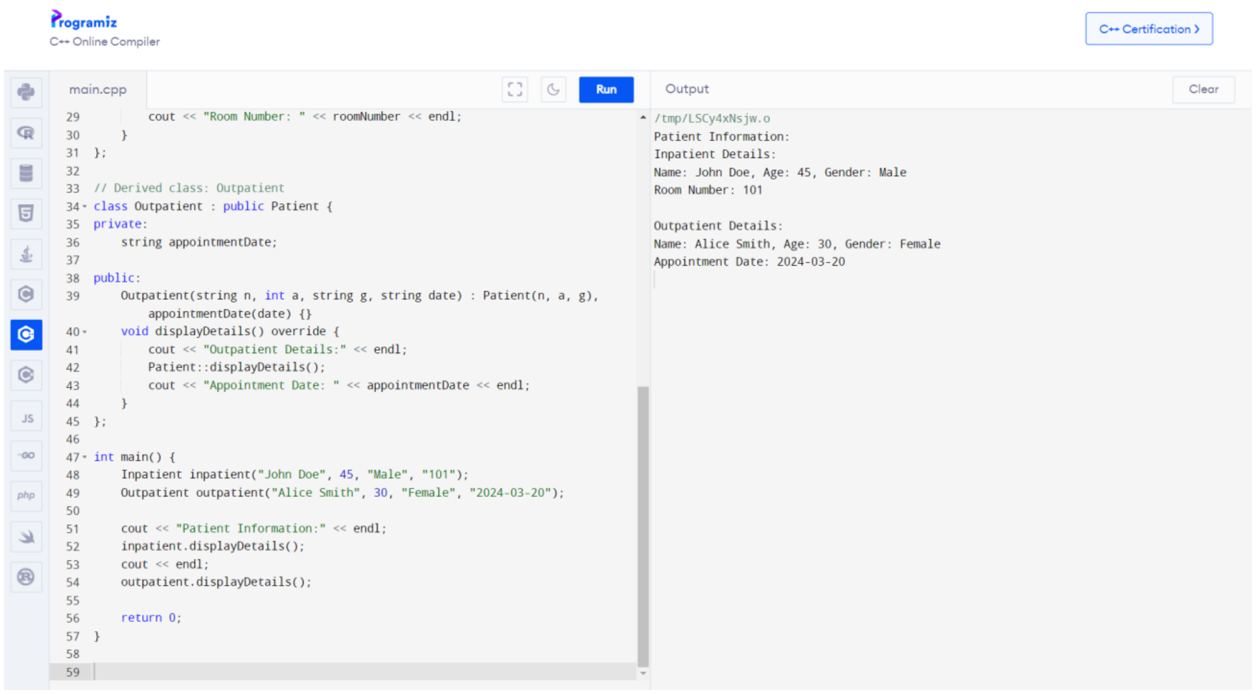
4. Write a C++ program to sort the given list of elements in ascending using pointer



```cpp
    for (int i = 0; i < n - 1; ++i) {
        for (int j = 0; j < n - i - 1; ++j) {
            if (*(arr + j) > *(arr + j + 1)) {
                swap(arr + j, arr + j + 1);
            }
        }
    }
}

int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;

    int arr[n];
    cout << "Enter " << n << " elements: ";
    for (int i = 0; i < n; ++i) {
        cin >> arr[i];
    }

    // Sort thehe array using bubble sort
    bubbleSort(arr, n);

    cout << "Sorted array in ascending order: ";
    for (int i = 0; i < n; ++i) {
        cout << *(arr + i) << " ";
    }
    cout << endl;

    return 0;
}

```

```
/tmp/VAWU65wN2X.o
Enter the number of elements: 5
Enter 5 elements: 2
4
9
37
7
Sorted array in ascending order: 2 3 4 7 9
```

5. Write a C++ Program for Enter Patient details using Inheritance



```cpp
        cout << "Room Number: " << roomNumber << endl;
    }
};

// Derived class: Outpatient
class Outpatient : public Patient {
private:
    string appointmentDate;

public:
    Outpatient(string n, int a, string g, string date) : Patient(n, a, g),
        appointmentDate(date) {}
    void displayDetails() override {
        cout << "Outpatient Details:" << endl;
        Patient::displayDetails();
        cout << "Appointment Date: " << appointmentDate << endl;
    }
};

int main() {
    Inpatient inpatient("John Doe", 45, "Male", "101");
    Outpatient outpatient("Alice Smith", 30, "Female", "2024-03-20");

    cout << "Patient Information:" << endl;
    inpatient.displayDetails();
    cout << endl;
    outpatient.displayDetails();

    return 0;
}

```

```
/tmp/LSCy4xNsjw.o
Patient Information:
Inpatient Details:
Name: John Doe, Age: 45, Gender: Male
Room Number: 101

Outpatient Details:
Name: Alice Smith, Age: 30, Gender: Female
Appointment Date: 2024-03-20
```