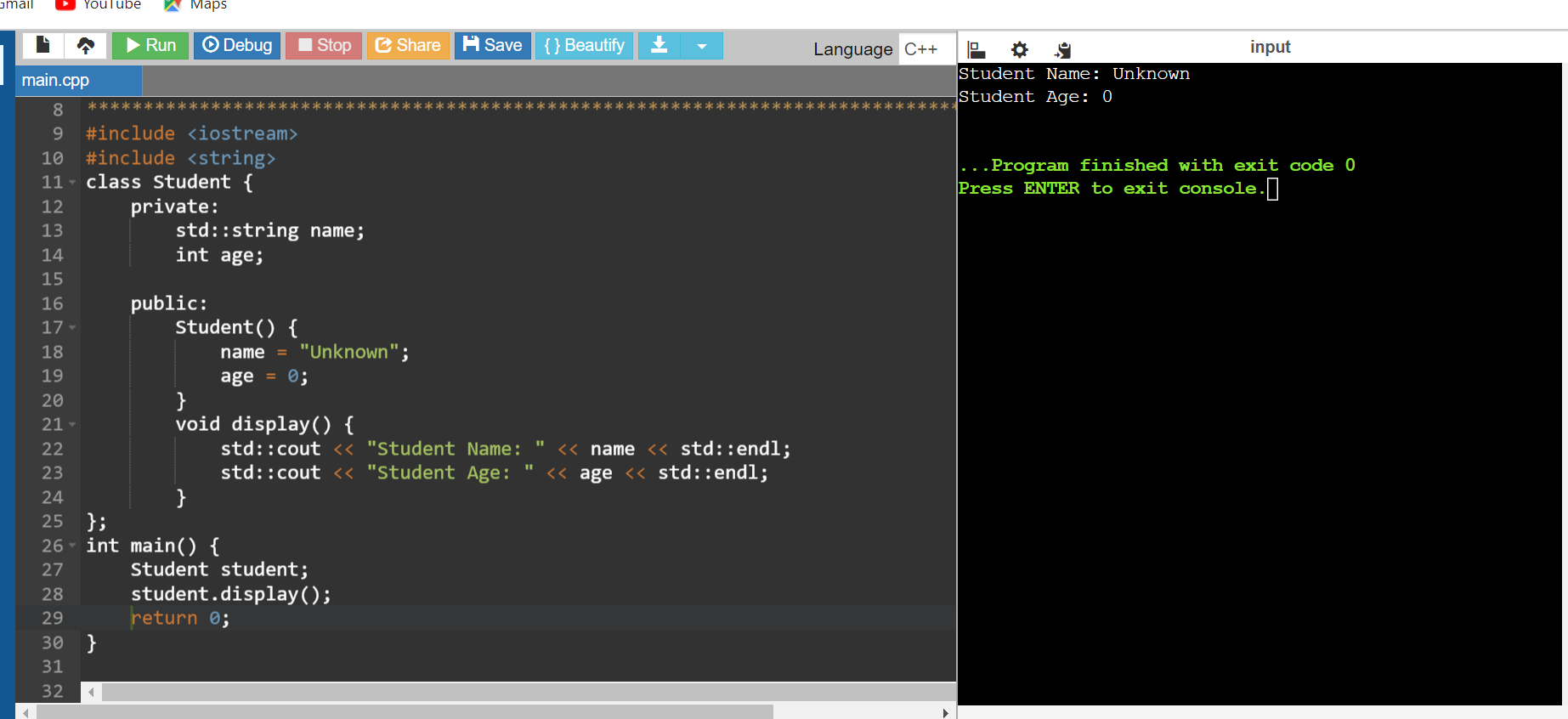
Q.1. Default Constructor:

Write a class Student with a default constructor that initializes the student's name to "Unknown" and age to 0. Add a method display to print the student's details.

Explanation :

The Student class is defined with two private member variables: name of type std::string and age of type int.

The default constructor Student() is defined to initialize name to "Unknown" and age to 0. A public method display is defined to print the student's details.



Q.2. Parameterized Constructor:

Write a class Rectangle with a parameterized constructor that initializes the length and width. Add a method area that returns the area of the rectangle.

The Rectangle class is defined with two private member variables: length and width of type double.

The parameterized constructor Rectangle(double l, double w) is defined to initialize length and width with the provided values.

A public method area is defined to calculate and return the area of the rectangle by multiplying length and width. The const qualifier ensures that this method does not modify any member variables.



Q.3 Write a class Book that has both a default constructor and a parameterized constructor. The default constructor should set the title to "Unknown" and the number of pages to 0. The parameterized constructor should initialize the title and pages with given values.

The Book class is defined with two private member variables: title of type std::string and pages of type int.The default constructor Book() initializes title to "Unknown" and pages to 0.The parameterized constructor Book(std::string t, int p) initializes title and pages with the provided values t and p respectively.A public method display is defined to print the book's details. The const qualifier ensures that this method does not modify any member variables.

#include <iostream>

#include <string>

class Book {

private:

std::string title;

int pages;

public:

Book() {

title = "Unknown";

pages = 0;

}

Book(std::string t, int p) {

title = t;

pages = p;

}

void display() const {

std::cout << "Title: " << title << std::endl;

std::cout << "Pages: " << pages << std::endl;

}

};

int main() {

Book defaultBook;

std::cout << "Default Book:" << std::endl;

defaultBook.display();

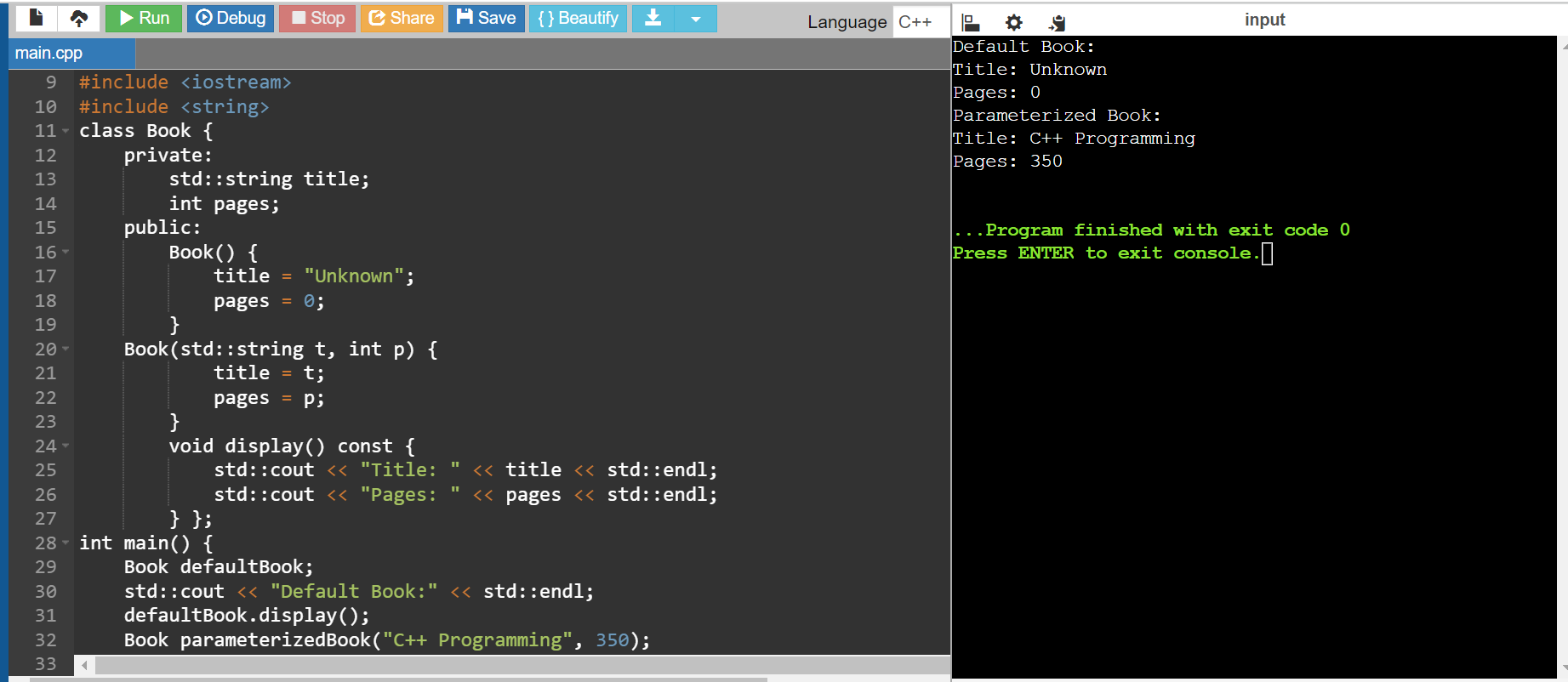
Book parameterizedBook("C++ Programming", 350);

std::cout << "Parameterized Book:" << std::endl;

parameterizedBook.display();

return 0;

}



Q.4 Write a class Complex that represents complex numbers. Implement a default constructor that sets both real and imaginary parts to 0, and a parameterized constructor that takes two arguments to initialize the real and imaginary parts.

The Complex class is defined with two private member variables: real and imaginary of type double.

The default constructor Complex() initializes real to 0.0 and imaginary to 0.0.

The parameterized constructor Complex(double r, double i) initializes real and imaginary with the provided values r and i respectively.

A public method display is defined to print the complex number in the form a + bi or a - bi, depending on the sign of the imaginary part. The const qualifier ensures that this method does not modify any member variables.

#include <iostream>

class Complex {

private:

double real;

double imaginary;

public:

Complex() {

real = 0.0;

imaginary = 0.0;

}

Complex(double r, double i) {

real = r;

imaginary = i;

}

void display() const {

std::cout << "Complex number: " << real;

if (imaginary >= 0) {

std::cout << " + " << imaginary << "i" << std::endl;

} else {

std::cout << " - " << -imaginary << "i" << std::endl;

}

}

};

int main() {

Complex defaultComplex;

std::cout << "Default Complex Number:" << std::endl;

defaultComplex.display();

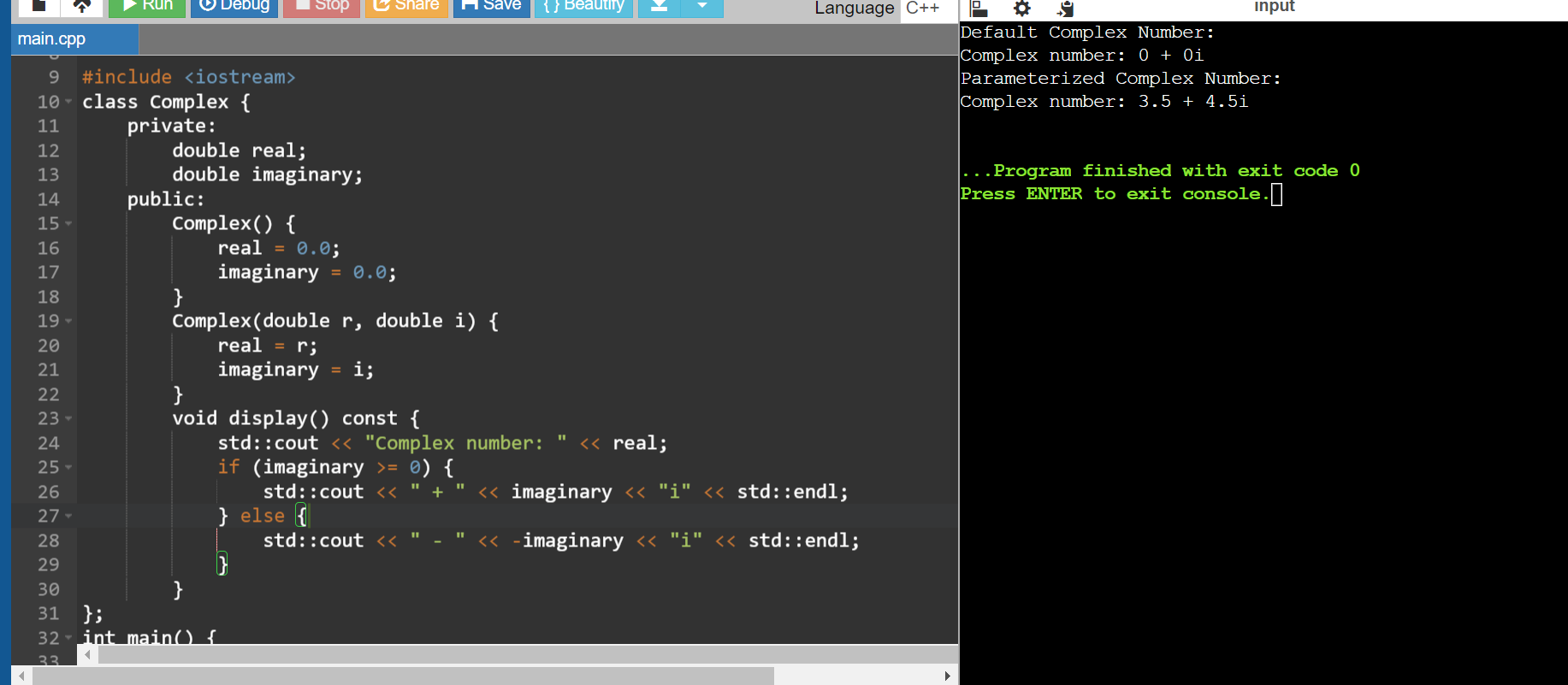
Complex parameterizedComplex(3.5, 4.5);

std::cout << "Parameterized Complex Number:" << std::endl;

parameterizedComplex.display();

return 0;

}



Q.5 Pointer to an Integer:\*

Write a function increment that takes a pointer to an integer and increments its value by 1 Demonstrate the function in the main program.



Q.6 Pointer to a Class:

Write a class Circle with a method area. Create a pointer to an object of this class and call the area method using the pointer

The Circle class is defined with a private member variable radius of type double.

A constructor Circle(double r) is provided to initialize the radius with the given value r.

A public method area is defined to calculate and return the area of the circle using the formula π \* radius^2. The const qualifier ensures that this method does not modify any member variables.

#include <iostream>

using namespace std;

class circle

{

private:

double radius;

public:

circle(double r)

{

radius = r;

}

double area()

{

return 3.14 \* radius \* radius; //formula for calculate area of circle

}

};

int main()

{

circle \*pcircle;

pcircle = new circle(5.0);

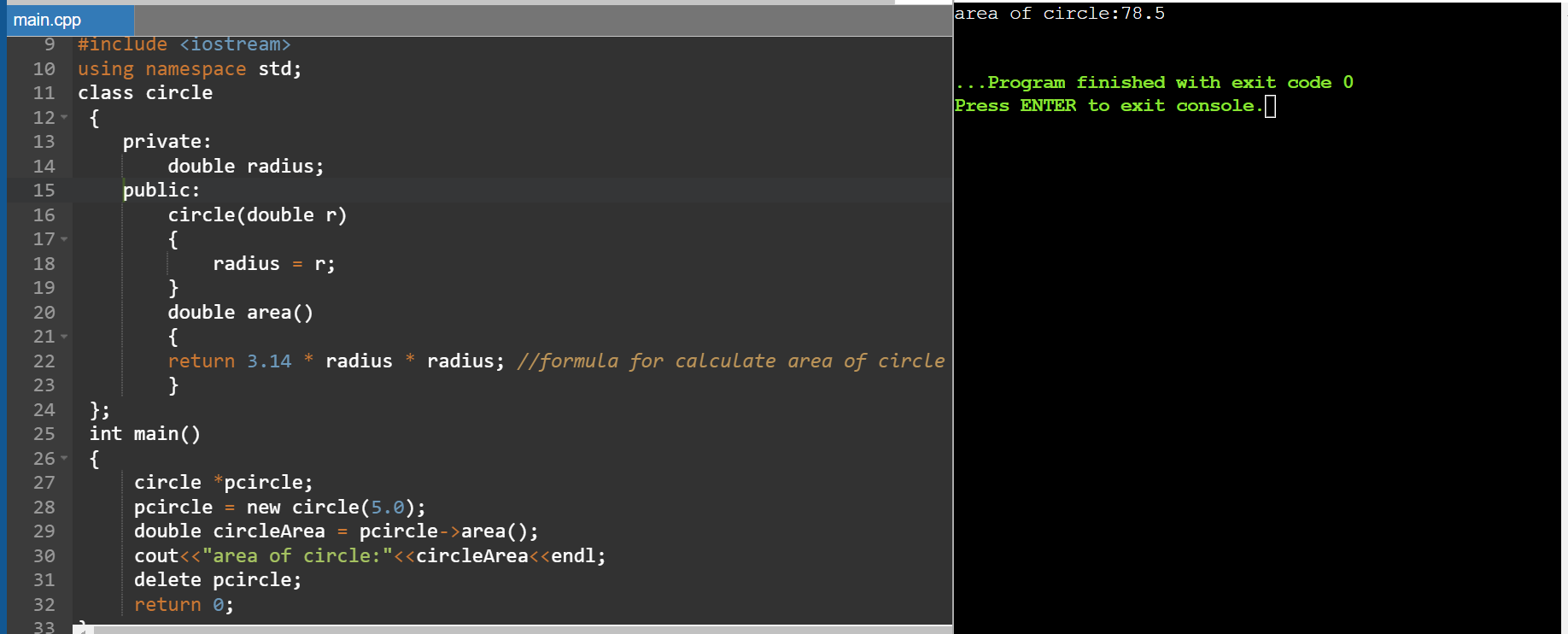
double circleArea = pcircle->area();

cout<<"area of circle:"<<circleArea<<endl;

delete pcircle;

return 0;

}



Q.7 Array of Pointers:\*

Write a program that creates an array of pointers to integers. Initialize the array with values and print them using the pointers.

#include <stdio.h>

int main() {

int size = 5;

int \*arr[size];

int values[] = {1, 2, 3, 4, 5};

for (int i = 0; i < size; i++) {

arr[i] = &values[i];

}

printf("Values in the array using pointers:\n");

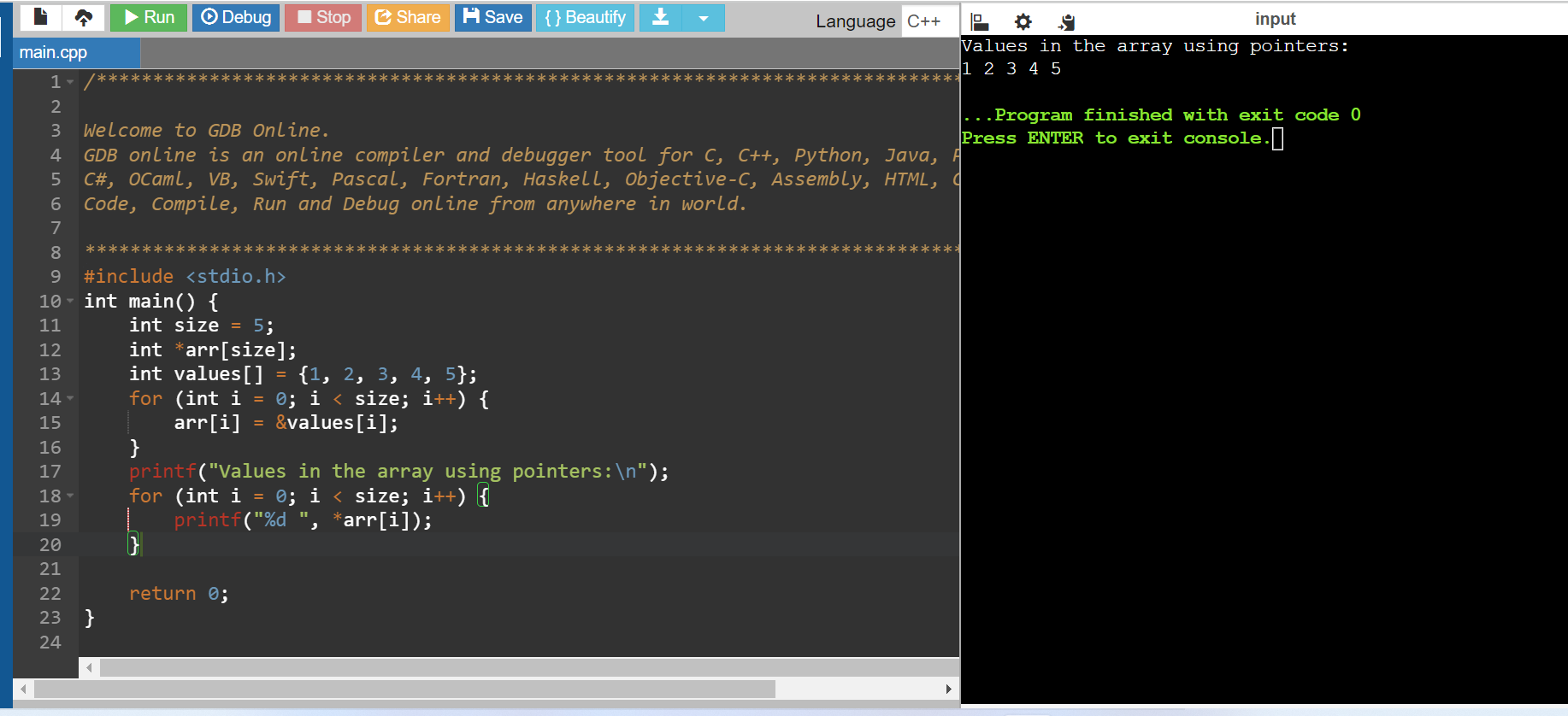
for (int i = 0; i < size; i++) {

printf("%d ", \*arr[i]);

}

return 0;

}



Q.8 Array of Pointers:

Write a program that creates an array of pointers to integers. Initialize the array with values and print them using the pointers.

#include <stdio.h>

void printArray(int \*arr, int size) {

for (int i = 0; i < size; i++) {

printf("%d ", arr[i]);

}

printf("\n");

}

int main() {

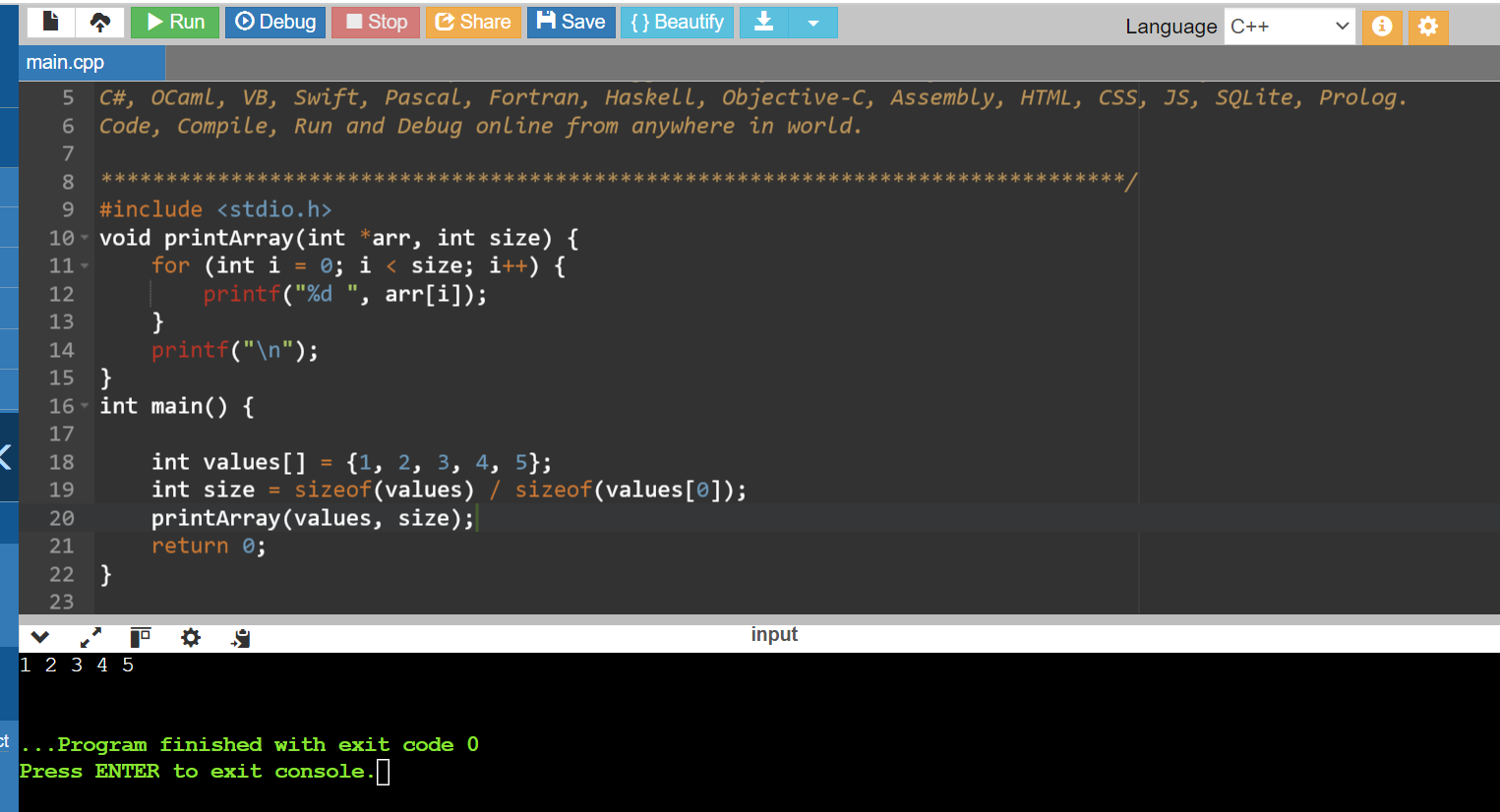
int values[] = {1, 2, 3, 4, 5};

int size = sizeof(values) / sizeof(values[0]);

printArray(values, size);

return 0;

}



Q.9 17. \*Pass by Value:\*

Write a function addTen that takes an integer by value and adds 10 to it. Demonstrate how the original value is not changed after calling the function.

The main function initializes an integer variable value with a value of 5.

It prints the original value of value before calling the addTen function.

The addTen function is called, passing value by value. This means that a copy of value is passed to the function.

Inside the addTen function, the copied value is modified and printed, but this does not affect the original value in main.

After the function call, the original value is printed again to demonstrate that it has not changed.

#include <stdio.h>

void addTen(int num) {

num += 10;

printf("Value inside addTen function: %d\n", num);

}

int main() {

int value = 5;

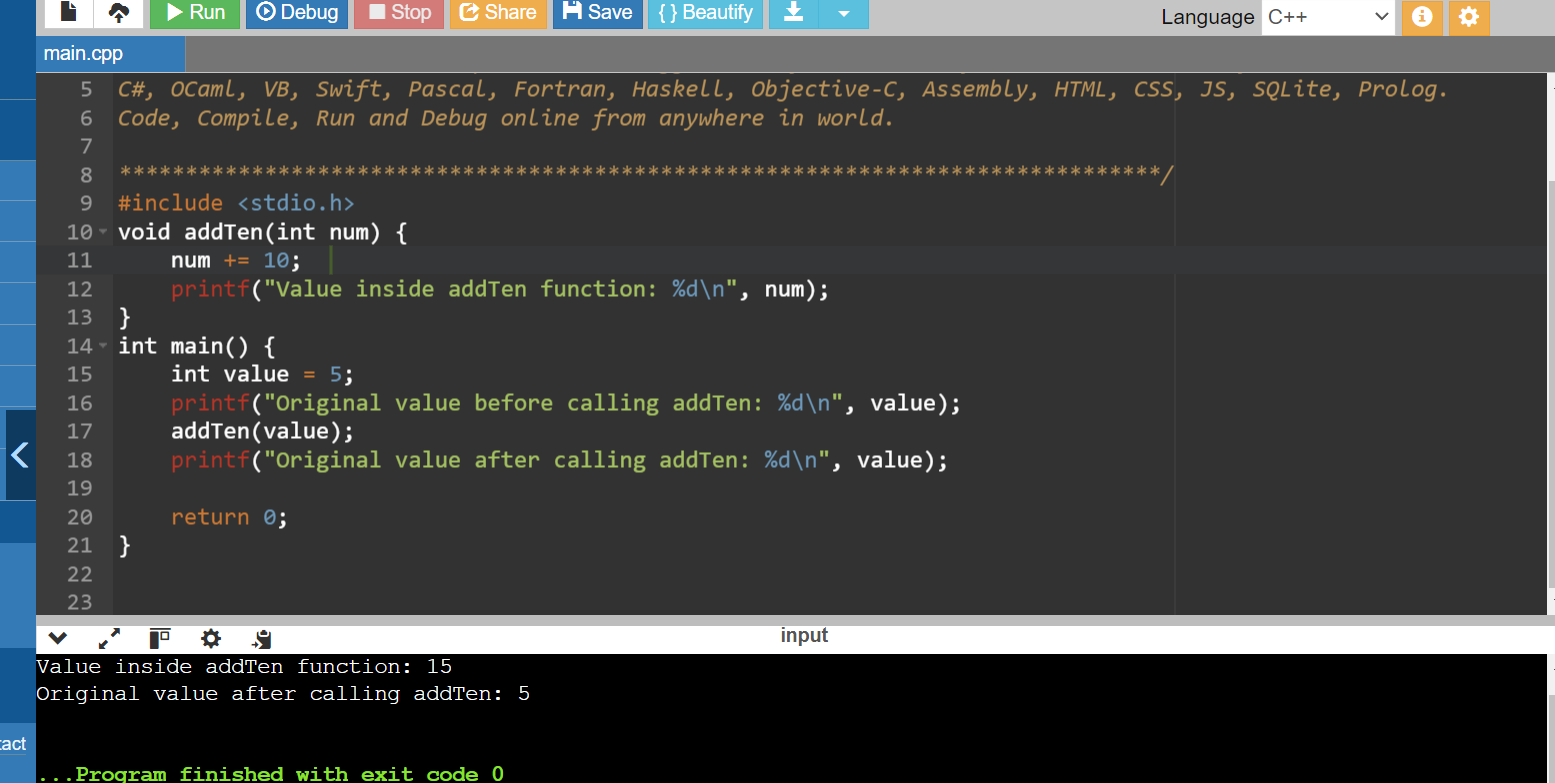
printf("Original value before calling addTen: %d\n", value);

addTen(value);

printf("Original value after calling addTen: %d\n", value);

return 0;

}



Q.10 Function Returning a Reference:\*

Write a function that returns a reference to a static variable. Modify the returned value in the main function and print it.

#include <iostream>

int& getStaticVariable() {

static int staticVar = 10;

return staticVar;

}

int main() {

int& ref = getStaticVariable();

std::cout << "Initial value of static variable: " << ref << std::endl;

ref += 5;

std::cout << "Modified value of static variable: " << ref << std::endl;

std::cout << "Value of static variable after calling getStaticVariable again: " << getStaticVariable() << std::endl;

return 0;

}

