



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

Programiz
C++ Online Compiler

[C++ Certification >](#)

```
main.cpp
4 class Person {
5 protected:
6     std::string name;
7     int age;
8
9 public:
10    Person(const std::string& n, int a) : name(n), age(a) {}
11    void display() {
12        std::cout << "Name: " << name << ", Age: " << age << std::endl;
13    }
14 };
15
16 class Student : public Person {
17 private:
18     int rollNumber;
19
20 public:
21    Student(const std::string& n, int a, int roll) : Person(n, a), rollNumber(roll) {}
22    void display() {
23        std::cout << "Student Details:" << std::endl;
24        Person::display();
25        std::cout << "Roll Number: " << rollNumber << std::endl;
26    }
27 };
28
29 int main() {
30     Student s("John Doe", 20, 101);
31     s.display();
32     return 0;
33 }
34
```

Output

```
/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

Programiz
C++ Online Compiler

C++ Certification >

main.cpp

Run

Clear

```
1 #include <string>
2
3
4 class Base {
5 private:
6     int baseData;
7
8 public:
9     Base(int data) : baseData(data) {}
10    void displayBase() {
11        std::cout << "Base Data: " << baseData << std::endl;
12    }
13 };
14
15 class Derived : private Base {
16 private:
17     int derivedData;
18
19 public:
20     Derived(int base, int derived) : Base(base), derivedData(derived) {}
21    void displayDerived() {
22        displayBase(); // Accessing the base class function
23        std::cout << "Derived Data: " << derivedData << std::endl;
24    }
25 };
26
27 int main() {
28     Derived d(10, 20);
29     d.displayDerived();
30     // d.displayBase(); // Error: displayBase() is private in Derived
31     return 0;
32 }
33
```

Output

Clear

/tmp/mfHbv1MSBa.o
Base Data: 10
Derived Data: 20

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

Programiz
C++ Online Compiler

C++ Certification >

main.cpp

Run

Clear

```
1 #include <iostream>
2
3 int main() {
4     int myVariable = 42;
5
6     // Print the address of myVariable
7     std::cout << "Address of myVariable: " << &myVariable << std::endl;
8
9     return 0;
10 }
11
```

Output

Clear

/tmp/hfx4EVD17M.o
Address of myVariable: 0x7ffff58b34c

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

Programiz
C++ Online Compiler

C++ Certification >

```
main.cpp
1 #include <iostream>
2 #include <string>
3
4 class Person {
5 protected:
6     std::string name;
7
8 public:
9     Person(const std::string& n) : name(n) {}
10    void displayPerson() {
11        std::cout << "Person Name: " << name << std::endl;
12    }
13 };
14
15 class Employee {
16 protected:
17     int employeeId;
18
19 public:
20     Employee(int id) : employeeId(id) {}
21    void displayEmployee() {
22        std::cout << "Employee ID: " << employeeId << std::endl;
23    }
24 };
25
26 class Manager : public Person, public Employee {
27 private:
28     std::string department;
29
30 public:
31     Manager(const std::string& n, int id, const std::string& dept)
32         : Person(n), Employee(id), department(dept) {}

```

Output

```
/tmp/CuVSUmRi00.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

Programiz
C++ Online Compiler

C++ Certification >

```
main.cpp
1 #include <iostream>
2 #include <cmath>
3
4 // Abstract class Shape
5 class Shape {
6 public:
7     // Pure virtual function to calculate area
8     virtual double calculateArea() = 0;
9 };
10
11 // Derived class Square
12 class Square : public Shape {
13 private:
14     double side;
15
16 public:
17     Square(double s) : side(s) {}
18    double calculateArea() override {
19        return side * side;
20    }
21 };
22
23 // Derived class Circle
24 class Circle : public Shape {
25 private:
26     double radius;
27
28 public:
29     Circle(double r) : radius(r) {}
30    double calculateArea() override {
31        return M_PI * radius * radius;
32    }

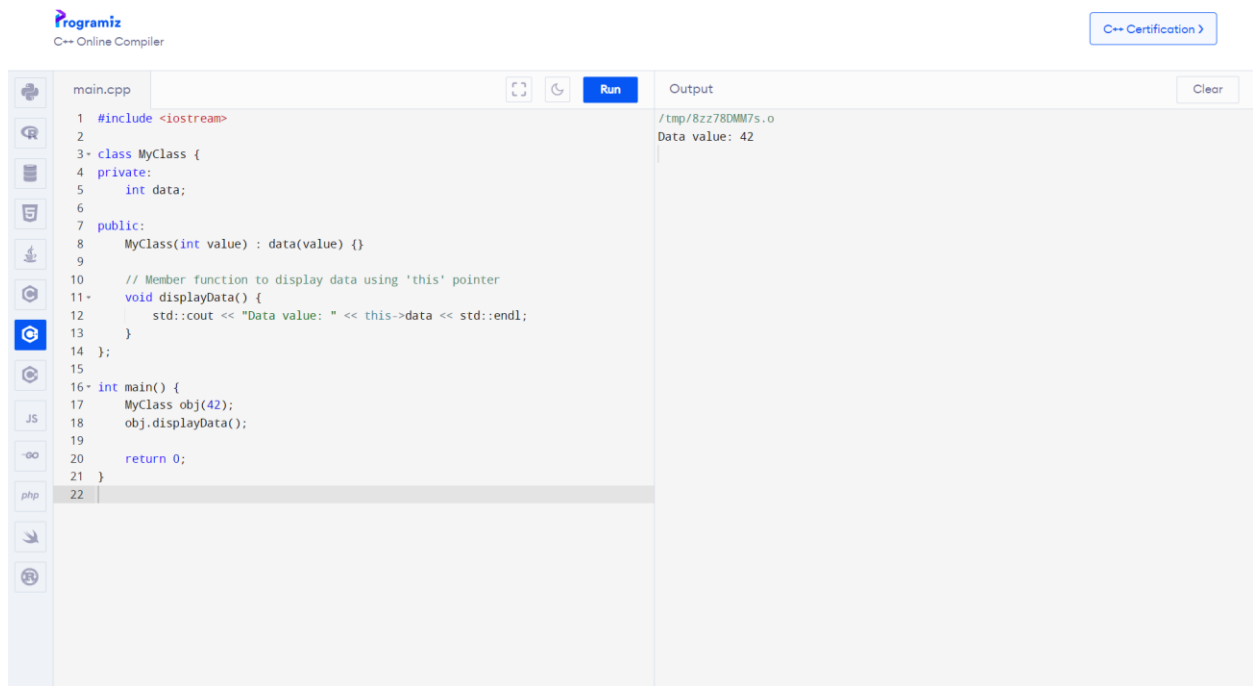
```

Output

```
/tmp/NnnNEGR0hc.o
Area of square: 25
Area of circle: 28.2743

```

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

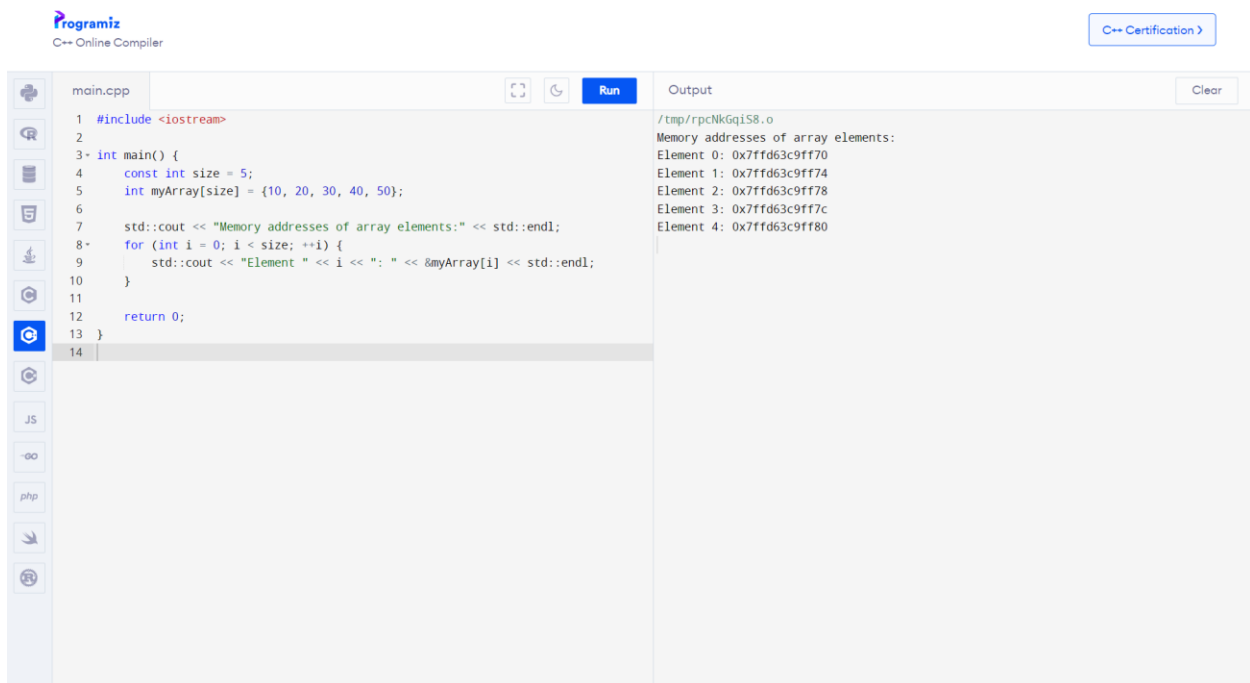


The screenshot shows the Programiz C++ Online Compiler interface. The code editor on the left contains a C++ program named `main.cpp`. The program defines a class `MyClass` with a private member `int data;` and a public constructor `MyClass(int value) : data(value) {}`. It also has a public member function `displayData()` that uses the `this` pointer to access the `data` member. The `main` function creates an object `obj(42)` and calls `obj.displayData()`. The output window on the right shows the result of the program execution.

```
1 #include <iostream>
2
3 class MyClass {
4 private:
5     int data;
6
7 public:
8     MyClass(int value) : data(value) {}
9
10    // Member function to display data using 'this' pointer
11    void displayData() {
12        std::cout << "Data value: " << this->data << std::endl;
13    }
14 };
15
16 int main() {
17     MyClass obj(42);
18     obj.displayData();
19     return 0;
20 }
21
22
```

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

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

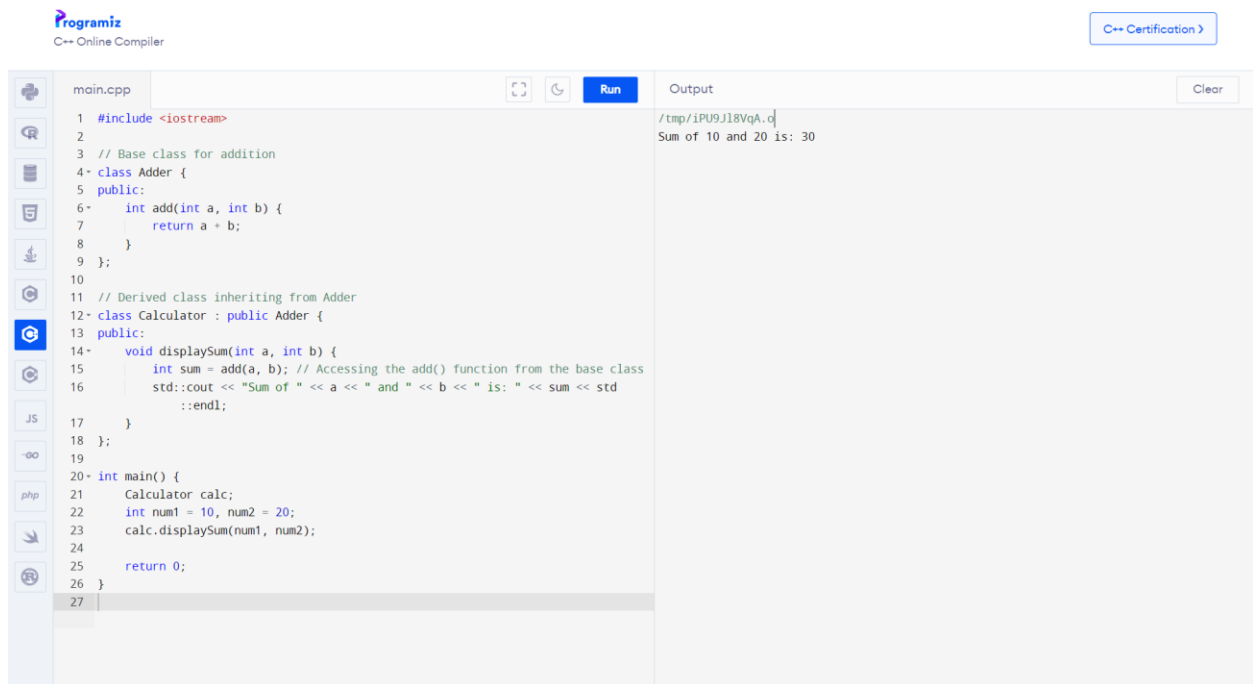


The screenshot shows the Programiz C++ Online Compiler interface. The code editor on the left contains a C++ program named `main.cpp`. The program defines a constant `int size = 5;` and an array `int myArray[size] = {10, 20, 30, 40, 50};`. It then uses a `for` loop to iterate over the array elements and print their memory addresses using the `&myArray[i]` expression. The output window on the right shows the result of the program execution.

```
1 #include <iostream>
2
3 int main() {
4     const int size = 5;
5     int myArray[size] = {10, 20, 30, 40, 50};
6
7     std::cout << "Memory addresses of array elements:" << std::endl;
8     for (int i = 0; i < size; ++i) {
9         std::cout << "Element " << i << ": " << &myArray[i] << std::endl;
10    }
11
12    return 0;
13 }
14
```

Output: /tmp/rpcNkGq158.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.

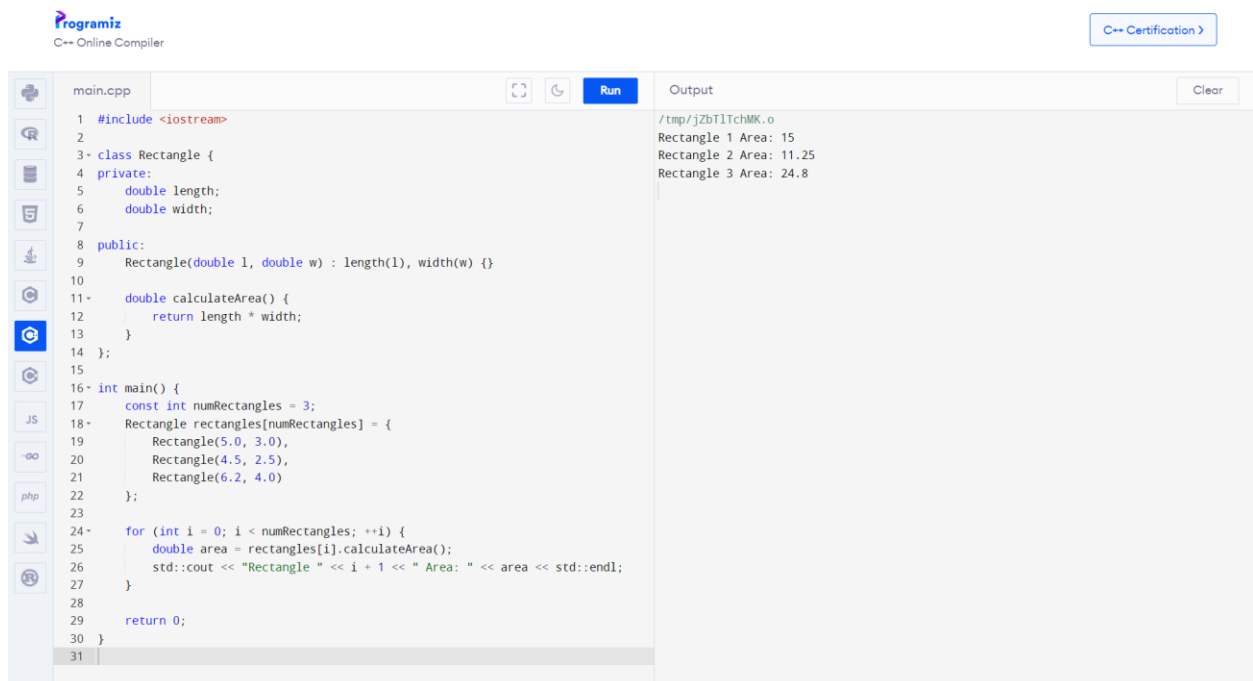


The screenshot shows the Programiz C++ Online Compiler interface. The code in main.cpp defines a base class 'Adder' with an 'add' function. A derived class 'Calculator' inherits from 'Adder' and has a 'displaySum' function that calls 'add' and prints the result. The 'main' function creates a 'Calculator' object and calls 'displaySum' with values 10 and 20.

```
1 #include <iostream>
2
3 // Base class for addition
4 class Adder {
5 public:
6     int add(int a, int b) {
7         return a + b;
8     }
9 };
10
11 // Derived class inheriting from Adder
12 class Calculator : public Adder {
13 public:
14     void displaySum(int a, int b) {
15         int sum = add(a, b); // Accessing the add() function from the base class
16         std::cout << "Sum of " << a << " and " << b << " is: " << sum << std::endl;
17     }
18 };
19
20 int main() {
21     Calculator calc;
22     int num1 = 10, num2 = 20;
23     calc.displaySum(num1, num2);
24
25     return 0;
26 }
```

The output shows: 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.

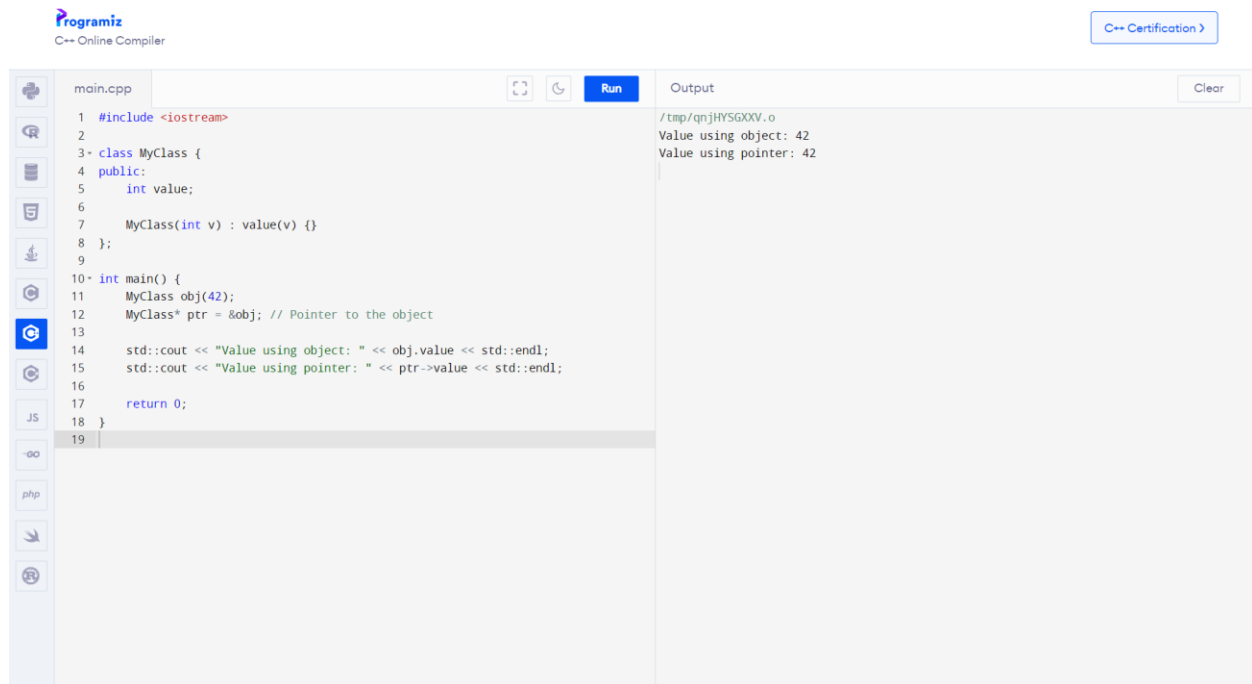


The screenshot shows the Programiz C++ Online Compiler interface. The code in main.cpp defines a 'Rectangle' class with 'length' and 'width' attributes and a 'calculateArea' function. The 'main' function creates an array of three 'Rectangle' objects with different dimensions and iterates through them to calculate and print their areas.

```
1 #include <iostream>
2
3 class Rectangle {
4 private:
5     double length;
6     double width;
7
8 public:
9     Rectangle(double l, double w) : length(l), width(w) {}
10
11     double calculateArea() {
12         return length * width;
13     }
14 };
15
16 int main() {
17     const int numRectangles = 3;
18     Rectangle rectangles[numRectangles] = {
19         Rectangle(5.0, 3.0),
20         Rectangle(4.5, 2.5),
21         Rectangle(6.2, 4.0)
22     };
23
24     for (int i = 0; i < numRectangles; ++i) {
25         double area = rectangles[i].calculateArea();
26         std::cout << "Rectangle " << i + 1 << " Area: " << area << std::endl;
27     }
28
29     return 0;
30 }
```

The output shows: 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



The screenshot shows the Programiz C++ Online Compiler interface. The code in main.cpp is as follows:

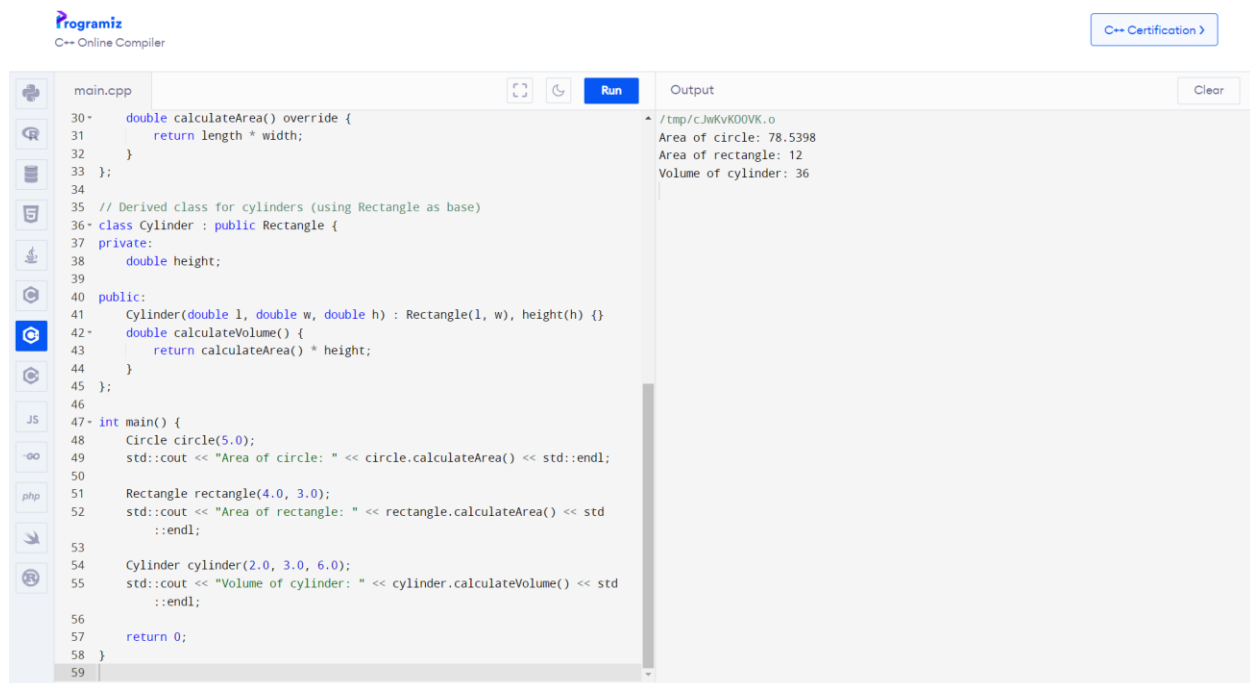
```
1 #include <iostream>
2
3 class MyClass {
4 public:
5     int value;
6
7     MyClass(int v) : value(v) {}
8 };
9
10 int main() {
11     MyClass obj(42);
12     MyClass* ptr = &obj; // Pointer to the object
13
14     std::cout << "Value using object: " << obj.value << std::endl;
15     std::cout << "Value using pointer: " << ptr->value << std::endl;
16
17     return 0;
18 }
19
```

The output of the program is:

```
/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.



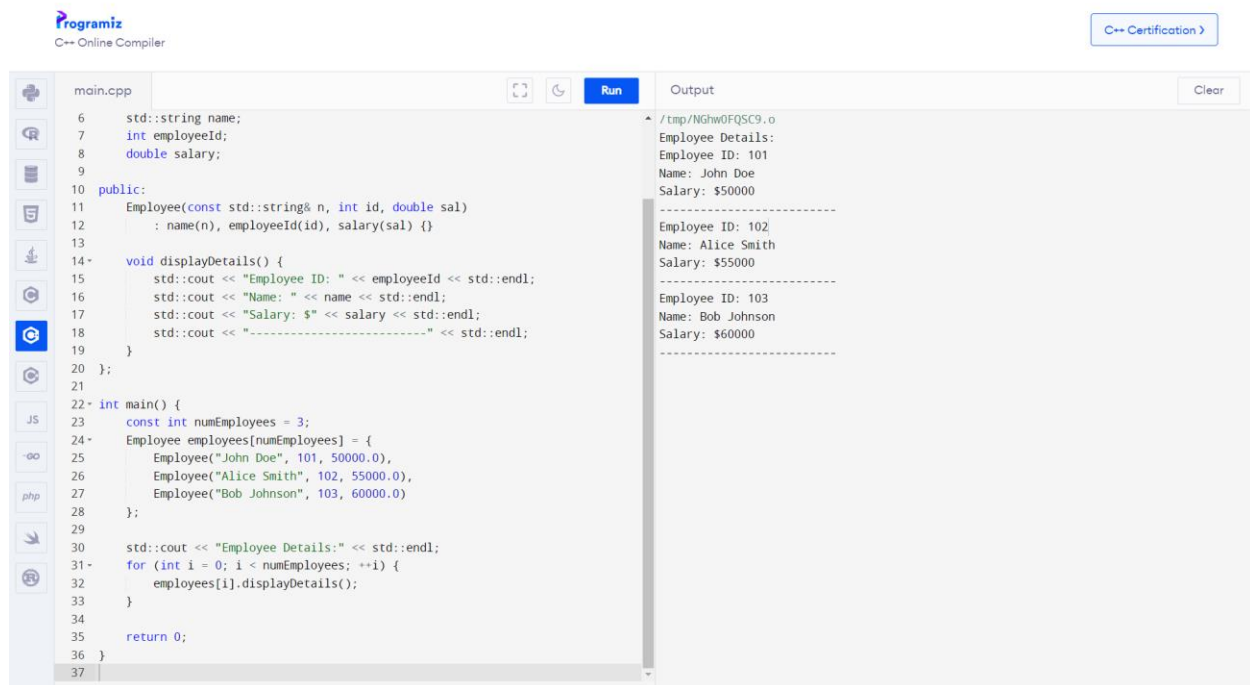
The screenshot shows the Programiz C++ Online Compiler interface. The code in main.cpp is as follows:

```
30 double calculateArea() override {
31     return length * width;
32 }
33 };
34
35 // Derived class for cylinders (using Rectangle as base)
36 class Cylinder : public Rectangle {
37 private:
38     double height;
39
40 public:
41     Cylinder(double l, double w, double h) : Rectangle(l, w), height(h) {}
42     double calculateVolume() {
43         return calculateArea() * height;
44     }
45 };
46
47 int main() {
48     Circle circle(5.0);
49     std::cout << "Area of circle: " << circle.calculateArea() << std::endl;
50
51     Rectangle rectangle(4.0, 3.0);
52     std::cout << "Area of rectangle: " << rectangle.calculateArea() << std::endl;
53
54     Cylinder cylinder(2.0, 3.0, 6.0);
55     std::cout << "Volume of cylinder: " << cylinder.calculateVolume() << std::endl;
56
57     return 0;
58 }
59
```

The output of the program is:

```
/tmp/cJmKvK00VK.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



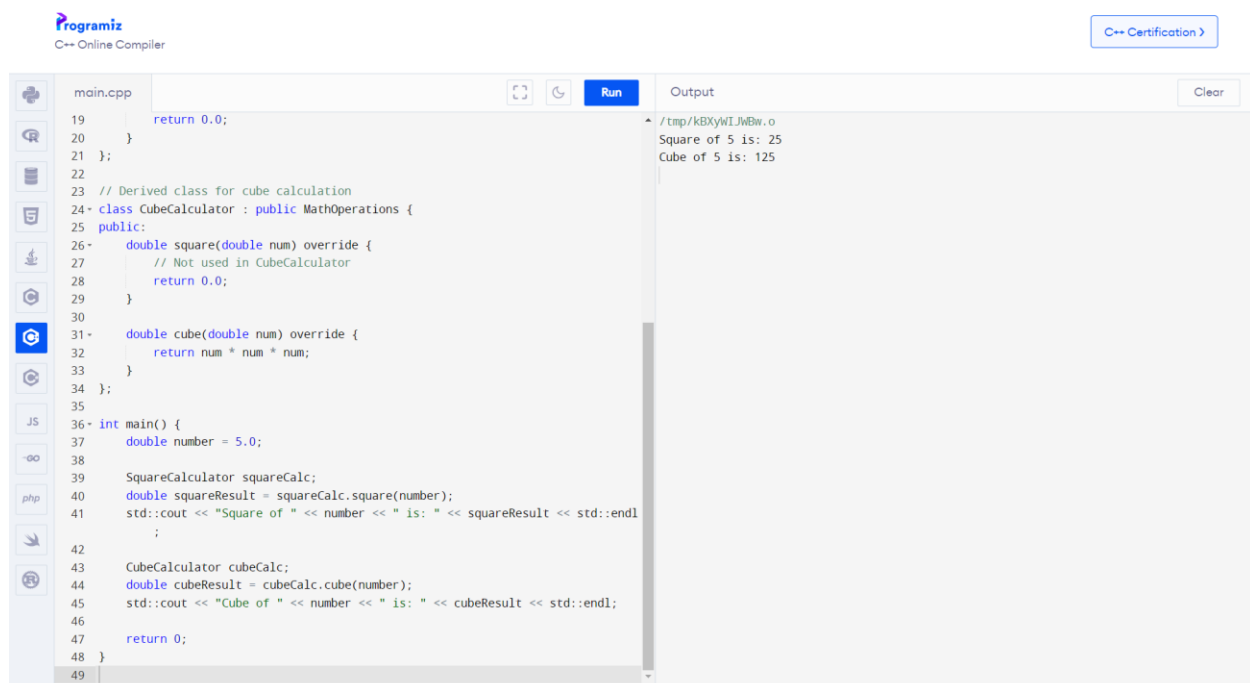
The screenshot shows the Programiz C++ Online Compiler interface. The code in main.cpp defines an Employee class with attributes name, employeeId, and salary. It includes a constructor, a displayDetails method, and a main function that creates an array of three Employee objects and prints their details.

```
main.cpp
6  std::string name;
7  int employeeId;
8  double salary;
9
10 public:
11     Employee(const std::string& n, int id, double sal)
12         : name(n), employeeId(id), salary(sal) {}
13
14     void displayDetails() {
15         std::cout << "Employee ID: " << employeeId << std::endl;
16         std::cout << "Name: " << name << std::endl;
17         std::cout << "Salary: $" << salary << std::endl;
18         std::cout << "-----" << std::endl;
19     }
20 };
21
22 int main() {
23     const int numEmployees = 3;
24     Employee employees[numEmployees] = {
25         Employee("John Doe", 101, 50000.0),
26         Employee("Alice Smith", 102, 55000.0),
27         Employee("Bob Johnson", 103, 60000.0)
28     };
29
30     std::cout << "Employee Details:" << std::endl;
31     for (int i = 0; i < numEmployees; ++i) {
32         employees[i].displayDetails();
33     }
34
35     return 0;
36 }
37
```

Output

```
/tmp/NGhw0FQSC9.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



The screenshot shows the Programiz C++ Online Compiler interface. The code defines a base class MathOperations and a derived class CubeCalculator. The CubeCalculator class overrides the square method to return 0.0 and implements the cube method. The main function creates a CubeCalculator object and prints the square and cube of the number 5.

```
main.cpp
19     return 0.0;
20 }
21 };
22
23 // Derived class for cube calculation
24 class CubeCalculator : public MathOperations {
25 public:
26     double square(double num) override {
27         // Not used in CubeCalculator
28         return 0.0;
29     }
30
31     double cube(double num) override {
32         return num * num * num;
33     }
34 };
35
36 int main() {
37     double number = 5.0;
38
39     SquareCalculator squareCalc;
40     double squareResult = squareCalc.square(number);
41     std::cout << "Square of " << number << " is: " << squareResult << std::endl;
42
43     CubeCalculator cubeCalc;
44     double cubeResult = cubeCalc.cube(number);
45     std::cout << "Cube of " << number << " is: " << cubeResult << std::endl;
46
47     return 0;
48 }
49
```

Output

```
/tmp/kBXyWtJwBw.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.

Programiz
C++ Online Compiler

C++ Certification >

main.cpp

Run

Output

Clear

```
1 #include <iostream>
2
3 int main() {
4     double num1, num2, num3;
5
6     // Input three numbers
7     std::cout << "Enter three numbers: ";
8     std::cin >> num1 >> num2 >> num3;
9
10    // Find the greatest number using pointers
11    double* ptr1 = &num1;
12    double* ptr2 = &num2;
13    double* ptr3 = &num3;
14
15    if (*ptr1 >= *ptr2 && *ptr1 >= *ptr3) {
16        std::cout << "The greatest number is: " << *ptr1 << std::endl;
17    } else if (*ptr2 >= *ptr1 && *ptr2 >= *ptr3) {
18        std::cout << "The greatest number is: " << *ptr2 << std::endl;
19    } else {
20        std::cout << "The greatest number is: " << *ptr3 << std::endl;
21    }
22
23    return 0;
24 }
25
```

/tmp/v8V1Sgu06L.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.

Programiz
C++ Online Compiler

C++ Certification >

main.cpp

Run

Output

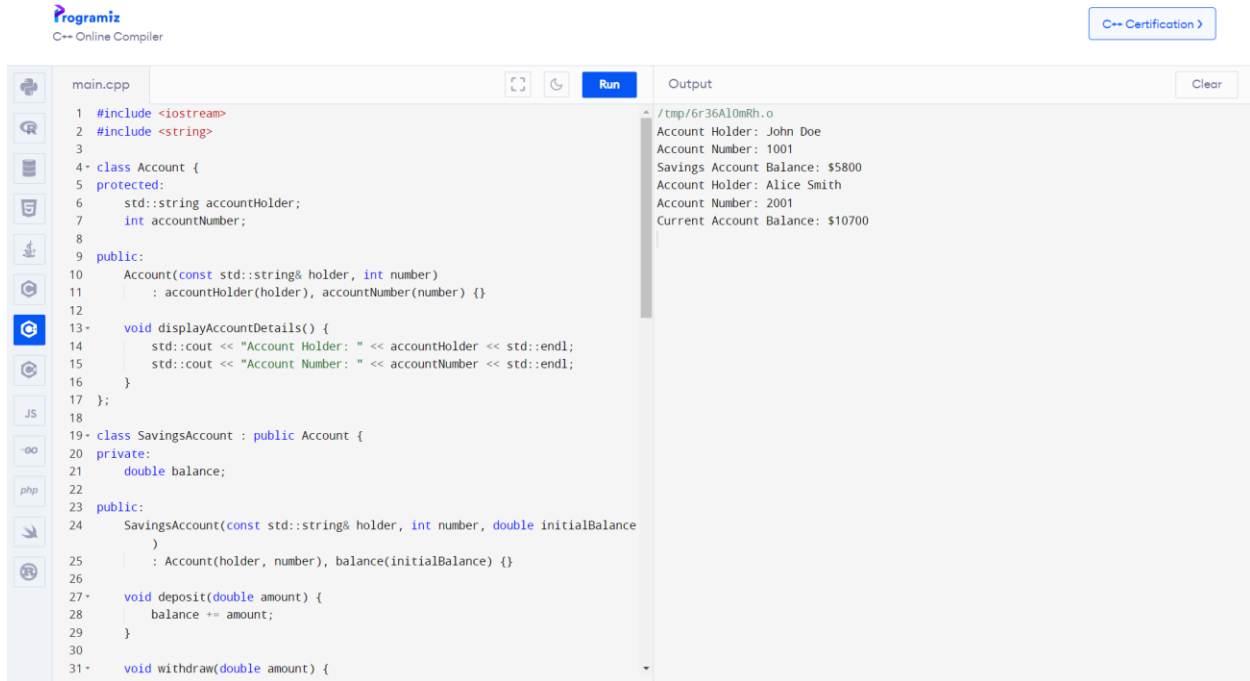
Clear

```
1 #include <iostream>
2
3 int main() {
4     const int size = 5;
5     int data[size];
6
7     // Input data using pointer notation
8     for (int i = 0; i < size; ++i) {
9         std::cout << "Enter data at index " << i << ": ";
10        std::cin >> *(data + i); // Equivalent to data[i]
11    }
12
13    // Display data using pointer notation
14    std::cout << "Entered data: ";
15    for (int i = 0; i < size; ++i) {
16        std::cout << *(data + i) << " "; // Equivalent to data[i]
17    }
18    std::cout << std::endl;
19
20    return 0;
21 }
22
```

/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.

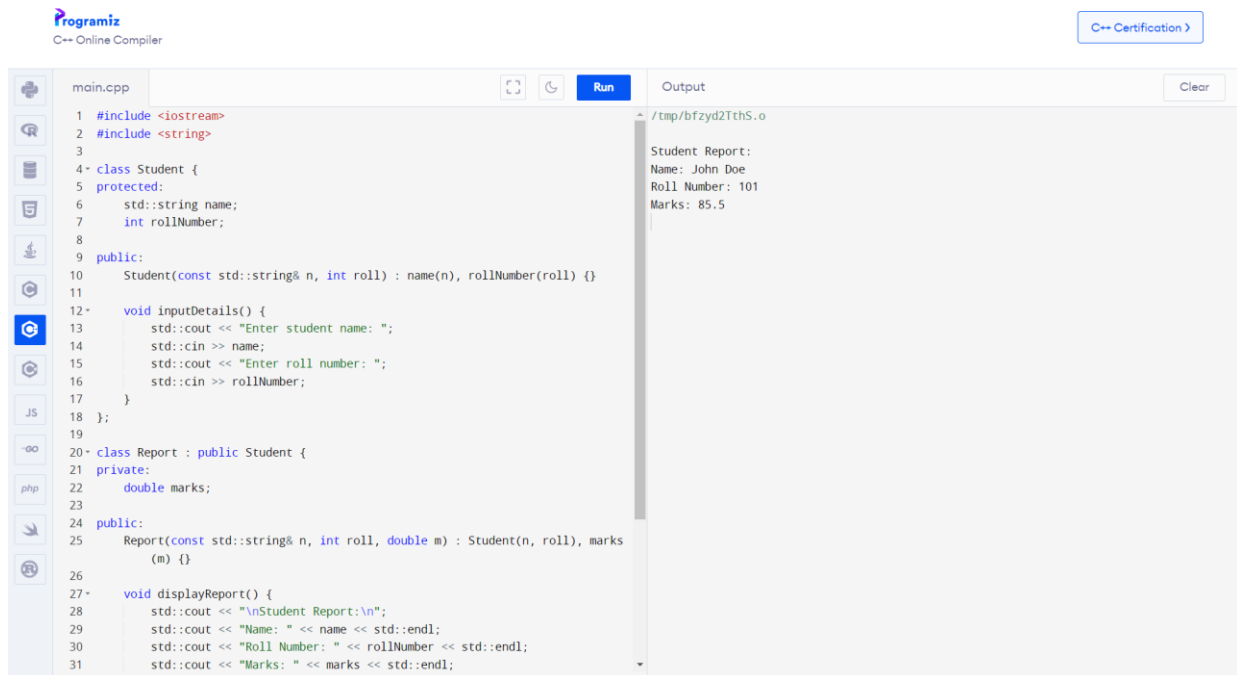


```
1 #include <iostream>
2 #include <string>
3
4 class Account {
5 protected:
6     std::string accountHolder;
7     int accountNumber;
8
9 public:
10     Account(const std::string& holder, int number)
11         : accountHolder(holder), accountNumber(number) {}
12
13     void displayAccountDetails() {
14         std::cout << "Account Holder: " << accountHolder << std::endl;
15         std::cout << "Account Number: " << accountNumber << std::endl;
16     }
17 };
18
19 class SavingsAccount : public Account {
20 private:
21     double balance;
22
23 public:
24     SavingsAccount(const std::string& holder, int number, double initialBalance)
25         : Account(holder, number), balance(initialBalance) {}
26
27     void deposit(double amount) {
28         balance += amount;
29     }
30
31     void withdraw(double amount) {
```

Output

```
/tmp/6r36A10mRh.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.



```
1 #include <iostream>
2 #include <string>
3
4 class Student {
5 protected:
6     std::string name;
7     int rollNumber;
8
9 public:
10     Student(const std::string& n, int roll) : name(n), rollNumber(roll) {}
11
12     void inputDetails() {
13         std::cout << "Enter student name: ";
14         std::cin >> name;
15         std::cout << "Enter roll number: ";
16         std::cin >> rollNumber;
17     }
18 };
19
20 class Report : public Student {
21 private:
22     double marks;
23
24 public:
25     Report(const std::string& n, int roll, double m) : Student(n, roll), marks(m) {}
26
27     void displayReport() {
28         std::cout << "\nStudent Report:\n";
29         std::cout << "Name: " << name << std::endl;
30         std::cout << "Roll Number: " << rollNumber << std::endl;
31         std::cout << "Marks: " << marks << std::endl;
```

Output

```
/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

Programiz
C++ Online Compiler

C++ Certification >

```
main.cpp
1 #include <iostream>
2 using namespace std;
3
4 // Base class: Employee
5 class Employee {
6 protected:
7     string name;
8     int empId;
9
10 public:
11     Employee(string n, int id) : name(n), empId(id) {}
12     virtual void calculateSalary() = 0; // Pure virtual function
13 };
14
15 // Derived class: PermanentEmployee
16 class PermanentEmployee : public Employee {
17 protected:
18     double basicSalary;
19
20 public:
21     PermanentEmployee(string n, int id, double salary) : Employee(n, id),
22         basicSalary(salary) {}
23     void calculateSalary() override {
24         double totalSalary = basicSalary + (0.1 * basicSalary); // Adding 10%
25         bonus
26         cout << "Permanent Employee: " << name << ", ID: " << empId << ",
27             Salary: $" << totalSalary << endl;
28     }
29 };
30
31 // Derived class: ContractEmployee
32 class ContractEmployee : public Employee {
```

Output

```
/tmp/GtQaux8YNZ.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

Programiz
C++ Online Compiler

C++ Certification >

```
main.cpp
13 for (int i = 0; i < n - 1; ++i) {
14     for (int j = 0; j < n - i - 1; ++j) {
15         if (*(arr + j) > *(arr + j + 1)) {
16             swap(arr + j, arr + j + 1);
17         }
18     }
19 }
20 }
21
22 int main() {
23     int n;
24     cout << "Enter the number of elements: ";
25     cin >> n;
26
27     int arr[n];
28     cout << "Enter " << n << " elements: ";
29     for (int i = 0; i < n; ++i) {
30         cin >> arr[i];
31     }
32
33     // Sort the array using bubble sort
34     bubbleSort(arr, n);
35
36     cout << "Sorted array in ascending order: ";
37     for (int i = 0; i < n; ++i) {
38         cout << *(arr + i) << " ";
39     }
40     cout << endl;
41
42     return 0;
43 }
44
```

Output

```
/tmp/VAMU6SmN2X.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

Programiz
C++ Online Compiler

C++ Certification >

main.cpp

```
29     cout << "Room Number: " << roomNumber << endl;
30 }
31 };
32
33 // Derived class: Outpatient
34 class Outpatient : public Patient {
35 private:
36     string appointmentDate;
37
38 public:
39     Outpatient(string n, int a, string g, string date) : Patient(n, a, g),
40         appointmentDate(date) {}
41     void displayDetails() override {
42         cout << "Outpatient Details:" << endl;
43         Patient::displayDetails();
44         cout << "Appointment Date: " << appointmentDate << endl;
45     }
46 };
47
48 int main() {
49     Inpatient inpatient("John Doe", 45, "Male", "101");
50     Outpatient outpatient("Alice Smith", 30, "Female", "2024-03-20");
51
52     cout << "Patient Information:" << endl;
53     inpatient.displayDetails();
54     cout << endl;
55     outpatient.displayDetails();
56
57     return 0;
58 }
59
```

Output

```
/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
```