**OOPS Assignment**

-What is OOP? List OOP concepts

OOPS stands for Object-Oriented Programming System. C++ is a popular programming language that supports object-oriented programming (OOP) concepts. Here are the key OOP concepts in C++:

1. **Class and Object:**
   * **Class:** A blueprint for creating objects. It defines the attributes and behaviors common to all objects of the class.
   * **Object:** An instance of a class. Objects are created based on the structure defined by the class.

**Encapsulation:**

* The bundling of data (attributes) and methods (functions) that operate on the data within a single unit (class).

**Inheritance:**

* A mechanism that allows a class to inherit properties and behaviors from another class.

**Polymorphism:**

* The ability to present the same interface for different data types.
* Includes function overloading and virtual functions.

**Abstraction:**

* The process of hiding the complex implementation details and showing only the essential features of an object. These are the fundamental OOP concepts in C++. Utilizing these concepts helps in creating modular, reusable, and efficient code.

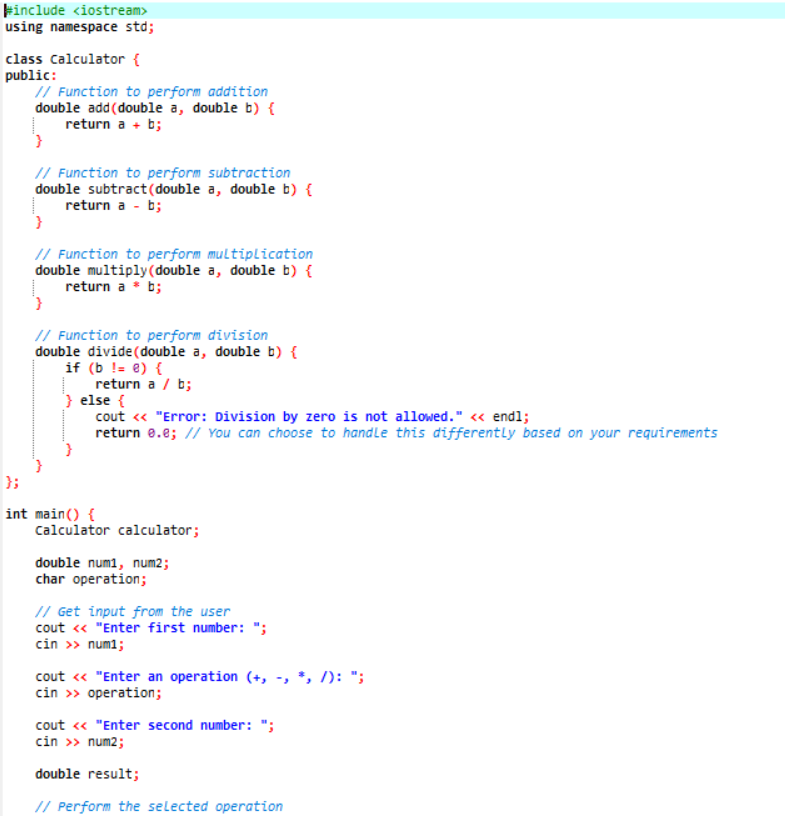
- What is the difference between OOP and POP?

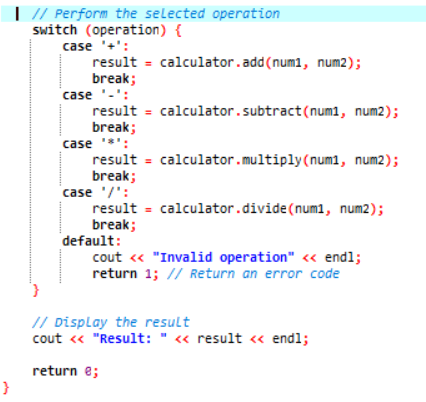
1. **Basic Unit:**
   * **OOP:** Basic unit is an "object," which combines data and functions (methods) that operate on the data.
   * **POP:** Basic unit is a "procedure" or "function," which operates on data.
2. **Data and Functions:**
   * **OOP:** Encapsulation is used to bundle data and functions into a single unit (class).
   * **POP:** Data and functions are separate; functions operate on data.
3. **Encapsulation:**
   * **OOP:** Encapsulation hides the internal details of an object and exposes a well-defined interface. Data is private to the object and can only be accessed through public methods.
   * **POP:** Encapsulation is less emphasized. Data may be global or shared among functions.
4. **Inheritance:**
   * **OOP:** Supports inheritance, allowing a class to inherit properties and behaviors from another class.
   * **POP:** Typically does not have built-in mechanisms for inheritance. Code reuse is achieved through functions.
5. **Polymorphism:**
   * **OOP:** Supports polymorphism, allowing objects of different types to be treated as objects of a common base type. This includes method overloading and overriding.
   * **POP:** Polymorphism may be achieved through function overloading and parameterized functions.
6. **Code Reusability:**
   * **OOP:** Promotes code reusability through the creation of reusable classes and objects.
   * **POP:** Code reuse is often achieved through the reuse of functions.
7. **Flexibility and Maintenance:**
   * **OOP:** Provides better support for large and complex systems, making it easier to maintain and extend. Changes to one part of the code are less likely to affect other parts.
   * **POP:** May become complex and harder to maintain as the size of the program increases. Changes to one part of the code may have a broader impact.
8. **Examples:**
   * **OOP:** Java, C++, Python.
   * **POP:** C, Pascal.

These differences highlight the contrasting approaches to structuring code and organizing logic in OOP and POP paradigms. While OOP is more focused on the organization of code into objects and classes, POP is centered around procedures or functions that operate on data. The choice between OOP and POP often depends on the nature of the problem being solved and the design goals of the software.

Top of Form

- WAP to create simple calculator using class.





- Define a class to represent a bank account. Include the following members:

1. Data Member:

-Name of the depositor

-Account Number

-Type of Account

-Balance amount in the account

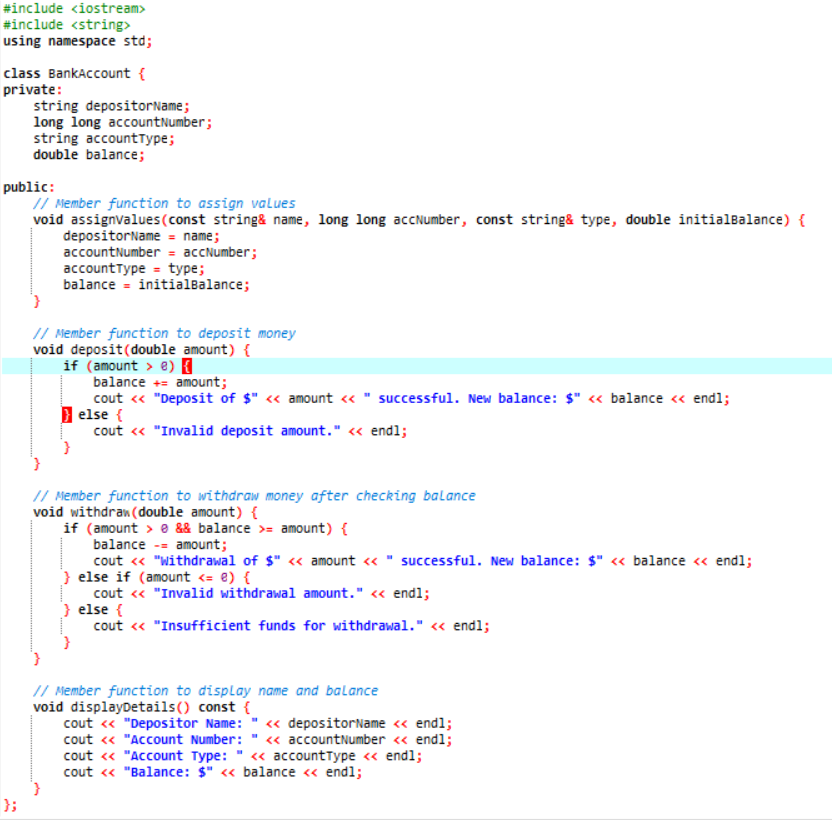
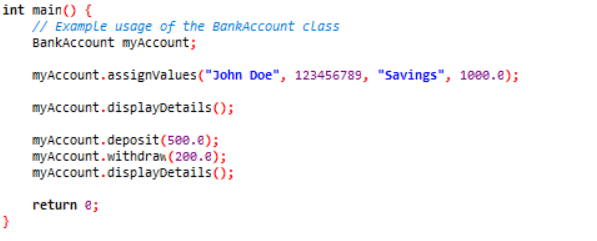
2. Member Functions

-To assign values

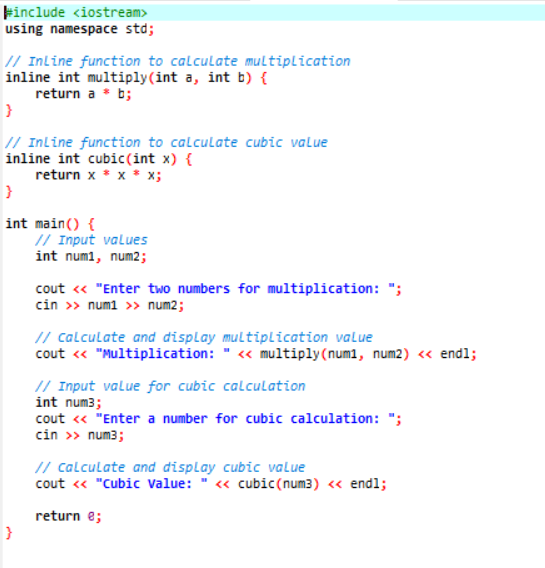
-To deposited an amount

-To withdraw an amount after checking balance

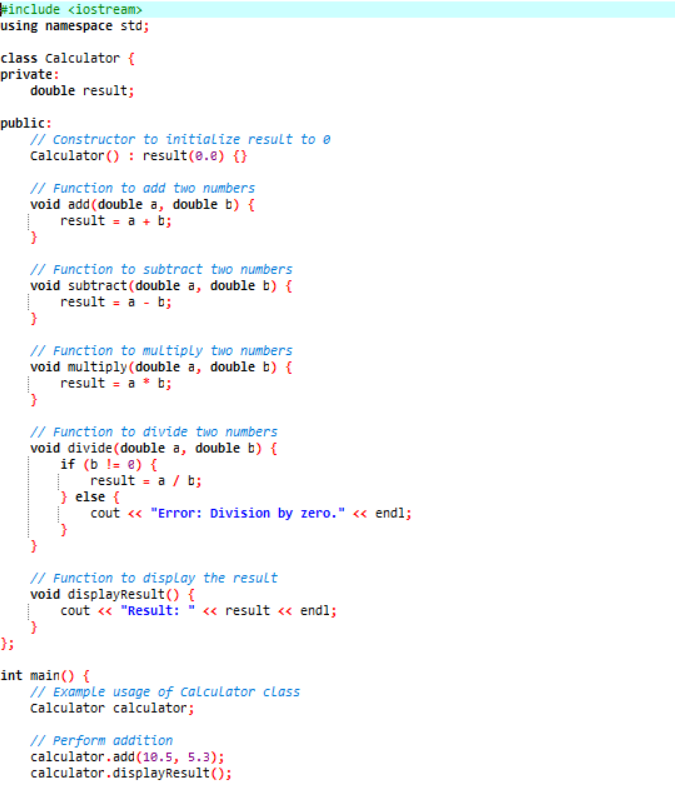
-To display name and balance

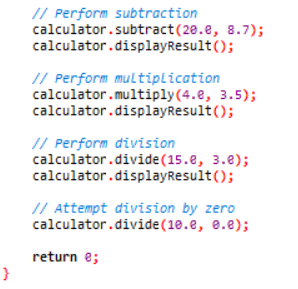
 

- Write a program to find the multiplication values and the cubic values using inline function.

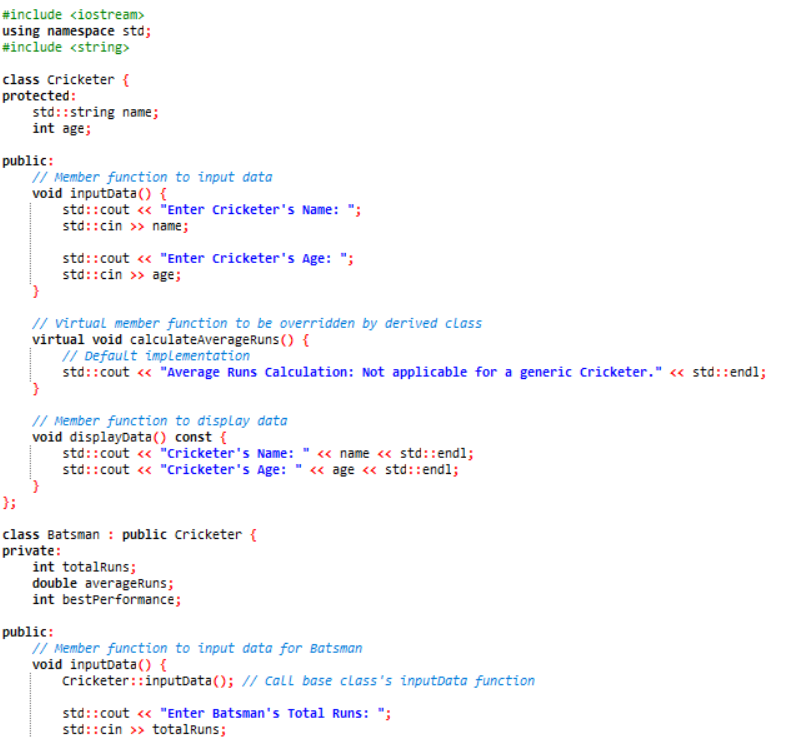


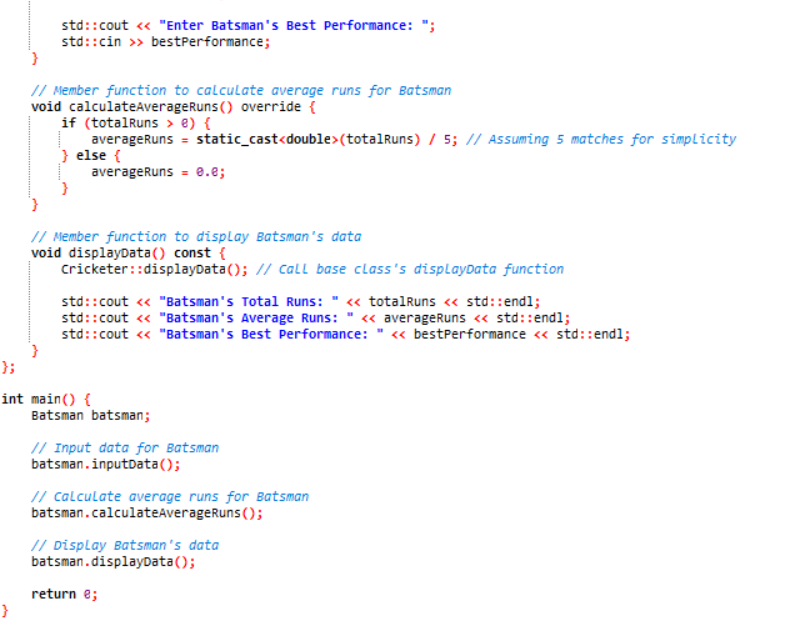
- Write a program of Addition, Subtraction, Division, Multiplication using constructor.



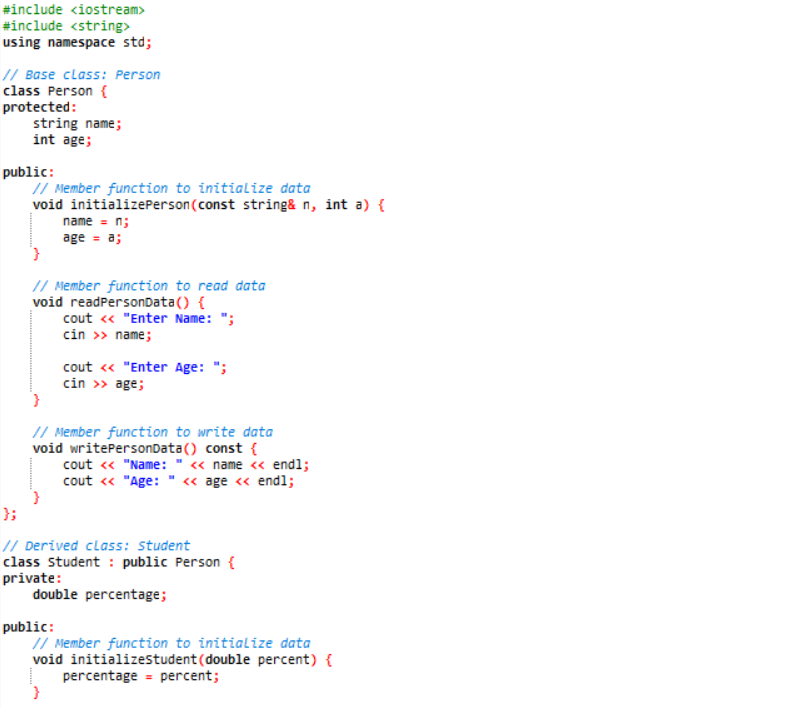


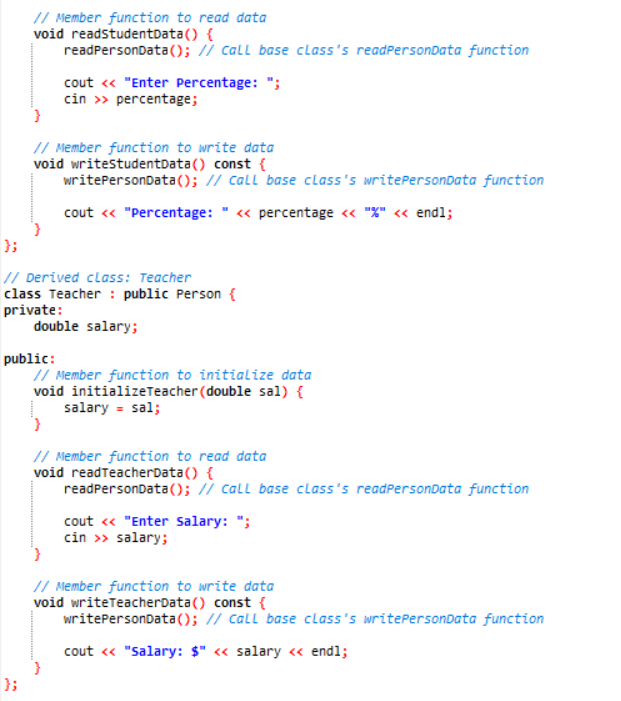
- Assume a class cricketer is declared. Declare a derived class batsman from cricketer. Data member of batsman. Total runs, Average runs and best performance. Member functions input data, calculate average runs, Display data. (Single Inheritance)

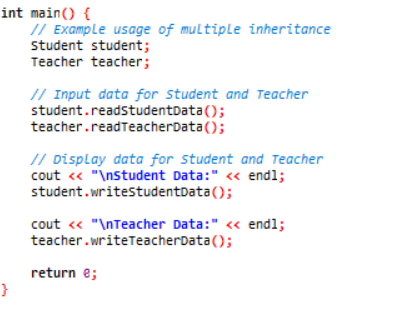




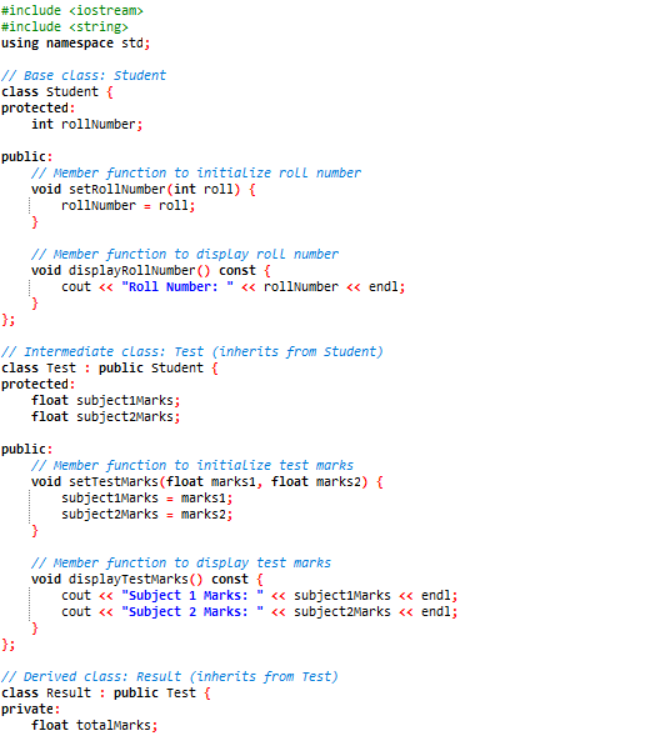
- Create a class person having members name and age. Derive a class student having member percentage. Derive another class teacher having member salary. Write necessary member function to initialize, read and write data. Write also Main function (Multiple Inheritance).

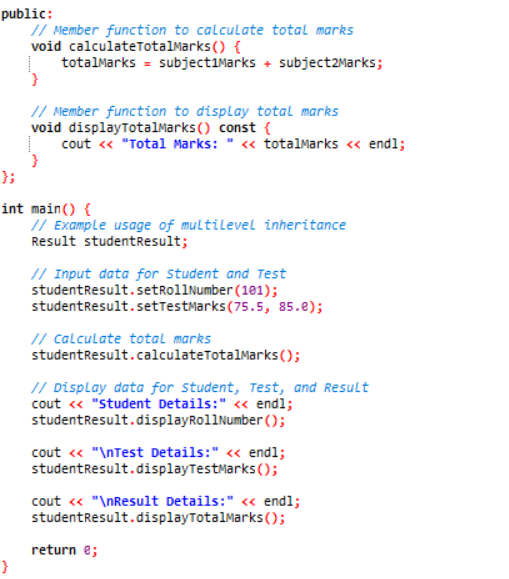




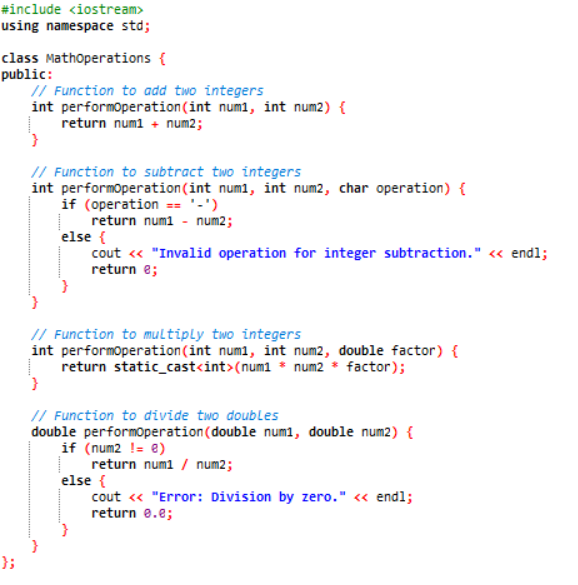


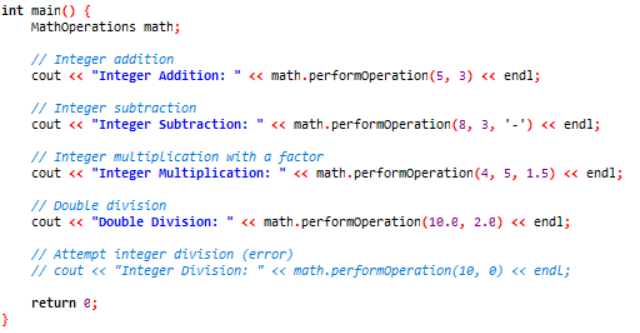
- Assume that the test results of a batch of students are stored in three different classes. Class Students are storing the roll number. Class Test stores the marks obtained in two subjects and class result contains the total marks obtained in the test. The class result can inherit the details of the marks obtained in the test and roll number of students. (Multilevel Inheritance).



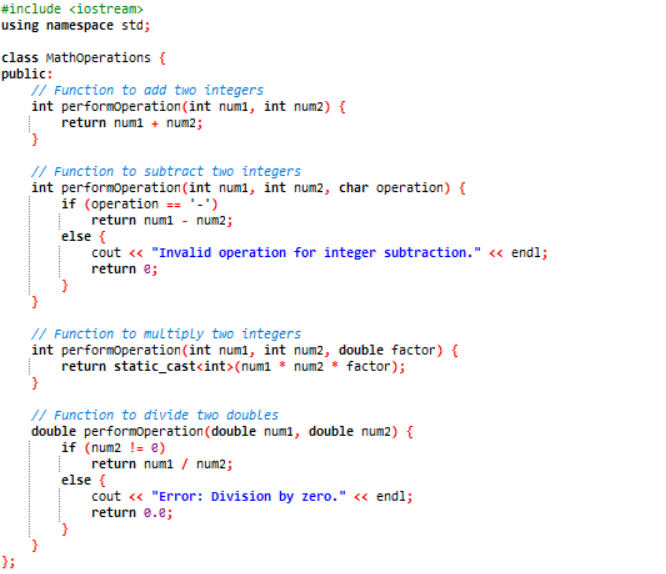


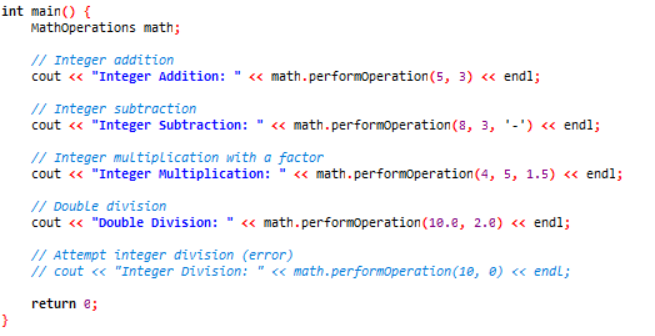
- Write a program to Mathematic operation like Addition, Subtraction, Multiplication, Division Of two number using different parameters and Function Overloading.



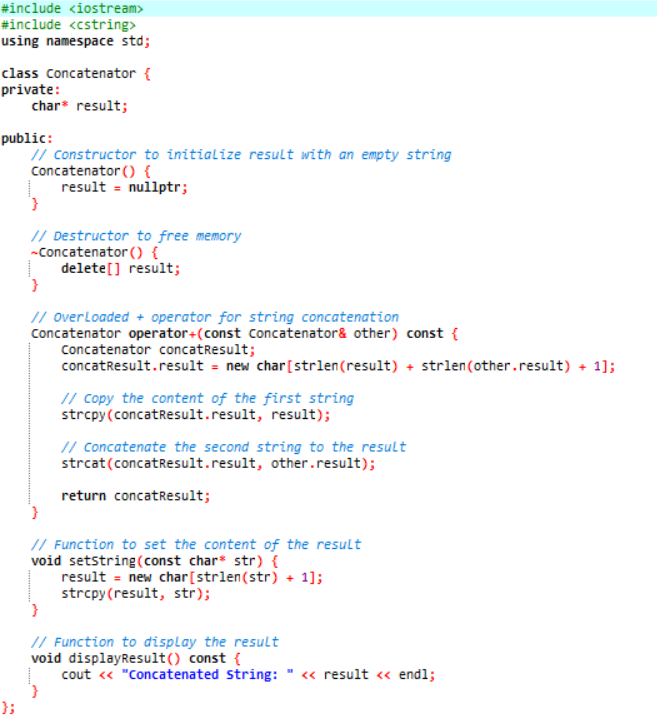


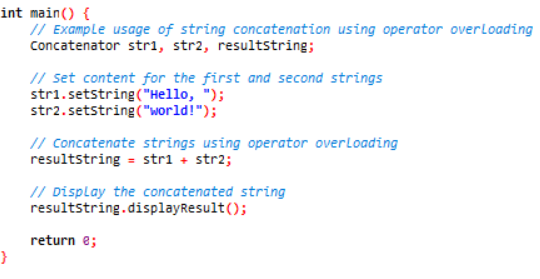
- Write a Program of Two 1D Matrix Addition using Operator Overloading.





- Write a program to concatenate the two strings using Operator Overloading.



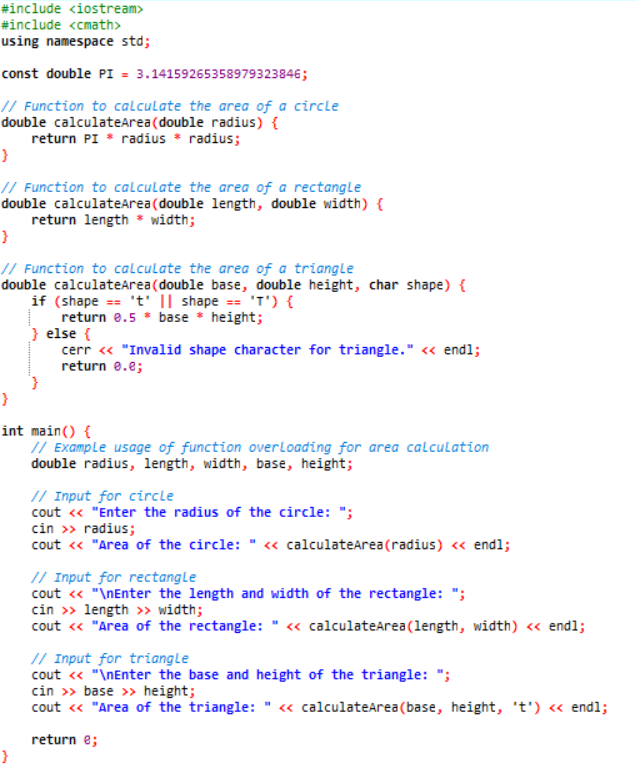


- Write a program to calculate the area of circle, rectangle and triangle using Function Overloading.

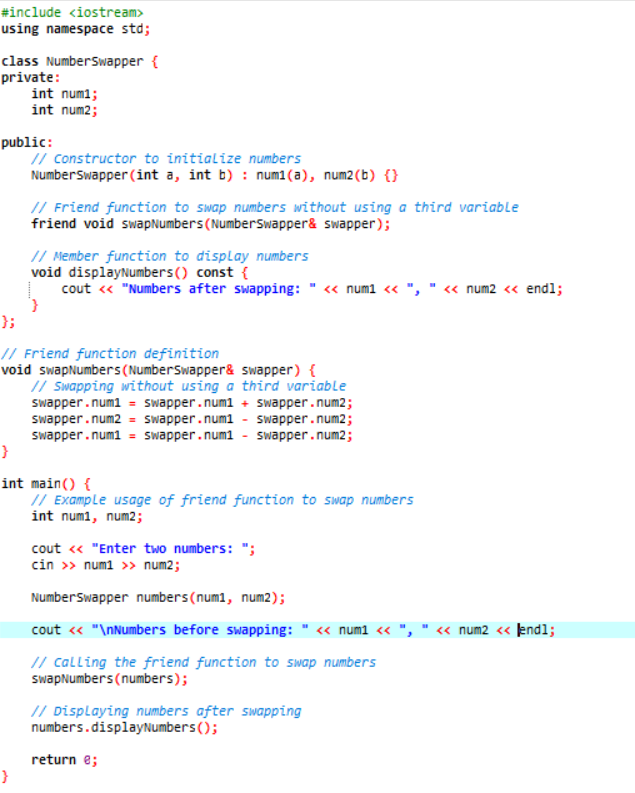
¬ Rectangle: Area \* breadth

¬ Triangle: ½ \*Area\* breadth

¬ Circle: Pi \* Area \*Area



- Write a program to swap the two numbers using friend function without using third variable.



- Write a program to find the max number from given two numbers using friend function.

