**Task: Implement a Custom Increment Operator in a Class**

**Objective**: Create a simple C++ program that demonstrates the use of operator overloading, specifically the increment operator (++). You will implement a class that manages an integer value and allows it to be incremented by a specific amount.

**Requirements:**

1. **Class Definition (A)**:
2. Create a class named A that contains a private member variable:
3. a: an integer that will store a value.
4. Implement the following:
5. A default constructor that initializes a to 0.
6. A public method show() that displays the current value of a.
7. Overload the prefix increment operator (++) so that it increases the value of a by 2 when the operator is used.
8. **Main Function**:
9. In the main() function, perform the following tasks:
10. Create an instance of the class A.
11. Call the show() method to display the initial value of a.
12. Use the overloaded increment operator (++) on the instance.
13. Call the show() method again to display the updated value of a.

**Example Interaction:**

When the program is executed, the output should look like this:

The value of a = 0

The value of a = 2

**Additional Notes:**

1. Ensure that the operator overloading function for ++ correctly modifies the value of a.
2. Pay attention to the visibility of class members and use access specifiers appropriately.

This task will help students understand the concepts of encapsulation and operator overloading in C++, reinforcing their skills in class design and member function implementation.

* Overload the prefix increment operator (++) so that it
* Returns a new object of type A that contains the updated value of a access specifiers.

Here’s a detailed task description based on the provided code that your students can follow to write their own implementation:

**Task: Implement a Class with Prefix Increment Operator Overloading**

**Objective**: Create a C++ program that demonstrates operator overloading by implementing a custom prefix increment operation in a class. This will help students understand how to define and use overloaded operators with class instances.

**Requirements:**

1. **Class Definition (A)**:
2. Create a class named A with the following specifications:
3. **Private Member Variable**:
4. int a: an integer to store a value.
5. **Constructor**:
6. Implement a default constructor that initializes a to 0.
7. **Member Function**:
8. Implement a public method show() that outputs the current value of a in the format: the value of a = <value>.
9. **Operator Overloading**:
10. Overload the prefix increment operator (++) to:
11. Increment the value of a by 2.
12. Create a new object of type A, set its a to the updated value, and return this new object.
13. **Main Function**:
14. In the main() function, perform the following tasks:
15. Create two instances of the class A named x and y.
16. Call the show() method on both instances to display their initial values.
17. Use the overloaded prefix increment operator on x and assign the result to y.
18. Call the show() method again on both x and y to display their updated values.

**Example Interaction:**

When the program is executed, the output should resemble the following:

The value of a = 0

The value of a = 0

The value of a = 2

The value of a = 2

**Additional Notes:**

1. Ensure that the operator overloading function correctly modifies the value of a and that the new object returned contains the incremented value.
2. Pay attention to encapsulation principles by keeping the member variable a private and providing public methods for interaction.
3. Use appropriate formatting and comments in your code to enhance readability.

**Task: Implement a Class with Operator Overloading for Addition**

**Objective**: Create a C++ program that demonstrates operator overloading by implementing a custom addition operation in a class. This will help students understand how to define and use overloaded operators to work with class objects.

**Requirements:**

1. **Class Definition (A)**:
2. Create a class named A with the following specifications:
3. **Private Member Variables**:
4. int a: to store the first integer value.
5. int b: to store the second integer value.
6. **Constructor**:
7. Implement a default constructor that initializes both a and b to 0.
8. **Member Functions**:
9. Implement a public method enter() that prompts the user to input values for a and b.
10. Implement a public method show() that displays the current values of a and b in the format:

The value of a = <value>

The value of b = <value>

1. **Operator Overloading**:
2. Overload the addition operator (+) to:
3. Create a new object of type A, where the new object's a is the sum of the a values of both objects, and the new object's b is the sum of the b values of both objects.
4. Return the new object.
5. **Main Function**:
6. In the main() function, perform the following tasks:
7. Create three instances of class A named x, y, and z.
8. Call the enter() method on x and y to allow the user to input values.
9. Call the show() method on both x and y to display their values.
10. Use the overloaded addition operator to add x and y, storing the result in z.
11. Call the show() method again on both x and y to display their values after the addition.

**Example Interaction:**

When the program is executed, the expected output should look like this:

Enter 1st value

5

Enter 2nd value

10

Enter 1st value

3

Enter 2nd value

7

The value of a = 5

The value of b = 10

The value of a = 3

The value of b = 7

The value of a = 8

The value of b = 17

**Additional Notes:**

1. Ensure that the operator overloading function correctly calculates the sum of the member variables a and b.
2. Use appropriate formatting and comments in your code to enhance readability.
3. Emphasize the importance of encapsulation by keeping member variables private and providing public methods for interaction.