
Project - 1 **MA517M-Basic Programming Laboratory** Last Date: 14 November 2026
Name _____ Roll No.: **MA26M105**

Quaternions using C++ Classes and Operator Overloading

Objective: To design a C++ program that implements quaternions using classes, supports arithmetic operations, demonstrates non-commutative multiplication, and uses operator overloading.

A quaternion is a four-dimensional number of the form

$$q = a + bi + cj + dk, \quad a, b, c, d \in \mathbb{R},$$

where the fundamental quaternion units satisfy

$$i^2 = j^2 = k^2 = ijk = -1.$$

The multiplication rules are

$$ij = k, \quad jk = i, \quad ki = j,$$

while reversing the order changes the sign:

$$ji = -k, \quad kj = -i, \quad ik = -j.$$

Therefore, quaternion multiplication is generally **non-commutative**:

$$q_1 q_2 \neq q_2 q_1.$$

Problem Description

Design a class `Quaternion` that represents quaternions of the form

$$q = a + bi + cj + dk.$$

The class should support quaternion addition, subtraction, multiplication, conjugation, norm calculation, equality comparison, and display using operator overloading.

In particular, demonstrate that quaternion multiplication is non-commutative by computing both

$$q_1 q_2 \quad \text{and} \quad q_2 q_1.$$

Class Specification

- **Class Name:** Quaternion
- **Private Data Members:**
 - double a;
 - double b;
 - double c;
 - double d;
- **Public Member Functions:**
 - Quaternion(double a=0, double b=0, double c=0, double d=0);
 - Quaternion conjugate() const;
 - double norm() const;
 - void display() const;

Quaternion Operations

For

$$q_1 = a_1 + b_1i + c_1j + d_1k$$

and

$$q_2 = a_2 + b_2i + c_2j + d_2k,$$

the following operations should be implemented.

Addition

$$q_1 + q_2 = (a_1 + a_2) + (b_1 + b_2)i + (c_1 + c_2)j + (d_1 + d_2)k.$$

Subtraction

$$q_1 - q_2 = (a_1 - a_2) + (b_1 - b_2)i + (c_1 - c_2)j + (d_1 - d_2)k.$$

Multiplication

The product is given by

$$\begin{aligned} q_1q_2 = & (a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2) \\ & + (a_1b_2 + b_1a_2 + c_1d_2 - d_1c_2)i \\ & + (a_1c_2 - b_1d_2 + c_1a_2 + d_1b_2)j \\ & + (a_1d_2 + b_1c_2 - c_1b_2 + d_1a_2)k. \end{aligned}$$

Students should verify that, in general,

$$q_1q_2 \neq q_2q_1.$$

Conjugate

For

$$q = a + bi + cj + dk,$$

the conjugate is

$$\bar{q} = a - bi - cj - dk.$$

Norm

The norm of a quaternion is

$$\|q\| = \sqrt{a^2 + b^2 + c^2 + d^2}.$$

Verify the important identity

$$\boxed{\|q_1 q_2\| = \|q_1\| \|q_2\|}.$$

Operator Overloading

The following operators should be overloaded:

- operator `+`(`()`) Quaternion addition
- operator `-`(`()`) Quaternion subtraction
- operator `*`(`()`) Quaternion multiplication
- operator `==(())`, `!=(())` Checks equality or inequality of two quaternions
- operator `<<()` Displays a quaternion in standard mathematical form

The following member functions should also be implemented:

- `conjugate()` — returns the conjugate of a quaternion
- `norm()` — returns the norm of a quaternion

Tasks

1. Create two quaternions $q_1 = 1 + 2i + 3j + 4k$ and $q_2 = 2 - i + 2j - 3k$.
2. Compute

$$q_1 + q_2, \quad q_1 - q_2.$$

3. Compute

$$q_1 q_2$$

using the overloaded multiplication operator.

4. Compute

$$q_2 q_1$$

and verify that

$$q_1 q_2 \neq q_2 q_1.$$

5. Compute the conjugates

$$\overline{q_1}, \quad \overline{q_2}.$$

6. Compute the norms

$$\|q_1\|, \quad \|q_2\|.$$

7. Verify numerically that

$$\|q_1 q_2\| = \|q_1\| \|q_2\|.$$

8. Compare q_1 and q_2 using the overloaded `==` and `!=` operators.

9. Display all results using the overloaded `<<` operator.

Expected Output Example

```
q1 = 1 + 2i + 3j + 4k
```

```
q2 = 2 - i + 2j - 3k
```

```
q1 + q2 = 3 + i + 5j + k
```

```
q1 - q2 = -1 + 3i + j + 7k
```

```
q1 * q2 = 15 + 3i + 9j - 6k
```

```
q2 * q1 = 15 - 3i - 9j + 6k
```

```
q1 == q2 : False
```

```
q1 != q2 : True
```

```
conjugate(q1) = 1 - 2i - 3j - 4k
```

```
conjugate(q2) = 2 + i - 2j + 3k
```

```
||q1|| = 5.47723
```

```
||q2|| = 4.24264
```

```
||q1 * q2|| = 23.2379
```

```
||q1|| * ||q2|| = 23.2379
```

Additional Challenge

Extend the `Quaternion` class to implement quaternion division.

For a non-zero quaternion q ,

$$q^{-1} = \frac{\overline{q}}{\|q\|^2}.$$

Then implement

$$q_1/q_2 = q_1 q_2^{-1}.$$

Finally, verify that

$$qq^{-1} = 1$$

for a non-zero quaternion q .

Conceptual Questions

1. Why is quaternion multiplication non-commutative?
2. What happens if the multiplication operator is implemented incorrectly by assuming

$$ij = ji?$$

3. Why is the quaternion conjugate useful in computing the inverse?
4. Verify the identity

$$q\bar{q} = \|q\|^2.$$

5. Explain why quaternion multiplication is a good example of operator overloading in C++.

Project - 2 : Word Shuffle Game Using C++ Classes

Problem Statement

Design and implement a **Word Shuffle Game** using **C++ classes**. The program should allow the user to unscramble letters to form meaningful words. The project should utilize object-oriented programming concepts such as classes, objects, encapsulation, and methods to handle word selection, shuffling, and user interaction.

Project Requirements

1. Create a **Word** class to represent a word in the game.
 - (a) Include a method to randomly select a word from a predefined list.
 - (b) Include a method to shuffle the letters of the word.
 - (c) Include a method to display the shuffled word to the user.
2. Create a **Game** class to manage gameplay.
 - (a) Display options for the user: **Play** or **Solution**.
 - (b) If the user chooses **Play**, allow them to enter guesses.
 - (c) Validate user input and provide feedback if the guess is correct or incorrect.
 - (d) Keep track of the number of attempts.
 - (e) Allow the user to quit by entering a special command (e.g., **Q/q**) and confirm before exiting.
3. If the user chooses **Solution**, display the original word and the correct sequence of letters.
4. Ensure proper encapsulation of word logic and gameplay operations within the respective classes.

Suggested Class Structure

1. Word Class:

- Data member: `string originalWord`, `string shuffledWord`
- Methods: `selectWord()`, `shuffleWord()`, `displayWord()`

2. Game Class:

- Data member: `Word` object, user choice, attempt counter
- Methods: `playGame()`, `showSolution()`, `processGuess(string guess)`, `confirmQuit()`