Tutorial 53 - this Pointer in C++

Definition:

- The `this` pointer is an implicit pointer available in every non-static member function of a class.
- It points to the object that invokes the member function.

Code Examples:

1. Basic Usage of `this` Pointer:

```
#include<iostream>
using namespace std;
class A {
    int a;
public:
    void setData(int a) {
        this->a = a; // Assigns parameter to the private member
    }
    void getData() {
        cout << "The value of a is " << a << endl;
    }
};
int main() {
    A obj;
    obj.setData(4); // Sets value of `a` to 4
    obj.getData();  // Outputs: "The value of a is 4"
    return 0;
}

```

Explanation:

1. **Class Structure**:
   - **Private Data Member**: `a`.
   - **Member Functions**:
     - `setData(int a)`: Uses `this->a` to distinguish between the parameter and the member variable.
     - `getData()`: Prints the value of `a`.
2. `this` **Pointer**:
   - Used in `setData()` to refer to the member variable `a` of the calling object.

---

2. Returning Reference to Invoking Object:

```
class A {
    int a;
public:
    A &setData(int a) {
        this->a = a;
        return *this; // Returns reference to the current object
    }
    void getData() {
        cout << "The value of a is " << a << endl;
    }
};
int main() {
    A obj;
    obj.setData(4).getData(); // Chained function call
    return 0;
}

```

Explanation:

1. **Returning Object Reference**:
    - The `setData()` function returns a reference to the invoking object (`*this`).
2. **Chaining Function Calls**:
    - Allows chaining: `obj.setData(4).getData();`
    - The returned reference is used to call `getData()`.

Key Points:

1. **Purpose of `this` Pointer**:
    - Points to the invoking object.
    - Helps resolve name conflicts (e.g., member variable vs. function parameter with the same name).
2. **Returning Object Reference**:
    - Allows method chaining by returning `*this`.
3. **Usage**:
    - Commonly used in setters and fluent APIs.

Short Notes for Notebook:

Definition:

- `this` **pointer**: Implicit pointer in every non-static member function pointing to the invoking object.

Basic Example:

```
class A {
    int a;
public:
    void setData(int a) { this->a = a; }
    void getData() { cout << "Value: " << a << endl; }
};

```

- `this->a`: Refers to the class member `a`.

Chaining Example:

```
class A {
    int a;
public:
    A &setData(int a) { this->a = a; return *this; }
};

```

- `setData()` returns `*this` for chaining: `obj.setData(4).getData();`

Benefits:

1. Resolves name conflicts (e.g., `this->a = a`).
2. Enables method chaining for concise code.