📘 Static vs. Non-Static Member Functions in C++

In C++, member functions of a class can be either **static** or **non-static**. Understanding the difference between them is crucial for designing efficient and logical object-oriented programs.

🔹 Non-Static Member Functions

Definition:

Non-static member functions are associated with an **instance** of a class. They can access both **instance variables** and **static variables** of the class.

Characteristics:

- ❗ Require an object to be called. ❗
- Have access to the `this` pointer (i.e., the current object).
- Can access and modify instance (non-static) data members.

Example:

```
#include <iostream>
using namespace std;

class Student {
private:
    string name;
    int age;

public:
    void setDetails(string n, int a) {
        name = n; //non-static methods can access and modify the data members of the class
        age = a;
    }

    void display() {
        cout << "Name: " << name << ", Age: " << age << endl;
    }
};

int main() {
    //
    // Step 1. Create an object of the Student class
    Student s1;

    //
    // Step 2. Use the Student variable to call the non-static methods
    s1.setDetails("Alice", 20);
    s1.display();  // Output: Name: Alice, Age: 20


    return 0;
}
```

🔹 Static Member Functions

Definition:

Static member functions are associated with the **class itself**, not with any particular object. They can only access **static data members** or other **static member functions**.

Characteristics:

- Can be called using the class name (no object needed).
- Do **not** have access to the `this` pointer.
- Cannot access non-static members directly.

Example:

```
#include <iostream>
using namespace std;

class Counter {
private:
    static int count;

public:
    static void increment() {
        count++;
    }

    static void showCount() {
        cout << "Count: " << count << endl;
    }
};

// Definition of static member
int Counter::count = 0;

int main() {
    //
    // Use ClassName:: to call static methods
    Counter::increment();
    Counter::increment();
    Counter::showCount();

    return 0;
}
```

🔸 Key Differences

Feature	Non-Static Function	Static Function
Called using	Object	Class name or object
Access to `this` pointer	Yes	No
Access to instance data	Yes	No
Access to static data	Yes	Yes
Memory allocation	Per object	Shared across all instances

✅ When to Use What?

- Use **non-static** functions when behavior depends on object-specific data.
- Use **static** functions for utility operations or when behavior is independent of object state.

🎯 Quiz!

Here's a short quiz on the topic: quiz

🔭 Markdown Viewer

How to view the markdown files in a browser...

- Markdown Viewer

🧠 Lecture Practices

Here are the lecture Practices...

- Day 1
- Day 2
- Day 3

🔍 Lecture Quizzes

Here are the lecture quizzes...

- Day 1
- Day 2
- Day 3

⚙️ Weekly Topics

Here are the topics for the week...

- Calling Methods
- Calling Methods 2
- Creating Methods
- Iterators
- Vectors
- References
- Const
- Erasing in a Loop
- Default Parameters