Q . Operator Overloading

Rectangle Class: Define a class Rectangle with member variables for width and height. Overload the + operator to return a new Rectangle object representing the sum of the areas of two rectangles.

Fraction Class: Create a class Fraction with numerator and denominator. Overload the arithmetic operators (+, -, \*, /) for fraction addition, subtraction, multiplication, and division.

Money Class: Design a class Money to store currency amount and type (e.g., USD, EUR). Overload the comparison operators (==, !=, <, >, <=, >=) for Money objects, considering currency types and exchange rates.

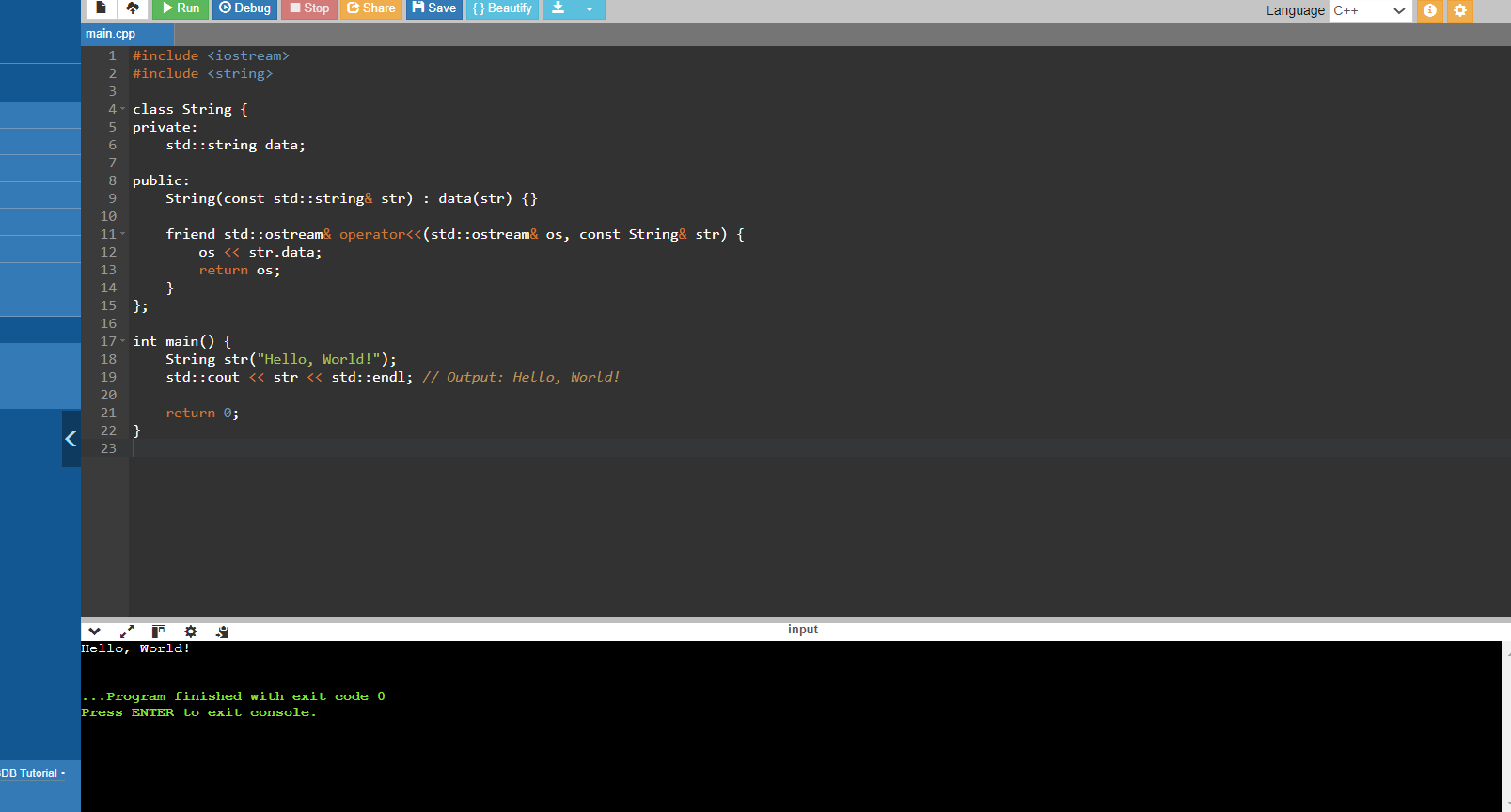
String Stream Insertion: Overload the stream insertion operator (<<) for a custom String class to allow easy printing of strings to standard output.

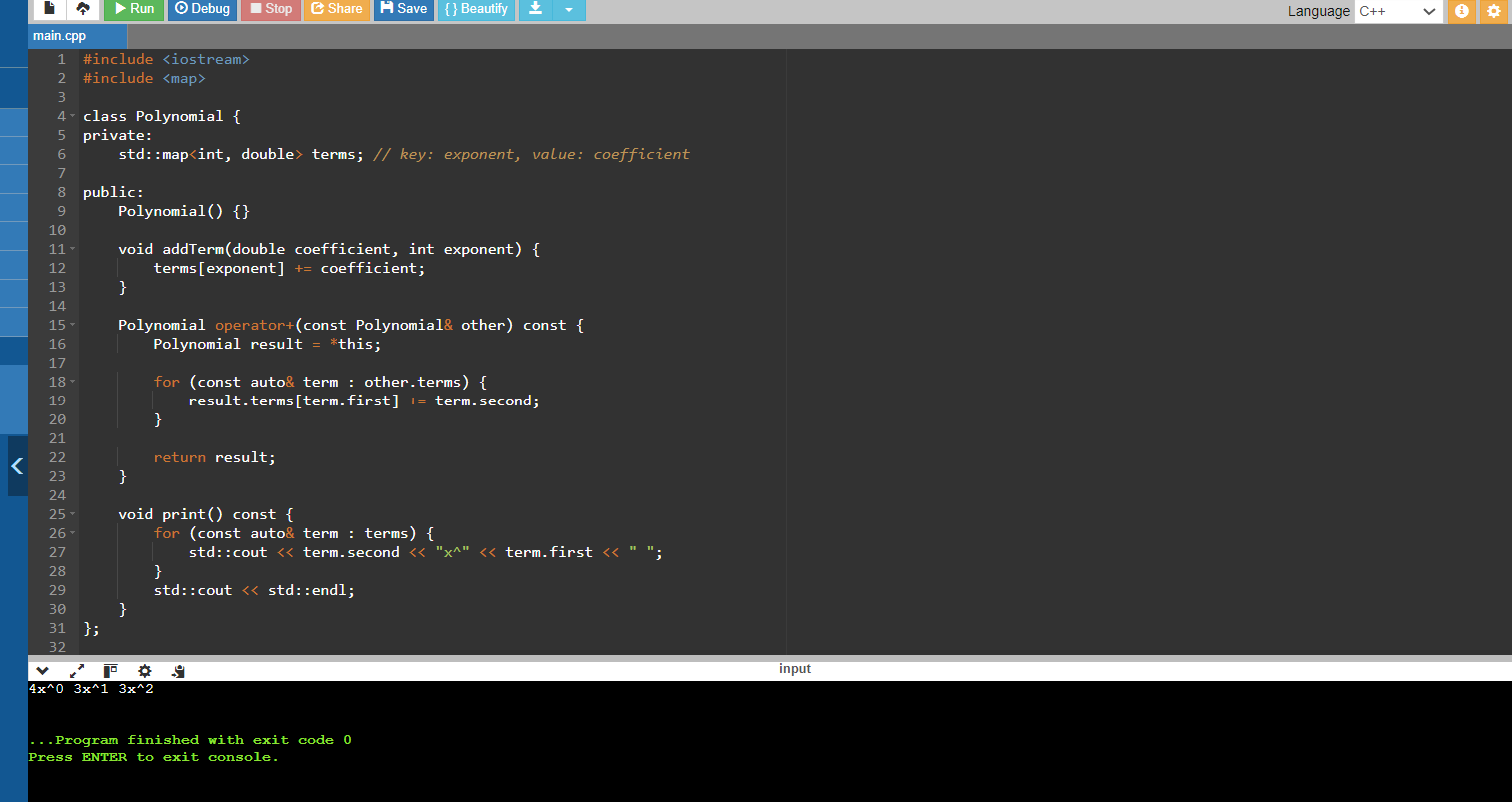
Polynomial Addition: Implement a class Polynomial to represent polynomials with terms (coefficient and exponent). Overload the + operator to add two Polynomial objects and return a new Polynomial with the combined terms.

Function Overloading









Inventory Management: Implement a class Item with properties like name, price, and quantity. Overload the << operator for easy printing of item details to the console.

Custom Container: Design a class CustomList that behaves like a list but overloads the subscript operator ([]) to perform boundary checking and prevent out-of-bounds access.

Smart Pointers: Define a smart pointer class MySmartPtr that overloads the dereference operator (\*) and arrow operator (->) for memory management and safe access to the pointed-to object.

Template Class (Vector): Implement a template class Vector that can store elements of any data type and overload operators (+, -, []) to work with vectors of different types.

Matrix Operations (Challenge): Create a class Matrix to store a 2D array and overload arithmetic operators (+, -, \*) for matrix addition, subtraction, and multiplication (considering matrix dimensions).