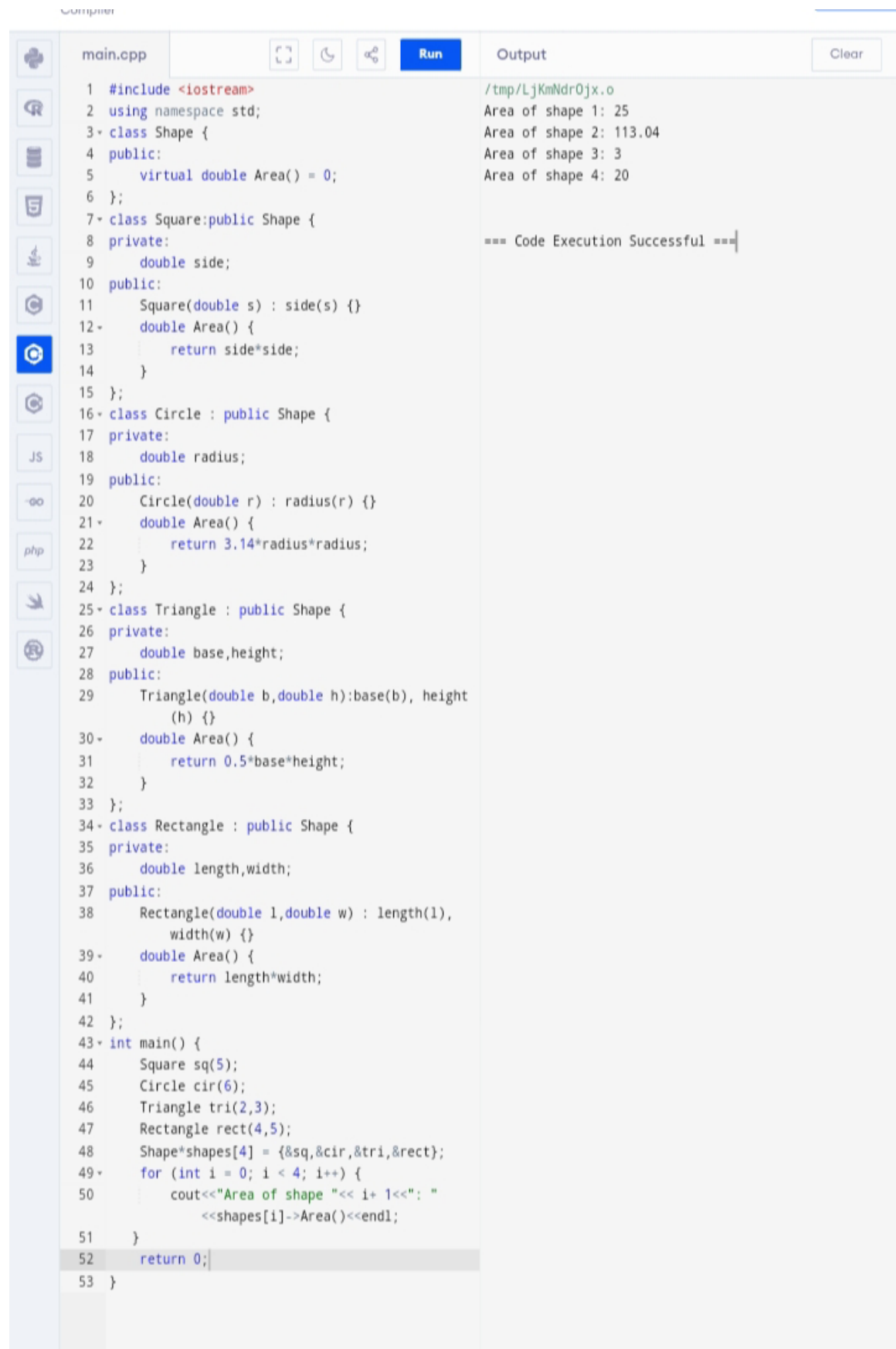


1. Write C++ program to overload Area() function to calculate area of different shapes like square, circle, triangle and rectangle.



The screenshot shows a C++ IDE with a file named 'main.cpp'. The code defines a base class 'Shape' with a virtual 'Area()' function. It then defines four derived classes: 'Square', 'Circle', 'Triangle', and 'Rectangle', each with its own 'Area()' implementation. The 'main()' function creates instances of these shapes and prints their areas. The output window shows the results of the program execution.

```
1 #include <iostream>
2 using namespace std;
3 class Shape {
4 public:
5     virtual double Area() = 0;
6 };
7 class Square:public Shape {
8 private:
9     double side;
10 public:
11     Square(double s) : side(s) {}
12     double Area() {
13         return side*side;
14     }
15 };
16 class Circle : public Shape {
17 private:
18     double radius;
19 public:
20     Circle(double r) : radius(r) {}
21     double Area() {
22         return 3.14*radius*radius;
23     }
24 };
25 class Triangle : public Shape {
26 private:
27     double base,height;
28 public:
29     Triangle(double b,double h):base(b), height
        (h) {}
30     double Area() {
31         return 0.5*base*height;
32     }
33 };
34 class Rectangle : public Shape {
35 private:
36     double length,width;
37 public:
38     Rectangle(double l,double w) : length(l),
        width(w) {}
39     double Area() {
40         return length*width;
41     }
42 };
43 int main() {
44     Square sq(5);
45     Circle cir(6);
46     Triangle tri(2,3);
47     Rectangle rect(4,5);
48     Shape*shapes[4] = {&sq,&cir,&tri,&rect};
49     for (int i = 0; i < 4; i++) {
50         cout<<"Area of shape "<< i+ 1<<": "
            <<shapes[i]->Area()<<endl;
51     }
52     return 0;
53 }
```

Output:

```
/tmp/LjKmNdr0jx.o
Area of shape 1: 25
Area of shape 2: 113.04
Area of shape 3: 3
Area of shape 4: 20

=== Code Execution Successful ===
```

2. Create a class Student with data members regno, Name, mobile, dep[10], sem, year and cgpa. Create member function StudDetails() and friend function maxmark().  
 StudDetails()- Displays the student details.  
 Maxmark()- Finds and prints the student who scored maximum cgpa.  
 Create 2 Student objects, display their details and print the maximum mark scored student.

```

main.cpp
1  #include <iostream>
2  using namespace std;
3  class Student{
4      char*regno;
5      char*name;
6      char*mobile;
7      char*dep;
8      int sem;
9      int year,cgpa;
10 public:
11     Student(){}
12     Student(char*r,char*n,char*m,char*d,int s,
13             int y,int c){
14         regno = r;
15         name = n;
16         mobile = m;
17         dep = d;
18         sem = s;
19         year = y;
20         cgpa = c;
21     }
22     void StudDetails(){
23         cout <<"Regno : "<<regno<<"\nName : "<<
24             name<<"\nMobile : "<< mobile
25             <<"\nDepartment : "<<dep<<"\nSemester
26             "<<sem<<"\nYear : "<<year<<"\nCGPA : "<<
27             cgpa<<endl;
28     }
29     friend void Maxmark(Student a, Student b);
30 };
31 void Maxmark(Student a, Student b){
32     Student c;
33     if(a.cgpa > b.cgpa){
34         c.cgpa = a.cgpa;
35         c.name = a.name;
36     }
37     else{
38         c.cgpa = b.cgpa;
39         c.name = b.name;
40     }
41     cout<<"The Max mark Scored Student Details :
42         <<endl<<"Name : "<<c.name<<endl<<"cgpa
43         : "<<c.cgpa;
44 }
45 int main()
46 {
47     Student s1,s2;
48     s1 = Student("61772231031","Suresh",
49                 "9876543210","ECE", 5, 4, 9.15);
50     s2 = Student("61772231015","Ramesh",
51                 "9123456780","ECE", 4, 5, 10);
52     s1.StudDetails();
53     cout<<"\n\n\n";
54     s2.StudDetails();
55     cout<<"\n\n\n";
56     Maxmark(s1,s2);
57 }
  
```

```

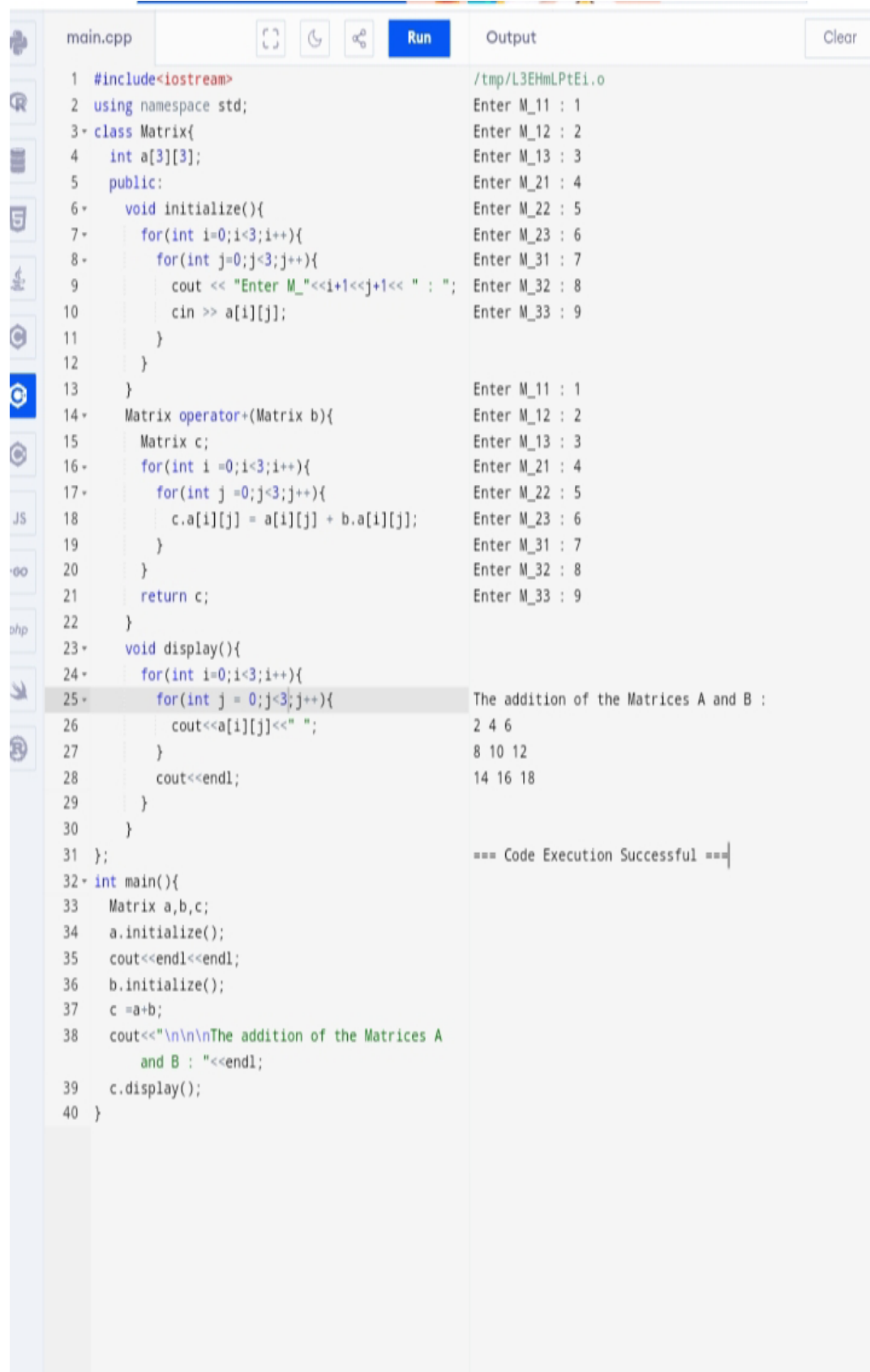
/tmp/kv7myJiAAI.o
Regno : 61772231031
Name : Suresh
Mobile : 9876543210
Department : ECE
Semester : 5
Year : 4
CGPA : 9

Regno : 61772231015
Name : Ramesh
Mobile : 9123456780
Department : ECE
Semester : 4
Year : 5
CGPA : 10

The Max mark Scored Student Details :
Name : Ramesh
cgpa : 10

=== Code Execution Successful ===
  
```

3. Implement a C++ program to calculate the sum of two matrices using operator overloading.



The screenshot shows a C++ IDE with a file named `main.cpp`. The code defines a `Matrix` class with a `3x3` integer array `a`. It includes an `initialize()` method for input, a `display()` method for output, and an `operator+` overload that adds two matrices. The `main` function creates two matrices `a` and `b`, initializes them, and then calculates their sum `c = a + b`, displaying the result.

```
1 #include<iostream>
2 using namespace std;
3 class Matrix{
4     int a[3][3];
5     public:
6     void initialize(){
7         for(int i=0;i<3;i++){
8             for(int j=0;j<3;j++){
9                 cout << "Enter M_"<<i+1<<j+1<< " : ";
10                cin >> a[i][j];
11            }
12        }
13    }
14    Matrix operator+(Matrix b){
15        Matrix c;
16        for(int i =0;i<3;i++){
17            for(int j =0;j<3;j++){
18                c.a[i][j] = a[i][j] + b.a[i][j];
19            }
20        }
21        return c;
22    }
23    void display(){
24        for(int i=0;i<3;i++){
25            for(int j = 0;j<3;j++){
26                cout<<a[i][j]<<" ";
27            }
28            cout<<endl;
29        }
30    }
31 };
32 int main(){
33     Matrix a,b,c;
34     a.initialize();
35     cout<<endl<<endl;
36     b.initialize();
37     c =a+b;
38     cout<<"\n\n\nThe addition of the Matrices A
39     and B : "<<endl;
40     c.display();
41 }
```

The output window shows the execution results, including prompts for matrix elements and the final sum matrix.

```
/tmp/L3EHmLPtEi.o
Enter M_11 : 1
Enter M_12 : 2
Enter M_13 : 3
Enter M_21 : 4
Enter M_22 : 5
Enter M_23 : 6
Enter M_31 : 7
Enter M_32 : 8
Enter M_33 : 9

Enter M_11 : 1
Enter M_12 : 2
Enter M_13 : 3
Enter M_21 : 4
Enter M_22 : 5
Enter M_23 : 6
Enter M_31 : 7
Enter M_32 : 8
Enter M_33 : 9

The addition of the Matrices A and B :
2 4 6
8 10 12
14 16 18

=== Code Execution Successful ===
```

4. Define a class String that could work as a user-defined string type. Include constructors that will enable us to create an uninitialized string String s1; and also to initialize an object with a string constant at the time of creation like String s2("Welcome"); Include a function that adds two strings to make a third string.

Write a complete program to test your class to see that it does the following tasks:

- (a) Creates uninitialized string objects.
- (b) Creates objects with string constants.
- (c) Concatenates two strings properly.
- (d) Displays a desired string object.



```
main.cpp
1 #include<iostream>
2 using namespace std;
3 class String{
4     char *ch;
5 public:
6     String(){
7         ch = new char[50];
8     }
9     String(char *c){
10         ch =c;
11     }
12     String add(String b){
13         String c;
14         int i;
15         for(i = 0;ch[i]!='\0';i++){
16             c.ch[i]=ch[i];
17         }
18         for(int j=0;b.ch[j]!='\0';j++){
19             c.ch[i]=b.ch[j];
20             i++;
21         }
22         return c;
23     }
24     void display(){
25         for(int i=0;ch[i]!='\0';i++){
26             cout<<ch[i];
27         }
28     }
29 };
30 int main()
31 {
32     String s1,s2;
33     s1 = String(" Good");
34     s2 = String(" Morning");
35     String s3=s1.add(s2);
36     cout<<"String 1 : ";
37     s1.display();
38     cout<<endl<<"String 2 : ";
39     s2.display();
40     cout<<endl<<"String 1 + String 2 : ";
41     s3.display();
42 }
```

```
/tmp/pvZNcPX8Ha.o
String 1 : Good
String 2 : Morning
String 1 + String 2 : Good Morning

=== Code Execution Successful ===
```

5. Apply multiple inheritance concept using employee details and show their data.



```
main.cpp
1 #include <iostream>
2 using namespace std;
3 class Manager{
4     protected:
5     char* name;
6     int age;
7     int salary;
8     public:
9     Manager(){ }
10    Manager(char* n,int a,int s){
11        name = n;
12        age = a;
13        salary = s;
14    }
15 };
16 class Worker{
17     protected:
18     char* name;
19     int age;
20     int salary;
21     public:
22     Worker(){ }
23    Worker(char* n,int a,int s){
24        name = n;
25        age = a;
26        salary = s;
27    }
28 };
29 class Details : public Manager , public Worker{
30     public:
31     Details(char* n1, int a1, int s1, char* n2,
32             int a2, int s2) : Manager(n1,a1,s1),
33                             Worker(n2,a2,s2){ }
34    void display(){
35        cout << "Details of Manager : "<<endl<<
36             "Name : "<<Manager::name<<endl<< "Age : "
37             "<<Manager::age<<endl<<"Salary : "
38             "<<Manager::salary<<endl<<endl;
39        cout << "Details of Worker : " <<endl
40             "<<"Name : "<<Worker::name<<endl<<"Age : "
41             "<<Worker::age<<endl<<"Salary : "
42             "<<Worker::salary;
43    }
44 };
45 int main()
46 {
47     Details d("Suresh",50,100000,"Ramesh",30,50000);
48     d.display();
49 }
```

/tmp/16uiN4h4v7.o

Details of Manager :

Name : Suresh

Age : 50

Salary : 100000

Details of Worker :

Name : Ramesh

Age : 30

Salary : 50000

=== Code Execution Successful ===