DOI 10.31891/2307-5732-2021-297-3-34-38
УДК 004.9

А. І. КОВАЛЬ, О. М. ЯШИНА, Г. І. РАДЕЛЬЧУК, Ю. В. ФОРКУН
Хмельницький національний університет

ПОРІВНЯННЯ ОБ'ЄКТНО-ОРІЄНТОВАНОЇ ТА ФУНКЦІЙНОЇ ПАРАДИГМ ПРОГРАМУВАННЯ У ПРОЕКТУВАННІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

У статті описано та досліджено дві парадигми проектування – об'єктно-орієнтовану та функційну. Проаналізовано доречність використання кожної з них з посиланням на їх відмінності та переваги для конкретних цілей. Незважаючи на змінні тенденції популярності цих парадигм, зробити висновок стосовно актуальності котроїсь з них залишається неможливим.

Хоча як об'єктно-орієнтоване програмування, так і функціональне програмування є суттєвими парадигмами, які мають одну і ту ж мету – розробити зрозумілі та безпомилкові програми, їх підходи різні. ООП дотримується імперативної моделі програмування, яка базується на наборі примітивів, яку надає мова програмування. Функціональна парадигма, навпаки, тісно пов'язана з декларативним стилем, що означає, що визначається лише те, що потрібно виконати, не вказуючи, як це зробити.

Здається, загальний консенсус полягає в тому, що об'єктно-орієнтована парадигма та функціональна парадигма ефективні в будь-якій певній ситуації, тому розробники завжди повинні вибрати парадигму програмування, яка робить процес продуктивним та простим.

Об'єктно-орієнтовані мови хороші, коли є фіксований набір операцій над речами, і коли код розвивається, в першу чергу додаються нові речі. Цього можна досягти, додавши нові класи, що реалізують існуючі методи, а існуючі класи залишаються самі.

Функціональні мови хороші, коли є фіксований набір речей, і коли код розвивається, в першу чергу додаються нові операції над існуючими речами. Цього можна досягти, додавши нові функції, які обчислюються з існуючими типами даних, а існуючі функції залишаються в спокої.

Об'єктно-орієнтоване програмування, так само як і функційне, має своє місце у сучасній розробці програмного забезпечення. В той час, як розвиток технологій Big Data дав новий поштовх для використання функційного програмування, об'єктно-орієнтоване програмування, у свою чергу, залишається актуальним задля роботи для відображення в коді об'єктів реального світу.

Ключові слова: парадигма програмування, об'єкт, функція, об'єктно-орієнтоване програмування, функційне програмування.

A. I. KOVAL, O. M. YASHYNA,
G. I. RADELCHUK, Y. V. FORKUN
Khmelnytskyi National University

COMPARISON OF OBJECT-ORIENTED AND FUNCTIONAL PROGRAMMING PARADIGMS IN SOFTWARE DESIGN

This article describes two types of paradigms – object-oriented and functional paradigms. Paradigm stands for a style and an approach to perform any kind of coding activities. Relevance of usage for each of them as well as their differences and benefits were analyzed. Regardless of ever-changing tendencies in popularity of both paradigms it is impossible to acknowledge any of them to deprecate another one. Although object-oriented programming as well as functional programming are essential in their approaches they have the same goal – to make comprehensive programs without possible mistakes. OOP follows imperative programming model which is based on a set of primitives the given language provides. Functional paradigm is intertwined with declarative style which imply what is to be done, but not how to do it. We may come to conclusion that the consensus is that object oriented paradigm and functional paradigm can be effective in a peculiar situation. Therefore, developers are to choose and pick the programming paradigm for a given task to make a process as simple and productive as it can be.

Object-oriented programming languages are a good choice when you have a fixed set of operations on things and you add new things for your code to evolve. You can achieve it by adding new classes that implement existing methods while existing classes remain the same. Functional languages may be a better choice if you have a fixed set of things and you add new operations on existing things for your code to evolve. You can achieve it by adding new functions which are to be computed with existing data types while existing functions remain the same. Object-oriented programming as well as functional programming has its place in modern approaches to software development. Meanwhile the functional programming is being reconsidered and used much more often due to Big Data Technologies while object-oriented programming remains popular to perform representation of real-life objects in the code.

Keywords: programming paradigm, object, function, object-oriented programming, functional programming.

Вступ

Парадигма програмування – це стиль або «спосіб» програмування. Іншими словами – це система ідей і понять, які визначають стиль написання комп'ютерних програм, а також спосіб мислення програміста [1]. Парадигми програмування відрізняються одна від одної залежно від особливостей та стилю, який вони підтримують. Існує декілька особливостей, що визначають парадигму програмування, а саме: модульність, об'єкти, переривання або події, керування потоком тощо. Кожна парадигма програмування має власні переваги, про які варто знати перед тим, як її використовувати.

Дві найпопулярніші парадигми програмування при розробці програмного забезпечення (ПЗ) – це об'єктно-орієнтоване програмування (ООП) та функційно-орієнтоване програмування (ФОП). Саме ці дві парадигми найчастіше використовують розробники, дизайнери та проектувальники при розробці програмного забезпечення.

Об'єктно-орієнтовані мови хороші, коли у розробника є фіксований набір операцій; і коли програмний код розширюється, він у першу чергу додає нові речі. Цього можна досягти, додавши нові класи, які реалізують існуючі методи.

Функційні мови хороші, коли у розробника є фіксований набір речей; і коли код розвивається, він у першу чергу додає нові операції над існуючими речами. Цього можна досягти, додавши нові функції, які обчислюються з існуючими типами даних. Також є можливим використання обох парадигм (відповідно до власних потреб) за допомогою мультипарадигмових мов програмування, які підтримують як об'єктно-орієнтовану концепцію, так і функційну.

Метою статті є дослідження і порівняльний аналіз особливостей об'єктно-орієнтованої та функційної парадигм програмування на предмет їх ефективності та зручності використання при проектуванні ПЗ.

Виклад основного матеріалу

Об’єктно-орієнтоване програмування. Об’єктно-орієнтоване програмування – одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція. Відповідно до парадигми ООП кожен об'єкт здатний отримувати повідомлення, обробляти дані та надсилати повідомлення іншим об'єктам. Кожен об'єкт – це своєрідний незалежний автомат з окремим призначенням та відповідальністю.

Парадигма ООП використовує об’єкти для репрезентації речей у програмі (до прикладу, структури даних). Інколи ці об’єкти можуть бути і реальними речами. Кожний об’єкт має атрибути, що містять дані, якими можна маніпулювати за допомогою властивих йому методів чи функцій. Наприклад, існує об’єкт під назвою «**Особа**», який має різні атрибути людини, такі як вага, зріст, колір шкіри, колір волосся тощо. Окрім атрибутів, об’єкт також має функції. Наприклад, об’єкт «**Людина**» може мати такі функції, як їжа, сон, прогулянка тощо. Ці функції використовують дані, які об’єкт зберігає як атрибути.

Об'єктно-орієнтоване проектування – це стиль програмування, який допомагає програмістам моделювати реальні сценарії, а отже, при ООП існує прямий перехід від реальних об’єктів до фактичного коду. Деякі приклади мов програмування, які використовуються в ООП, – C++, Java, Python, C # та ін. [2].

Функційне програмування. Функційне програмування – парадигма програмування, яка розглядає програму як обчислення математичних функцій та уникає станів і змінних даних. Іншими словами, функційне програмування є способом створення програм, в яких єдиною дією є виклик функції, єдиним способом розбиття програми є створення нового імені функції та задання для цього імені виразу, що обчислює значення функції, а єдиним правилом композиції є оператор суперпозиції функцій.

Важливим поняттям у функційному програмуванні є референційна прозорість, яка означає, що для заданої функції та вхідного значення вона поверне той самий результат, незалежно від порядку програми (тобто незалежно від того, коли і з якої точки програми вона викликається) [3]. У функційному програмуванні вихід функції повністю покладається на аргументи функції. Так, наприклад, якщо викликається функція Sum (), яка обчислює суму двох змінних як вхідних даних і повертає цю суму, то вихідний результат завжди буде однаковим. Отже, у функційно-орієнтованому програмуванні функції програми є максимально передбачуваними.

У цій парадигмі програмування є невеликі функції, які виконують лише свою частину, а отже, код у програмуванні, орієнтованому на функції, є модульним і чистим. Також до програм, написаних у межах ФОП, доволі легко застосувати модульне тестування.

До функційно-орієнтованих мов програмування відносять мови Lisp, Haskell, Clojure, F # та деякі інші.

Недоліки об’єктно-орієнтованої та функційної парадигм програмування. Перша проблема для об'єктно-орієнтованого програмування полягає в тому, що деякі функції залежать від свого класу, і, отже, важко використовувати ці функції з іншим класом [4]. Також відомо, що об’єктно орієнтоване програмування є менш ефективним, але складнішим для роботи програмістів.

З іншого боку, є низка недоліків, пов’язаних із функційним програмуванням. Перш за все, функційно-орієнтоване програмування не є загальним методом. Програмістам легше мислити з точки зору об'єктно-орієнтованого програмування порівняно з ФОП. Об'єктно-орієнтоване програмування полягає в моделюванні об'єктів реального життя в коді, з іншого боку, маніпулювання даними є найважливішим у функційному програмуванні. Рівень складності функційно-орієнтованого програмування гарантує, що небагато програмістів використовують цей стиль програмування. Це також означає, що стає менше розробників, які вносять свій внесок у спільноту, а отже, менше інформації та контенту про функційно-орієнтоване програмування.

Відмінності об’єктно-орієнтованого та функційного програмування. В обох парадигмах програмування кінцевою метою є створення програм та уможливлення легкої і швидкої розробки з мінімальною кількістю помилок. Проте для спільних цілей ці дві парадигми застосовують різні методи зберігання та обробки даних. В об'єктно-орієнтованому програмуванні дані зберігаються в атрибутах об'єктів, і маніпулювання ними здійснюється за допомогою функцій об'єкта. Функційне ж програмування полягає у перетворенні даних шляхом створення нових версій цих даних та маніпулювання ними.

Фундаментальною є різниця концепцій між двома парадигмами. У функційно-орієнтованому дизайні функція програми є найбільш суттєвою. З іншого боку, в об'єктно-орієнтованій програмі основна

увага приділяється даним та їх маніпулюванню, а не функціям. Об'єкти в об'єктно-орієнтованому програмуванні – це незалежні сутності, які зберігають усі стани та можуть швидко змінюватися.

Ще одна відмінність між парадигмами програмування полягає в тому, що функційно-орієнтоване програмування використовує ітеративну процедурну декомпозицію, що є стратегією зверху вниз [5] Функційно-орієнтована програма трактується як ієрархія зростаючих рівнів деталей, де існує першочерговий опис функції, і з кожним наступним етапом він уточнюється. Це означає, що програма розробляється, починаючи від висококонцептуальної моделі, а далі – переходячи до деталей нижчого рівня. Цей процес повторюється до тих пір, поки не буде досягнуто рівень атомарності окремої функції.

З іншого боку, ООП більше пов’язане з реальними речами, які переносяться на об’єкти в об’єктно-орієнтованому програмуванні. Кожен об'єкт має характеристики з точки зору його атрибутів та поведінки. Ці об'єкти можуть бути розподілені, і, отже, вони виконуються послідовно або паралельно [6].

Ще однією темою для обговорення є абстракція. У ФОП абстракція є функціями реального світу, тоді як в ООП абстракція даних – це сутності реального світу. Крім того, у функційно-орієнтованому програмуванні функції групуються разом, за допомогою чого отримується функція вищого рівня, тоді як в об'єктно-орієнтованому, функції групуються разом на основі своїх даних, а класи асоціюються з їх методами. У ФОП інформація про стан представлена в централізованій спільній пам'яті, а в об'єктно-орієнтованому програмуванні інформація про стан реалізується або розподіляється між об'єктами.

Поряд з усіма іншими відмінностями ще однією різницею між обома стилями програмування є їх використання. Функційно-орієнтоване програмування є популярним в обчислювальних додатках, тоді як об'єктно-орієнтоване програмування використовується в системі, що розвивається, імітує бізнес чи бізнес-кейси.

На основі результатів аналізу основні характеристики об’єктно-орієнтованої та функційної парадигм програмування зведені у порівняльну таблицю 1.

Таблиця 1

Порівняння основних характеристик об’єктно-орієнтованої та функційної парадигм програмування

Характеристика	Парадигма	
	Об’єктно-орієнтована	Функційна
Фокус	На понятті об’єктів	На оцінці функції
Дані	Використовує змінні дані	Використовує незмінні дані
Модель	Імперативна модель	Декларативна модель
Паралельне програмування	Не підтримує	Підтримує
Виконання	Оператори виконуються в певному порядку	Твердження може виконуватися в будь-якому порядку
Ітерація	В основному використовує цикли	Використовує рекурсію
Основний елемент	Об’єкти та методи	Функції та змінні

Тенденції використання об’єктно-орієнтованого та функційного програмування. Хоча функційному програмуванню і надають перевагу в аспекті паралельності та іммутабельності коду, твердження стосовно витіснення ним об’єктно-орієнтованої парадигми є невірним. Багаторазове використання коду, абстрагування даних, ефективне вирішення проблем та гнучкість поліморфізму досягаються лише за допомогою об’єктно-орієнтованого програмування. Що стосується системної безпеки, то функційне програмування перевершує об’єктно-орієнтоване програмування, особливо беручи до уваги, що великі компанії залежать від машинного навчання та штучного інтелекту у своїх бізнес-додатках. Неможливо переоцінити значення функційного програмування при написанні коду для розробки та тренування моделей машинного навчання.

Динаміка популярності об’єктно-орієнтованого та функційного програмування наочно продемонстровані на рис. 1 та 2 [7].

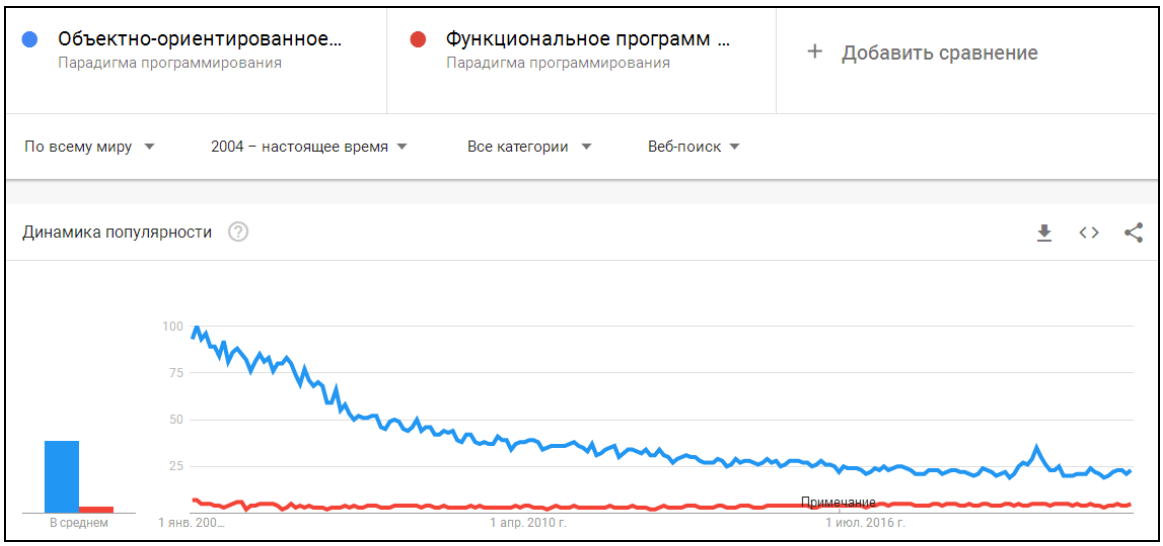

Рис. 1. Діаграма частоти Google-запитів за 2004 року по теперішній час [7]

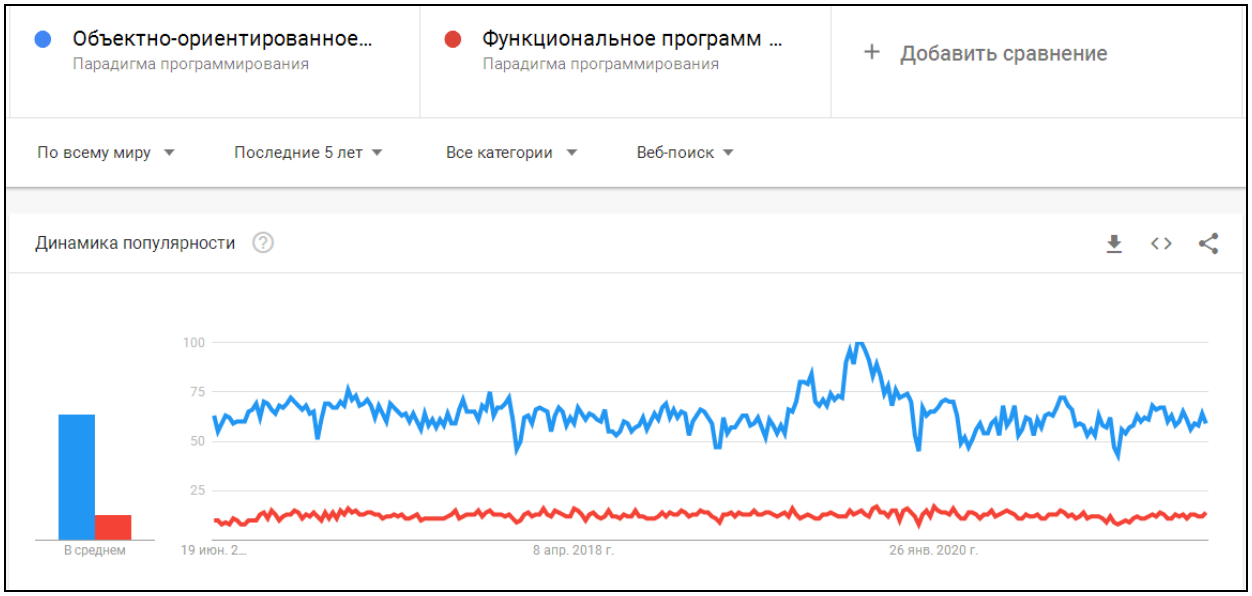

Рис. 2. Діаграма частоти Google-запитів за останні п’ять років [7]

Висновки

На основі опрацювання літератури та проведеного дослідження можна зробити наступні висновки.

Описані парадигми вимагають різних підходів до їх використання. Для вирішення проблем функційно-орієнтоване програмування використовує підхід зверху вниз, тоді як об’єктно-орієнтоване використовує підхід знизу догори. Ще один момент, на який слід звернути увагу, це те, що розробка за функційною парадигмою розпочинається з визначення сценаріїв та діаграм використання, а об’єктно-орієнтоване програмування розпочинається з визначення об’єктів і класів. Декомпозиція також відрізняється в обох парадигмах: у функціонально-орієнтованому програмуванні декомпозиція здійснюється на рівні функції/процедури, тоді як в об’єктно-орієнтованому програмуванні – на рівні класів.

Незважаючи на те, що функціональне та об'єктно-орієнтоване програмування є досить протилежними поняттями, вони переслідують одну й ту ж мету – створення простих для розуміння, без помилок та ефективних програм, – і обидва роблять свою роботу добре.

Хоча і ООП, і ФОП є двома важливими парадигмами програмування, які мають спільну мету створення зрозумілих, гнучких та працездатних програм, вони використовують два різні підходи до створення цих програм. ООП дотримується імперативної моделі програмування, що спирається на набір примітивів, які надає відповідна мова програмування. За допомогою ООП визначається, ЯК її потрібно впроваджувати, не вказуючи, ЩО потрібно виконати. З іншого боку, функціональне програмування тісно пов’язане з декларативним стилем програмування, який лише визначає, ЩО потрібно виконати, не вказучи, ЯК.

Отже, правильне питання полягає не в тому, який метод кращий, а в тому, який з них найкраще вирішує поставлену проблему. І навіть якщо використовується оптимальна парадигма, розроблювана програма ризикує «розвалитися», якщо вона стане занадто складною. Проблема полягає в тому, що досить важко уникнути складності у великих проектах, у яких бере участь багато людей. У таких випадках може знадобитися гібридний підхід, який об’єднує переваги обох парадигм, і, відповідно, мультипарадигмова мова програмування.

Література

1. Programming paradigm. Wikipedia. the free encyclopedia. URL: https://en.wikipedia.org/wiki/Programming_/paradigm

2. Гамма Э. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон, Дж. Влиссидес ; пер. с англ. – СПб : Питер, 2017. – 368 с.

3. Берд Р. Жемчужины проектирования алгоритмов. Функциональный подход. С примерами на языке Haskell / Р. Берд. – М. : Изд-во «ДМК-Пресс»? 2015. – 330 с.

4. Вайсфельд М. Объектно-ориентированное мышление / М. Вайсфельд ; пер. с англ. – СПб : Питер, 2018. – 304 с.

5. Окасаки К. Чисто функциональные структуры данных / К. Окасаки ; пер. с англ. – М. : Изд-во «ДМК-Пресс», 2016. – 252 с.

6. Буч Г. Объектно-ориентированный анализ и проектирование с примерами приложений / Г. Буч, Роберт, А. Максимчук и др. ; пер. с англ. – 3-е изд. – М. : ИД «Вильямс», 2020. – 720 с.

7. Динамика популярности объектно-ориентированного и функционального программирования [Электронный ресурс] // Google Trends. – Режим доступа : https://trends.google.ru/trends/explore?date=/today%205-y&q=%2Fm%2F05prj,%2Fm%2F02ykw

References

1. Programming paradigm. Wikipedia. the free encyclopedia. URL: https://en.wikipedia.org/-wiki/Programming_paradigm
2. Gamma E. Priemy obektno-orientirovannogo proektirovaniya. Patterny proektirovaniya / E. Gamma, R. Helm, R. Dzhonson, Dzh. Vlissides ; per. s angl. – SPb : Piter, 2017. – 368 s.
3. Berd R. Zhemchuzhiny proektirovaniya algoritmov. Funkcionalnyj podhod. S primerami na yazyke Haskell / R. Berd. – M. : Izd-vo «DMK-Press»? 2015. – 330 c.
4. Vajsfeld M. Obektno-orientirovannoe myshlenie / M. Vajsfeld ; per. s angl. – SPb : Piter, 2018. – 304 s.
5. Okasaki K. Chisto funkcionalnye struktury dannyh / K. Okasaki ; per. s angl. – M. : Izd-vo «DMK-Press», 2016. – 252 s.
6. Buch G. Obektno-orientirovannyj analiz i proektirovanie s primerami prilozhenij / G. Buch, Robert, A. Maksimchuk i dr. ; per. s angl. – 3-e izd. – M. : ID «Vilyams», 2020. – 720 s.
7. Dinamika populyarnosti obektno-orientirovannogo i funkcionalnogo programmirovaniya [Elektronnyj resurs] // Google Trends. – Rezhim dostupa : https://trends.google.ru/trends/explore?date=today%205-y&q=%2Fm%2F05prj,%2Fm%2F02ykw

А. І. КОВАЛЬ andrey1998t29@gmail.com
О. М. ЯШИНА ORCID ID: 0000-0001-7816-1662 ksusha.ja@gmail.com
Г. І. РАДЕЛЬЧУК gal_2015@ukr.net
Ю. В. ФОРКУН ORCID ID: 0000-0002-7906-4191 forkun@ridne.net

Рецензія/Peer review : 04.05.2021 р. Надрукована/Printed :30.06.2021 р.