**Функционално програмиране**

Парадигмите за програмирането са набор от идеи и концепции, които определят стила на писане на компютърни програми. Това е начин на концептуализация, който определя организацията на изчисленията и структурирането на работата, извършвана от компютър.

Една програмна парадигма не е еднозначно дефинирана от език за програмиране. Почти всички съвременни езици за програмиране до известна степен позволяват използването на различни парадигми. Така в C, който не е обектно-ориентиран, е възможно да се работи в съответствие с принципите на обектно-ориентираното програмиране, въпреки че това е свързано с определени трудности. Функционалното програмиране може да се използва при работа на всеки императивен език, който има функции и т.н.

Освен това съществуващите парадигми често се пресичат помежду си в детайли (например модулно и обектно-ориентирано програмиране), така че е възможно да се намерят ситуации, в които различни автори използват имена от различни парадигми, докато всъщност говорят за едно и също явление.

Най-известните функционални езици за програмиране са:

Lisp (John McCarthy, 1958) и много от неговите разновидности, като най-известните са Scheme, Clojure и Common Lisp;

Erlang (Joe Armstrong, 1986) е функционален език с поддръжка на процеси, както и неговият пряк наследник Elixir;

APL е предшественикът на съвременните научни изчислителни среди като MATLAB;

ML (Robin Milner, 1979) и основните му разновидности Standard ML и OCaml;

F# е функционален език от семейството на ML за платформата .NET;

Scala е език на JVM платформа, който съчетава възможностите на функционалното и обектно-ориентираното програмиране;

Миранда (Дейвид Търнър, 1985) и неговият пряк наследник, чистият функционален език Haskell;

Nemerle е хибриден функционално-императивен език.

Все още не напълно функционалните първоначални версии както на Lisp, така и на APL имат специален принос за създаването и развитието на функционалното програмиране. По-късните версии на Lisp, като Scheme, както и различни варианти на APL, поддържат всички функции и концепции на функционален език.

По правило интересът към функционалните езици за програмиране, особено към чисто функционалните, е по-скоро научен, отколкото комерсиален. Въпреки това известни езици като Erlang, OCaml, Haskell, Scheme (след 1986 г.), както и специфичните R (статистика), Wolfram (символна математика), J и K (финансов анализ) и XSLT (XML) имат своя път в индустрията за комерсиално програмиране. Широко разпространените декларативни езици като SQL и Lex/Yacc съдържат някои елементи на функционално програмиране, като например неизползване на променливи. Езиците за електронни таблици също могат да се считат за функционални, тъй като клетките на електронните таблици съдържат набор от функции, които обикновено зависят само от други клетки, и ако искате да моделирате променливи, трябва да си прибягва до възможностите на императивен макро език.

**Източници:**

<https://ru.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F>

<https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5>

Александър Бинчев 12а