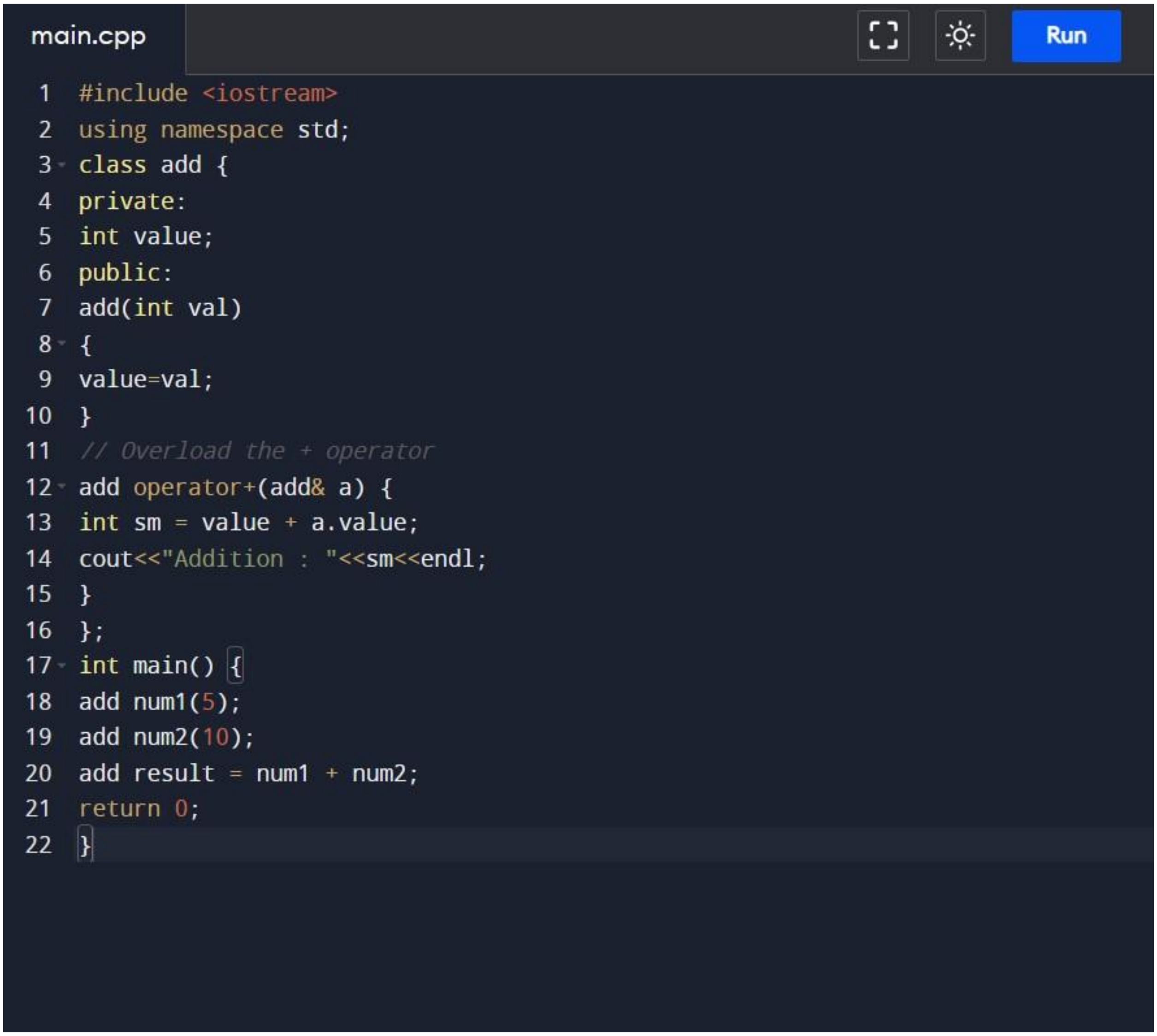
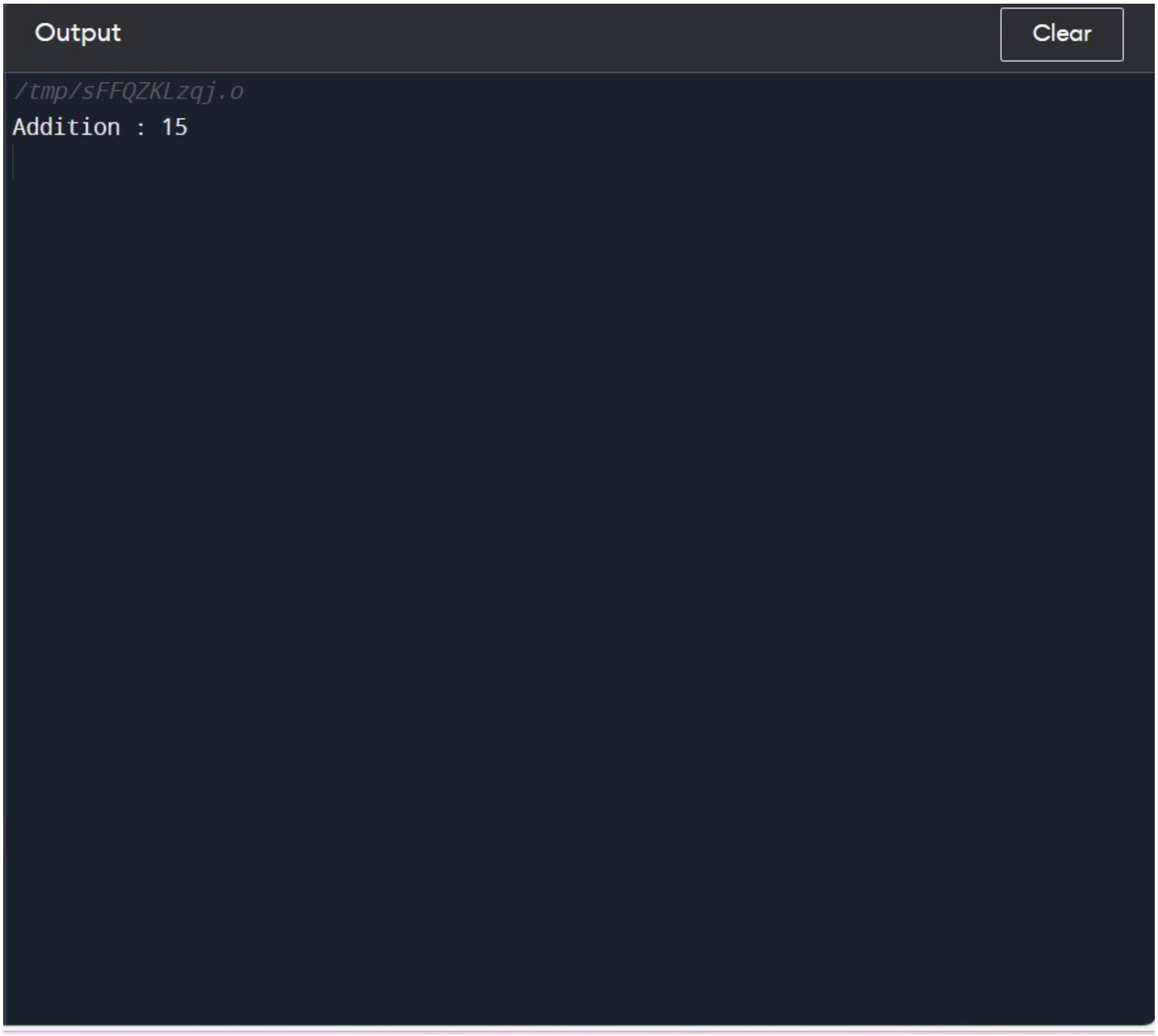
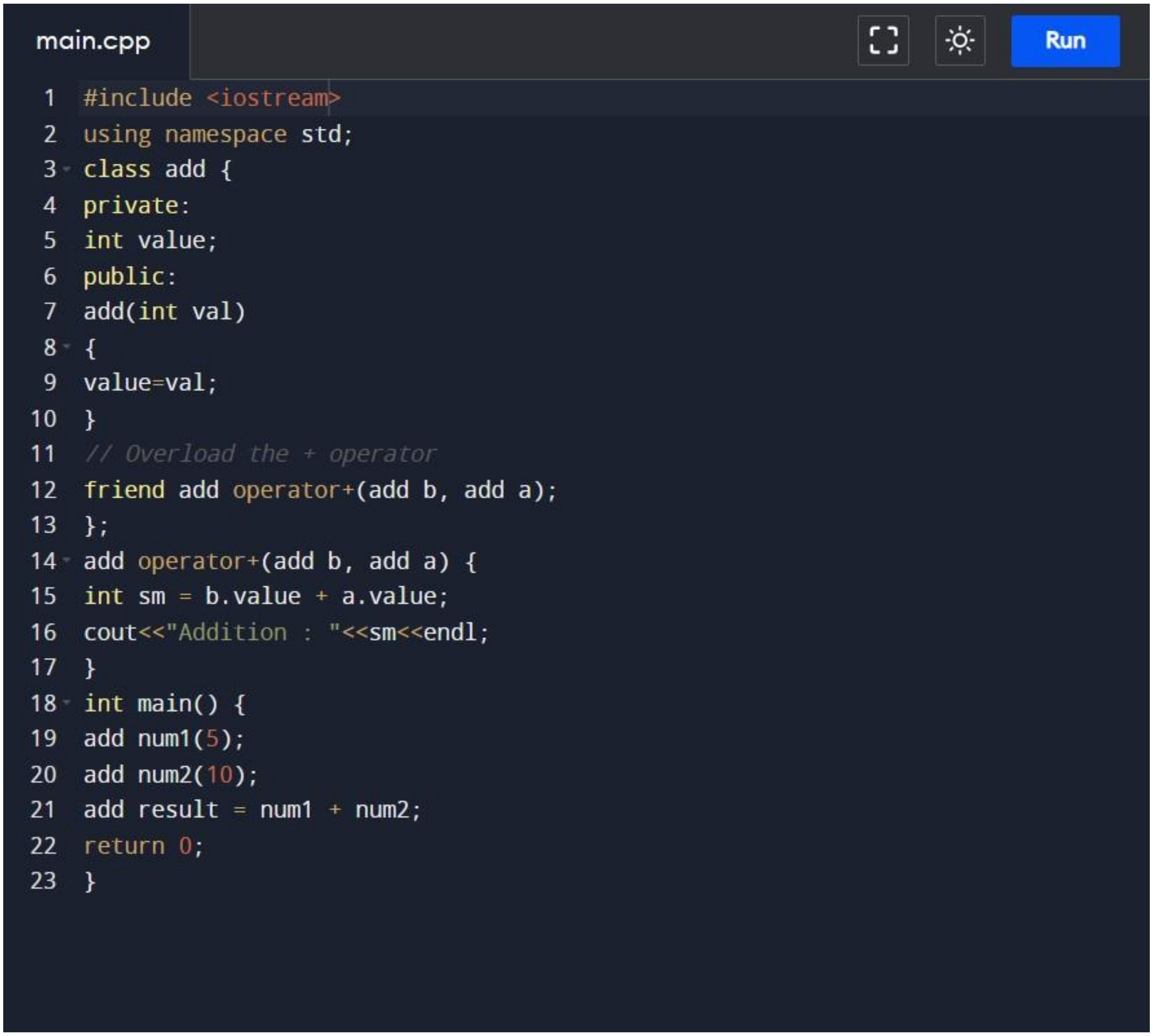


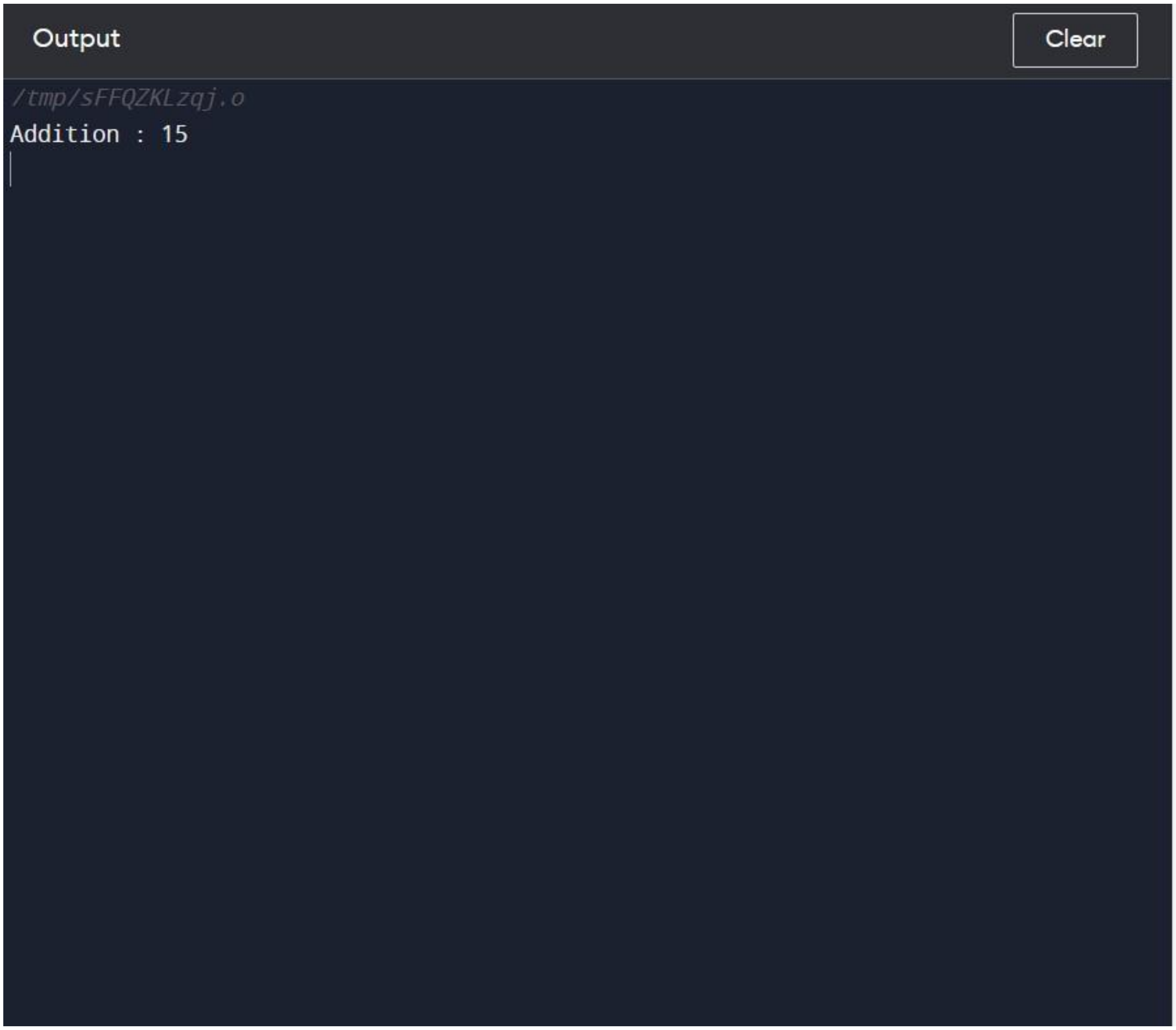
```cpp
#include <iostream>
using namespace std;
class add {
private:
int value;
public:
add(int val)
{
value=val;
}
// Overload the + operator
add operator+(add& a) {
int sm = value + a.value;
cout<<"Addition : "<<sm<<endl;
}
};
int main() {
add num1(5);
add num2(10);
add result = num1 + num2;
return 0;
}
```



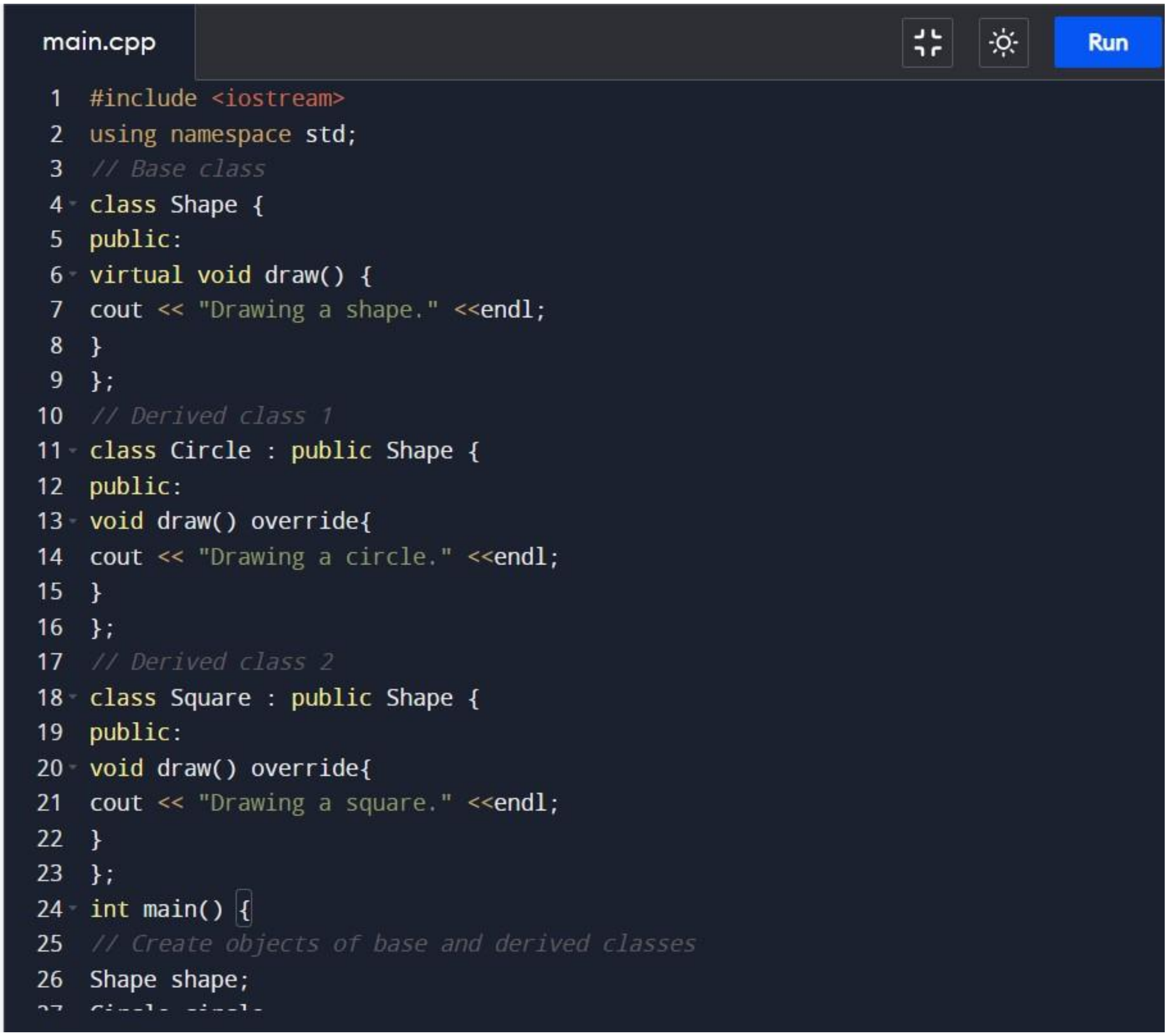
```
/tmp/sFFQZKLzqj.o
Addition : 15
```



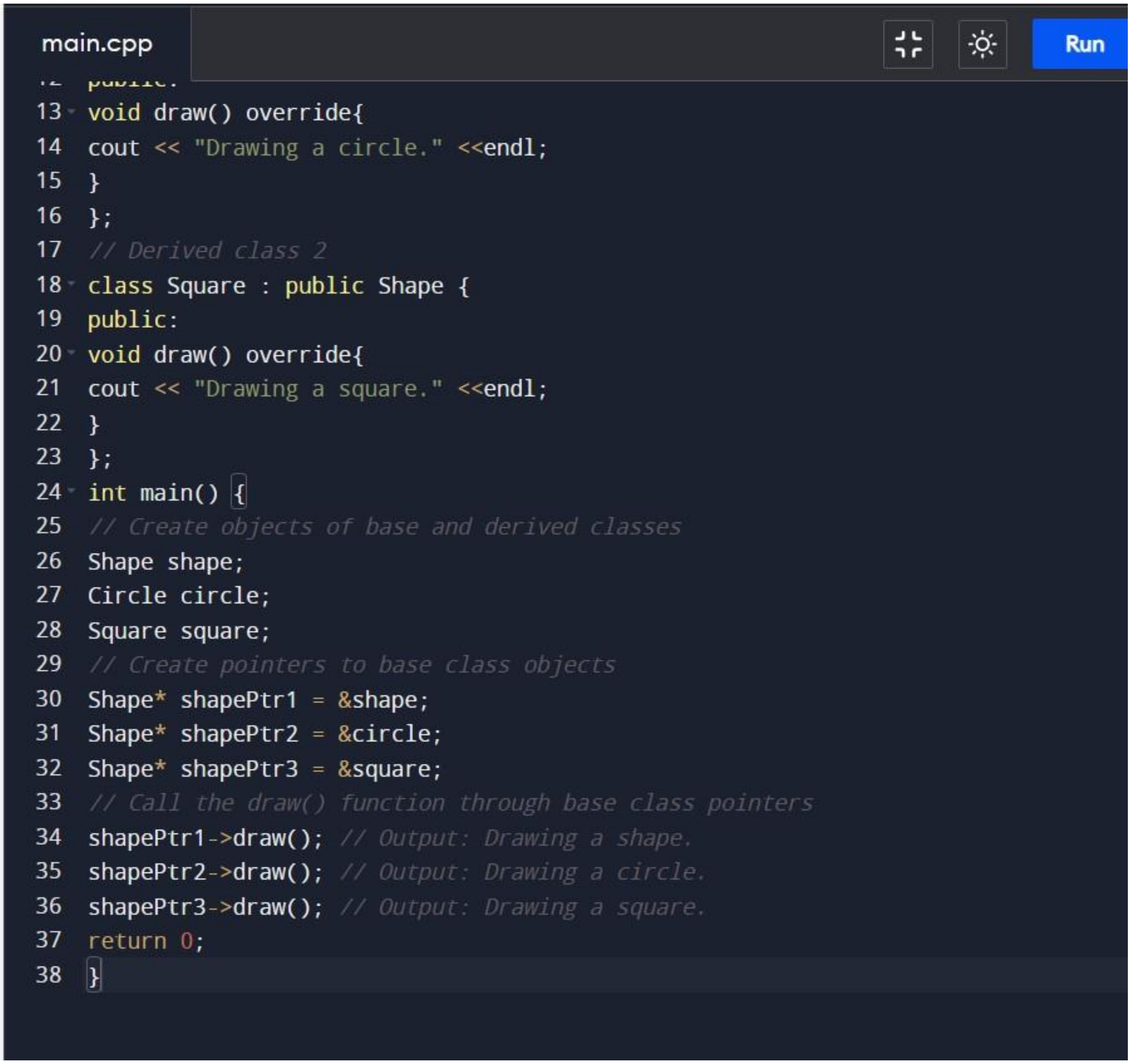
```cpp
#include <iostream>
using namespace std;
class add {
private:
int value;
public:
add(int val)
{
value=val;
}
// Overload the + operator
friend add operator+(add b, add a);
};
add operator+(add b, add a) {
int sm = b.value + a.value;
cout<<"Addition : "<<sm<<endl;
}
int main() {
add num1(5);
add num2(10);
add result = num1 + num2;
return 0;
}
```



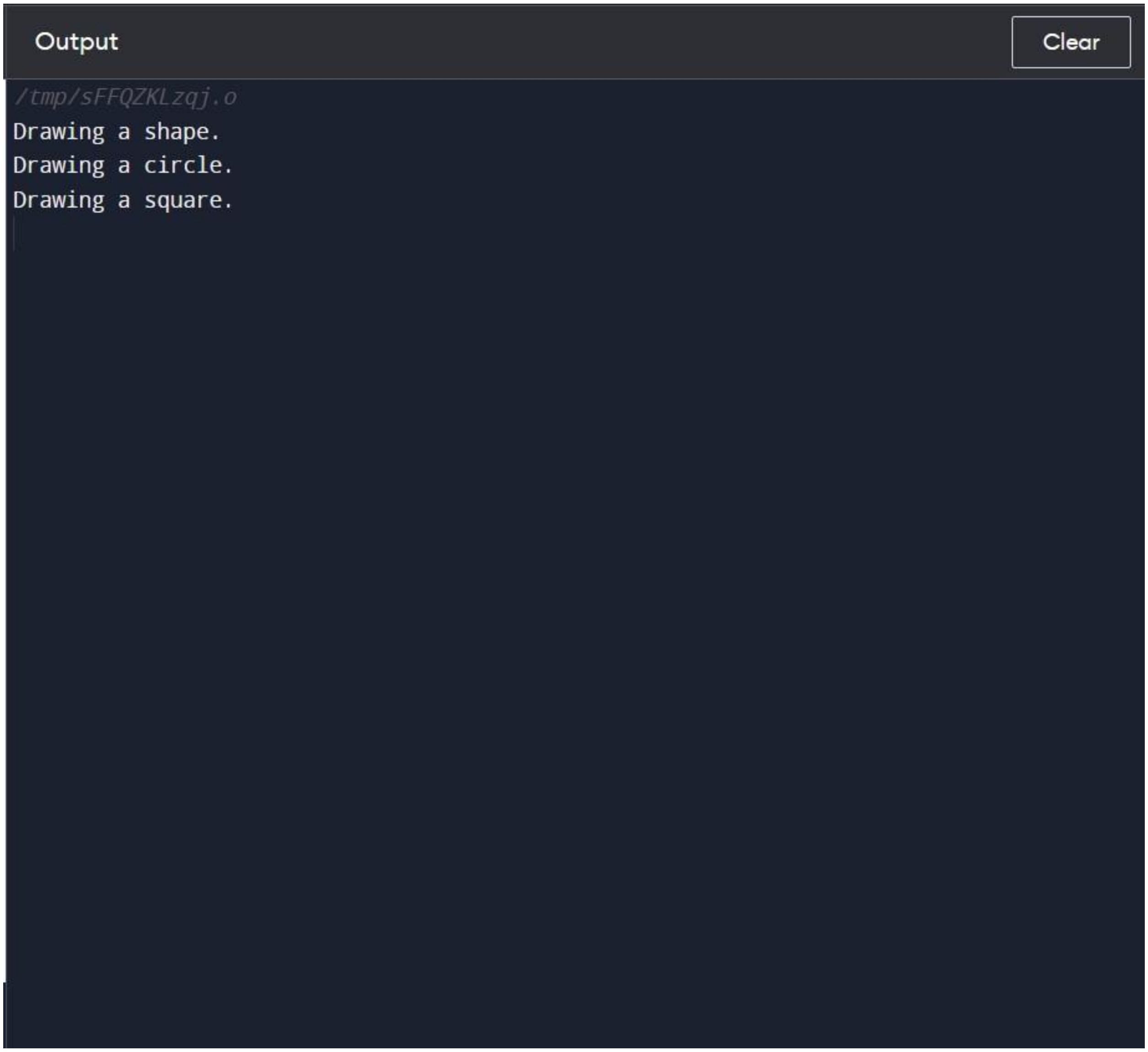
```
/tmp/sFFQZKLzqj.o
Addition : 15
```



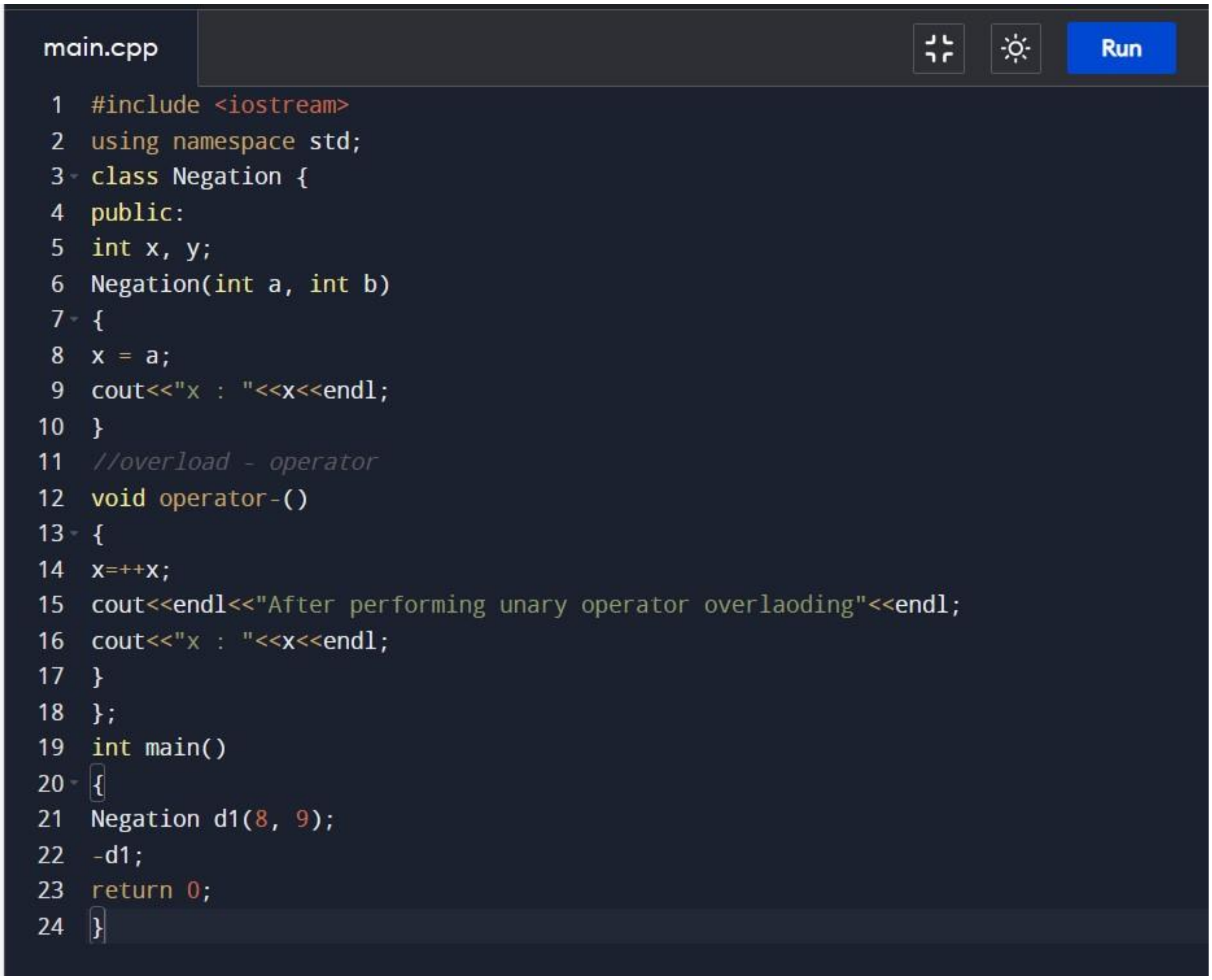
```cpp
#include <iostream>
using namespace std;
// Base class
class Shape {
public:
virtual void draw() {
cout << "Drawing a shape." <<endl;
}
};
// Derived class 1
class Circle : public Shape {
public:
void draw() override{
cout << "Drawing a circle." <<endl;
}
};
// Derived class 2
class Square : public Shape {
public:
void draw() override{
cout << "Drawing a square." <<endl;
}
};
int main() {
// Create objects of base and derived classes
Shape shape;
Circle circle
```



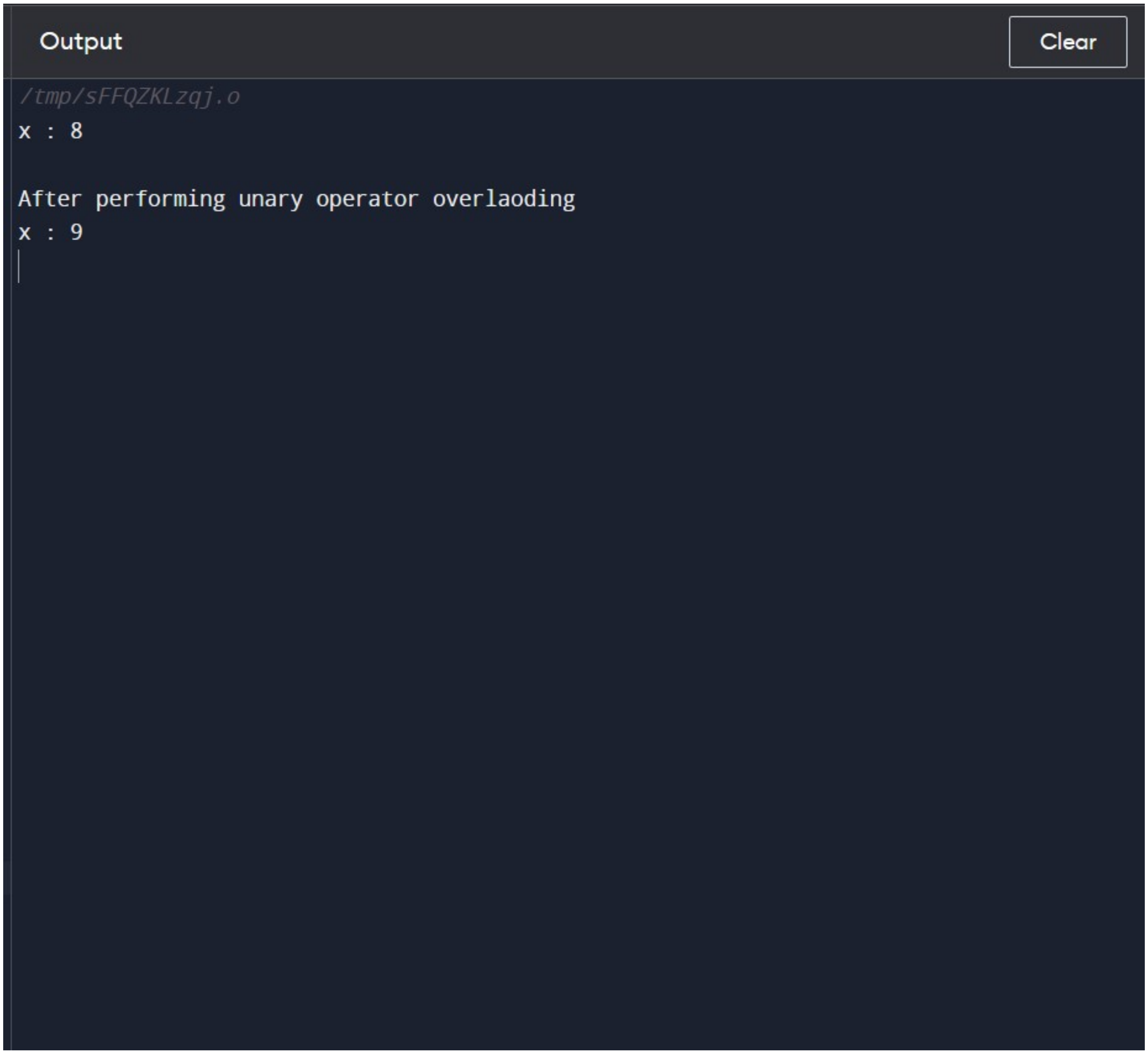
```
public:
void draw() override{
cout << "Drawing a circle." <<endl;
}
};
// Derived class 2
class Square : public Shape {
public:
void draw() override{
cout << "Drawing a square." <<endl;
}
};
int main() {
// Create objects of base and derived classes
Shape shape;
Circle circle;
Square square;
// Create pointers to base class objects
Shape* shapePtr1 = &shape;
Shape* shapePtr2 = &circle;
Shape* shapePtr3 = &square;
// Call the draw() function through base class pointers
shapePtr1->draw(); // Output: Drawing a shape.
shapePtr2->draw(); // Output: Drawing a circle.
shapePtr3->draw(); // Output: Drawing a square.
return 0;
}
```



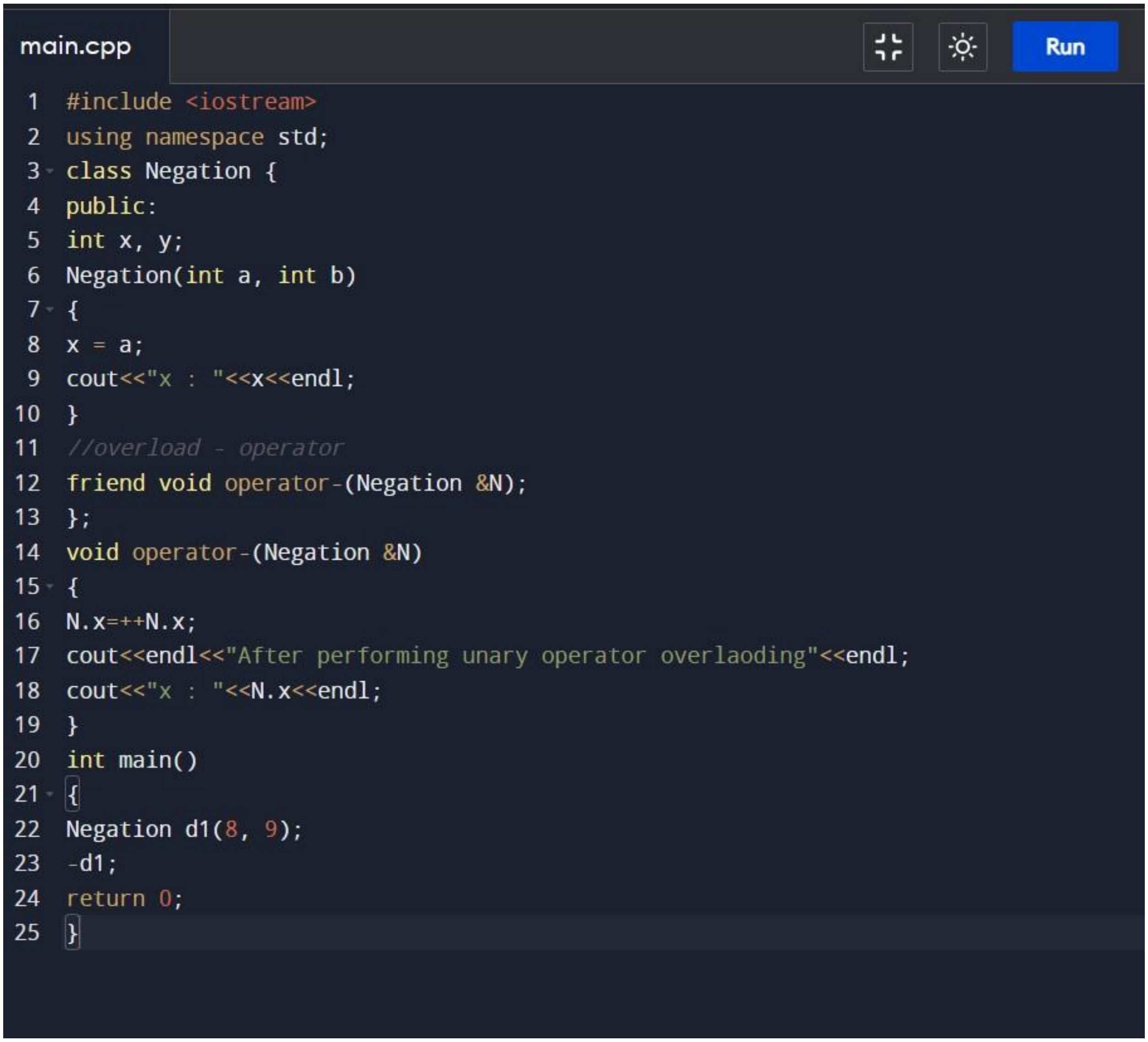
```
/tmp/sFFQZKLzqj.o
Drawing a shape.
Drawing a circle.
Drawing a square.
```



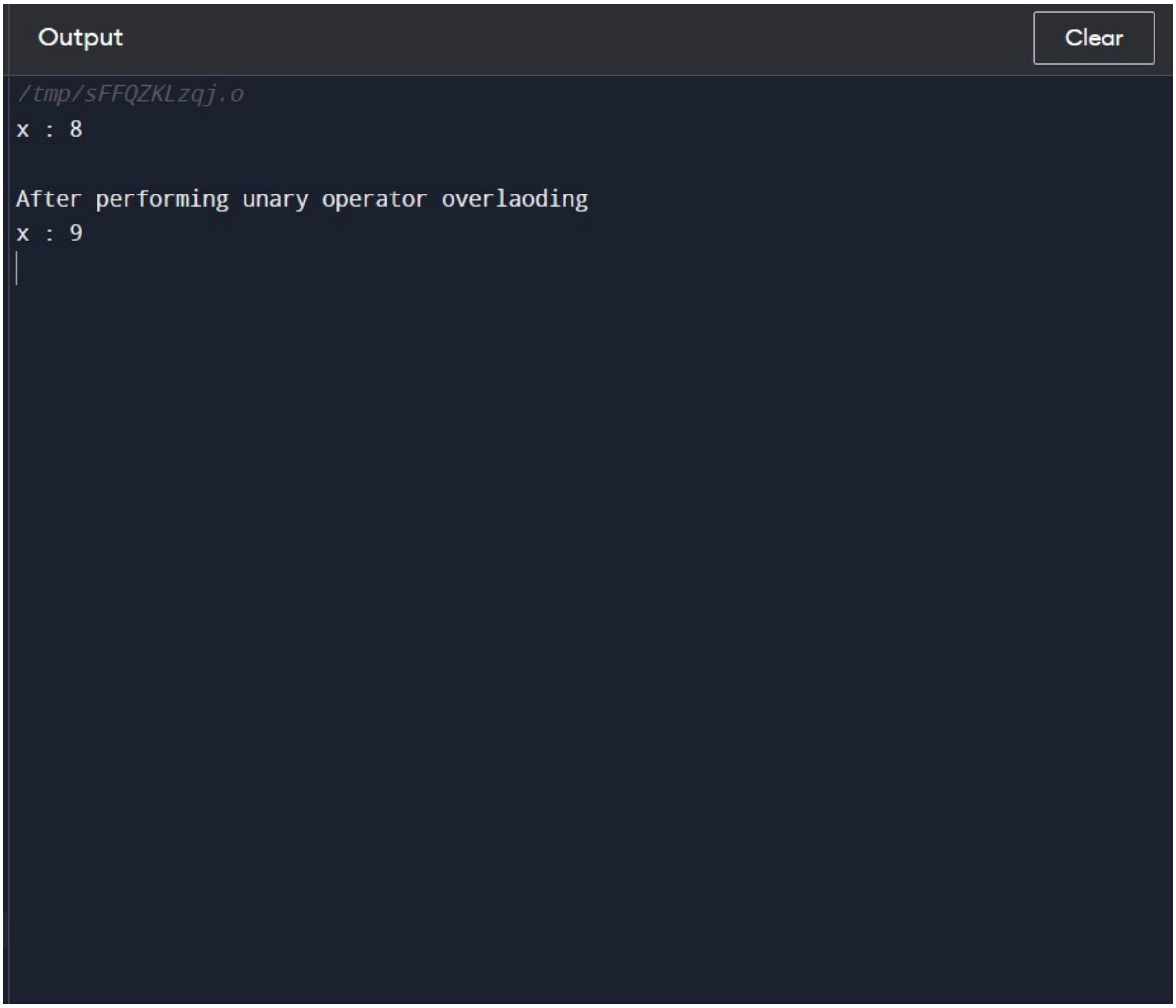
```cpp
#include <iostream>
using namespace std;
class Negation {
public:
int x, y;
Negation(int a, int b)
{
x = a;
cout<<"x : "<<x<<endl;
}
//overload - operator
void operator-()
{
x=++x;
cout<<endl<<"After performing unary operator overlaoding"<<endl;
cout<<"x : "<<x<<endl;
}
};
int main()
{
Negation d1(8, 9);
-d1;
return 0;
}
```



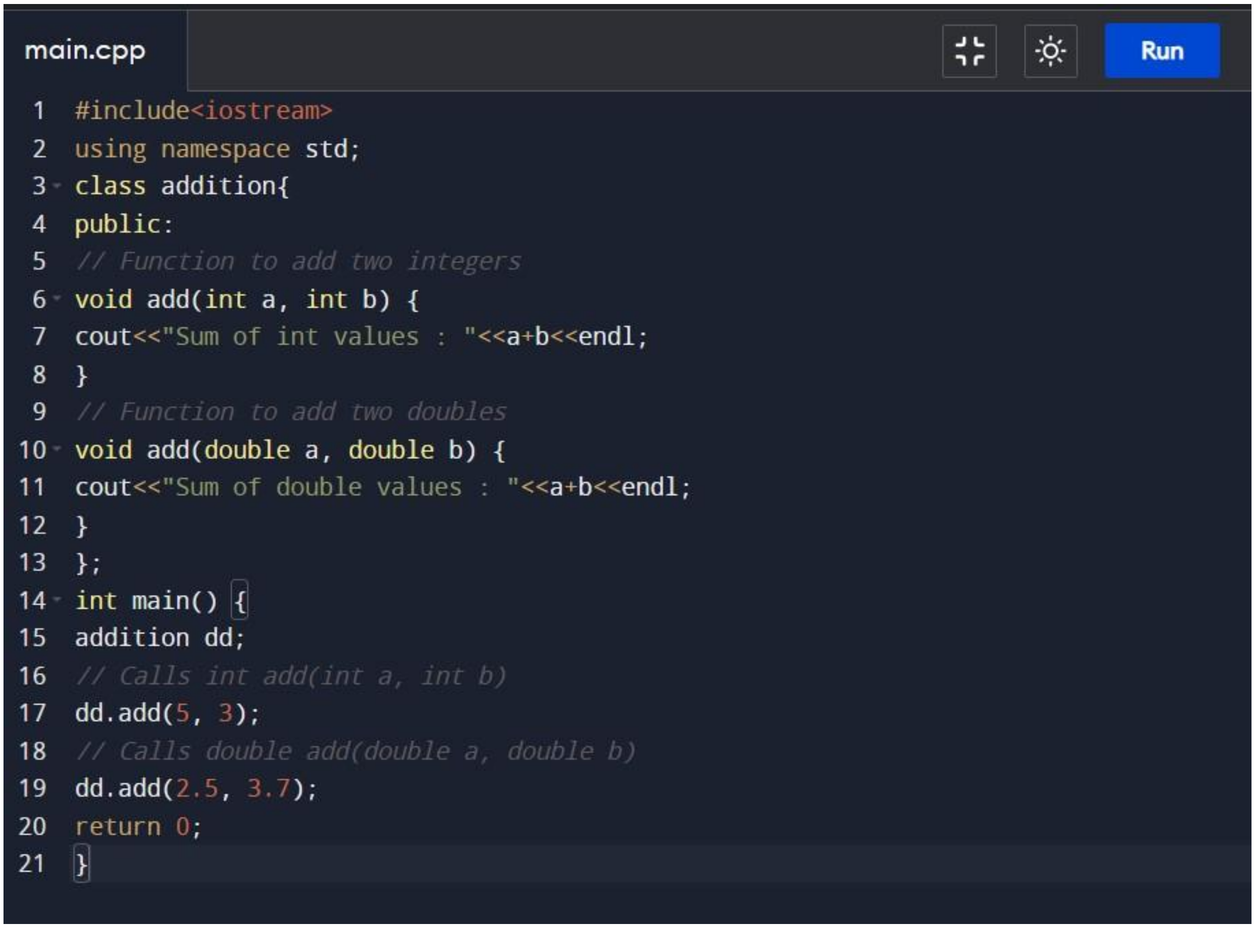
```
/tmp/sFFQZKLzqj.o
x : 8

After performing unary operator overlaoding
x : 9
```



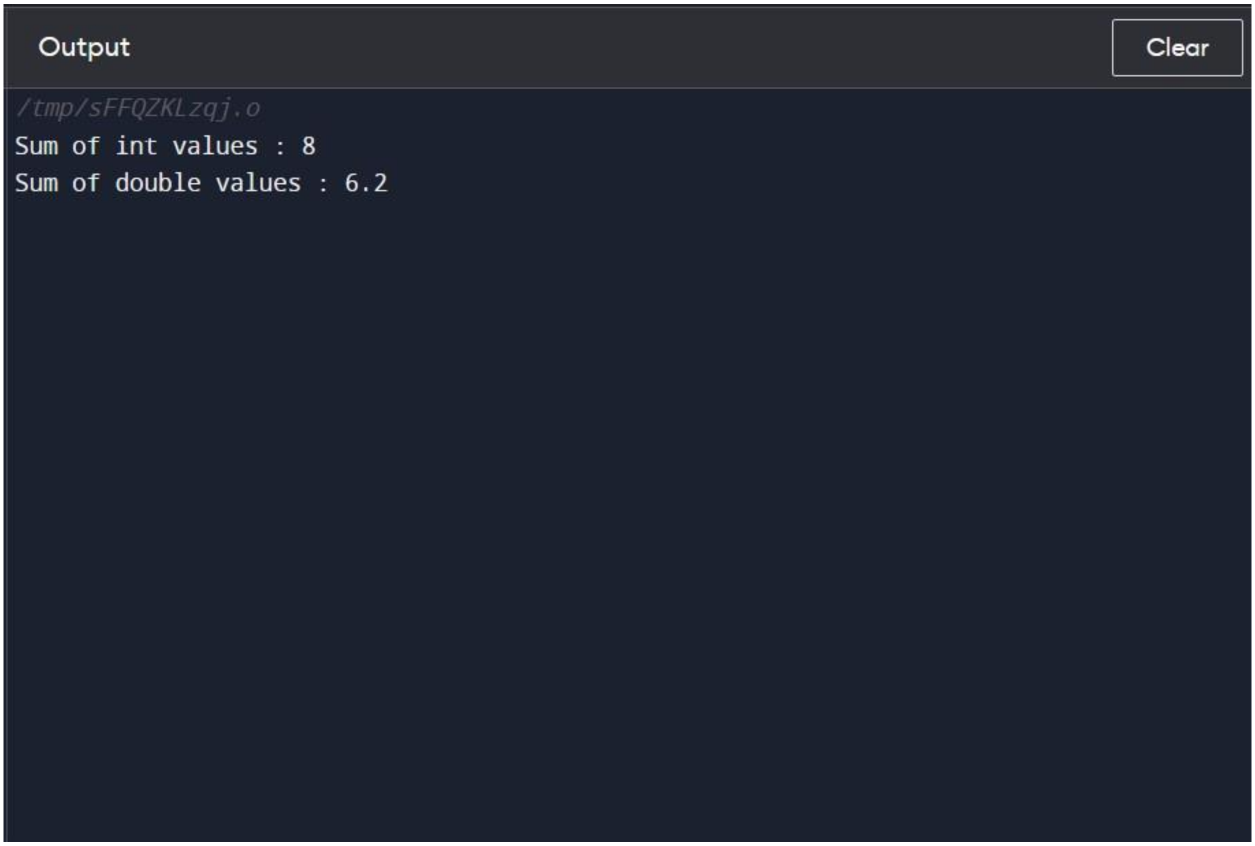
```cpp
#include <iostream>
using namespace std;
class Negation {
public:
int x, y;
Negation(int a, int b)
{
x = a;
cout<<"x : "<<x<<endl;
}
//overload - operator
friend void operator-(Negation &N);
};
void operator-(Negation &N)
{
N.x=++N.x;
cout<<endl<<"After performing unary operator overlaoding"<<endl;
cout<<"x : "<<N.x<<endl;
}
int main()
{
Negation d1(8, 9);
-d1;
return 0;
}
```



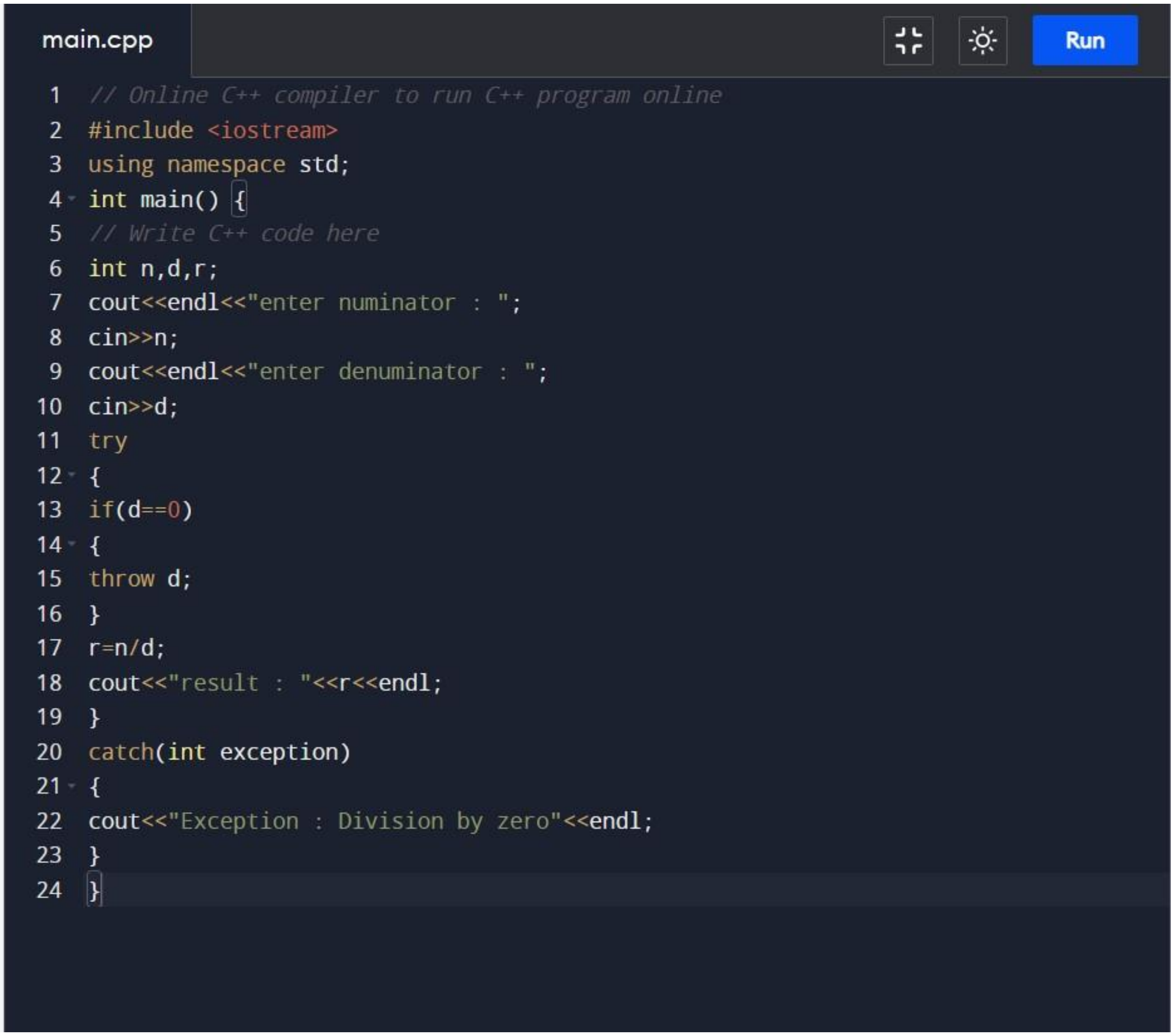
```
/tmp/sFFQZKLzqj.o
x : 8

After performing unary operator overlaoding
x : 9
```



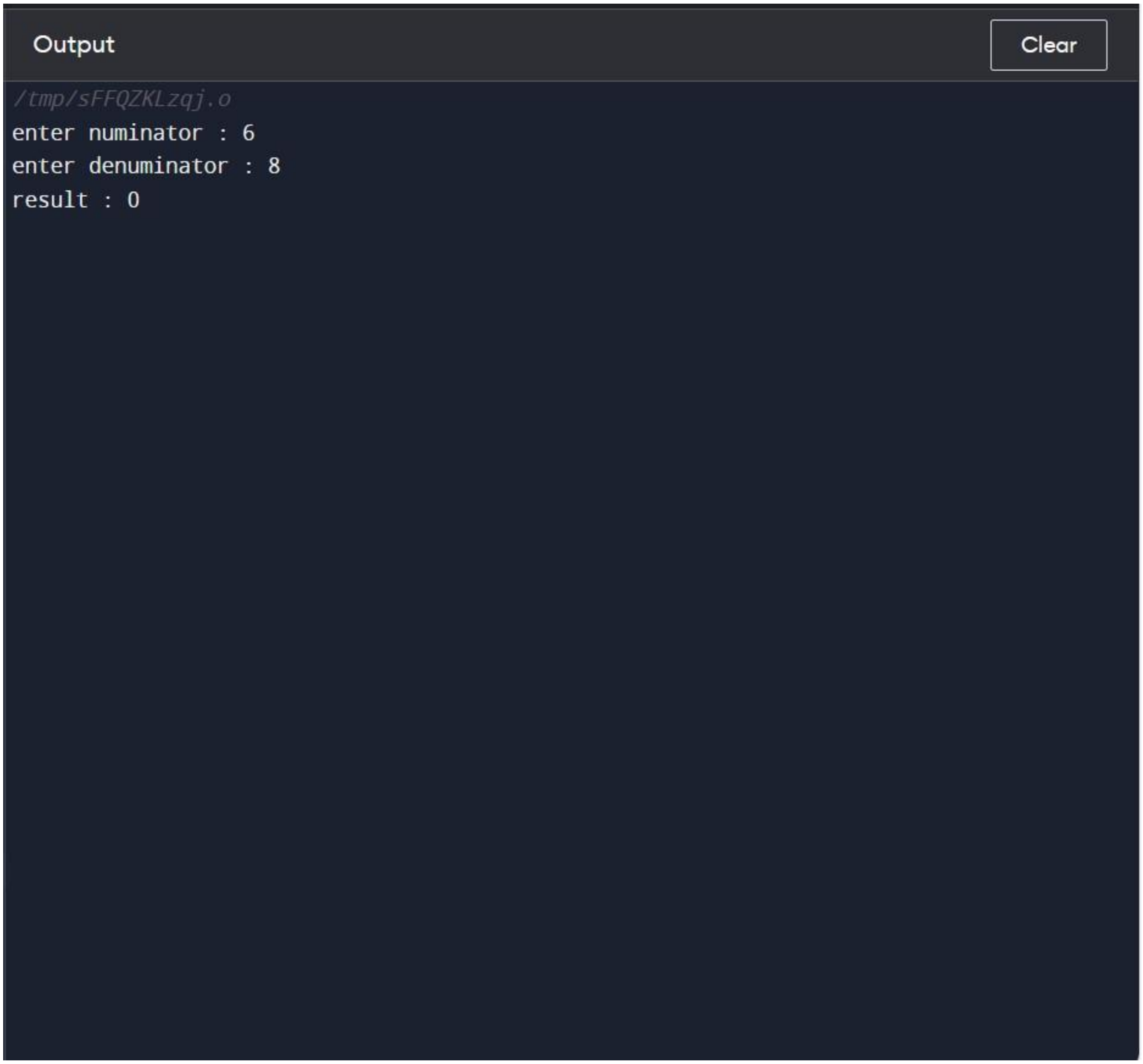
```cpp
#include<iostream>
using namespace std;
class addition{
public:
// Function to add two integers
void add(int a, int b) {
cout<<"Sum of int values : "<<a+b<<endl;
}
// Function to add two doubles
void add(double a, double b) {
cout<<"Sum of double values : "<<a+b<<endl;
}
};
int main() {
addition dd;
// Calls int add(int a, int b)
dd.add(5, 3);
// Calls double add(double a, double b)
dd.add(2.5, 3.7);
return 0;
}
```



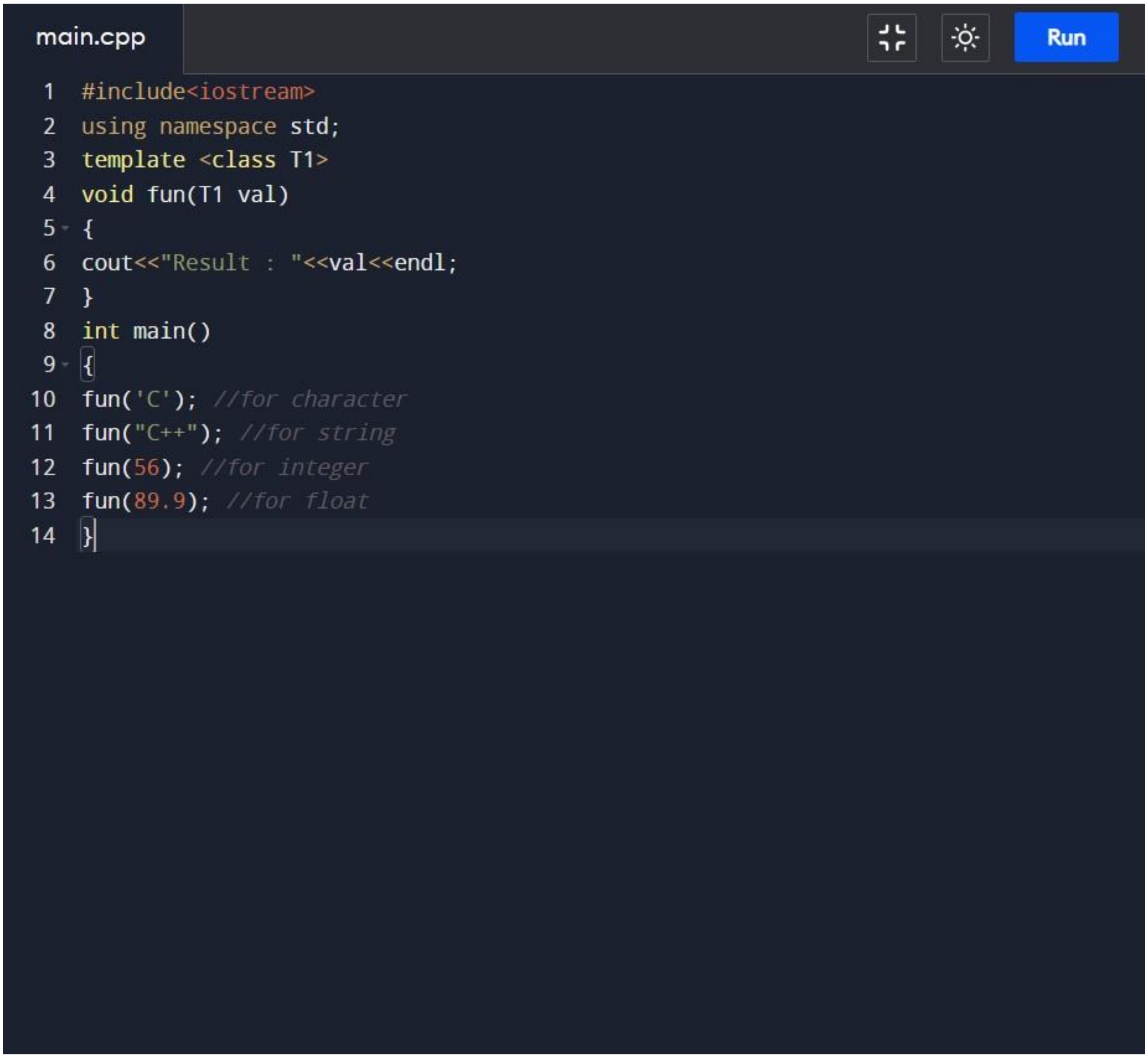
```
/tmp/sFFQZKLzqj.o
Sum of int values : 8
Sum of double values : 6.2
```



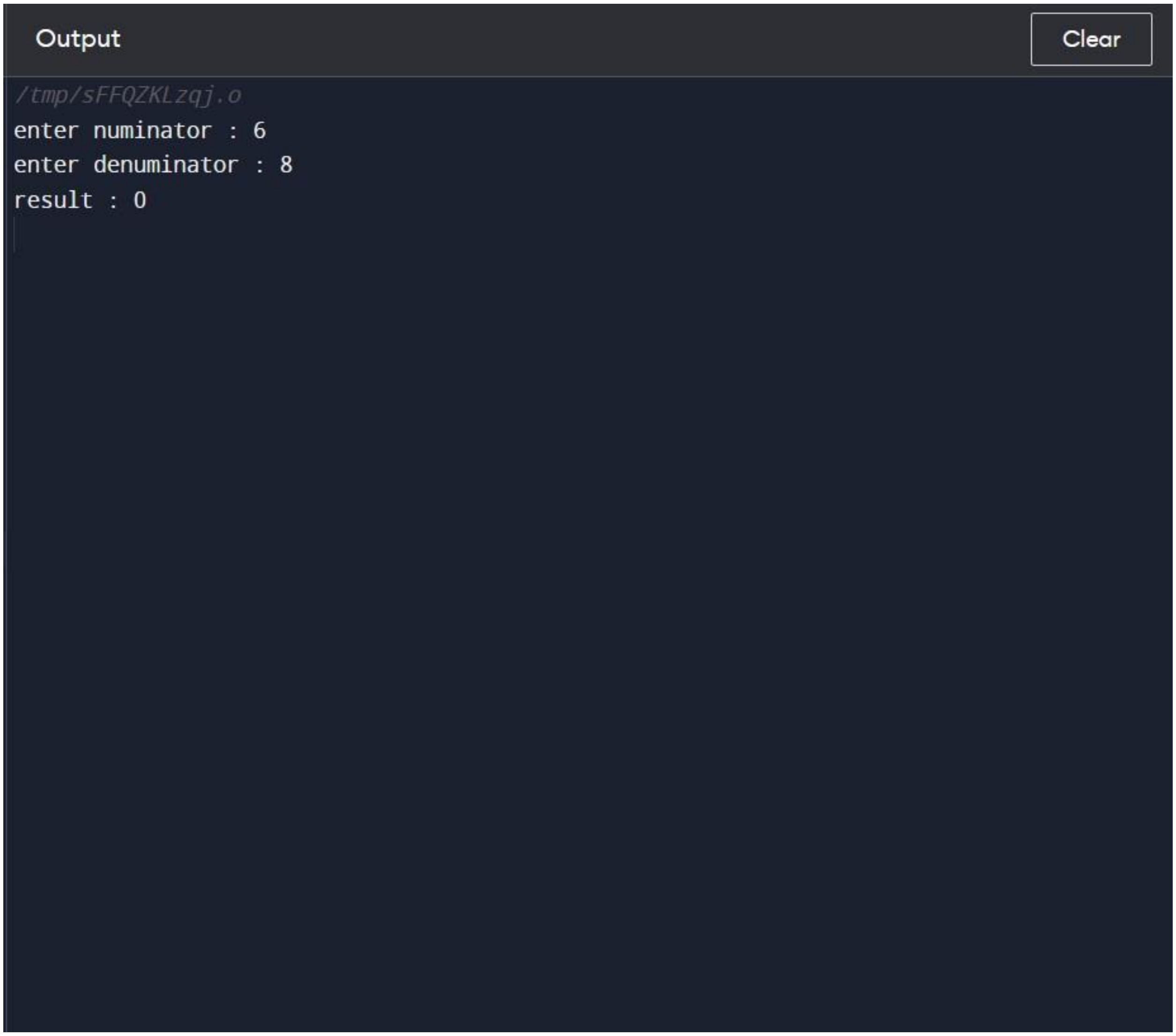
```cpp
// Online C++ compiler to run C++ program online
#include <iostream>
using namespace std;
int main() {
// Write C++ code here
int n,d,r;
cout<<endl<<"enter numinator : ";
cin>>n;
cout<<endl<<"enter denuminator : ";
cin>>d;
try
{
if(d==0)
{
throw d;
}
r=n/d;
cout<<"result : "<<r<<endl;
}
catch(int exception)
{
cout<<"Exception : Division by zero"<<endl;
}
}
```



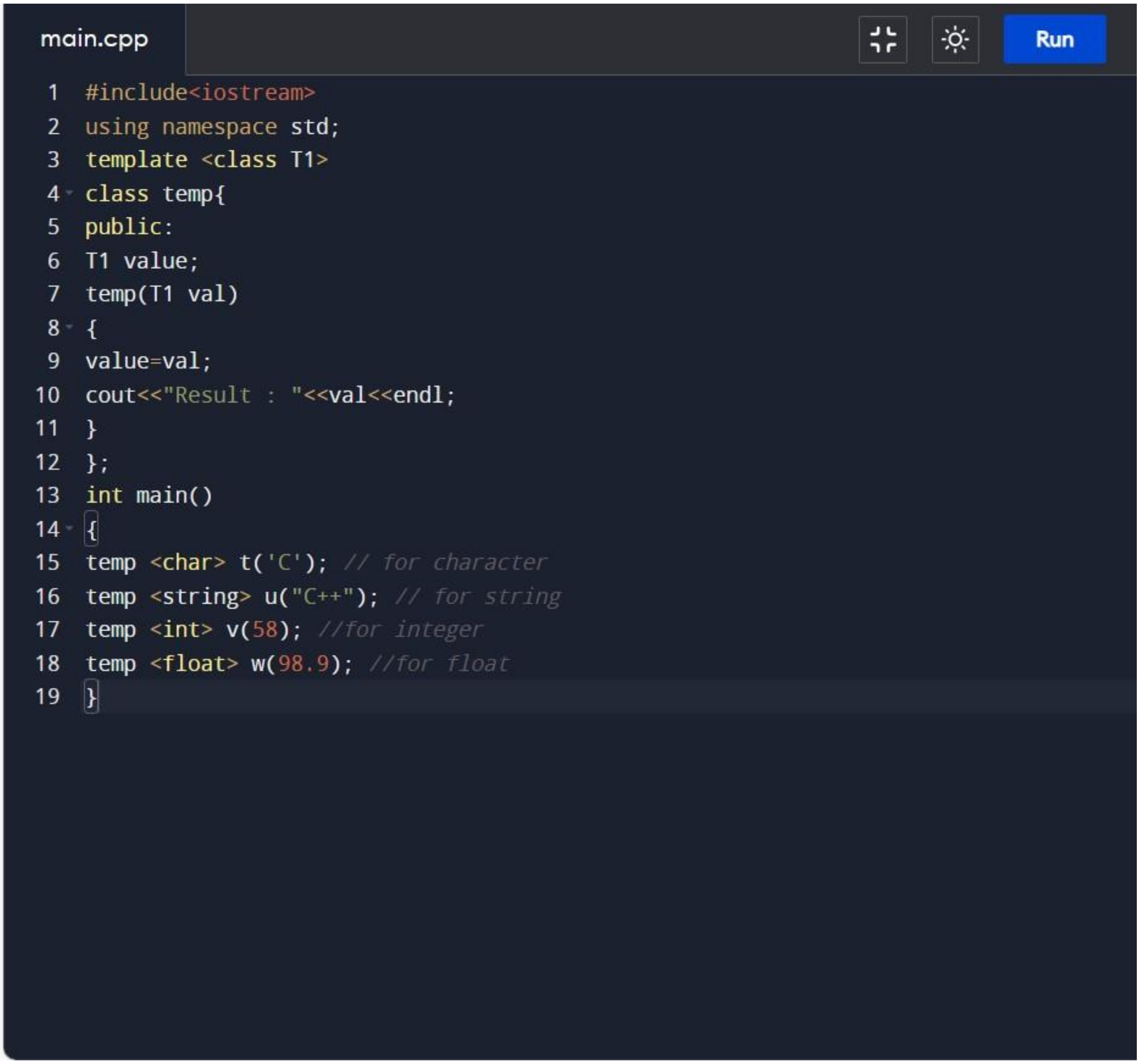
```
/tmp/sFFQZKLzqj.o
enter numinator : 6
enter denuminator : 8
result : 0
```



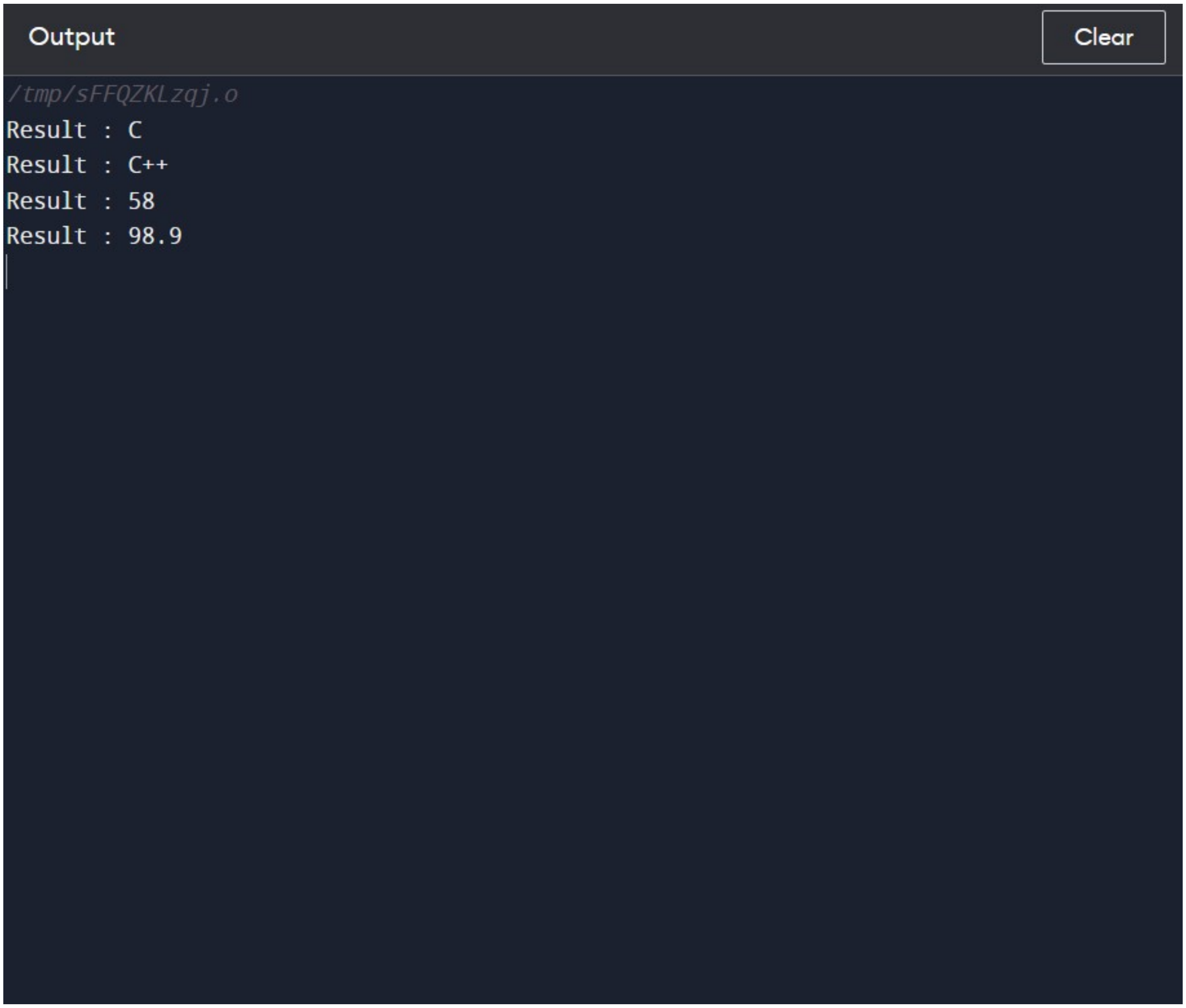
```cpp
#include<iostream>
using namespace std;
template <class T1>
void fun(T1 val)
{
cout<<"Result : "<<val<<endl;
}
int main()
{
fun('C'); //for character
fun("C++"); //for string
fun(56); //for integer
fun(89.9); //for float
}
```



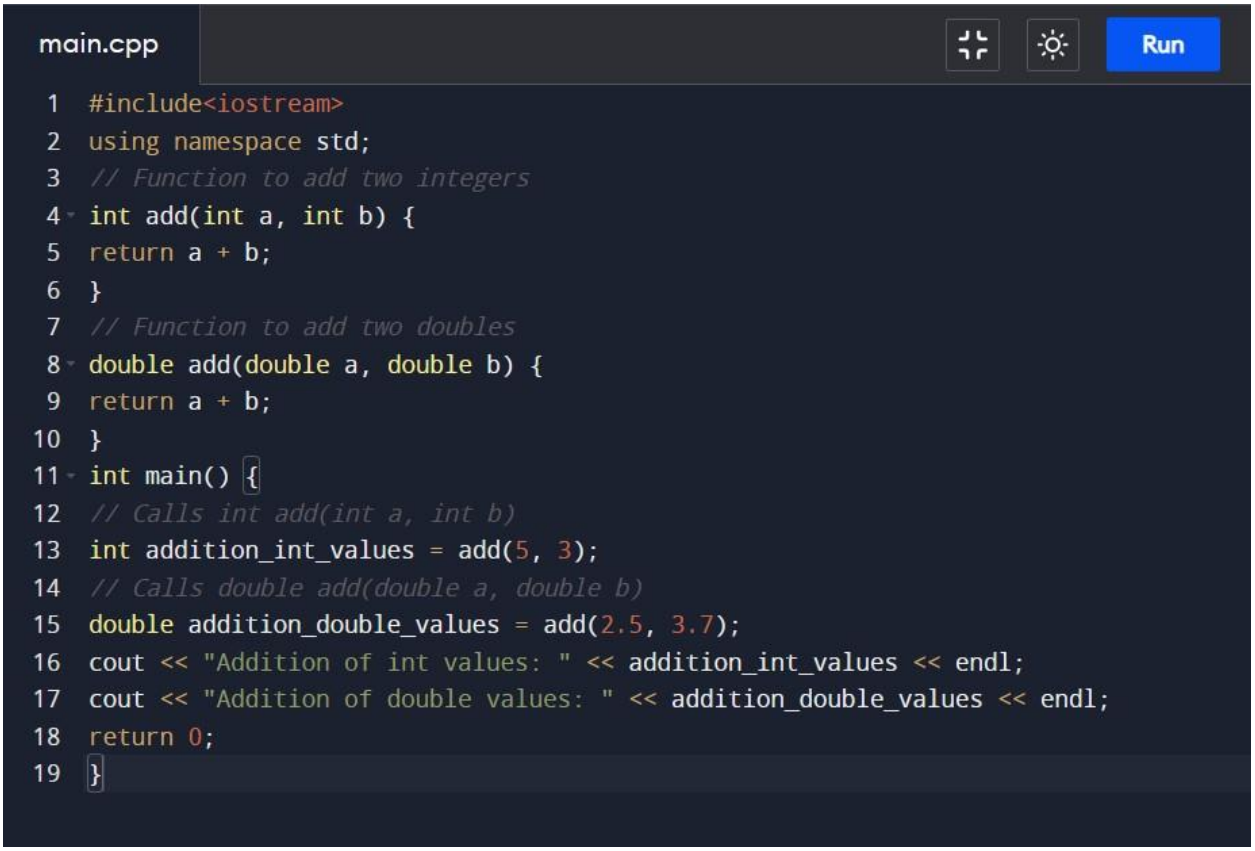
```
/tmp/sFFQZKLzqj.o
enter numinator : 6
enter denuminator : 8
result : 0
```



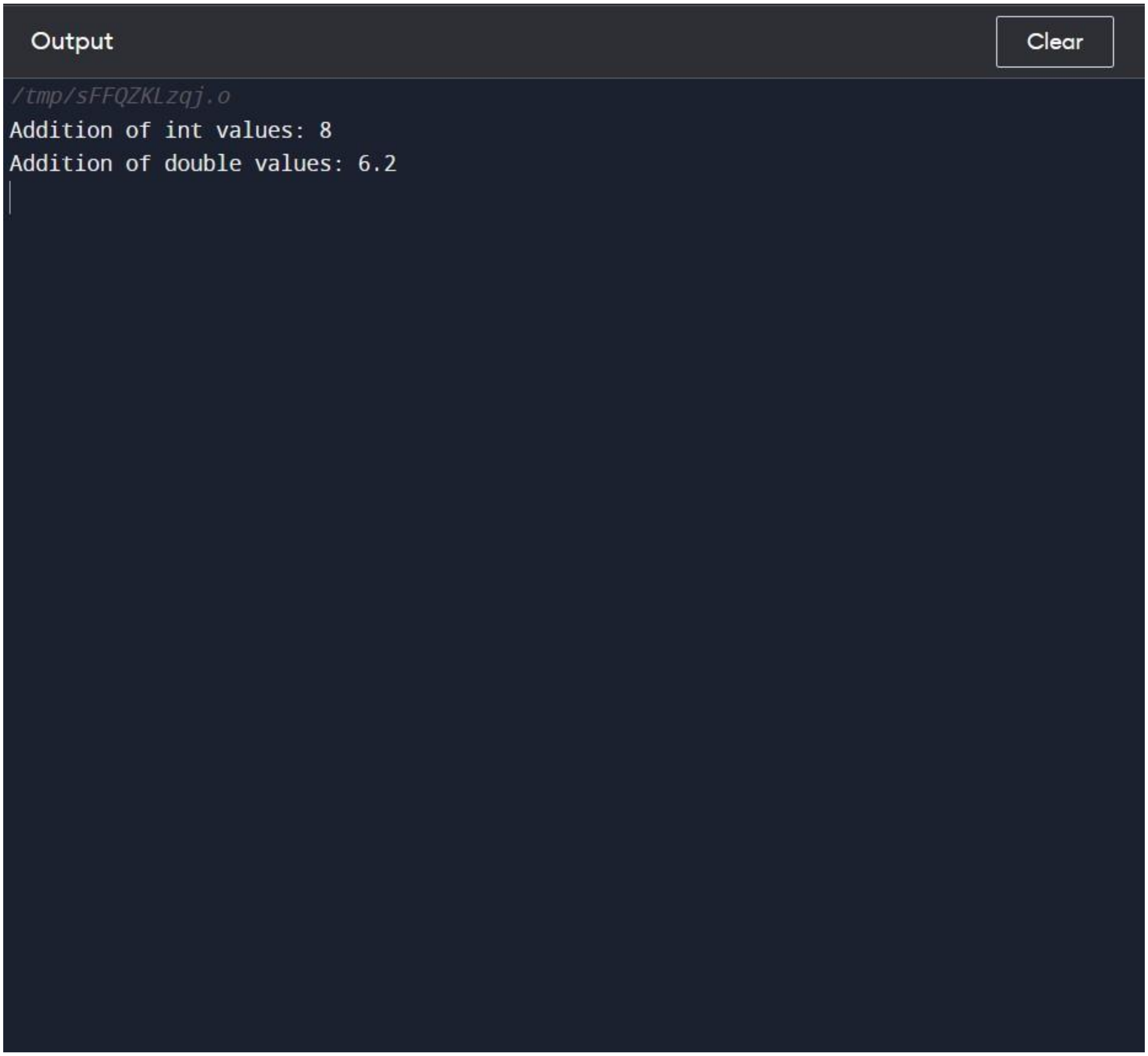
```cpp
#include<iostream>
using namespace std;
template <class T1>
class temp{
public:
T1 value;
temp(T1 val)
{
value=val;
cout<<"Result : "<<val<<endl;
}
};
int main()
{
temp <char> t('C'); // for character
temp <string> u("C++"); // for string
temp <int> v(58); //for integer
temp <float> w(98.9); //for float
}
```



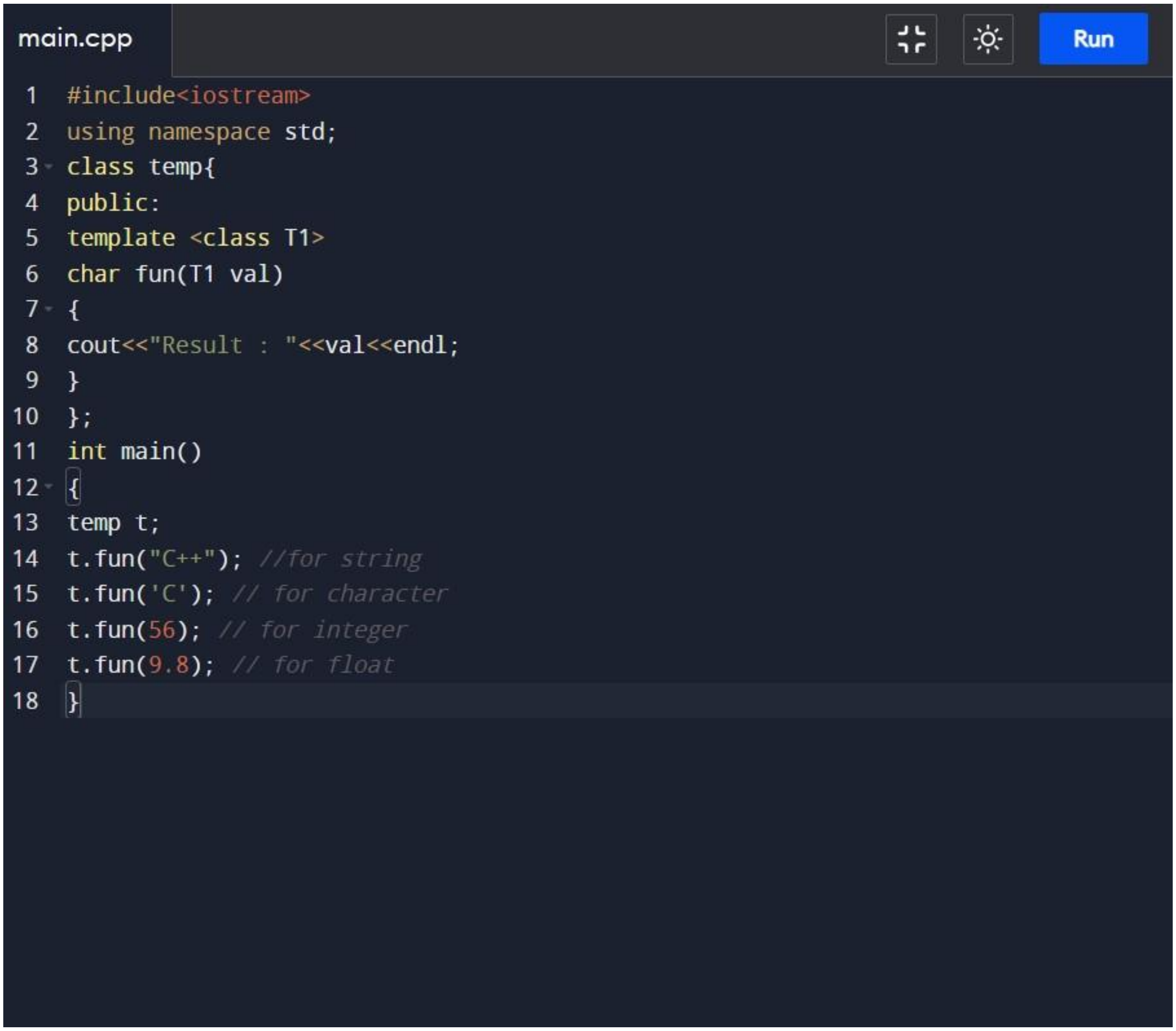
```
/tmp/sFFQZKLzqj.o
Result : C
Result : C++
Result : 58
Result : 98.9
```



```cpp
#include<iostream>
using namespace std;
// Function to add two integers
int add(int a, int b) {
return a + b;
}
// Function to add two doubles
double add(double a, double b) {
return a + b;
}
int main() {
// Calls int add(int a, int b)
int addition_int_values = add(5, 3);
// Calls double add(double a, double b)
double addition_double_values = add(2.5, 3.7);
cout << "Addition of int values: " << addition_int_values << endl;
cout << "Addition of double values: " << addition_double_values << endl;
return 0;
}
```

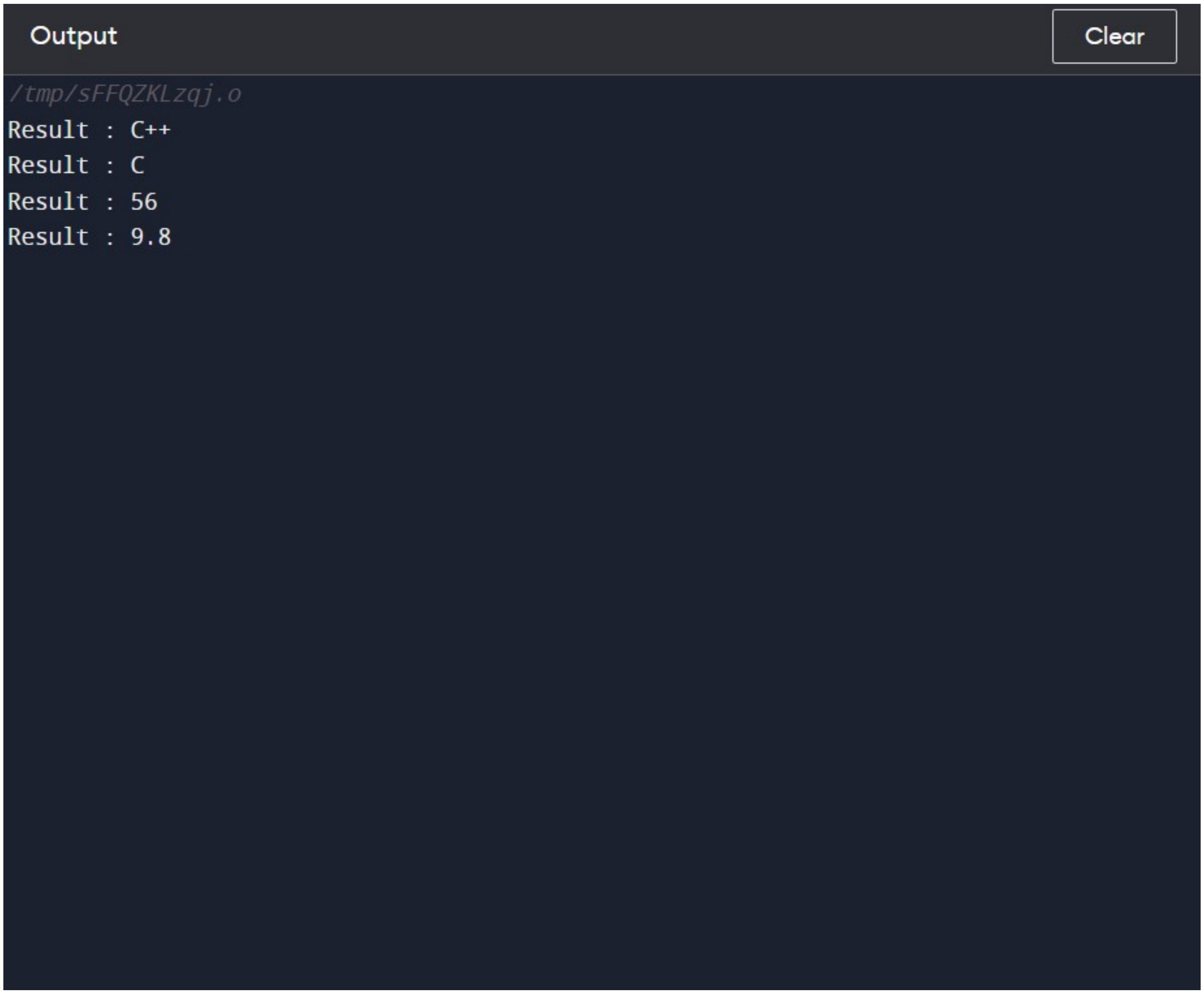


```
/tmp/sFFQZKLzqj.o
Addition of int values: 8
Addition of double values: 6.2
```



```cpp
#include<iostream>
using namespace std;
class temp{
public:
template <class T1>
char fun(T1 val)
{
cout<<"Result : "<<val<<endl;
}
};
int main()
{
temp t;
t.fun("C++"); //for string
t.fun('C'); // for character
t.fun(56); // for integer
t.fun(9.8); // for float
}
```

v



```
/tmp/sFFQZKLzqj.o
Result : C++
Result : C
Result : 56
Result : 9.8
```