Procedural vs. Object-Oriented Programming

Understanding the Differences

Introduction

- Programming paradigms define how we structure and organize code.
- Two common paradigms are **Procedural Programming (PP)** and **Object-Oriented Programming (OOP)**.
- Understanding their differences helps in choosing the right approach for a project.

What is Procedural Programming?

- **Definition:**
  Procedural Programming follows a **linear** step-by-step approach, using functions and procedures to operate on data.
- **Key Features:**
- Focuses on **functions** and **procedures**.
- Code execution follows a **top-to-bottom** structure.
- Data is **separate** from functions and can be accessed directly.
- **Examples:**
- C, Pascal, Fortran

What is Object-Oriented Programming?

- **Definition:**
  Object-Oriented Programming (OOP) is based on the concept of **objects**, which encapsulate data and behavior.
- **Key Features:**
- Organizes code into **classes** and **objects**.
- Promotes **encapsulation**, **inheritance**, and **polymorphism**.
- Data is **hidden** within objects and accessed via methods.
- **Examples:**
  - Java, C++, Python (supports both PP and OOP)

Procedural Programming in C

```
#include <stdio.h>

// Function to display a message
void greet() {
    printf("Hello, World!\n");
}

int main() {
    greet();
    return 0;
}
```

Object-Oriented Programming in C++

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

class Greeter {
public:
    void greet() {
        cout << "Hello, World!" << endl;
    }
};

int main() {
    Greeter obj;
    obj.greet();
    return 0;
}
```

Advantages & Disadvantages

Paradigm	Advantages	Disadvantages
C (PP)	Simple, fast	Hard to manage large projects
C++ (OOP)	Reusable, secure	More complex

Conclusion

- **C (Procedural Programming)** is best for **simple tasks and system-level programming**.

- **C++ (Object-Oriented Programming)** is better for **large, complex applications**.

Thank You! 😊

- Any questions?