Functional programming

In [computer science](https://en.wikipedia.org/wiki/Computer_science), **functional programming** is a [programming paradigm](https://en.wikipedia.org/wiki/Programming_paradigm)—a style of building the structure and elements of [computer programs](https://en.wikipedia.org/wiki/Computer_program)—that treats [computation](https://en.wikipedia.org/wiki/Computation) as the evaluation of [mathematical functions](https://en.wikipedia.org/wiki/Function_(mathematics)) and avoids changing-[state](https://en.wikipedia.org/wiki/Program_state) and [mutable](https://en.wikipedia.org/wiki/Immutable_object) data. It is a [declarative programming](https://en.wikipedia.org/wiki/Declarative_programming) paradigm, which means programming is done with [expressions](https://en.wikipedia.org/wiki/Expression_(computer_science)) or declarations instead of [statements](https://en.wikipedia.org/wiki/Statement_(computer_science)). In functional code, the output value of a function depends only on the [arguments](https://en.wikipedia.org/wiki/Function_argument) that are passed to the function, so calling a function *f* twice with the same value for an argument *x* produces the same result *f(x)* each time; this is in contrast to [procedures](https://en.wikipedia.org/wiki/Subroutine) depending on a [local](https://en.wikipedia.org/wiki/Local_state) or [global state](https://en.wikipedia.org/wiki/Global_variable), which may produce different results at different times when called with the same arguments but a different program state. Eliminating [side effects](https://en.wikipedia.org/wiki/Side_effect_(computer_science)), i.e., changes in state that do not depend on the function inputs, can make it much easier to understand and predict the behavior of a program, which is one of the key motivations for the development of functional programming.

Functional programming has its origins in [lambda calculus](https://en.wikipedia.org/wiki/Lambda_calculus), a [formal system](https://en.wikipedia.org/wiki/Formal_system) developed in the 1930s to investigate [computability](https://en.wikipedia.org/wiki/Computability), the [Entscheidungsproblem](https://en.wikipedia.org/wiki/Entscheidungsproblem" \o "Entscheidungsproblem), function definition, [function application](https://en.wikipedia.org/wiki/Function_application), and [recursion](https://en.wikipedia.org/wiki/Recursion). Many functional [programming languages](https://en.wikipedia.org/wiki/Programming_language) can be viewed as elaborations on the lambda calculus. Another well-known declarative programming paradigm, [*logic programming*](https://en.wikipedia.org/wiki/Logic_programming), is based on [relations](https://en.wikipedia.org/wiki/Relation_(logic)).

In contrast, [imperative programming](https://en.wikipedia.org/wiki/Imperative_programming) changes state with commands in the [source code](https://en.wikipedia.org/wiki/Source_code), the simplest example being [assignment](https://en.wikipedia.org/wiki/Assignment_(computer_science)). Imperative programming does have functions—not in the mathematical sense—but in the sense of [subroutines](https://en.wikipedia.org/wiki/Subroutine). They can have [side effects](https://en.wikipedia.org/wiki/Side_effect_(computer_science)) that may change the value of program state. Functions without [return values](https://en.wikipedia.org/wiki/Return_value) therefore make sense. Because of this, they lack [referential transparency](https://en.wikipedia.org/wiki/Referential_transparency_(computer_science)), i.e., the same language expression can result in different values at different times depending on the state of the executing program.

Functional programming languages have largely been emphasized in [academia](https://en.wikipedia.org/wiki/Academic) rather than in commercial software development. However, prominent programming languages that support functional programming such as [Common Lisp](https://en.wikipedia.org/wiki/Common_Lisp), [Scheme](https://en.wikipedia.org/wiki/Scheme_(programming_language)), [Clojure](https://en.wikipedia.org/wiki/Clojure), [Wolfram Language](https://en.wikipedia.org/wiki/Wolfram_Language) (also known as [Mathematica](https://en.wikipedia.org/wiki/Mathematica" \o "Mathematica)), [Racket](https://en.wikipedia.org/wiki/Racket_(programming_language)), [Erlang](https://en.wikipedia.org/wiki/Erlang_(programming_language)" \o "Erlang (programming language)), [OCaml](https://en.wikipedia.org/wiki/OCaml), [Haskell](https://en.wikipedia.org/wiki/Haskell_(programming_language)), and [F#](https://en.wikipedia.org/wiki/F_Sharp_(programming_language)" \o "F Sharp (programming language))have been used in industrial and commercial applications by a wide variety of organizations. [JavaScript](https://en.wikipedia.org/wiki/JavaScript), one of the world's most widely distributed languages, has the properties of an untyped functional language, in addition to imperative and object-oriented paradigms. Functional programming is also supported in some [domain-specific programming languages](https://en.wikipedia.org/wiki/Domain-specific_programming_language) like [R](https://en.wikipedia.org/wiki/R_(programming_language)) (statistics), [J](https://en.wikipedia.org/wiki/J_(programming_language)), [K](https://en.wikipedia.org/wiki/K_(programming_language)) and [Q from Kx Systems](https://en.wikipedia.org/wiki/Q_(programming_language_from_Kx_Systems)) (financial analysis), [XQuery](https://en.wikipedia.org/wiki/XQuery)/[XSLT](https://en.wikipedia.org/wiki/XSLT) ([XML](https://en.wikipedia.org/wiki/XML)), and [Opal](https://en.wikipedia.org/wiki/Opal_(programming_language)). Widespread domain-specific declarative languages like [SQL](https://en.wikipedia.org/wiki/SQL) and [Lex](https://en.wikipedia.org/wiki/Lex_(software)" \o "Lex (software))/[Yacc](https://en.wikipedia.org/wiki/Yacc" \o "Yacc) use some elements of functional programming, especially in eschewing [mutable values](https://en.wikipedia.org/wiki/Mutable_object).

Programming in a functional style can also be accomplished in languages that are not specifically designed for functional programming. For example, the imperative [Perl](https://en.wikipedia.org/wiki/Perl) programming language has been the subject of a book describing how to apply functional programming concepts. This is also true of the [PHP](https://en.wikipedia.org/wiki/PHP) programming language. [C++11](https://en.wikipedia.org/wiki/C%2B%2B11), [Java 8](https://en.wikipedia.org/wiki/Java_8), and [C# 3.0](https://en.wikipedia.org/wiki/C_Sharp_3.0) all added constructs to facilitate the functional style. The [Julia](https://en.wikipedia.org/wiki/Julia_(programming_language)) language also offers functional programming abilities. An interesting case is that of [Scala](https://en.wikipedia.org/wiki/Scala_(programming_language)" \o "Scala (programming language)) – it is frequently written in a functional style, but the presence of side effects and mutable state place it in a grey area between imperative and functional languages.

**Функциональное программирование**

В информатике функциональное программирование представляет собой парадигму программирования - стиль построения структуры и элементов компьютерных программ, который рассматривает вычисления как оценку математических функций и избегает изменяющихся состояний и изменяемых данных. Это парадигма декларативного программирования, что означает, что программирование выполняется с помощью выражений или деклараций вместо операторов. В функциональном коде выходное значение функции зависит только от аргументов, которые передаются функции, поэтому вызов функции f дважды с тем же значением для аргумента x производит один и тот же результат f (x) каждый раз; это контрастирует с процедурами, зависящими от локального или глобального состояния, которое может давать разные результаты в разное время при вызове с теми же аргументами, но в другом состоянии программы. Устранение побочных эффектов, то есть изменение состояния, которое не зависит от входных функций, может значительно облегчить понимание и прогнозирование поведения программы, что является одним из ключевых мотивов для разработки функционального программирования.

Функциональное программирование исходит из лямбда-исчисления, формальной системы, разработанной в 1930-х годах для исследования вычислимости, проблемы Entscheidungs, определения функции, применения функций и рекурсии. Многие языки функционального программирования можно рассматривать как разработки по лямбда-исчислению. Другая известная парадигма декларативного программирования, логическое программирование, основана на отношениях.

Напротив, императивное программирование меняет состояние с командами в исходном коде, самым простым примером является назначение. Императивное программирование имеет функции - не в математическом смысле, а в смысле подпрограмм. Они могут иметь побочные эффекты, которые могут изменить значение состояния программы. Поэтому функции без возвращаемых значений имеют смысл. Из-за этого им не хватает ссылочной прозрачности, т. Е. Одно и то же выражение языка может приводить к различным значениям в разное время в зависимости от состояния исполняемой программы.

Функциональные языки программирования в основном были подчеркнуты в академических кругах, а не в разработке коммерческого программного обеспечения. Тем не менее, известные языки программирования, которые поддерживают функциональное программирование, такие как Common Lisp, Scheme, Clojure, Wolfram Language (также известный как Mathematica), Racket, Erlang, OCaml, Haskell и F #, используются в промышленных и коммерческих приложениях в широком разнообразии организаций. JavaScript, один из самых распространенных в мире языков, обладает свойствами нетипизированного функционального языка, в дополнение к императивным и объектно-ориентированным парадигмам. Функциональное программирование также поддерживается на некоторых языках программирования, таких как R (статистика), J, K и Q из Kx Systems (финансовый анализ), XQuery / XSLT (XML) и Opal. Широко распространенные специфические для домена декларативные языки, такие как SQL и Lex / Yacc, используют некоторые элементы функционального программирования, особенно в предотвращении изменчивых значений.

Программирование в функциональном стиле также может быть выполнено на языках, которые специально не предназначены для функционального программирования. Например, императивный язык программирования Perl был предметом книги, описывающей, как применять концепции функционального программирования. Это также относится к языку программирования PHP. C ++ 11, Java 8 и C # 3.0 все добавленные конструкции для облегчения функционального стиля. Язык Julia также предлагает функциональные возможности программирования. Интересный случай - это Scala - он часто написан в функциональном стиле, но наличие побочных эффектов и изменяемое состояние помещают его в серой области между императивными и функциональными языками.