Vikram
@code._learning

# C++ OOPS

## Object-Oriented Programming in C++

Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects," which can contain data and code that manipulates that data. C++ supports OOP, and it involves key concepts such as classes, objects, inheritance, polymorphism, encapsulation, and abstraction.

## Classes and Objects

A class is a blueprint for creating objects (instances). It defines a data structure by bundling data (attributes) and methods (functions) that operate on the data.

@code._learning

```
---- Defining a Class: ----

class Rectangle {
public:
    int width, height;

    int area() {
        return width * height;
    }
};


---- Creating Objects: ----

int main() {
    Rectangle rect;  // Create an object of Rectangle
    rect.width = 5;
    rect.height = 10;

    cout << "Area: " << rect.area() << endl;  // Output: Area: 50
    return 0;
}
```

# Member Functions

Member functions are functions defined inside a class. They operate on the data members of the class.

```
  ---- Defining Member Functions: ----

class Rectangle {
public:
    int width, height;

    int area() {
        return width * height;
    }

    void setDimensions(int w, int h) {
        width = w;
        height = h;
    }
};


  ---- Using Member Functions: ----

int main() {
    Rectangle rect;
    rect.setDimensions(5, 10);

    cout << "Area: " << rect.area() << endl;  // Output: Area: 50
    return 0;
}
```

# Constructors and Destructors

Constructors and destructors are special member functions that are automatically called when an object is created and destroyed, respectively.

## 1. Constructors:

- Constructors initialize objects. They have the same name as the class and no return type.

```
class Rectangle {
public:
    int width, height;

    Rectangle(int w, int h) {   // Constructor
        width = w;
        height = h;
    }

    int area() {
        return width * height;
    }
};
```

## 2. Destructors:

- Destructors clean up resources. They have the same name as the class, preceded by a tilde (~), and no return type.

```cpp
class Rectangle {
public:
    int width, height;

    Rectangle(int w, int h) {
        width = w;
        height = h;
    }

    ~Rectangle() {  // Destructor
        cout << "Rectangle destroyed" << endl;
    }

    int area() {
        return width * height;
    }
};

int main() {
    Rectangle rect(5, 10);  // Constructor called
    cout << "Area: " << rect.area() << endl;  // Output: Area: 50
    return 0;  // Destructor called when rect goes out of scope
}
```

## Access Specifiers (public, private, protected)

Access specifiers define the accessibility of class members.

1. **public**: Members are accessible from outside the class.

2. **private**: Members are accessible only within the class.

3. **protected**: Members are accessible within the class and by derived classes.

## This Pointer

The **this** pointer is an implicit pointer to the object that invokes a member function. It is used to access the object's members and to return the object itself from a member function.

Example On Next Page

@code._learning

```cpp
class Rectangle {
private:
    int width, height;

public:
    Rectangle(int w, int h) {
        this->width = w;
        this->height = h;
    }

    Rectangle& setWidth(int w) {
        this->width = w;
        return *this;
    }

    Rectangle& setHeight(int h) {
        this->height = h;
        return *this;
    }

    int area() const {
        return width * height;
    }
};

int main() {
    Rectangle rect(5, 10);
    rect.setWidth(7).setHeight(8);  // Method chaining
    cout << "Area: " << rect.area() << endl;  // Output: Area: 56
    return 0;
}
```