
ISSN 2710-1177 (online)
DOI: 10.61718/crp
Since 1989
КРЕАТИВНИЙ ПРОСТІР
Науковий журнал
Електронне видання

ISSN 2710-1177 (online)
DOI: 10.61718/crp
Publisher ID: 7886

КРЕАТИВНИЙ ПРОСТІР™

Науковий журнал

Електронне видання

2025 • № 27

Освіта, фізична культура, спорт, культура, мистецтво, журналістика, гуманітарні та соціальні науки, бізнес, адміністрування, право, готельно-ресторанна справа, туризм, рекреація, транспорт та безпека, інженерія, виробництво, будівництво, природничі науки, математика та статистика, охорона здоров'я та соціальне забезпечення

КРЕАТИВНИЙ ПРОСТІР ™

НАУКОВИЙ ЖУРНАЛ

Засновано 2020 року. Журнал є електронним мультидисциплінарним науковим виданням. Міжнародний стандартний номер періодичного видання ISSN 2710-1177 (online). Видавцем журналу використовується офіційно зареєстрований знак для товарів та послуг (торгова марка) Креативний простір ™, що підтверджує та гарантує високу якість наданих послуг (Свідоцтво № 342949 видане Державною організацією «Український національний офіс інтелектуальної власності та інновацій»). Свідоцтво про внесення суб'єкта видавничої справи до державного реєстру видавців,виготовлювачів і розповсюджувачів видавничої продукції ДК № 8013 від 22.11.2023. Зареєстровано у глобальному реєстрі видавців Global Register of Publishers. Ідентифікатор видавця 7886. Видання отримує власний DOI. Видавництво зареєстровано у Crossref із власним префіксом 10.61718. Засновник та видавець: Соціально-гуманітарна науково-творча майстерня «Новий курс» (рік заснування – 1989) є науковою установою. E-mail: creatspace2020@gmail.com, info@newroute.org.ua. Сайт: www.newroute.org.ua Контент розміщується в базі даних інформаційного ресурсу «Наукова періодика України», в пошуковій системі наукових публікацій «Google Scholar», в репозитарії на сайті засновника.Індексується за показниками h-індекс (Google Scholar), i10-індекс (Google Scholar). Включено до каталогу наукових ресурсів відкритого доступу ROAD, до академічної бази даних ResearchBid, до бази наукових публікацій Google Scholar, до каталогу наукової періодики України з питань освіти, педагогічної, психологічної та соціальних наук Державної науково-педагогічної бібліотеки України імені В. О. Сухомлинського Національної академії педагогічних наук України, до реєстру наукових видань України державної наукової установи України «Український інститут науково-технічної експертизи та інформації». Публікація у журналі є науковою працею, яка опублікована у вітчизняному електронному науковому періодичному виданні та підтверджує апробацію наукових досліджень автора. У журналі можуть бути розміщені наукові повідомлення про розробки та праці, в т. ч. інформація про підготовку дисертаційних досліджень та творчих мистецьких проектів тощо. Видання не входить до переліку фахових видань України. У журналі розміщуються наукові публікації за повним переліком галузей знань. Автори несуть відповідальність за зміст (авторство та самостійність досліджень), точність та достовірність викладеного матеріалу. Редакція може не поділяти точку зору авторів. Наукові публікації оприлюднюються в рамках проведення науково-практичної конференції «Креативна трансформація та модернізація сучасного суспільства», м. Харків, Україна. Розміщення публікації автора у виданні є підтвердженням участі автора у конференції. За результатами проведення конференції та оприлюднення рукописів, автори отримують електронний сертифікат (30 годин – 1 ECTS credits). Сертифікати оприлюднюються за адресою www.newroute.org.ua (згідно Порядку підвищення кваліфікації педагогічних і науково-педагогічних працівників, затвердженого постановою Кабінету Міністрів України від 21 серпня 2019 р. № 800).

Редакційна колегія:
Кучин Сергій Павлович, головний редактор, доктор економічних наук, професор, академік Національної академії наук вищої освіти України
Акіншина Ірина Миколаївна, кандидат філологічних наук, доцент
Березовська-Чміль Олена Борисівна, кандидат політичних наук, доцент
Внукова Ольга Миколаївна, кандидат педагогічних наук, доцент
Гетьман Ірина Анатоліївна, кандидат технічних наук, доцент
Гришко Світлана Вікторівна, кандидат географічних наук, доцент
Доброєр Наталія Вікторівна, кандидат культурології, доцент
Дубовик Наталія Анатоліївна, кандидат політичних наук, доцент
Єрошенко Олена Віталіївна, кандидат мистецтвознавства, доцент
Калініна Ольга Сергіївна, кандидат культурології
Карпинський Борис Андрійович, доктор економічних наук, професор
Кислюк Любов Вікторівна, кандидат наук із соціальних комунікацій, доцент
Кожедуб Олена Василівна, кандидат соціологічних наук, доцент
Коробчук Людмила Іванівна, кандидат педагогічних наук, доцент
Кучин Павло Захарович, заслужений артист України
Кучина Тетяна Ігорівна, відповідальний секретар. магістр з маркетингу
Мкртічян Оксана Альбертівна, доктор педагогічних наук, професор
Пашкова Надія Ігорівна, кандидат філологічних наук, доцент
Підлісна Ольга Вікторівна, кандидат мистецтвознавства, доцент
П'ятакова Галина Павлівна, доктор педагогічних наук, професор
Рассомахіна Ольга Андріївна, кандидат юридичних наук
Сафонова Наталія Анатоліївна, кандидат філологічних наук, доцент
Стефанишин Олена Василівна, кандидат історичних наук, доцент
Сторож Олена Василівна, кандидат психологічних наук, доцент
Тарасюк Лариса Сергіївна, доктор філософських наук, професор
Федоренко Микола Олександрович, кандидат філософських наук, доцент
Харченко Артем Вікторович, кандидат історичних наук, доцент
Хожило Ірина Іванівна, доктор наук з державного управління, кандидат медичних наук
Ціватий Вячеслав Григорович, кандидат історичних наук, доцент, заслужений працівник освіти України
Шевчук Інна Володимирівна, доктор наук з державного управління, професор
Шептуха Олена Михайлівна, кандидат економічних наук, доцент
Штулер Ірина Юріївна, доктор економічних наук, професор

Креативний простір: електрон. наук. журн. – № 27. – Харків: СГ НТМ «Новий курс», 2025. – 80 с.
УДК 001 • ISSN 2710-1177 (online) • DOI: 10.61718/crp • www.newroute.org.ua
Опубліковано на основі ліцензії Creative Commons Attribution License.
© СГ НТМ «Новий курс», 2025
© Автори, 2025

ЗМІСТ

Стор.

Освіта, фізична культура та спорт, культура та мистецтво, гуманітарні та соціальні науки ... 7

Козубовська Ірина Василівна, Палкуш Віталія Петрівна, Тополянський Сергій Іванович
ДЕЯКІ АСПЕКТИ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ В СИСТЕМІ ВИЩОЇ ОСВІТИ США ... 7

Нідзельський Станіслав Петрович
ЦИКЛІЧНІСТЬ І ФОЛЬКЛОРНА СИМВОЛІКА В САКСОФОННОМУ ЦИКЛІ «КОЛО» ВАЛЕРІЇ ВИНОГРАДОВОЇ ... 9

Хричова Діана Олегівна
КРЕАТИВНА ТРАНСФОРМАЦІЯ ТА МОДЕРНІЗАЦІЯ В ОБРАЗОТВОРЧОМУ МИСТЕЦТВІ ... 10

Козубовська Ірина Василівна, Милян Жанна Іванівна, Козубовський Максим Ростиславович
ДЕЯКІ ОСОБЛИВОСТІ ПРЕВЕНТИВНОЇ РОБОТИ З НЕПОВНОЛІТНІМИ ГРУПИ РИЗИКУ: ЗАРУБІЖНИЙ ДОСВІД ... 11

Шведа Анастасія, Панасюк Дарина, Гаврилюк Анастасія, Тартачник Богдана, Цуп Андрій
ДРІППІНГ – СПОНТАННИЙ ЖИВОПИС. ЕКСПРЕСІЯ ТА КОНЦЕПЦІЯ У ГРАФІЧНОМУ ДИЗАЙНІ ... 13

Ярушевський Тарас Васильович
ФЛЕЙТА, ФАГОТ І ФОРТЕПІАНО ЯК КАМЕРНИЙ АНСАМБЛЬ: ІСТОРИКО-СТИЛЬОВА ЕВОЛЮЦІЯ ТА ІНТЕРПРЕТАЦІЯ В «ЕКЗЕРСИСАХ» ГЕННАДІЯ ЛЯШЕНКА ... 15

Отрох Юліана Павлівна
ВПЛИВ ВИКОРИСТАННЯ СОЦІАЛЬНИХ МЕРЕЖ НА РОЗВИТОК ДЕПРЕСИВНИХ СИМПТОМІВ У МОЛОДІ ... 16

Романова Анна Сергіївна
ІДЕАЛЬНА КАРТИНКА У СОЦМЕРЕЖАХ ТА ЇЇ ВПЛИВ НА САМОСПРИЙНЯТТЯ ... 17

Клімова Олена Ігорівна
ОПТИМАЛЬНА МОДЕЛЬ КОМАНДНОЇ ВЗАЄМОДІЇ У СПОРТИВНО-БАЛЬНОМУ ТАНЦІ: ПОЄДНАННЯ ПРОЄКТНОЇ, КРОС-ФУНКЦІОНАЛЬНОЇ ТА КОМАНДИ ДІЇ ... 17

Лебедь Андрій Сергійович
АКТОРСЬКА МАЙСТЕРНІСТЬ ЯК ЗАСІБ РОЗВИТКУ ЕМОЦІЙНОГО ІНТЕЛЕКТУ: КРЕАТИВНІ ВИКЛИКИ ЧАСУ ... 18

Боденчук Богдан Іванович
ВПЛИВ ЦИФРОВИХ ТЕХНОЛОГІЙ НА ОРГАНІЗАЦІЮ ГАЗЕТНО-ЖУРНАЛЬНОГО ВИРОБНИЦТВА ... 20

Бінєєва Алла Махмудівна, Сомов Лев Миколайович, Лисій Олег Степанович
ЗАПАМ'ЯТОВУВАННЯ НАГОЛОСІВ У СЦЕНІЧНІЙ МОВІ ЧЕРЕЗ КАРТКОВІ ІГРИ ТА МЕТОД SPACED REPETITION ... 21

Романова Анна Сергіївна
РОЛЬ ЕМОЦІЙНОГО СТАНУ У СПРИЙНЯТТІ ІДЕАЛЬНОГО ВІЗУАЛЬНОГО КОНТЕНТУ В СОЦІАЛЬНИХ МЕРЕЖАХ ... 23

Отрох Юліана Павлівна
ВПЛИВ РЕГУЛЯРНОЇ ПРАКТИКИ ВДЯЧНОСТІ НА РІВЕНЬ СУБ'ЄКТИВНОГО БЛАГОПОЛУЧЧЯ СЕРЕД МОЛОДИХ ЛЮДЕЙ ... 24

Гориславец Володимир, Гориславец Євген
ЖЕНЬКА РУСАБРОВ – МІЙ ДРУГ, ТОВАРИШ, ОПОНЕНТ ... 25

Мох Юлія Андріївна
ОСОБЛИВОСТІ РОБОТИ З РІЗНОРІВНЕВОЮ ГРУПОЮ У НЕМОВНОМУ ЗВО ... 29

Боцула Світлана Павлівна
ДЕДЛАЙН У ПЕДАГОГІЧНІЙ ДІЯЛЬНОСТІ ... 31

Штогринець Христина Андріївна
ПСИХОСОЦІАЛЬНІ АСПЕКТИ ДЕПРЕСИВНОЇ СИМПТОМАТИКИ ЗА ТРАНЗАКЦІЙНИМ АНАЛІЗОМ ... 32

Сліпченко Єлизавета Вікторівна
ІННОВАЦІЙНІ СТРАТЕГІЇ АДАПТАЦІЇ ІВЕНТ-ІНДУСТРІЇ ДО УМОВ НЕСТАБІЛЬНОСТІ ... 36
Стрічевська Світлана Миколаївна
РОЗРОБКА КОМУНІКАЦІЙНОЇ ПОЛІТИКИ МОЛОДІЖНОЇ ДРАМАТИЧНОЇ СТУДІЇ ... 38
Трофімова Альона Сергіївна
ЕВОЛЮЦІЯ УКРАЇНСЬКОГО ВИКОНАВСТВА НА ЄВРОПЕЙСЬКИХ МУЗИЧНИХ КОНКУРСАХ ТА УПРАВЛЯННЯ НЕЮ ... 39
Дяченко Анна Валентинівна
ФОРМУВАННЯ ТА РЕАЛІЗАЦІЯ МАРКЕТИНГОВОЇ ПОЛІТИКИ ПРОСУВАННЯ ІВЕНТ-ЗАХОДІВ ... 40
Малех Олена
ВПЛИВ РІВНЯ СТРЕСУ ТА ОСОБИСТІСНИХ ХАРАКТЕРИСТИК НА ВИКОРИСТАННЯ МЕТОДІВ САМОРЕГУЛЯЦІЇ ТА ВІДПОЧИНКУ ... 41

Бізнес, адміністрування та право, безпека, готельно-ресторанна справа, туризм та рекреація ... 43

Білик Дар'я Сергіївна
МЕТОДИ ЗНИЖЕННЯ ФІНАНСОВИХ РИЗИКІВ У РЕСТОРАННОМУ БІЗНЕСІ ... 43
Вознюк Вікторія Сергіївна
РОЛЬ МАРКЕТИНГОВИХ ІННОВАЦІЙ У ТРАНСФОРМАЦІЇ БІЗНЕС-МОДЕЛЕЙ ... 44
Козаченко Віталій Валерійович, Сахно Дар'я Олександрівна
ПРОФІЛЮВАННЯ ЗҐВАЛТУВАНЬ: КРИМІНОЛОГІЧНИЙ ПІДХІД ДО ВИВЧЕННЯ ОСОБИ ЗЛОЧИНЦЯ ТА ЗАПОБІГАННЯ СЕКСУАЛЬНОМУ НАСИЛЬСТВУ ... 46
Болдижева Аліна Романівна, Демиденко Дар'я Олександрівна, Лещенко Катерина Олегівна, Тищук Артур Вячеславович
ТУРИСТИЧНІ КЛАСТЕРИ ЯК СТРАТЕГІЯ ІННОВАЦІЙНОГО ЗРОСТАННЯ РЕГІОНІВ ... 48
Козаченко Віталій Валерійович
ПЕРЕДЗЛОЧИННА ПОВЕДІНКА ЯК ОБ'ЄКТ КРИМІНОЛОГІЧНОГО АНАЛІЗУ ... 50
Khimchenko Anna Vyacheslavivna
THE ROLE OF THE UNITED NATIONS ORGANIZATION IN RESOLVING INTERNATIONAL CONFLICTS ... 51
Рибачек Валентин Кіндратович
ОРГАНІЗАЦІЙНО-ПРАВОВЕ РЕГУЛЮВАННЯ ПОВОДЖЕННЯ З ПОБУТОВИМИ ВІХОДАМИ ВЕЛИКИХ МІСТ В УКРАЇНІ ... 53
Трофимов Єгор Романович
РЕАЛІЗАЦІЯ УПРАВЛІНСЬКИХ РІШЕНЬ В СФЕРІ ІНДУСТРІЇ МОДИ ... 57
Якименко Павло Павлович
ПРОБЛЕМАТИКА ДОМАШНЬОГО НАСИЛЬСТВА В УМОВАХ ВОЄННОГО СТАНУ ... 59
Рубльова Оксана Володимирівна
ЗАРУБІЖНИЙ ДОСВІД ПРАВОВОГО РЕГУЛЮВАННЯ ЮРИДИЧНОЇ ВІДПОВІДАЛЬНОСТІ ЗА ЕКОЛОГІЧНІ ПРАВОПОРУШЕННЯ У СФЕРІ ЗЕМЕЛЬНИХ ВІДНОСИН ... 61

Інженерія, виробництво, транспорт, будівництво, природничі науки, математика та статистика ... 62

Пронь Валентин Васильович, Безгачнюк Юрій Володимирович, Белей Оксана Ігорівна, Штаєр Лідія Омелянівна
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБЛЕННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ЖАЛЮЗІ ... 62
Романюк Олександр Никифорович, Майданюк Володимир Павлович, Рейда Олександр Миколайович, Катєльніков Денис Іванович, Черняк Олександр Іванович
ТРЕНДИ МОВ ПРОГРАМУВАННЯ ... 64
Яременко Андрій Леонідович
ПІДВИЩЕННЯ ДОВГОВІЧНОСТІ ТА ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІДЦЕНТРОВИХ НАСОСІВ У СИСТЕМАХ ЗРОШЕННЯ ... 71

Онищук Олексій Андрійович
ПІДВИЩЕННЯ КОМФОРТУ УТРИМАННЯ ВРХ ЧЕРЕЗ УДОСКОНАЛЕННЯ АВТОМАТИЧНОЇ ПРИВ'ЯЗІ ... 72
Немченко Юрій Владиславович
ВІДНОВЛЮВАЛЬНІ ДЖЕРЕЛА ЕНЕРГІЇ ТА СТІЙКІСТЬ ОБ'ЄДНАНОЇ ЕНЕРГОСИСТЕМИ УКРАЇНИ ... 72

Охорона здоров'я та соціальне забезпечення ... 75

Давиденко Віталій Володимирович, Жукова Ольга Анатоліївна, Рябіщук Вероніка Анатоліївна
ЗАРУБІЖНИЙ ДОСВІД СОЦІАЛЬНОЇ ПІДТРИМКИ СІМЕЙ У СКЛАДНИХ ЖИТТЄВИХ ОБСТАВИНАХ ... 75
Хомин Аліна Романівна
ВПЛИВ СОЦІАЛЬНИХ МЕРЕЖ НА САМОІДЕНТИФІКАЦІЮ ОСОБИСТОСТІ ... 77

УДК 004.43

Романюк Олександр Никифорович
Доктор технічних наук, професор
Вінницький національний технічний університет
Майданюк Володимир Павлович
Кандидат технічних наук, доцент
Вінницький національний технічний університет
Рейда Олександр Миколайович
Кандидат технічних наук, доцент
Вінницький національний технічний університет
Катєльніков Денис Іванович
Кандидат технічних наук, доцент
Вінницький національний технічний університет
Черняк Олександр Іванович
Кандидат технічних наук, доцент
Вінницький національний технічний університет

ТРЕНДИ МОВ ПРОГРАМУВАННЯ

У галузі програмного забезпечення постійно з'являються нові мови програмування [1-20], що є реакцією на зміну вимог до безпеки, ефективності, паралелізму, масштабованості, а також потреб у спрощенні розробки. Аналіз нових мов програмування дозволяє оцінити їхню релевантність, конкурентні переваги, а також потенціал до широкого впровадження в IT-галузі.

Rust [1, 2] – це сучасна мова програмування системного рівня, розроблена для забезпечення високої продуктивності та безпеки, особливо в контексті управління пам'яттю. Вона була створена у 2010 році командою Mozilla Research і швидко здобула визнання в IT-спільноті завдяки своїм унікальним особливостям. Головна відмінність Rust від традиційних мов, таких як C чи C++, полягає у використанні інноваційної системи володіння, яка дозволяє компілятору гарантувати відсутність помилок доступу до пам'яті ще на етапі компіляції без необхідності застосування збирача сміття.

Однією з важливих сторін Rust є її здатність забезпечити – захист від переповнення буфера, гонок даних, використання вже звільненої пам'яті тощо. Це особливо важливо для критичних систем, таких як операційні системи, веб-браузери, вбудовані пристрої та блокчейн-технології. Наприклад, у проєкті Mozilla Firefox вже використовуються компоненти, написані на Rust, як більш безпечна альтернатива C++. Rust дозволяє використовувати багатопоточність лише там, де це гарантовано безпечно, інакше компілятор не допустить помилкових операцій. Це робить багатопотоковий код менш схильним до гонок даних і складних помилок, які складно відловити в інших мовах. Ще однією перевагою Rust є її гнучка система типів і потужна модель узагальнень , що дозволяє будувати високопродуктивні та типобезпечні абстракції. Водночас, мова досить вимоглива до початківців: концепції володіння, запозичення , життєвих циклів можуть здатися складними на перших етапах навчання. Rust має активну й динамічну спільноту, а також добре розвинену екосистему з централізованим менеджером пакетів Cargo і реєстром бібліотек crates.io. Компанії на кшталт Microsoft, Amazon, Dropbox, Cloudflare та інших уже активно впроваджують Rust у своїх продуктах, що свідчить про її стабільне місце в майбутньому програмної інженерії.

Go (або Golang) – це мова програмування, створена в Google у 2009 році з метою забезпечити просту, ефективну та масштабовану платформу для розробки серверного програмного забезпечення, мережевих сервісів і хмарних застосунків. Її основними розробниками є Роб Пайк, Кен Томпсон та Роберт Ґрісемер – відомі постаті в історії комп'ютерних наук, зокрема творці Unix та мови C. Go поєднує у собі продуктивність компільованих мов із легкістю синтаксису, притаманною скриптовим мовам, таким як Python. Однією з головних особливостей Go є вбудована підтримка конкурентного програмування через механізм goroutines – легковагових потоків, які значно дешевші у виконанні, ніж класичні системні потоки. Завдяки канальній моделі синхронізації, Go дозволяє писати масштабовані паралельні програми без складної синхронізації. Мова свідомо уникає складних об'єктно-орієнтованих конструкцій, таких як спадкування або узагальнення. Цей мінімалізм дозволяє швидше освоїти мову, полегшує читання чужого коду та зменшує кількість стилістичних варіацій, що особливо цінується у командній розробці. Go компілюється у самостійні бінарні файли, які не потребують зовнішніх залежностей, що спрощує розгортання програм, особливо в середовищах із обмеженими ресурсами або в Docker-контейнерах.

Однією з переваг Go є офіційна підтримка Google, а також те, що багато компаній – зокрема Uber, Dropbox, Cloudflare, Kubernetes, Terraform, Docker – використовують її у своїх високонавантажених проєктах. Проте Go має і певні недоліки. Її система типів доволі проста, а обмежена підтримка узагальнень ускладнює написання універсального коду. Крім того, Go не має винятків у класичному розумінні — обробка помилок виконується вручну, що вимагає більше коду, хоча і підвищує контроль над логікою.

Kotlin [5, 6] – це сучасна статично типізована мова програмування, розроблена компанією JetBrains і вперше представлена у 2011 році. Вона призначена для роботи з Java Virtual Machine, однак також підтримує компіляцію до JavaScript та нативного коду, що робить її універсальним інструментом для створення як бекенд-систем, так і мобільних або десктопних застосунків. Популярність Kotlin значно зросла після того, як у 2017 році компанія Google оголосила її офіційною мовою для Android-розробки. Головна мета Kotlin – забезпечити розробникам потужну, виразну та безпечну мову, яка зберігає повну сумісність із Java, але усуває її недоліки. Однією з ключових переваг Kotlin є система null-безпеки, що дозволяє уникати однієї з найпоширеніших помилок у Java – NullPointerException. Синтаксис Kotlin більш лаконічний і виразний, ніж у Java. Наприклад, оголошення класів, геттерів, сеттерів, анонімних функцій і колекцій відбувається набагато простіше, що суттєво скорочує кількість шаблонного коду. Крім того, мова підтримує розширення, лямбда-вирази, корутини для асинхронного програмування та domain-specific language синтаксис, що дозволяє створювати зрозумілі та декларативні API. Однією з визначальних особливостей Kotlin є корутини – легковагові одиниці асинхронного виконання, які дозволяють зручно організовувати конкурентні або фонові операції. На відміну від потоків у Java, корутини значно економніші у використанні ресурсів. Завдяки повній сумісності з Java, Kotlin може використовувати всю наявну екосистему Java-бібліотек і фреймворків (Spring, Hibernate, Retrofit, та ін.), що спрощує інтеграцію у вже існуючі проєкти. Мова має офіційний плагін для IntelliJ IDEA (і Android Studio), а також активно підтримується великими компаніями – JetBrains, Google, Square, Atlassian, Netflix, та іншими. Серед обмежень Kotlin варто зазначити відносно повільнішу компіляцію порівнянно з Java, а також складність деяких функціональних конструкцій. Проте ці недоліки компенсуються гнучкістю мови, високим рівнем абстракції.

Julia [7, 8] – це високорівнева, динамічна мова програмування, створена для вирішення завдань, що потребують інтенсивних чисельних обчислень. Вона була розроблена групою науковців з Массачусетського технологічного інституту . Метою створення Julia було об'єднання продуктивності мов, таких як C і Fortran, із простотою мов високого рівня, таких як Python або MATLAB. Це дозволяє Julia виконувати чисельні розрахунки на високій швидкості без потреби писати критичні фрагменти коду на низькорівневих мовах. Головна перевага Julia – висока продуктивність, яка досягається завдяки використанню JIT-компілятора (Just-In-Time) на основі LLVM. Це дозволяє Julia досягати швидкостей, близьких до C/C++, навіть при динамічному типуванні. Крім того, Julia підтримує векторизацію, розпаралелювання й розподілені обчислення без необхідності звернення до зовнішніх бібліотек або платформ.

Julia має синтаксис, подібний до MATLAB і Python, що робить її легкою для вивчення дослідниками, інженерами й аналітиками. Вона підтримує множинну диспетчеризацію, що дає змогу більш гнучко керувати перевантаженням функцій, залежно від типів аргументів. Julia була спеціально спроєктована для чисельного аналізу, машинного навчання, статистики, обробки сигналів, моделювання фізичних процесів, а також задач в галузі фінансів і біоінформатики. Вона підтримує типи з плаваючою точкою високої точності, комплексні числа, багатовимірні масиви, матричні операції та обчислення з GPU-прискоренням. Однак Julia має певні недоліки. Її екосистема ще не така велика, як у Python або R, і в деяких випадках підтримка сторонніх бібліотек може бути недостатньою.

Solidity [14] – це спеціалізована мова програмування, створена для розробки смартконтрактів, які виконуються на блокчейн-платформі Ethereum. Вона була запропонована у 2014 році Гевіном Вудом, одним із співзасновників Ethereum, і з того часу стала стандартом для створення децентралізованих застосунків . Solidity є мовою високого рівня, орієнтованою на об’єктно-орієнтовану та контрактно-орієнтовану парадигми, і має синтаксис, подібний до JavaScript, що полегшує її вивчення для розробників із веб-фоном.

Основне призначення Solidity – визначення логіки смартконтрактів, які автоматично виконуються в розподіленому середовищі Ethereum Virtual Machine. Контракт у Solidity – це набір функцій і станів, які зберігаються у блокчейні, і які можуть бути викликані користувачами або іншими контрактами. Solidity забезпечує механізми для управління станом, обмеження доступу, обробки подій, а також взаємодії з криптовалютними токенами й іншими смартконтрактами. Мова підтримує вбудовану обробку чисел фіксованої точності, булеві типи, структури, масиви, мапи , модифікатори доступу та винятки. Однією з характерних властивостей Solidity є неможливість зміни контракту після його розгортання. Це накладає відповідальність на розробника за безпеку та правильність логіки ще до публікації контракту в мережі. З цієї причини широко

використовуються патерни проектування, такі як проксі-контракти бібліотеки та шаблони оновлення логіки. Популярність Solidity зростає разом із розвитком децентралізованих фінансів , NFT-проєктів, DAO та інших рішень, заснованих на смартконтрактах. Вона підтримується такими середовищами розробки, як Remix IDE, Truffle, Hardhat, які забезпечують компіляцію, тестування та інтеграцію з блокчейном. Solidity не має повноцінної підтримки floating-point чисел, що ускладнює реалізацію деяких математичних алгоритмів. Також недосвідчені розробники можуть допускати критичні помилки безпеки, які не можуть бути виправлені після запуску контракту. Саме тому важливими є аудит коду, формальна верифікація та покриття тестами.

Zig [9, 10] – це сучасна низькорівнева мова програмування, орієнтована на продуктивність, передбачуваність і точний контроль над ресурсами. Вона була представлена у 2016 році Ендрю Келлі як альтернатива мовам C і C++, і швидко здобула популярність серед системних програмістів завдяки своїй простоті, надійності та чіткому фокусу на безпечне керування пам'яттю без автоматичних механізмів, як-от збирач сміття

Основна ідея Zig [9, 10] – це повернення до принципів явного, зрозумілого та мінімалістичного програмування, де розробник має повний контроль над тим, що відбувається в програмі. Мова дозволяє точно керувати розподілом пам'яті, використовувати кастомні алокатори, уникати прихованих витрат часу виконання.

Zig не має препроцесора, макросистеми чи класичного ООП, але замість цього пропонує потужну систему компіляційного часу, яка дозволяє виконувати складні обчислення, перевірки й генерацію коду ще до запуску програми. Це наближає Zig до концепцій метапрограмування, не вводячи при цьому складності, притаманної C++ шаблонам або макросистемам.

У контексті продуктивності, Zig може компілюватися до дуже ефективного машинного коду й повністю інтегрується з C – як у використанні C API, так і у прямій компіляції C-файлів. Мова здатна замінити C у ролі базової інфраструктурної мови для розробки ядра операційної системи, драйверів, компіляторів, графічних рушіїв або прошивок для вбудованих систем.

Застосування Zig охоплює компілятори (наприклад, Zig сам себе компілює), ігрові рушії, високопродуктивні бібліотеки, embedded-пристрої, розробку ПЗ безпосередньо під ОС або навіть її повноцінне створення (є проєкти OS, написані повністю на Zig). Мова активно використовується в проектах, де необхідна оптимізація без зайвих накладних витрат.

Незважаючи на молодий вік, Zig має активну спільноту та постійно розвивається. Її філософія нагадує модернізовану версію C: вона не намагається бути універсальною, натомість надає розробникам зрозумілий і передбачуваний інструмент без «магії», і з глибокою прозорістю кожної операції.

Наведемо порівняльну таблиця сучасних мов програмування за базовими критеріями.

Таблиця 1

Характеристики сучасних мов програмування за базовими критеріями

Мова	**РРік випуску**	**Основне призначення**	**Продуктив-ність**	**Безпека пам'яті**	**Паралелізм**	**Легкість навчання**	**Підтримка спільноти**
Rust	22010	Системне програму-вання	Висока	Так	Обмежена через async/await	Середня	Висока
Go	22009	Сервери, мережі, мікросервіси	Висока	Частково (GC)	Вбудований goroutine	Висока	Висока
Kotlin	22011	Android, JVM-розробка	Середня	Так (null safety)	Через JVM	Висока	Висока
Julia	22012	Наукові обчислення, ML	Дуже висока	Ні	Обмежена	Середня	Зростає
Solidity	22015	Блокчейн, смартконтракти	Низька	Ні	Немає	Середня	Середня
Zig	22016	Альтернатива C, вбудовані системи	Висока	Частково	В розробці	Середня	Невелика, активна

Станом на 2024-2025 роки активно розробляються кілька нових мов програмування, кожна з яких спрямована на вирішення конкретних проблем, пов'язаних з продуктивністю, безпекою, паралелізмом або новими парадигмами програмування. Розглянемо основні з них.

Carbon [11] –- це експериментальна мова програмування, ініційована компанією Google у 2022 році як потенційна наступниця C++, яка має усунути його історичні недоліки, зберігаючи при цьому сумісність і високу продуктивність. Її основним розробником є Чендлер Каррат, один із провідних фахівців у сфері компіляторів LLVM. C++ за понад 40 років свого розвитку накопичив велику кількість функцій, особливостей і спадщини, що ускладнюють як розробку нових програм, так і подальшу підтримку існуючого коду. Carbon

пропонує більш чіткий синтаксис і структуру, які мають сприяти кращій зрозумілості та скороченню кількості помилок. Мова розробляється з акцентом на explorable design, тобто кожне рішення у синтаксисі чи поведінці обґрунтовано і відкрито для аналізу спільнотою.

Однією з головних цілей Carbon є інтероперабельність із C++: програми на C++ зможуть викликати функції, написані на Carbon, і навпаки, що особливо важливо для великих систем, де повна міграція на нову мову технічно й економічно складна. Carbon також передбачає автоматичну обробку ресурсів, однак без повноцінного збирача сміття (GC), на відміну від Java чи Go. Це дозволяє контролювати продуктивність і пам'ять на низькому рівні, як у C++ чи Rust, але з меншими ризиками.

Carbon підтримує сучасну систему типів із параметризованими типами, яка усуває складність шаблонів C++ та забезпечує перевірки типів на етапі компіляції. Обробка помилок в Carbon також спрощена і передбачає безпечні конструкції для роботи з результатами функцій (наприклад, через тип Result або механізми pattern matching, що наближає її до Rust або Swift.

Ще одна важлива риса Carbon – швидка компіляція і простота вбудовування в існуючі компіляційні ланцюги. Оскільки Carbon компілюється через LLVM, вона зберігає ту ж саму ефективність та оптимізаційні можливості, що й C++ або Rust.

На момент 2025 року Carbon залишається у стадії активної розробки. Вона ще не досягла стабільного релізу і не має широкої індустріальної підтримки, проте активно тестується у спільноті та отримує значну увагу як потенційна основа для довгострокової модернізації великих C++-систем. У майбутньому її успішність залежатиме від підтримки з боку інструментів (IDE, build-систем), спільноти й здатності реалізувати сумісність на практиці.

У підсумку, Carbon – це амбітна спроба створити сучасну, безпечну та ефективну мову для системного програмування, яка дозволяє органічно переходити від C++ до нових парадигм, не руйнуючи наявну кодову базу. Вона має всі шанси зайняти нішу між C++, Rust і Swift, особливо там, де важлива продуктивність, контроль над ресурсами та довготривала підтримка великого проєкту.

Vale [15, 16] – це експериментальна мова програмування, яка створюється з метою об'єднати високу продуктивність мов системного програмування з автоматизованим управлінням пам'яттю та безпекою на рівні компілятора. Вона з'явилась як реакція на обмеження C++, складність Rust і потребу в безпечному конкурентному програмуванні без втрати ефективності. Автор мови, Райан Спітцер (Ryan Spicer), сформулював мету Vale як створення мови, що дозволяє програмістам писати швидкий, безпечний і паралельний код. Ключовою інновацією мови є регійне управління пам'яттю. Це означає, що об'єкти створюються в логічних «регіонах» пам'яті, які можуть бути очищені одночасно, коли більше не потрібні. Такий підхід поєднує переваги ручного управління пам'яттю (як у C/C++) з безпечністю та автоматичністю, властивою мовам із збирачем сміття, без накладних витрат на runtime. Крім того, Vale використовує перевірки на етапі компіляції (compile-time borrow checker), схожі на механізм у Rust, але більш гнучкі та менш суворі для розробника.

Vale також орієнтована на конкурентне програмування. Вона підтримує безпечну роботу з багатопоточними даними за допомогою строгого відстеження володіння (ownership), одночасного доступу та життєвих циклів. дозволяє ефективно масштабувати програми, особливо в середовищах з високим навантаженням, таких як ігрові рушії, графічні обчислення, обробка сигналів або мережеві сервіси.

Синтаксис Vale схожий на сучасні мови – він поєднує елементи Rust, Swift і Kotlin. Мова підтримує функціональний стиль, шаблони (generics), модулі, типи з контролем нульових значень, шаблонне зіставлення (pattern matching) та інші сучасні засоби. Вона також орієнтована на автоматизацію рутинного коду за рахунок макросистеми та можливостей компіляційного часу.

Vale розробляється відкрито як проект із відкритим кодом, зі спільнотою розробників, які тестують і впроваджують мову в нішевих проектах. Хоча вона ще не досягла стабільної версії, її архітектура вже зараз демонструє здатність об'єднати контроль, безпеку й продуктивність у спосіб, недоступний для традиційних мов.

Серед потенційних сфер застосування Vale – розробка ігор, вбудованих систем, графічних рушіїв, обробка великих обсягів даних, розподілені системи. Її філософія «оптимізуй усе, залишайся безпечним» дозволяє розробникам створювати високопродуктивне ПЗ із гарантіями відсутності помилок пам'яті.

V Language [12] – це сучасна компільована мова програмування, яка розробляється з метою забезпечити надзвичайно швидку компіляцію, безпечний синтаксис та мінімалістичну структуру коду. Її автором є Александр Медведніков, а перша публічна версія була представлена у 2019 році. Мову позиціонують як інструмент для розробників, які хочуть писати надійний, продуктивний і простий у підтримці код із дуже коротким циклом компіляції.

Однією з найсильніших особливостей мови є надшвидка компіляція – навіть великі проєкти можуть компілюватися за кілька секунд або часток секунди. Це досягається завдяки використанню однопрохідного

компілятора, що генерує C-код, який потім компілюється через GCC або Clang. У мові заборонені багато типових джерел помилок: небезпечні операції з пам'яттю, нульові вказівники, приховані приведення типів. Обробка помилок здійснюється через строго типізовані значення, що дозволяє виявляти потенційні винятки ще на етапі компіляції. У V немає збирача сміття за замовчуванням – замість нього використовується Automatic Reference Counting, або розробник може керувати пам'яттю вручну або через custom-алокатори.

Мова має лаконічний і легко читаємий синтаксис. Вона нагадує Go, але з суворішою типізацією і більшим акцентом на функціональний стиль. V також підтримує модулі, інтерфейси, уніфіковану систему пакетів, генерацію документації та вбудовані інструменти для форматування та тестування коду.

Ще одна цікава особливість: фреймворк для створення кросплатформних графічних інтерфейсів користувача з єдиною кодовою базою для Windows, macOS, Linux, Android та Web (через WebAssembly). Також розробляються інструменти для компіляції в JavaScript, що дозволяє використовувати V у веб-розробці.

Попри молодість, V уже застосовується для створення CLI-утиліт, серверів, системних бібліотек і навіть невеликих ОС. Мова має потенціал для заміни C/C++ у проектах, де важлива швидкість, компактність і безпечне управління ресурсами. Оскільки компілятор V написаний на самій мові V (bootstrapping), це демонструє зрілість її функціональності.

Однак V ще не є завершеною або широко поширеною мовою. Деякі функції перебувають на стадії розробки або бета-тестування. Її екосистема поки що обмежена, але активно зростає, і вона підтримується невеликою, але ентузіастичною спільнотою.

Bosque [19] – це експериментальна мова програмування, розроблена Microsoft Research під керівництвом Марка Марселя (Mark Marron) з метою створення абсолютно нового підходу до розробки програм: передбачуваного, чистого та формально керованого. Мова з'явилася у 2019 році й вирізняється тим, що відмовляється від традиційних імперативних конструкцій, таких як змінні, цикли, і побудови з побічними ефектами. Назва «Bosque» (з іспанської – «ліс») символізує структуру з чіткою архітектурою та відсутністю «хаосу» у вигляді непередбачуваних шляхів виконання. Основна мета Bosque – спростити розробку, верифікацію та обслуговування програмного забезпечення шляхом виключення джерел помилок, які часто зустрічаються в існуючих мовах. Серед них – неконтрольовані цикли, зміни глобального стану, виклики з побічними ефектами тощо. Замість цього Bosque пропонує функціональний стиль програмування, в якому відсутнє поняття змінної в класичному сенсі – усі значення є незмінними, що полегшує аналіз і перевірку коду.

Однією з ключових концепцій мови є активне усунення мовних конструкцій, які додають складності, але не мають достатнього виправдання з точки зору користі. Зокрема, мова не підтримує цикли for чи while, натомість використовує узагальнені конструкції обробки колекцій (наприклад, map, filter, fold), що дозволяє зосередитись на суті обчислень без контролю над ітераторами. Bosque також запроваджує концепцію value flow programming - програмування, яке не побудоване на зміні стану, а на передачі та трансформації значень. Такий підхід полегшує побудову аналізаторів, статичну перевірку, автоматичну оптимізацію, та формальну верифікацію програм.

Синтаксис мови нагадує JavaScript чи TypeScript, що робить її знайомою для багатьох сучасних розробників. При цьому Bosque має строго типізовану структуру, що дозволяє компілятору надійно виявляти помилки та попереджати про небезпечні дії ще до запуску програми. Також мова підтримує конструкції для обробки помилок, умовного виконання, узагальнення (generics), типові алгебраїчні типи (union, record, enum) і механізми pattern matching.

Bosque [19] не призначена для створення традиційних високопродуктивних систем або драйверів, як C або Rust. Натомість вона орієнтована на розробку надійного прикладного програмного забезпечення, у якому важлива читабельність, правильність і формальна перевірка. Це може бути бізнес-логіка, веб-застосунки, алгоритмічні служби, що потребують точного контролю та високої якості.

Оскільки мова експериментальна, вона поки що не має широкої підтримки в індустрії. Існує інтерпретатор і базовий компілятор, що дозволяє запускати програми Bosque, а також документація й специфікація синтаксису. Основна увага наразі приділяється дослідженню нових парадигм програмування, а не промисловій адаптації.

Darklang [14] це інноваційна мова програмування та одночасно повноцінна платформа для створення бекенд-застосунків, що кардинально змінює підхід до розробки програмного забезпечення. Вона була започаткована колишнім розробником з Facebook Полом Біггером (Paul Biggar) і набула популярності як спроба усунути складність DevOps, інфраструктури, CI/CD, серверного розгортання та взаємодії з базами даних. Головна ідея Darklang – зробити розробку вебзастосунків такою ж простою, як редагування тексту в Google Docs.

На відміну від традиційних мов, Darklang – це не лише мова, а ціла інтегрована хмара, що включає інтерпретатор, редактор коду, базу даних, серверну логіку, логування, CI/CD та миттєве розгортання в одному

середовищі. Це дозволяє розробникам уникати налаштування серверів, docker-файлів, CI-пайплайнів і зовнішніх баз даних. Код автоматично оновлюється й виконується в реальному часі – без очікування компіляції чи деплою.

Мова Darklang [14] заснована на функціональному підході, має простий та передбачуваний синтаксис, схожий на Elm або ReasonML. Усі функції чисті, а всі побічні ефекти – контрольовані. Вбудовані абстракції підтримують HTTP-запити, фонові задачі, взаємодію з базами даних та обробку подій без потреби в налаштуванні фреймворків. Важливо, що кожна функція автоматично логуються, версіюється і може бути викликана з frontend через URL, без окремого API Gateway чи proxy.

Одна з ключових функцій Darklang – live programming: розробник бачить виконання кожної частини коду в режимі реального часу, із візуалізацією значень вхідних і вихідних параметрів. Це дозволяє миттєво тестувати, відлагоджувати й покращувати логіку, не покидаючи браузерного редактора. Усе це доповнюється повною інтроспекцією, яка дає змогу відстежувати шлях виконання функцій на живих даних.

Darklang також пропонує інфраструктуру як код, яка працює без видимих серверів. Весь бекенд-код автоматично компілюється та виконується в sandbox-оточенні на хмарі Darklang. Ви не турбуєтесь про масштабування, безпеку, доступність чи інциденти – усе це бере на себе платформа.

Проте Darklang має свої обмеження. По-перше, це закрите середовище: код, який пишеться в Darklang, не може бути легко перенесений або запущений поза платформою. Це створює vendor lock-in, що може бути критично для деяких проєктів. По-друге, мова ще молода й активно розвивається, тому її функціональність та документація можуть бути неповними або нестабільними.

Darklang особливо підходить для стартапів, прототипування, MVP та швидкої ітеративної розробки. Це інструмент для невеликих команд, які хочуть зосередитися на бізнес-логіці замість управління інфраструктурою.

Jai [12] – це експериментальна мова програмування, розроблена Джонатаном Блоу , відомим геймдизайнером і програмістом, автором ігор Braid та The Witness. Метою створення Jai є створення ідеального інструмента для розробки високопродуктивного, передбачуваного і контрольованого програмного забезпечення – передусім у галузі розробки ігор, рушіїв, графічних систем, а також іншого системного ПЗ. Мова орієнтована на максимальну продуктивність і контроль над пам'яттю при збереженні простоти й читабельності коду. Jai не використовує збирач сміття, а натомість надає розробнику прямий контроль над виділенням і звільненням ресурсів, подібно до C, але з кращими інструментами для цього — зокрема, через механізм зоною пам'яті.

Одна з основних переваг Jai – компіляція на льоту, що дозволяє програмам генерувати або змінювати власний код ще під час компіляції. Це надає розробнику потужні засоби метапрограмування без складності C++-шаблонів чи макросів. Компілятор Jai підтримує виконання будь-якої частини коду під час компіляції, що дозволяє створювати структури даних, алгоритми або таблиці без окремих генераторів чи препроцесорів.

Jai уникає об'єктно-орієнтованого програмування у класичному розумінні. Замість цього вона підтримує прості й потужні структури даних, композицію замість спадкування, функціональний стиль з можливістю передачі поведінки через функції або делегати. Це дозволяє уникати перевантажених ієрархій та забезпечує краще управління залежностями в коді.

Синтаксис Jai простий, але виразний. Мова запозичує інтуїтивні конструкції з Pascal, C, Python та інших мов, але уникає складного синтаксису або багатозначності. Важливою особливістю є відсутність undefined behavior, що властиве C/C++, Jai прагне зробити поведінку коду максимально передбачуваною і контрольованою навіть при низькорівневих операціях.

Jai підтримує модульну структуру коду, автоматичне виведення типів, шаблони даних, управління пам'яттю на рівні платформи, роботу з SIMD, багатопотоковість, а також пряму інтеграцію з API операційної системи або апаратного забезпечення. Вона не має стандартної бібліотеки в традиційному сенсі – все будується як гнучкі інструменти, які розробник використовує за потребою, без примусу.

Мова поки що не є відкритою або стабільною: її розробка ведеться в закритому режимі невеликою командою. Доступ до неї отримують обмежені групи бета-тестувальників, переважно з індустрії відеоігор. Головний фокус поки що зосереджений на стабільності, компіляторі, інтеграції з low-level API та демо-проєктах, які демонструють її можливості.

F* – це сучасна функціональна мова програмування з залежними типами, розроблена для написання формально перевіреного коду, особливо в сферах, де критично важлива безпека, правильність та математична доказовість програм. Вона поєднує синтаксис, схожий на OCaml або ML, з потужною системою типів, яка дозволяє формалізувати властивості програм і доводити їх правильність на рівні компілятора. Розробка F* ведеться в Microsoft Research у співпраці з INRIA (Франція) та іншими академічними установами, і активно використовується в проєктах із криптографії та інформаційної безпеки.

Головна ідея F* полягає у тому, що розробник може не просто написати функцію, а формально описати її властивості у вигляді логічних умов – передумов, постумов, інваріантів, тощо. Компілятор F* потім автоматично перевіряє ці властивості за допомогою SMT-солвера (зазвичай Z3), що дозволяє виявляти помилки ще до запуску програми. Таким чином, програма на F* – це одночасно і реалізація, і її математичне доведення.

Система типів F* заснована на залежних типах – тобто типах, що можуть залежати від значень. Це дозволяє, наприклад, визначити тип масиву певної довжини або функцію, яка приймає лише непорожній список. Такі конструкції особливо цінні в системах, де необхідна сувора перевірка правильності – криптографічні алгоритми, протоколи передачі даних, обчислювальні ядра.

Ще одна важлива концепція F* – це ефекти, які дозволяють контролювати побічні ефекти (наприклад, доступ до пам'яті, стану або IO) у стилі чистого функціонального програмування. Це дозволяє зберігати коректність доказів навіть у програмах, які мають взаємодію з навколишнім середовищем.

F* підтримує екстракцію коду в C, OCaml, F# та інші мови, що дозволяє використовувати доведений код у реальних системах. Наприклад, HACL* – криптографічна бібліотека, написана на F* і використана у Firefox та інших проектах – автоматично компілюється у C-код, який пройшов формальну перевірку на коректність, відсутність переповнення буфера тощо.

Синтаксис F* подібний до OCaml, однак є складнішим через розширені можливості системи типів. Вивчення мови потребує підготовки з логіки, теорії типів і функціонального програмування, тому вона насамперед орієнтована на академічну спільноту, формальну верифікацію, захищене програмування, або критичні галузі (авіація, оборона, фінанси, блокчейн).

Станом на 2025 рік, F* є одним із найпотужніших інструментів для доведення правильності коду, і продовжує впливати на розвиток інших мов.

Elvish – це сучасна кросплатформна мова програмування оболонки (shell), яка поєднує в собі можливості традиційних Unix shell'ів (наприклад, Bash або Zsh) з елементами сучасних мов програмування, таких як Python, Go чи Lisp. Вона створена для інтерфейсної взаємодії з операційною системою, автоматизації завдань, обробки даних та інтерактивного керування середовищем – але з набагато вищим рівнем виразності, передбачуваності та структурованості, ніж у класичних shell-мов. Elvish вирізняється уніфікованою системою типів – на відміну від Bash, де все є рядками, у Elvish кожне значення має конкретний тип: список, карта (словник), функція, файл, число тощо. Це дозволяє уникати класу помилок, пов'язаних із неправильною обробкою тексту, і робить скрипти надійнішими та легшими для відлагодження. Elvish підтримує повноцінні структури даних, включно з вкладеними колекціями, а також можливість програмного доступу до них без складних парсингів. Ще однією характерною рисою є реактивний і живий інтерфейс командного рядка. Elvish має вбудовану підтримку автодоповнення, інспекції змінних, командного редагування, історії з фільтрацією, підсвічування синтаксису – усе це інтегровано прямо в оболонку. Elvish дозволяє зручно створювати композиційні команди: вивід однієї функції можна безпосередньо передавати до іншої, не лише як текст, а як структуроване значення.

Мова підтримує функціональне програмування, включно з вищими порядками функцій, замиканнями (closures), чистими функціями без побічних ефектів, а також обробку потоків даних без необхідності створення тимчасових файлів. Наприклад, дані можна передавати через пайплайни як списки або об'єкти, а не як текстові рядки, що полегшує багато типових задач адміністрування. Elvish має уніфіковану REPL-платформу – усі введені команди одразу виконуються в інтерактивному середовищі, з можливістю змінювати контекст, динамічно переглядати змінні, повторно запускати функції або зберігати стан сеансу. Це дуже зручно для дослідницької роботи, налагодження сценаріїв або просто для щоденного використання.

Скрипти Elvish підтримують модульну структуру, підключення сторонніх бібліотек, оголошення змінних і функцій із чіткою областю видимості. Окрім цього, мова підтримує асинхронне виконання, що дозволяє обробляти дані в потоках і будувати реактивні пайплайни. На відміну від Bash або Zsh, Elvish є повноцінною скриптовою мовою, здатною замінити Python або Perl у багатьох системних задачах, з більшим рівнем інтерактивності й меншим порогом помилок.

Сучасні мови програмування орієнтовані на підвищення безпеки, ефективності та простоти розробки, підтримуючи парадигми функціонального, об'єктно-орієнтованого й реактивного програмування. Вони активно адаптуються до потреб штучного інтелекту, хмарних обчислень і багатопотокових систем.

Джерела.

1. Blandy, J., Orendorff, J., & Tindall, L. (2021). Programming Rust: Fast, Safe Systems Development (2nd ed.). O'Reilly Media. 784 с.
2. Klabnik, S., & Nichols, C. (2021). The Rust Programming Language (2nd ed.). No Starch Press. 552 с.
3. Donovan, A. A., & Kernighan, B. W. (2023). The Go Programming Language (2nd ed.). Addison-Wesley. 480 с.
4. Cox, R. (2021). Go and the future of programming. ACM Queue, 19(1), 30-45. https://doi.org/10.1145/3441962.3441970 16 с.
5. Jemerov, D., & Isaksson, S. (2021). Kotlin in Action (2nd ed.). Manning Publications. 384 с.

6. Nystrom, R. (2021). Programming Kotlin: Create Elegant, Expressive, and Performant JVM and Android Applications. Pragmatic Bookshelf. 496 с.
7. Bezanson, J., Edelman, A., Karpinski, S., & Shah, V. B. (2020). Julia: A fresh approach to numerical computing. SIAM Review, 62(1), 45–98. https://doi.org/10.1137/141000671 54 с.
8. Perkel, J. M. (2021). Julia: The great scientific computing experiment. Nature, 589, 197–198. https://doi.org/10.1038/d41586-021-00074-9 2 с.
9. Kelley, A. (2022). Systems Programming with Zig. [Онлайн ресурс]. https://ziglang.org бл. 320 с. (оцінка за змістом онлайн-версії)
10. Lattner, C. (2023). Why Zig? Comparing Modern Systems Languages. Modern Compilers Conf. [Матеріали конференції]. 15 с.
11. Carruth, C. (2022). Carbon Language: An experimental successor to C++. CppNorth 2022. https://github.com/carbon-language/carbon-lang.
12. Medvednikov, A. (2022). The V Programming Language: Simple. Fast. Safe. https://vlang.io бл. 250 с.
13. Blow, J. (2023). Jai Language Overview. https://jai-lang.org ~50 cBalasubramaniam, G., & Bierman, G. (2020). The Bosque Programming Language. Microsoft Research. https://github.com/microsoft/BosqueLanguage 80 с.
14. Trott, P. (2021). The Darklang Vision: Eliminating Accidental Complexity. Future of Coding Symposium. 100 с.
15. Protzenko, J., Swamy, N., et al. (2022). Secure Programming with F*. Springer. 312 с.
16. Horrocks, A. (2022). Vale: Towards memory-safe systems programming without garbage collection. https://vale.dev бл. 150 с.
17. Elvish Developers. (2021). Elvish Shell Programming Guide. https://elv.sh 110 с.
18. Tikhomirov, S. (2022). Mastering Ethereum Smart Contracts: The Ultimate Guide to Building Secure, Fast, and Efficient Blockchain Applications with Solidity (2nd ed.). Packt Publishing. 442 с.
19. Balasubramaniam, G., & Bierman, G. (2020). The Bosque Programming Language: A Grounded Approach to Software Development. Microsoft Research. Retrieved from https://github.com/microsoft/BosqueLanguage/
20. Swamy, N., Chen, J., Fournet, C., Strub, P.-Y., Bhargavan, K., & Yang, J. (2020). Secure Distributed Programming with Value-Dependent Types. In Proceedings of the 16th ACM SIGPLAN International Conference on Functional Programming (pp. 266-278).

УДК 621.65

Яременко Андрій Леонідович
Поліський національний університет
Науковий керівник: Савченко Василь Миколайович, кандидат технічних наук, доцент,
Поліський національний університет

ПІДВИЩЕННЯ ДОВГОВІЧНОСТІ ТА ЕКОНОМІЧНОЇ ЕФЕКТИВНОСТІ ВІДЦЕНТРОВИХ НАСОСІВ У СИСТЕМАХ ЗРОШЕННЯ

У тезах розглядається економічна доцільність удосконалення конструкції відцентрового насоса шляхом впровадження біонічного дизайну лопатей. Проведено техніко-економічний аналіз, який показав приріст довговічності насоса на 5,5% та зниження витрат на експлуатацію. Встановлено позитивний вплив модернізації на термін служби та енергетичну ефективність обладнання. Ключові слова: насос, довговічність, енергоефективність, ремонт, економія, зрошення.

Ефективність функціонування систем зрошення значною мірою залежить від технічного стану насосного обладнання. Однією з головних проблем при експлуатації відцентрових насосів є їх недостатня довговічність, обумовлена впливом кавітації та зношення конструктивних елементів. Часті простої та ремонти призводять до фінансових втрат аграрних підприємств.

В роботі проведено розрахунок приросту довговічності насоса в результаті впровадження біонічного дизайну лопатей. Зменшення зони низького тиску на 11% сприяє зниженню кавітаційного зносу, що продовжує строк служби обладнання з 15000 до 15825 годин. Такий приріст дозволяє зменшити частоту замін і ремонтів, а також збільшити надійність зрошувальних систем.

Окремо проведено аналіз витрат на енергію та обслуговування. Розрахунки показали, що впровадження нової моделі насоса дозволяє зменшити загальні витрати на 4068 грн за один робочий цикл. Зниження витрат досягається як за рахунок меншої кількості ремонтів, так і через покращену гідравлічну ефективність, яка зменшує енергоспоживання.

Таким чином, запропоновані зміни мають не лише технічне, а й вагоме економічне обґрунтування. Удосконалення конструкції відцентрових насосів підвищує ефективність аграрних підприємств і забезпечує стійке функціонування систем зрошення в умовах енергетичних та ресурсних обмежень.

Джерела.
1. Смирнов А. В. Довговічність насосного обладнання. Київ: Політехніка, 2022. – 234 с.
2. Літвін І. І., Ткаченко О. П. Економічна оцінка модернізації насосних систем. Львів: ЛНУ ім. Івана Франка, 2021. – 187 с.
3. KSB Group. Офіційний сайт компанії KSB. Доступно: https://www.ksb.com

Наукове видання
КРЕАТИВНИЙ ПРОСТІР ™
Науковий журнал
Електронне видання
№ 27
ISSN 2710-1177 (online). DOI: 10.61718/crp
Видання змішаними мовами
Відповідальний секретар – Кучина Т. І.
Формат 60x90 1/8, А4. Гарнітура «Times New Roman»
Авторські аркуші – 8,3
Опубліковано на основі ліцензії
Creative Commons Attribution License

Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції: ДК № 8013 від 22.11.2023. Зареєстровано у Global Register of Publishers. Ідентифікатор видавця 7886.
Видавцем використовується офіційно зареєстрований знак для товарів та послуг (торгова марка) Креативний простір ™ Свідоцтво № 342949 видане Державною організацією «Український національний офіс інтелектуальної власності та інновацій».
Видавець СГ НТМ «Новий курс» (наукова установа)
Пр. Перемоги, 77, оф. 179, м. Харків, 61174, Україна.
Тел.: +380500301905, +380962250903. Telegram, Viber: +380970440309 Сайт: www.newroute.org.ua

Запрошуємо
Національна академія наук і мистецтв України www.newroute.org.ua/nanmu

Наші пропозиції
Збірник «Соціально-гуманітарний вісник» www.newroute.org.ua/sgv
Журнал «Креативний простір» www.newroute.org.ua/crp
Часопис «Нотатки сучасної науки» www.newroute.org.ua/nsn
Журнал «Advanced top technology» www.newroute.org.ua/att
Колективна монографія www.newroute.org.ua/kmnk
Підвищення кваліфікації www.newroute.org.ua/pkv
Видавництво www.newroute.org.ua/pub
Творчі конкурси www.newroute.org.ua/art

Приєднуйтесь
Підписка http://surl.li/vvfqkp
Фейсбук-сторінка www.facebook.com/newroute1989
Телеграм https://t.me/newroute1989
Інстаграм www.instagram.com/newroute1989
Вайбер http://surl.li/nbtqz
Фейсбук-група www.facebook.com/groups/secnr

Будемо раді подальшій співпраці!