DOI: https: doi.org 10.36910 6775-2524-0560-2024-57-09
УДК 004.4
**Кожаєв Володимир Вікторович,** аспірант
https: orcid.org 0009-0005-8941-6348
Луцький національний технічний університет, м. Луцьк, Україна

# ПОРІВНЯННЯ МОВ ПРОГРАМУВАННЯ НА ОСНОВІ ПАРАДИГМ: КІЛЬКІСНИЙ ПІДХІД ТА ЕКСПЕРИМЕНТАЛЬНІ РЕЗУЛЬТАТИ

**Кожаєв В.В. Порівняння мов програмування на основі парадигм: кількісний підхід та експериментальні результати.** Предметом вивчення в статті є властивості та відмінності мов програмування, важливість властивостей для розробників, поняття парадигми мови програмування її кількісного значення та обрахування кількісної різниці між мовами. Метою статті є розробка математичної моделі відстані між мовами програмування на базі введеного поняття парадигми мови програмування та її кількісного значення що важливе зокрема для добору мов для добору мов для диверсного програмування з властивостями що максимально не збігаються, створення бази для майбутніх досліджень кількісних показників мов програмування в залежності від їх властивостей Завдання: з'ясувати важливість окремих властивостей мов програмування за допомогою експертного опитування, ввести та обґрунтувати поняття парадигми мови програмування як кортежу властивостей впорядкованого за важливістю для прикладного розробника, отримати алгоритм однозначного відображення парадигм мов програмування на множину цілих чисел, обрахування дистанції між мовами програмування на основі введеного поняття парадигми мови програмування та її кількісного значення як модуль різниці кількісних значень парадигми. Використовуваними методами є:опитування експертів, аналіз специфікацій мов програмування, біекція множин. Отримані результати статті: сформовано поняття парадигми мови програмування та її кількісного значення на базі важливості окремих властивостей мов для розробника що з'ясовані за допомогою експертного опитування та дистанції між мовами як модуля різниці між кількісним значенням парадигм Висновки. Наукова новизна полягає в з'ясуванні важливості окремих властивостей мов програмування для прикладного розробника, формалізації поняття різниці між мовами програмування, введенні та обгрунтуванні поняття парадигми мови програмування та її кількісного значення, створенні алгоритму його обрахування, створенні бази для майбутніх досліджень зв'язку кількісного значення парадигми та швидкості роботи програм написаних на певній мові, трудоємності трансляції тексту програм на різних мовах з одної на іншу.

**Ключові слова:** диверсне програмування, ключові характеристики мов програмування, відстань між мовами, парадигми мов програмування, біокція

**Kozhaev V. A comparison of paradigm-based programming languages: A quantitative approach and experimental results.** The subject of study in the article is the properties and differences of programming languages, the importance of properties for developers, the concept of a programming language paradigm, its quantitative value, and the calculation of the quantitative difference between languages. The purpose of the article is to develop a mathematical model of the distance between programming languages based on the introduced concept of a programming language paradigm and its quantitative value, which is important in particular for the selection of languages for diverse programming with properties that do not coincide as much as possible, creating a basis for future research on quantitative indicators of programming languages depending on their properties. Tasks: to find out the importance of individual properties of programming languages using expert survey, to introduce and substantiate the concept of a programming language paradigm as a tuple of properties ordered by importance for the application developer, to obtain an algorithm for unambiguously mapping programming language paradigms to a set of integers, to calculate the distance between programming languages based on the introduced concept of a programming language paradigm and its quantitative value as the modulus of the difference in quantitative values of the paradigm. The methods used are: expert survey, analysis of programming language specifications, set bijection. The results of the article: the concept of a programming language paradigm and its quantitative value was formed based on the importance of individual properties of languages for the developer, which were clarified by means of an expert survey and the distance between languages as the module of the difference between the quantitative value of the paradigms. Conclusions. The scientific novelty lies in clarifying the importance of individual properties of programming languages for the application developer, formalizing the concept of the difference between programming languages, introducing and substantiating the concept of a programming language paradigm and its quantitative value, creating an algorithm for its calculation, creating a base for future research on the relationship between the quantitative value of the paradigm and the speed of programs written in a certain language, the laboriousness of translating the text of programs in different languages from one to another.

**Keywords:** diverse programming, key characteristics of programming languages, distance between languages, programming language paradigms, bijection

**Постановка наукової проблеми.** Дефекти програмного забезпечення (ПЗ) систем критичного застосування можуть призводити до збоїв та відмов. Наслідки таких інцидентів мають різні ступені важкості – від незначних порушень та фінансових втрат до техногенних катастроф та людських жертв. Наведемо кілька прикладів таких випадків.

3 лютого 2022 року український оператор мобільного зв'язку «Київстар» зазнав масштабного збою, користувачі скаржаться на відсутність зв'язку та списання коштів. За повідомленням користувачів, при спробі зателефонувати на інший номер з'являється повідомлення про брак коштів на балансі. Перевірити баланс при цьому не вдається, а після поповнення коштів повідомлення про нестачу балансу залишається.

У 2021 році стався збій Instagram, Facebook та WhatsApp у Росії, США, Німеччині, Італії, Іспанії, Великобританії та інших країнах. Проблеми спостерігалися протягом кількох годин, включаючи неможливість надсилання повідомлень, некоректне з'єднання з сервісом, неможливість відкрити сайти та сервіси тощо. Все це спричинило збитки в мільярди доларів США для глобального ринку. Причому точну причину збою так і не було ідентифіковано. Експерти говорили про проблеми з серверами Facebook, а також про проблеми в роботі глобальних провайдерів. Все це в результаті призвело не тільки до прямих збитків, а й до непрямих. Так, індекс біржі NASDAQ упав за підсумками торгів цього дня на 2,14%. Акції Facebook тоді впали більш ніж на 5%. Збій у роботі соціальної мережі став найсерйознішим та найтривалішим за 13 років її роботи. Усього за день було зафіксовано понад 10,6 млн. скарг на некоректну роботу Instagram, Facebook та WhatsApp. Це був найбільший системний збій банківської системи останнім часом. Британський банк TSB переходив на нову платформу, після чого відбувся системний збій. Незабаром стало відомо, що помилка при оновленні комп'ютерної системи призвела до аномально високої кількості шахрайських атак. Кіберзлочинці намагалися отримати доступ до рахунків понад 2200 клієнтів, і в результаті 1300 осіб втратили гроші. Суми, які втратили клієнти банку, досить високі – близько 10-20 тис. фунтів стерлінгів. У зв'язку із системною помилкою банку надійшло загалом 93 700 скарг. Проблеми почалися у банку після переходу з однієї комп'ютерної системи (яка контролювалася колишнім власником банку Lloyds Banking Group) на нову платформу, розроблену Sabadell. Нова система не впоралася з напливом клієнтів, і в результаті 1,9 млн людей на кілька днів втратили доступ до інтернет-банку. Незважаючи на всі спроби вирішити проблему, вони спостерігалися протягом кількох тижнів. У 2022 році стало відомо, що загальна сума збитків банку від збою зросла до $502 млн. Банк витратив $433 млн. на відновлення, плюс багато мільйонів було витрачено на компенсації клієнтам.

Окремим випадком техногенних катастроф є авіація, адже до звичайної небезпеки додається падіння й швидкість. Так 10 березня 2018 року літак Boeing 737 Max 8 авіакомпанії Ethiopian Airlines, що виконував рейс з Аддіс-Абеби до Найробі, зазнав катастрофи за 60 км на схід від столиці Ефіопії. Авіакатастрофа забрала життя 157 осіб, серед яких були громадяни 35 країн.

Ця катастрофа стала другою катастрофою літака цього типу за півроку - наприкінці жовтня 2018 року такий же «Боїнг» індонезійської компанії Lion Air впав у Яванське море невдовзі після вильоту з аеропорту Джакарти. Загинули усі 189 людей на борту.

Причиною масових затримок рейсів American Airlines 28 квітня став «глюк» в одній із навігаційних програм iPad. За допомогою програми командири екіпажів та їхні помічники мали прокласти маршрут, а також отримати інформацію про розрахунковий час руху борту. За даними ЗМІ, пристрій на якийсь час вийшов з ладу, що змусило відразу кількох пілотів скасувати зліт. Частині льотчиків вдалося усунути проблему, не залишаючи смуги. Інші були змушені піднятися у повітря із суттєвим запізненням. Весь цей час у соціальних мережах поширювалися повідомлення, в яких найпопулярнішою версією того, що відбувається, була чутка про несправність в одній із конкретних моделей авіалайнерів перевізника. Пізніше припущення було спростовано, проте понервувати пасажирам, що знаходилися на борту, довелося неабияк.

Помилка, виявлена трохи пізніше, в ході лабораторних тестів програмного забезпечення «Боїнгів-787», виявилася серйознішою за несподівано підведені пілоти планшетів Apple. Як повідомили представники Федерального управління цивільної авіації США (U.S. Federal Aviation Administration, FAA), код однієї із систем, що управляють лайнером, вимагав обов'язкового перезавантаження після 248 днів безперервної роботи. В іншому випадку рейс прямо в ході польоту ризикував перейти в так званий стійкий до відмови і повністю знеструмити сам себе. Живлення автоматично могли втратити дві пари генераторів, встановлених на двигунах повітряного судна, а також два запасні генератори.

Незважаючи на високу потенційну небезпеку бага, він не загрожує безпеці пасажирів у реальних умовах, стверджували співробітники FAA: під час експлуатації «Боїнгів» усі системи постійно перезавантажуються і ніколи не працюють поспіль 248 днів. Керівництво «Боїнг» також

поспішило заспокоїти пасажирів. У компанії наголосили, що всі літаки оснащені додатковою системою безпеки, яка дозволяє запускати двигуни в аварійному режимі у разі відмови живлення.

У коментарі для преси співробітник одного з підрозділів компанії в Сіетлі підтвердив, що баг має бути усунений найближчим часом і справді міг представляти хіба що теоретичну загрозу. За словами фахівця, в умовах експлуатації ВО «Боїнгів» перезавантажується набагато частіше за критичний термін.

Загальною причиною таких наслідків є недостатній рівень надійності програмного забезпечення. Одним із шляхів підвищення надійності ПЗ є підхід, заснований на диверсній розробці програмного забезпечення. Він полягає у застосуванні кількох мов програмування при розробленні ПЗ (як правило два) та подальшим арбітражем керуючих впливів для систем критичного застосування.

Застосування мультипрограмного підходу потребує вирішення завдання вибору (або відбору) мов програмування для подальшої диверсної розробки програмного забезпечення. Критерієм вибору може бути максимальна відстань між мовами програмування, тобто. принцип масового розмаїття. Виникає питання вибору мов для диверсного програмування. Існує гіпотеза, що мови найбільш відрізняються одна від одної. Таким чином, актуальна проблема відстані між мовами.

Сьогодні налічується понад 8 тисяч мов програмування. Це мови програмування як загального універсального застосування (C++, Java, Python), і спеціалізовані (Go). Вони мають певні ознаки подібності та відмінності. Проведемо аналіз існуючих робіт, в яких автори пропонували варіанти критеріїв відмінності мов програмування. З відкритих джерел відомо [1], що сьогодні існує більш ніж вісім тисяч мов програмування. Кожна з мов створювалася для своєї конкретної мети, відповідно підтримує різні можливості. Наприклад, дві мови можуть підтримувати функціональну парадигму, але одна з них має статичну, а інша – динамічну типізацію і таких комбінацій дуже багато, відповідно актуальною є задача чисельного порівняння мов програмування – визначення відстані між мовами. Ця відстань має безліч застосувань, зокрема оцінку надійності багатоверсійних програмних систем: що далі мови відстоять одна від одної, то вище ймовірність попередження помилок за допомогою багатоверсійності.

Одним із методів підвищення надійності є n-версійне програмування: розробляється кілька версій програми, що управляє, які працюють незалежно один від одного. Поступаючі від програм сигнали управління аналізуються програмою-арбітром, що видає початковий сигнал.
Так у [2] розглядається розробка ПЗ мовами Rust і Forth. У [3] аналізуються інші методи збільшення точності та надійності виведення. Одним із них є реалізація конкуруючих частин програми різними мовами.

У цьому версії ПЗ можуть оброблятися незалежно, але у разі вартість розробки кратно зростає. Іншим підходом є генерація коду різними мовами з будь-якої мови уявлення предметної області (так званий DSL). У цьому випадку мову подання предметної області можна перевірити на відповідність специфікації за допомогою певного мат. апарату, наприклад, інерційного моделювання [4]. На даний момент розроблено велику кількість мов програмування. Таким чином, виникає проблема вибору мов програмування для генерації коду. Існує гіпотезапро те, що різноманітність мов створить розмаїтість у дефектах ПЗ.

У [5] досліджується застосування методу блоків відновлення підвищення надійності програмного забезпечення. Блоки доцільно реалізовувати різними мовами програмування, з метою збільшити ймовірність коректного спрацювання блоку.

Крім того, існує завдання трансляції застарілих мов програмування на сучасні. І відповідно оцінка кількості роботи у годиннику. Очевидно, крім кваліфікації розробників оцінка залежить від того, наскільки мови, що транслюються, відрізняються один від одного. Так перенос із мови java 1.4 на 1.8 досягається тривіально, зміною версії компілятора, без додаткових дій. З іншого боку, перенесення з мови ProLog C# не має чіткого алгоритму, оскільки другий не підтримує логічного програмування. Таким чином, важлива оцінка відстані між мовами.

**Аналіз досліджень.** У [6] наводиться порівняння мов програмування з енергоефективності. В [7]розглядається різниця кількості помилок під час програмування за допомогою об'єктно-орієнтованого та функціонального підходу.

Вочевидь, що у існуючих роботах мови програмування порівнюються лише без дослідження більш детальних чинників . Слід зазначити, що у існуючих роботах відсутні метрики кількісної оцінки відстані між мовами программирования[7] десять мов програмування (C++, C#, Java, Groovy, JavaScript, PHP, Schalar, Scheme, Haskell та AspectJ.) порівнюються на основі наступних параметрів:

1. Безпечні практики програмування;
2. Бібліотеки для веб-програмування;
3. Створення та дизайн веб-сервісів;
4. ООП-дизайн;
5. Використання рефлексії;
6. Аспектно орієнтоване програмування;
7. Функціональне програмування;
8. Декларативне програмування;
9. Сценарії командної оболонки;
10. Прототипування інтерфейсу користувача.

У роботі [8] пропонується порівнювати мови за парадигмою, властивої мові програмування сукупності ідей та понять, що визначають стиль написання комп'ютерних програм (підхід до програмування), також автор висловлює ідею створення автоматизованої системи порівняння мов виходячи з онтологій, проте жодних практичних підходів до створення вищезазначеного кошти не вказується.

Однак у жодній роботі відомої автору немає інтегрального порівняння відстані між мовами. Немає відповіді на запитання: наскільки схожі, чи різні дві мови програмування загалом. Наприклад, деякі компілятори дозволяють створити з коду мовою java - бінарний код, що виконується безпосередньо операційною системою, як зроблено для мови Pascal. Чи означає це, що мова Java близька до Pascal? З іншого боку, програмуючи на J2ME (версія Java для мобільних пристроїв з низькою продуктивністю, була популярна до появи О.С. Android) доводиться працювати з пам'яттю на низькому рівні, у ряді випадків використовувати цілочисленну арифметику для моделювання обчислень з реальними числами і так далі. Тобто, методика роботи з мовою програмування J2ME в даному випадку ближче до програмування мовою Сі, ніж Java Enterprise Edition.

Під парадигмою розуміється сукупність ідей та понять, що визначають стиль написання комп'ютерних програм(підхід допрограмування). Це спосіб концептуалізації, Що визначає організацію обчислень та структурування роботи, що виконується комп'ютером.

Таким чином, **метою** цієї роботи є пропозиція підходу (або методики) порівняння мов програмування за парадигмою.

**Метою** є розробка математичної моделі відстані між мовами програмування.

Для її виконання поставлено та розв'язано наступні **завдання**:

- З'ясувати важливість властивостей мов програмування
- Ввести та обґрунтувати поняття парадигми мови програмування як кортежу властивостей впорядкованого за важливістю для розробника
- Отримати алгоритм однозначного відображення парадигм мов програмування на множину цілих чисел, обрахування дистанції між мовами програмування на основі введеного поняття парадигми мови програмування та її кількісного значення.

**Поняття парадигми мови програмування.** З вищевказаного найперспективнішим автору бачиться порівняння мови з парадигми, проте є така проблема: більшість сучасних мов програмування є мультипарадигменними, тобто підтримують кілька підходів до програмування. Так наприклад, мова Java є об'єктно-орієнтованою, проте підтримує процедурний та функціональний підхід.

Крім цього різні аспекти мов очевидно мають неоднакову цінність для прикладних програмістів. Як приклад наведемо енергоефективність. Енергоефективність - це очевидно що в більшості випадків вона має меншу значущість, ніж є мова об'єктно орієнтованою.

Представимо етапність пропонованого підходу. По-перше, сформувати набір ознак за якими можна чітко сказати, чи підтримує його ця мова програмування, чи ні: чи підтримує мову компіляцію в бінарний код, інтерпретацію, чи є віртуальна машина (список ознак буде запропонований нижче)

Оскільки важливість ознак для програмістов не є однозначною отсортуймо їх методом експертної оцінки. Вектор ознак(впорядкований за важливістю) ми назвемо **парадигмою мови програмування**.

Парадигме поставимо у відповідність бінарне число у кожному розряді якого буде нуль, якщо мова не підтримує зазначену властивість і одиниця якщо підтримує. Це число ми назвемо чисельним виразом парадигми

Якщо чисельні вирази парадигм рівні, очевидно мови близькі одна до одної, називатимемо їх у одній парадигмі.

Різницю між чисельними значеннями парадигм називатимемо різницею між мовами за парадигмою. Далі, оберемо ознаки, що формують парадигму. При виборі ознак автор пропонує керуватися наступними критеріями:

- однозначність. За кожною з ознак можна чітко визначити, чи підтримує його мову;
- значущість для прикладного програмування. Виконуване «під капотом» не має значення для практичного написання коду
- конкретність. Обговорюється лише стандарт мови. Звичайно, можливі нові реалізації, проте це буде вже інша мова в іншій парадигмі.

Таким чином, **парадигмою мови програмування називається кортеж, що характеризують його однозначних властивостей, відсортованих за важливістю.**

**Список ознак для експерименту.** Розглянемо запропоновані ознаки для формування парадигми, які є необхідними для виконання експерименту. **Спосіб виконання програми.** Мови поділяються на чотири типи: інтерпретатори, компілятори, що транслюються в байт-код виконуваний віртуальною машиною і транслюються в іншу мову програмування. Інтерпретацією називається рядковий аналіз, обробка та виконання вихідного коду програми чи запиту. Приклади Lua[9], GNU Lisp[10] Примітка, GNU Lisp є прикладом мультипарадигменного методом виконання мови, оскільки підтримує інтерпретацію і трансляцію в мову Сі.

Транслятори передають програмний код програму виконувану безпосередньо операційною системою. Приклад C++[11], Delphi[12]

Транслюються в бінарний код (так званий байт-код), який виконується спеціальною програмою званої «віртуальна машина». Приклади Java [ß13], C# [14]. Таким чином, переваги компілюваної мови поєднуються з кросплатформністю, оскільки віртуальні машини існують для всіх популярних операційних систем.

Найчастіше дизайн базової мови програмування не влаштовує розробника через різні причини. Однак обсяг розробленого коду базовою мовою і велика кількість існуючих бібліотек роблять перехід на нову мову дуже дорогою. У цих випадках розробляється нова мова з більш прогресивним дизайном, але транслюється в базову. Приклад XTend[14] і вже згадуваний GNU Common Lisp.

**Типи ООП.** Мови поділяються на три основних типи: без ООП зовсім (приклад, мова Сі), без ООП у класичному розумінні, але з можливістю його емуляції (Lua, JavaScript та класичні мови ООП. Приклад:вже згадуваний Java, C++ та Scala [15]. Різниця між мовами без ООП та з емуляцією ООП є значною мірою умовною, тому різниця між ними визначається значною мірою практикою застосування.

**Типізація.** Розглянемо типізацію. Як відомо тип даних це характеристика набору даних, що визначає діапазон його можливих значень, список допустимих операцій, а також спосіб зберігання набору даних. Типізація буває статичною (коли тип змінної або методу оголошується при оголошенні класу) та динамічною (коли тип фіксується при ініціалізації). Інакше кажучи, статична компіляція означає, що перевірка на сумісність типів виконується етапі компіляції перевірки помилок, динамічна - етапі виконання.

Також система може типів може бути слабкою та сильною. Система типів називається «сильною», якщо вона виключає можливість виникнення помилки узгодження типів часу виконання, інакше кажучи, забезпечує типо-безпеку (відсутність неконтрольованих помилок приведення типів часу виконання) лише на рівні мовами[14]. Відповідно мова може підтримувати:

Статична (Java, C++, C#), чи динамічна (Python[16] PHP[17]), типізація.Сильна(Java, Python) або слабка (JavaScript [18]). Проте практично більшість програмістів розрізняють підтримку статичної і динамічної типізації.

**Функціональне програмування.** Під функціональним програмуванням мається на увазі парадигма програмування[19], у якій процес обчислення сприймається як обчислення значень функцій у математичному розумінні останніх (на відміну функцій як підпрограм у процедурному програмуванні).Протиставляється парадигмі імперативного програмування, яка описує процес обчислень як послідовну зміну станів (у значенні, подібно до такого в теорії автоматів). При необхідності, у функціональному програмуванні вся сукупність послідовних станів обчислювального процесу представляється явно, наприклад, як список. Центральним поняттям у літературі про функціональні мови є чисті функції - тільки повертають значення, без побічних ефектів. Однак їх можна створити у всіх відомих автору мовах програмування, що підтримують функції в принципі. Тому пропонуються такі критерії приналежності до функціональної парадигми

1. Наявність функціонального типу: Java 8 і більше, TypeScript [19];
2. Наявність згорток над даними Scala, GNU Common Lisp.
3. Багатопотокове програмування. Наступним є багатопотокове програмування. По багатопоточності мови можна поділити на три типи
4. Не підтримують багатопоточність (проте з можливість емуляції багатопоточності, наприклад за подіями таймера) ActionScript 3]
5. Підтримуючі базові можливості багатопоточності (Java 5)
6. Розвинені бібліотеки роботи з багатопоточністю (Java 8 та старше)
7. Макрос – цесимвольне ім'я, що замінюється під час обробки препроцесором на послідовність програмних інструкційДілимо мови на макроси, що підтримують і не підтримують.

**Логічне програмування.** Парадигма програмування, заснована на математичній логіці полягає у тому, що програми у ній задаються у формі логічних тверджень та правил виведення. Найбільш відома мова логічного програмування ProLog [21]. Ділимо мови на підтримуючі та не підтримуючі та логічне програмування.

**Типи класифікації мов.** Узагальнено, представимо усі показники та їх складові у вигляді єдиної таблиці (табл. 1):

Таблиця 1. Класифікуючі ознаки мов програмування

| Назва ознаки | Можливі значення |
|---|---|
| Спосіб виконання програми, які підтримує мову | Підтримка компіляції |
| | Інтерпретація |
| | Трансляція в байт-код та виконання його віртуальною машиною |
| | Трансляція в іншу мову програмування |
| Вид типізації | Сильна статична |
| | Сильна динамічна |
| | Слабка статична |
| | Слабка динамічна |
| Підтримка ООП | Не підтримує |
| | Підтримує рахунок емуляції |
| | Підтримує повністю |
| Підтримка інтерфейсів | Не підтримує |
| | Підтримує |
| Функціональне програмування | Наявність функціонального типу |
| | Наявність операції згортки над структурами даних |
| Багатопоточність | Не підтримує |
| | Підтримує |
| Макроси | Не підтримує |
| | Підтримує |
| Логічне програмування | Не підтримує |
| | Підтримує |

**Подання методики та постановка експерименту.** Метою експерименту є визначення ваги ознак, що характеризують мову для обчислення парадигми. Опитування проводилося у спільнотах програмістів різних предметних областей: розробка web-додатків, компіляторів та трансляторів, ігор, автоматизації бізнес-процесів, мобільних додатків. Вагою мови вважатимемо моду розподілу. Експертам під час анонімного опитування було запропоновано проранжувати важливість властивостей мов програмування (за шкалою від 1 до 10). Властивості вибиралися однозначними, тобто по кожному можна однозначно сказати, чи підтримує його мову, чи ні. Далі властивості були відсортовані за ступенем важливості за допомогою експертного опитування і таким чином отримано оцінку ваги кожної властивості.

Опитування було анонімним, за допомогою заповнення гугл-форми. В опитуванні брало участь 63 програмісти з досвідом не менше п'яти років.

Результати опитування мають значення викладені на рисунках 1-13, в якості ваги відповідного аспекту парадигми було взято значення середньої арифметичної [27] округленої до натурального значення разряду.

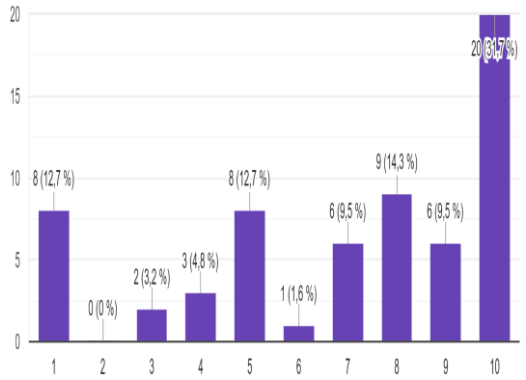


Рис 1. Підтримка компіляції

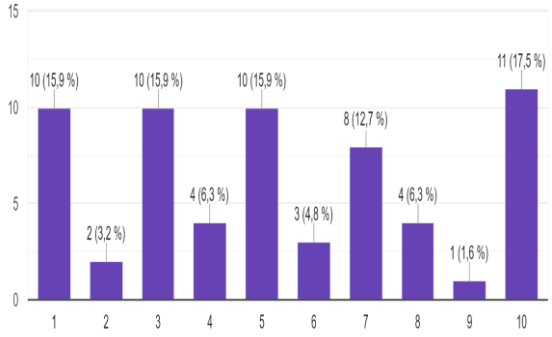


Рис 2. Підтримка інтерпретації

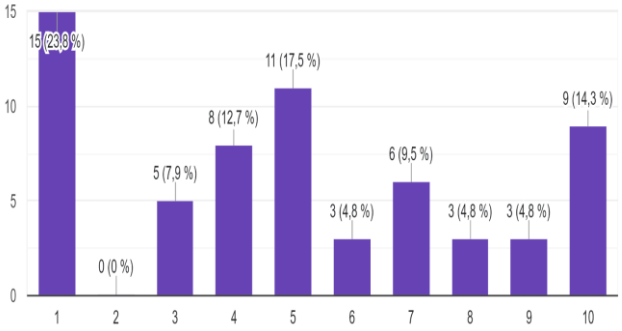


Рисунок 3.Трансляція в байт-код
та його виконання віртуальной машиной

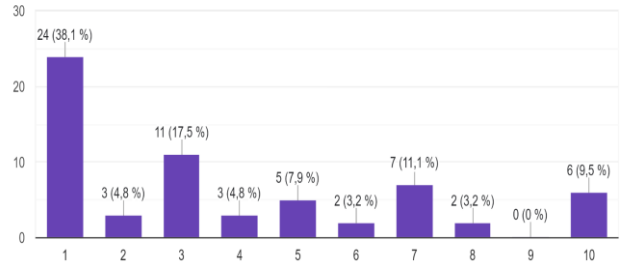


Рис 4. Трансляція в
іншу мову програмування

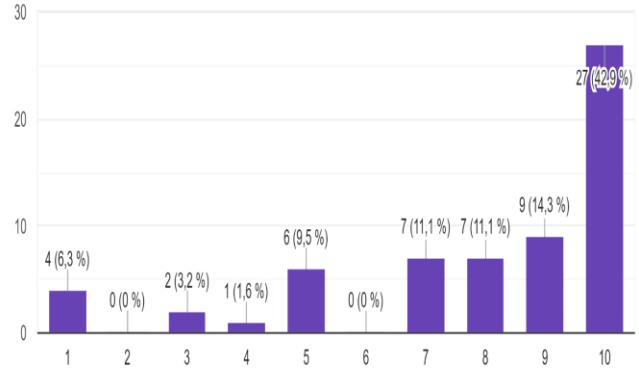


Рис 5. Вкажіть важливість типу типізації (сильна статична, сильна динамічна, слаба статична, слаба динамічна)

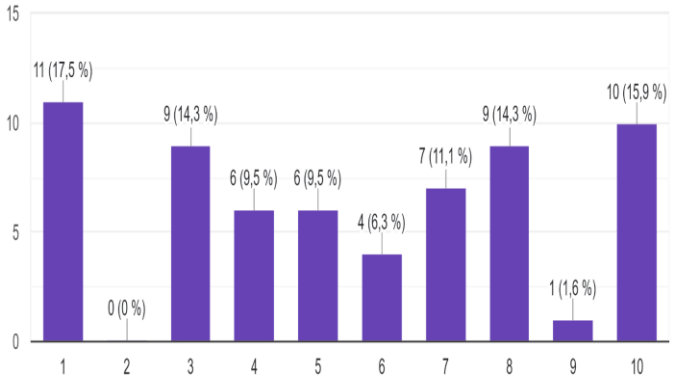


Рис 6. Вкажіть будь ласка важливість рівня підтримки мовою ООП(не підтримує, підтримує за рахунок емуляції, підтримує повністю)

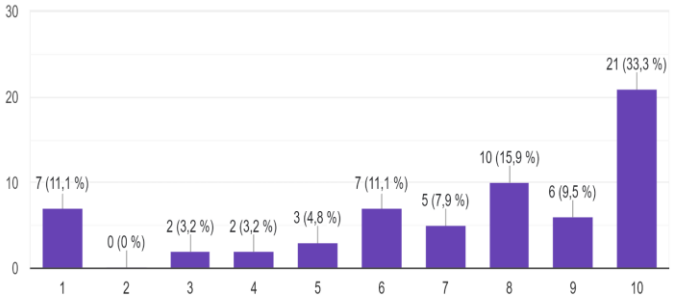


Рис 7. Важливість підтримки інтерфейсів

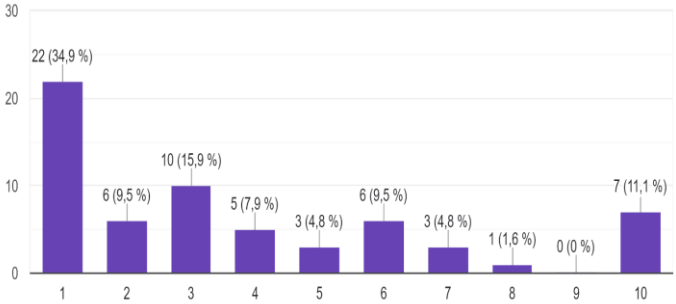


Рис 8. Важливість підтримки множинного наслідування

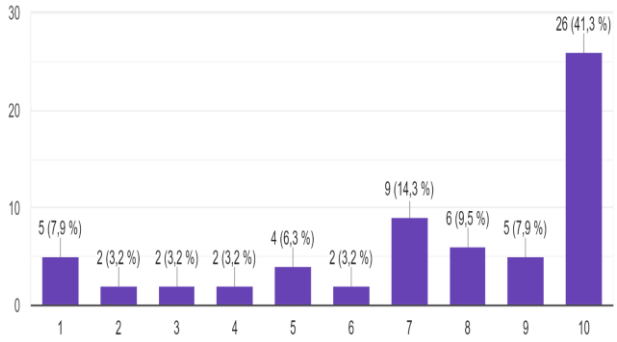


Рис 9. Важливість підтримки функціонального типу

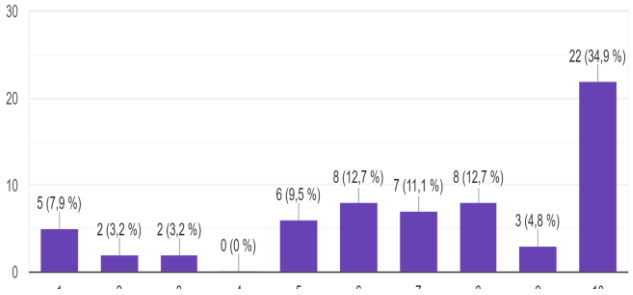


Рис 10. Присутність операції згортки над структурами даних

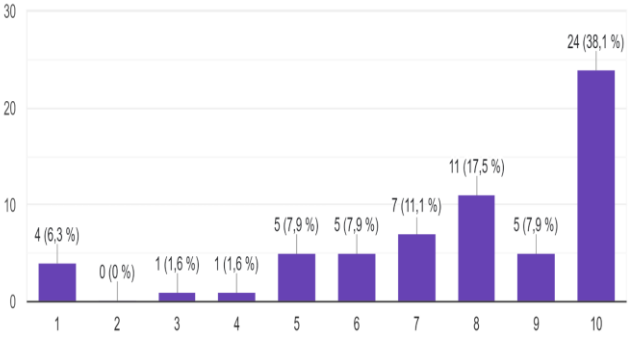


Рис 11. Вкажіть будь ласка рівень багатопоточного програмування, що підтримує мова (не підтримує, підтримує за рахунок емуляції, базові можливості багатопоточності, просунуті можливості багатопоточності, мови що підтримують shared memory)

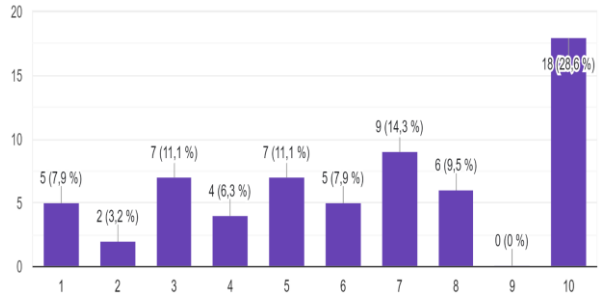


Рисунок 12. Вкажіть важливість підтримки макросів мовою

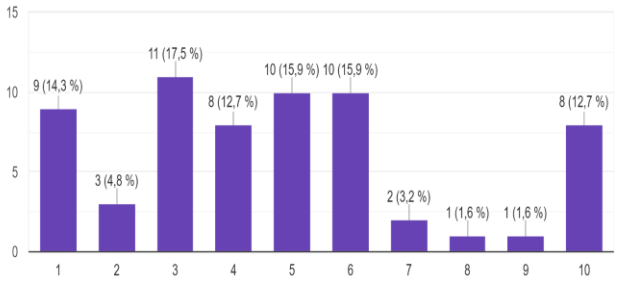


Рисунок 13. Вкажіть важливість підтримки мовою логічного програмування

Як видно з графіків, деякі властивості мають одне і те ж значення середнього розподілу, їх автор розподілив відповідно до внутрішнього уявлення проважливість. Найбільш значущими експерти вважають типи типізації, підтримку інтерфейсів (не плутати з наявністю ООП, див. JavaScript) та компіляції. .

**Коефіцієнти парадигми.** В результаті досліджень були узагальнені отримані значення порівняння мов програмування за парадигми у вигляді таблиці 2.

Таблиця 2. Порядок властивостей для порівняння мов за парадигмою

| | |
|---|---|
| Підтримка компіляції | 10 |
| Інтерпретація | 10 |
| Вид типізації | 10 |
| Підтримка інтерфейсів | 10 |
| Підтримка функціонального типу | 10 |
| Операція згортки даних | 10 |
| Ступінь підтримки багатопоточності | 10 |
| Підтримка макросів | 10 |
| Підтримка логічного програмування | 3 |
| Трансляція в байт-код та виконання його віртуальною машиною | 1 |
| Трансляція в іншу мову програмування | 1 |
| Рівень підтримки ООП | 1 |
| Підтримка множинного успадкування | 1 |

**Порівняння мов з парадигмою.** Для порівняння були обрані такі мови: Haskell, Delphi, C++, C, Scala, GNU Lisp, PHP 5.4, Ruby, SWI prolog, Java 8, C #, XTend, Java 5. Аналіз проводився за допомогою розробленого програмного забезпечення - плагіна для IDE Eclipse . Вихідний код розташований за адресою https: github.com vladimirkozhaev LanguageDiversityPluginNew.git

Данні за результатами дослідження по мовах програмування розподілились наступним чином (Табл 3). Додаткові характеристики мов були взяти з наступних джерел [23], відомості про мову Сі – [24], відомості про типи даних у мовах – [25], відомості про мову Java – [26], відомості про багатопотокове програмування – [27], а також додаткові відомості про мову Java

Таблиця 3. Обрахування парадигм мов програмування

| | CompIlation 10.0 | Interpretation 10.0 | Static Tipisation 10.0 | Interfaces Or trates Support 10.0 | Functional Type 10.0 | Convolution Operations 10.0 | Basic MultiThreading:10.0 | Macroses :10.0 | Dynamic Tipisation :10.0 | Class or objects Inhertane:10.0 | Advanced MultiThreading :4.0 | Logical Programming :3.0 | Bytecode Execution :1.0 | Translation to another Language :1.0 | Paradigm Value |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Haskell | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 14848 |
| Delphi | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 12176 |
| C++ | 1 | 0 | | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 12112 |
| C | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 10994 |
| Scala | 0 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 7066 |
| GNU Lisp | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 6137 |
| PHP 5.4 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 6034 |
| Ruby | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 3000 |
| SWI Prolog | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 4268 |
| Java 8 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 3994 |
| C# | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 3994 |
| XTend | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 3985 |
| Java 5 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 3218 |

З таблиці 3 видно, що мова Java 8 і C# знаходяться в одній парадигмі, мови C, C++ і Delphi близькі, так само досить близько знаходяться мови GNU LISP і Scala, що відповідає інтуїтивному уявленню про близькість мов.

**Висновки.** В результаті проведених досліджень були отримані наступні результати. Було встановлено, що мови програмування відрізняються як якісно, а й кількісно. Іншими словами, допустимо поняття відстані між мовами. Парадигму мови програмування можна звернути до числа і розглядати різницю між мовами як модуль різниці чисел. Було одержано кількісне значення парадигми для декількох популярних мов програмування. Згідно запропонованної методики, не складно отримати кількісне значення парадигми і для інших мов.

**Перспективи подальших досліджень.** Окрім порівняння мов програмування як таких, перспективним є порівняння реалізації окремих задач, зокрема порівняння швидкості виконання алгоритмів, пов'язаних із штучним інтелектом. Цікаво встановити, чи є звязок парадигми з швидкістю работи тих самих алгоритмів, реалізованих різними мовами. І, якщо закономірність є, побудувати криву залежності відносного часу роботи від парадигми.

**Список бібліографічного опису**

1. Список відомих шикорому загалу мов програмування [Електронний ресурс] Вікіпедія. URL: en.wikipedia.org wiki List_of_programming_languages (дата звернення 23.08.2024)
2. Priya Gupta. Combining Forth and Rust: A Robust and Efficient Approach for Low-Level System Programming[Текст] Ravi Rahar,Rahul Kumar Yadav,Ajit Singh Ramandeep, Kumar Sunil Kumar Recent Advances in Science and Engineering -

3. Glenford J. Myers, Corey Sandler, Tom Badgett The Art of Software Testing 3rd Edition [Текст] Glenford J. Myers, Corey Sandler, Tom Badgett Amazon, 2011. 256 с.
4. Anderson T. Recovery blocks in action: A system supporting high reliability[Текст] Anderson T., Kerr R. Recovery Reliable Computer Systems. Springer, Berlin, Heidelberg; 1985. 440 с.
5. Rui Pereiraa, Ranking programming languages by energy efficiency[Текст] Rui Pereiraa, Marco Coutoc, Francisco Ribeiroc, Rui Ruac, Jácome Cunhac, João PauloFernandesd, João Saraivac. Modern Innovations, Systems and Technologies, 2(3), 0127–0138
6. Sleiman Rabah. Comparative Studies of 10 Programming Languages within 10 Diverse Criteria[Текст] Jiang Лі. Mingzhi Liu. Yuanwei Lai. Concordia University.
7. I. S. Anureev. On the problem of computer language classification [Текст] IS Anureev, EV Bodin, LV Gorodnaya, AG Marchuk, FA Murzin, NV Shilovю Bull. Nov. Comp. Center, Comp. Science, 28 (2008), 31–42
8. Документація мови луа [Електронний ресурс] режим доступу: https: www.lua.org about.html( дата звернення 23.08.2024)
9. *Bjarne Stroustrup* The C++ Programming Language: Special Edition [текст] Bjarne Stroustrup Amazon 2000. 1019 с.
10. Stephan Diehl.A Formal Introduction to Compilation of Java[текст] Stephan Diehl «SoftwarePractice and Experience», Vol. 28 (3), pages 297-327, March 1998.
11. Документація мови XTend https: www.eclipse.org xtend (дата звернення 10.09.2024) — назва с экрану
12. Martin Odersky , Programming in Scala
[Текст] Martin Martin Odersky, Lex Spoon , Bill Venners Paragon OS 2011. 720 с.
13. Abhishek Singh. Essential Python for Machine Learning[Текст] Abhishek Singh Amazon 2023 — 572 с.
14. Dagfinn Reiersol. PHP in Action: Objects, Design[Текст] Agility Dagfinn Reiersol, Marcus Baker, Chris Shiflett Amazon 2007 — 400 с
15. Документація мови TypeScript https: www.typescriptlang.org (дата звернення 14.08.2024) — назва з экрану
16. Joey Lott ActionScript 3.0 Cookbook [Текст] Joey Lott, Darron Schall, Keith Peters Released 2006 — 300 с.
17. Ivan Bratko.PROLOG Programming for Artificial Intelligence[ Текст] Ivan Bratko ThriftBooks-Chicago USA,1996 — 440 с.
18. Luca Cardelli.Typeful Programming [Текст] Luca Cardelli. Digital Equipment Corporation Systems Research Center. 130 Lytton Avenue, Palo Alto, CA 94301
19. Jeffrey Richter. Applied Microsoft® .NET Framework Programming in Microsoft® Visual Basic® .NET[Текст] Jeffrey Richter, Francesco Balena 2002. 656 с.
20. Керніган, Д. MMOBA Ci [Текст] Керніган, Д. Рітчі М. 2-ге видання, Науковий світ 2023 — 288 с.
21. *Anthony J. Field ,* Functional Programming (International Computer Science Series) [Текст] Anthony J. Field, Peter Harrison MEDIMOPS, 1988. — 612 с.
22. Raoul-Gabriel Urma Java in Action Raoul-Gabriel Urmaambdas, streams, functional and reactive programming 2nd Edition
[Текст] Raoul-Gabriel Urma , Mario Fusco , Alan Mycroft, Amazon 2018. 592 с.
23. Paul Butcher. Se ven Concurrency Models в Seven Weeks: When Threads Unravel (The Pragmatic Programmers) [Текст] Paul Butcher Amazon 2016, 1st Edition . 296с
24. R Harrison .Comparing Programming Paradigms: Evaluation of Functional and Object-Oriented Programs.[ Текст] R Harrison, LG Samaraweera, MR Dobie, PH LewisDept. of Electronics and Computer Science, Southampton University, SO17 1BJ, UK April 24, 2012
25. Шмельова Т.Ф. Методичні вказівки до проведення практичних занять з дисципліни «Основи теорії прийняття рішень» [Текст] Київ. Національний Авиіацйний Университет 2015. 296с

**References**

1. List of programming languages known to the general public [Electronic resource]. Wikipedia. URL: https: en.wikipedia.org wiki List_of_programming_languages (access date 23.08.2024)
2. Priya Gupta. Combining Forth and Rust: A Robust and Efficient Approach for Low-Level System Programming [Text] Ravi Rahar,Rahul Kumar Yadav,Ajit Singh Ramandeep, Kumar Sunil Kumar Recent Advances in Science and Engineering
3. Glenford J. Myers, Corey Sandler, Tom Badgett The Art of Software Testing 3rd Edition [Text] Glenford J. Myers, Corey Sandler, Tom Badgett Amazon, 2011. 256 p.
4. Anderson T. Recovery blocks in action: A system supporting high reliability [Text] Anderson T., Kerr R. Recovery Reliable Computer Systems. Springer, Berlin, Heidelberg; 1985. 440 p.
5. Rui Pereiraa, Ranking programming languages by energy efficiency [Text] Rui Pereiraa, Marco Coutoc, Francisco Ribeiroc, Rui Ruac, Jácome Cunhac, João PauloFernandesd, João Saraivac. Modern Innovations, Systems and Technologies, 2(3), 0127–0138
6. Sleiman Rabah. Comparative Studies of 10 Programming Languages within 10 Diverse Criteria [Text] Jiang Li. Mingzhi Liu. Yuanwei Lai. Concordia University.
7. I.S. Anureev. On the problem of computer language classification [Text] IS Anureev, EV Bodin, LV Gorodnaya, AG Marchuk, FA Murzin, NV Shilovyu Bull. Nov. Comp. Center, Comp. Science, 28 (2008), 31–42
8. Lua Language Documentation [Electronic resource] access mode: https: www.lua.org.html(access date 23.08.2024).
9. Bjarne Stroustrup The C++ Programming Language: Special Edition. [text] Bjarne Stroustrup Amazon 2000. 1019 p.
10. Sannikov E. V. Practical Programming Course in Delphi. Object-Oriented Programming [Text] Sannikov E. V. Solon 2013 – 188 p.

11. Stephan Diehl.A Formal Introduction to Compilation of Java[text] Stephan Diehl «SoftwarePractice and Experience», Vol. 28 (3), pages 297-327, March 1998.
12. XTend Language Documentation https: www.eclipse.org xtend (accessed 10.09.2024)
13. Martin Odersky , Programming in Scala [Text] Martin Martin Odersky, Lex Spoon , Bill Venners Paragon OS 2011. 720 p.
14. Abhishek Singh. Essential Python for Machine Learning [Text] Abhishek Singh Amazon 2023. 572 p.
15. Dagfinn Reiersol. PHP in Action: Objects, Design [Text] Agility Dagfinn Reiersol, Marcus Baker, Chris Shiflett Amazon 2007. 400 c
16. TypeScript Language Documentation https: www.typescriptlang.org (accessed 08 14 2024)
17. Joey Lott ActionScript 3.0 Cookbook [Text] Joey Lott, Darron Schall, Keith Peters Released 2006.300 p.
18. Ivan Bratko.PROLOG Programming for Artificial Intelligence
19. [ Text] Ivan Bratko ThriftBooks-Chicago USA,1996. 440 p.
20. Luca Cardelli.Typeful Programming [Text] Luca Cardelli. Digital Equipment Corporation Systems Research Center. 130 Lytton Avenue, Palo Alto, CA 94301
21. Jeffrey Richter. Applied Microsoft® .NET Framework Programming in Microsoft® Visual Basic® .NET[Text] Jeffrey Richter, Francesco Balena 2002. 656 p.
22. Kernighan, D. MMOVA C [Text] Kernighan, D. Ritchie M. 2nd edition, Scientific World 2023. 288 p.
23. Shmelyova T.F. Methodological instructions for conducting practical classes in the discipline "Fundamentals of Decision Theory" [Text] Shmelyova T.F. National Aviation University 2015 - 296 p.
24. Anthony J. Field , Functional Programming (International Computer Science Series) [Text] Anthony J. Field, Peter Harrison MEDIMOPS, 1988. 612 p.
25. Raoul-Gabriel Urma Java in Action Raoul-Gabriel Urmaambdas, streams, functional and reactive programming 2nd Edition [Text] Raoul-Gabriel Urma, Mario Fusco, Alan Mycroft, Amazon 2018 — 592 p.
26. Paul Butcher. Seven Concurrency Models in Seven Weeks: When Threads Unravel (The Pragmatic Programmers) [Text] Paul Butcher Amazon 2016, 1st Edition - 296s
27. R Harrison. Comparing Programming Paradigms: Evaluation of Functional and Object-Oriented Programs.[Text] R Harrison, LG Samaraweera, MR Dobie, PH LewisDept. of Electronics and Computer Science, Southampton University, SO17 1BJ, UK April 24, 2012