Programming Paradigms

introduction

In this walkthrough, we will explore in brief various standard techniques used in programming. We will see various advantages and critical features of one over the other.

IMPerative PRogramming

Imperative programming is one of the oldest created paradigms. It uses statements to change the program state. It focuses on how a program operates.

The major problem encountered in such a paradigm was regarding code reusability and unsafe nature of goto statements for implementing iterative blocks of code.

PROCEDURAL programming

This paradigm solves the problems that were encountered in the imperative programming. It involves the use of functions/procedures/subroutines that help in repeating the same series of computational steps to be carried out without repeating the code everytime it is reused. In this paradigm, all programs can be seen as composed of the following control-structures

1. sequential - ordered statements executed in a sequence
2. selective - execute a selected block of statements based on the state of a program (Eg. if then else...)
3. iterative - execute a block repeatedly till some state is reached (Eg. while/for loop...)
4. recursion - a statement is executed by repeatedly calling itself until some termination conditions are met

functional programming

This paradigm evolves over the previous procedural programming in the use of what is known as pure functions. Pure functions are those that do not use any state from other entities. Every function is a self-sufficient entity that just performs a complex transformation on given inputs to produce outputs. It doesn't modify the input and is practically isolated from rest of the program.

The need of this pure functions arose from the fact that impure functions lead to various problems. Since they don't produce the same output on the same input, it makes the code harder for others to understand, debug and extend.

Besides solving this problem, functional programming has various advantages.

1. Lazy Evaluations: Since pure functions directly operate on input and behave like a transformation, one can actually remember transformations and apply them only when needed. So, when an input goes through series of pure functions sequentially, one can evaluate all those later only when the result is asked for.
2. Parallelization: Pure functions can be easily parallelized as they are independent of each other.

Object-oriented Programming

Object-oriented programming (OOP) is a programming paradigm based on a concept of entities (objects), which contains data (member variables) and behaviors (member functions). In OOP, computer programs are designed by making them out of objects interacting with one another. Every behavior of an object can possibly change the state (member variables) of that object.

OOP has a lot of advantages

1. Code reusability - Features like classes and inheritance make a lot of code reusable
2. Encapsulation - Objects have abilities to hide a certain part of themselves from programmers. This prevents programmers tampering with things they shouldn't. Also, classes are designed to group related data implementing encapsulation.
3. Intuitiveness and Easy Design - It allows programmers to intuitively create large programs in a structured fashion.
4. Extensible - Code in OOP is readable and much easier to maintain and modify compared to other paradigms

Which ONE to use?

Now that we have understood what each major programming paradigm is, we can reason about which paradigm is better. Clearly, the only choices we have are functional and object-oriented programming. Both give way to a completely different style of programming and have their own advantages. We need not choose one over the other i.e one can employ an OOP program with various FP concepts.

FP can be advantageous in cases when we have a fixed set of *things*and as code evolves we add new *operations*. OOP can be advantageous when we have fixed set of *operations* on *things* and we add new *things* as code evolves.

conclusion

In this abstract, we have briefly touched upon various major paradigms in existence. We have learned the evolution of programming paradigms due to various necessities and faults in prior models. FP and OOP are the major paradigms in use today and should be used according to the specific problems encountered. We have seen various advantages of FP and OOP and thus need to learn these paradigms to design elegant and efficient programs.