Computer science

ФЕДОТОВ Кирило Володимирович,
випускник магістратури,
Дніпровський національний університет імені Олеся Гончара,
Україна

МОВИ ПРОГРАМУВАННЯ МАЙБУТНЬОГО: ТЕНДЕНЦІЇ, ОСОБЛИВОСТІ ТА ПАРАДИГМИ

Сфера мов програмування постійно розвивається, зумовлена технологічним прогресом і потребою в більш ефективних і виразних способах написання програмного забезпечення. Дивлячись у майбутнє, важливо вивчати тенденції, особливості та парадигми, які формуватимуть мови програмування майбутнього. У цій роботі досліджуються деякі ключові аспекти майбутніх мов програмування, спираючись на різні джерела, щоб забезпечити повне розуміння теми.

Підвищена увага до продуктивності та досвіду розробників

Однією з важливих тенденцій у майбутніх мовах програмування є збільшення уваги до продуктивності та досвіду розробника. Мови програмування розвиваються, щоб зробити розробку програмного забезпечення швидшою, інтуїтивно зрозумілішою та менш схильною до помилок. Згідно з дослідженням, проведеним JetBrains, провідною компанією з розробки програмного забезпечення, такі мови, як Kotlin і Rust, набули популярності завдяки своїй зосередженості на продуктивності розробника та функціях безпеки[1]. Ці мови забезпечують надійну систему типів, сучасний синтаксис і широку підтримку інструментів, покращуючи загальний досвід програмування.

Інтеграція штучного інтелекту

Оскільки штучний інтелект (ШІ) продовжує розвиватися, мови програмування розробляються для полегшення інтеграції технологій ШІ. Фреймворки машинного та глибокого навчання, такі як TensorFlow і PyTorch, уже широко використовуються, а також з'являються нові мови для спрощення розробки ШІ. Наприклад, Swift для TensorFlow[2], варіант мови програму-

1 JetBrains. (2021). The State of Developer Ecosystem 2020. URL : https://www.jetbrains.com/lp/devecosystem-2020/ (18.05.23).

2 TensorFlow. (n.d.). Swift for TensorFlow. URL: https://www.tensorflow.org/swift/guide/overview (10.05.23).

вання Swift, поєднує в собі простоту використання Swift із потужністю TensorFlow для створення програм ШІ.

Доменне-специфічні мови (DSL)

Доменно-орієнтовані мови (DSL) — це мови програмування, створені для вирішення конкретних проблемних областей. Вони забезпечують абстракції високого рівня та виразність для конкретних завдань, дозволяючи розробникам писати код більш стисло та точно. Очікується, що в майбутньому DSL стануть більш поширеними, що дозволить розробникам ефективно працювати в спеціалізованих областях. Прикладом популярного DSL є SQL (Structured Query Language), який використовується для запитів і керування реляційними базами даних[3].

Одночасність і паралелізм

Із зростаючою потребою в масштабованому та ефективному програмному забезпеченні майбутні мови програмування приділятимуть більшу увагу паралельності та паралелізму. Уміння писати код, який ефективно використовує декілька процесорів і розподілених систем, буде мати вирішальне значення. Такі мови, як Go з його легкими goroutines і Rust з його моделлю власності для безпечного паралелізму, вже привертають увагу своїми можливостями паралельного програмування[4].

Покращена безпека

З огляду на зростаючі кіберзагрози та потенційний вплив уразливостей програмного забезпечення, майбутні мови програмування надаватимуть пріоритет функціям безпеки та захисту. Rust, наприклад, забезпечує безпеку пам'яті та запобігає поширеним помилкам програмування, таким як розіменування нульового вказівника та змагання даних[5]. Інші мови також досліджують подібні підходи для мінімізації вразливостей безпеки та забезпечення більш надійного коду.

Парадигма функціонального програмування

Парадигма функціонального програмування, яка акцентує увагу на незмінності, чистих функціях і декларативному програмуванні, набирає по-

3 Oracle. (n.d.). SQL Language Reference. URL: https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Preface.html#GUID-0897B474-6033-4398-AA8A-922F1C5CAF53 (19.04.23).

4 The Go Programming Language. (n.d.). Goroutines. URL: https://go.dev/tour/concurrency/1 (09.04.23); The Rust Programming Language. (n.d.). Fearless Concurrency. URL: https://doc.rust-lang.org/book/ch16-00-concurrency.html (18.04.23).

5 The Rust Programming Language. (n.d.). Rust Language Safety. URL: https://doc.rust-lang.org/book/ch04-00-understanding-ownership.html (21.05.23).

пулярності. Функціональні мови програмування, такі як Haskell, Scala та Clojure, пропонують численні переваги, включаючи покращену читабельність коду, можливість тестування та модульність. Багато майбутніх мов програмування, ймовірно, включатимуть концепції функціонального програмування, щоб дозволити розробникам писати більш стислий код, який зручно підтримувати[6].

Інтероперабельність і міжмовна підтримка

Очікується, що в майбутньому мови програмування більше зосередяться на сумісності та міжмовній підтримці. Оскільки системи програмного забезпечення стають дедалі складнішими, розробникам часто доводиться комбінувати кілька мов, щоб використовувати сильні сторони кожної з них. Такі проекти, як WebAssembly, спрямовані на створення універсального бінарного формату, який дозволяє бездоганно взаємодіяти між різними мовами та ефективно працювати на різних платформах. Ця гнучкість дозволяє розробникам вибирати правильну мову для конкретного завдання, одночасно забезпечуючи плавну інтеграцію в більшу систему, та має потенціал для трансформації різних галузей, створюючи нові можливості для інновацій та зростання[7].

Розширені інструменти та підтримка

IDE Інструменти та підтримка IDE відіграють вирішальну роль у продуктивності та ефективності розробників. Майбутні мови програмування, швидше за все, надаватимуть пріоритет надійним екосистемам інструментів та інтегрованим середовищам розробки (IDE). Ці інструменти допомагають розробникам у завершенні коду, рефакторингу, налагодженні та оптимізації продуктивності. Такі мови, як TypeScript, з його потужним виведенням типів і підтримкою інструментів, набули популярності серед веб-розробників завдяки розширеному досвіду розробки, який вони надають[8].

Мови квантових обчислень

Оскільки квантові обчислення продовжують розвиватися, очікується поява спеціальних мов програмування для квантових систем. Мови квантового програмування, такі як Q#, розроблені для роботи з квантовими ал-

6 Peyton Jones, S., & Wadler, P. (Eds.). (1993). The Haskell 98 Language Report. URL: https://www.haskell.org/onlinereport/haskell2010/haskellch1.html (07.04.23).

7 WebAssembly. (n.d.). Faq. URL: https://webassembly.org/docs/faq/ (05.04.23).

8 TypeScript. (n.d.). Handbook. URL: https://www.typescriptlang.org/docs/handbook/intro.html (10.05.23).

горитмами та унікальними властивостями квантових комп'ютерів. Ці мови забезпечують абстракції та конструкції, які дозволяють розробникам ефективно писати квантовий код і використовувати потужність квантових обчислень[9]. Ймовірно, вони відіграватимуть життєво важливу роль у різних сферах, включаючи криптографію, оптимізацію та наукове моделювання.

Доступність та інклюзивність

У майбутньому мови програмування, ймовірно, зосередяться на доступності та інклюзивності, маючи на меті знизити бар'єри для входу та задовольнити ширше коло розробників.

Докладаються зусиль, щоб розробити мови та інструменти, зручні для початківців, забезпечити кращу документацію та пропонувати навчальні ресурси для різних спільнот. Такі проекти, як Scratch і Blockly, дозволяють молодим учням ознайомитися з концепціями кодування через візуальні інтерфейси програмування, сприяючи ранньому інтересу до програмування[10].

Зі зміною середовища розробки програмного забезпечення змінюватимуться й мови програмування майбутнього. Такі тенденції, як підвищена увага до продуктивності, інтеграція штучного інтелекту, доменно-орієнтовані мови, паралельність і паралелізм, покращена безпека, а також розвиток парадигм функціонального програмування, готові сформувати мови програмування майбутнього. Слідкуючи за цими тенденціями та впроваджуючи нові функції та парадигми, розробники можуть підвищити свою продуктивність, створювати надійніші програми та відкривати нові можливості у світі програмування, що постійно змінюється.

Мови програмування майбутнього будуть формуватися різними тенденціями, включаючи підвищену увагу до продуктивності та досвіду розробників, інтеграцію штучного інтелекту, предметно-орієнтовані мови, функціональні парадигми програмування, сумісність, розширені інструменти та підтримка IDE, мови квантового обчислення та доступність. Ці тенденції відображають постійні потреби розробників і вимоги нових технологій. Розуміючи та сприймаючи ці тенденції, програмісти можуть залишатися в авангарді інновацій і використовувати весь потенціал майбутніх мов програмування.

DOI: 10.51587/9798-9866-95990-2023-013-5-8

9 Microsoft Quantum. (n.d.). Q# Language. URL: https://learn.microsoft.com/en-us/azure/quantum/overview-what-is-qsharp-and-qdk (11.05.23).

10 Scratch. (n.d.). About Scratch. URL: https://scratch.mit.edu/about/ (02.05.23).