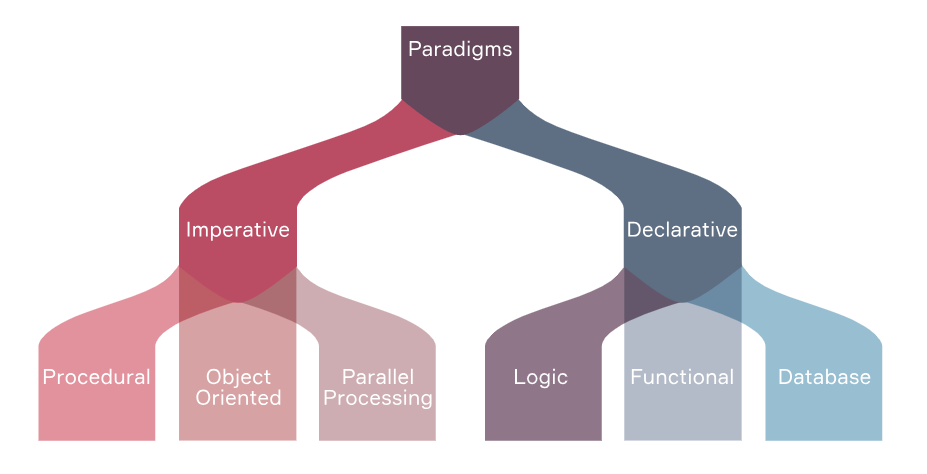
**Paradigms**

Different programming tasks can be solved in different ways: you might need to write a function, or create a separate class with methods, and so on. All such variants are combined into different programming approaches, which are also called paradigms. Below we will review the concept of paradigms and we will analyze two main paradigms: **imperative** and **declarative** and their types.



Almost all modern languages are multi-paradigm. They easily combine the capabilities of the imperative and declarative approaches. Nevertheless, in the section about each paradigm, we will give a list of languages where, among other paradigms, the one discussed in the section is also implemented.

1. **Imperative paradigm**

The**imperative paradigm** is one of the oldest programming paradigms. It is closely related to machine architecture. An imperative program is similar to the orders expressed by the imperative mood in natural languages. It is a sequence of instructions that the processor must execute step by step. The main focus of this paradigm is on how to achieve a goal. The paradigm consists of several statements, and after executing them all, the result is stored or displayed.

For example, you want to display the phrase "Hello, <username>!" on the screen. How should the program do it? Through the following steps:

* ask for the username
* read and remember the username
* display the result

Imperative programming is divided into three broad categories: **Procedural programming paradigm**, **Object-oriented programming**,and the**Parallel processing approach**. Let's talk about them in more detail.

1. **Procedural programming paradigm**

The**procedural programming paradigm**is based upon the concept of procedure calls, in which statements are structured into procedures also known as subroutines or functions. They are a list of instructions to tell the computer what to do step by step. In other words, the computer takes input data and processes it sequentially, remembering each new change.

Procedural Programming is suitable for general-purpose programming to complete common tasks. So, this can be a small computational problem, such as calculating a factorial, or finding the area of a figure, or displaying some information/phrase like "Hello, world!". Also, the code can be reused in different parts of the program, without the need to copy it.

Programming languages that have implemented the Procedural Programming paradigm are C, Java, C++, Kotlin, ColdFusion, and Pascal.

1. **Object-oriented programming**

**Object-oriented programming**or **OOP** is the paradigm where the program is written as a collection of classes. Each class has instances called objects.

A class is a way of describing an entity in general, defining the usual state and behavior that depends on that state, as well as the usual rules for interacting with this entity. Formally, a class is viewed as a set of data like fields, attributes, class members, and functions, i.e. methods for working with them.

For example, we have an entity cat and we want to describe it using a class. So, the cat will be an object of the corresponding Cat class. A cat has some attributes, for example, a tail, paws, claws, muzzle, ears, and whiskers. A cat's behavior is what it usually does, for example, it can run, jump, meow, eat, and rip off the wallpaper. All of these will be cat methods.

OOP can handle almost all kinds of common real-life problems where you need to model typical objects and work with them.

Programming languages that have implemented the OOP paradigm are Ruby, Kotlin, Java, C++, Python, Simula (the first OOP language), Smalltalk, Visual Basic .NET, and Objective-C.

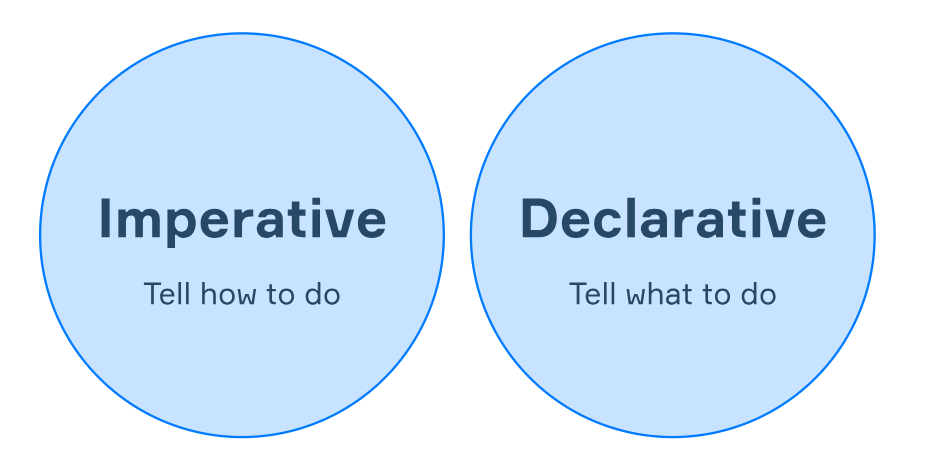
You could notice some overlap between lists of languages for procedural paradigm and OOP. This doesn't mean that all OOP languages are procedural and vice versa. It just as been stated before, most of modern programming languages can work with multiple paradigms.

1. **Parallel processing approach**

**Parallel processing** helps reduce instruction execution time. It does this by sharing or parallelizing instructions across multiple processors. The meaning of the approach can be summarized in one phrase: "divide and conquer". Examples of programming languages that support parallel processing are NESL (one of the oldest) and C / C++ (also supported due to some library functions)

1. **Declarative paradigm**

**Declarative programming** is a programming paradigm in which it is important to specify the problem and the expected result of its solution. That is, in contrast to the imperative paradigm, where it is necessary to answer the question "How to do this?" you need to ask the questions "What needs to be done?" and "What will be the result of the work?". So, rather than providing step-by-step instructions, you tell the system what you need and let it try to come up with a solution.



Declarative programming is divided into Logic, Functional, and Database paradigm types. We will describe them below.

1. **Logic programming paradigm**

**Logic programming** is a programming paradigm that is heavily based on formal logic. Any program written in a logical programming language is a set of sentences in a logical form that express facts and rules about a certain problem area.

So, the basic statements of logic programming are as follows:

* **Facts** are fundamental assertions about the problem domain, like "Socrates is a man."
* **Rules** allow to make inferences about the facts in the domain ("All men are mortal.")
* **Queries** are questions about that domain ("Is Socrates mortal?")

In general, the task here is to find the answer to the query based on facts and rules.

The major families of logic programming languages include Prolog, Answer Set Programming (ASP), and Datalog.

1. **Functional programming paradigm**

**Functional programming** is a programming paradigm in which the computation process is interpreted as the computation of the values of functions. The function, in this case, is similar to a mathematical one. That is, a function in which the input is an array that is not changed, and the output is a new array with new data. This makes a mathematical function different from a function in procedural programming, where a function is a sequence of actions that change the original data.

Here's a simple example: you might have a function that takes a list of numbers as input and returns a new list with the squares of those numbers. This does not change the original list of numbers.

Programming languages that have implemented the Functional programming paradigm are JavaScript, Haskell, Scala, Erlang, Lisp, ML, Clojure, OCaml, and F#.

1. **Database programming paradigm**

This programming methodology is based on working with data. The data is stored in the database and queries are made to this database in a special language, for example, SQL. With these languages, you can access the data for filtering, transformations, calculating statistics, and so on. Program statements are data-defined rather than a hard-coded series of steps.

The database program is the heart of the business information system, allowing for file creation, data entry, updating, querying, and reporting functions.

1. **Conclusion**

To sum up,

* Different approaches to creating programs are called paradigms.
* There are two main programming paradigms: imperative and declarative.
* The imperative paradigm focuses on achieving a result using step-by-step instructions that process the data sequentially.
* Imperative paradigm includes Procedural programming paradigm, Object-oriented programming, and Parallel processing approach.
* The declarative paradigm focuses on the task and tries to get an expected result.
* Declarative paradigm includes Logic, Functional, and Database paradigms.

You should remember that most modern programming languages can work with multiple paradigms. In the sections above we listed languages that are commonly associated with some paradigm. But this doesn't mean that they can only be used with one certain paradigm.